Multilayer Neuroadaptive Control of Two DOF Robot with Deadzone

Thesis submitted by

Ujjwal
2K19/C&I/12

under the guidance of

Prof. Mini Sreejeth

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY
IN
CONTROL AND INSTRUMENTATION

DEPARTMENT OF ELECTRICAL ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

2021

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

DECLARATION

I, **Ujjwal**, Roll No. 2K19/C&I/12 student of M.Tech. Control and Instrumentation(C&I), hereby declare that the dissertation/project titled **"Multilayer Neuroadaptive Control of Two DOF Robot with Deadzone"** under the supervision of Prof. Mini Sreejeth of Electrical Engineering Department, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology has not been submitted elsewhere for the award of any Degree.

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

This is to certify that the project titled **"Multilayer Neuroadaptive Control of Two DOF Robot with Deadzone"** created by **Ujjwal**, Roll No. 2K19/C&I/12 student of M.Tech. Control and Instrumentation(C&I), under my supervision and submitted to Electrical Engineering Department, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology and has not been submitted elsewhere for the award of any Degree.

Prof. Mini Sreejeth
Professor
Department of Electrical Engineering
Delhi Technological University

Place: New Delhi

Date: 12th July 2021

ACKNOWLEDGEMENTS

I am highly grateful to the Department of Electrical Engineering, Delhi Technological University(DTU) for providing this opportunity to carry out this project work.

The constant guidance and encouragement received from my supervisor Prof. Mini Sreejeth of Department of Electrical Engineering, DTU, has been of great help in carrying my present work and is acknowledged with reverential thanks.

Finally, I would like to expresses gratitude to all faculty members of Electrical Engineering Department, DTU for their intellectual support in my M.tech study at DTU.

Ujjwal

ABSTRACT

Robots generally suffer from nonlinearities such as deadzone, backlash and deadband. The inherent coupling effect between links of the robot along with nonlinearities puts considerable challenge to control the position of end effector. A two link manipulator is a basic robot in which most of the controllers are implemented to get a fair idea of the control performance. In this paper we use a adaptive neural network for controlling a two link rigid planar manipulator with deadzone at input. The deadzone at the input is not static but dynamic. To remove the effect of deadzone, inverse deadzone is used which just adds an offset to the control input. The offset is greater than or equal to the deadzone width and hence there could be some extra input going to the actuator which is modeled as disturbance in the system. Performance of the Multilayer Neuroadaptive (MN) controller is compared with typical Single Layer Neuroadaptive controller, PID and classical Adaptive Controller in simulation. Error norms and other parameters are calculated for different control performance are given to compare quantitatively.

Keywords: Neuroadaptive; Deadzone; Inverse Deadzone; Manipulator;

Contents

List of Tables

List of Figures

ABBREVIATIONS

EL	Euler-Lagrange
IE	Initial Excitation
PE	Persistence of Excitation
PD	Positive Definite
ND	Negative Definite
EMK	Exact Model Knowledge
UUB	Uniformly Ultimately Bounded
UGS	Uniformly Globally Stable
UGES	Uniformly Globally Asymptotically Stable
LS	Least Squares
RMS	Root Mean Square

NOTATION

q	trajectory followed by the robot
q_d	desired trajectory
θ	Parameter vector
Φ	Regressor
τ	Control torque input
e	tracking error in radians
r	filtered tracking error
ε	Estimation error
Γ_θ	Adaptation gain
V	Lyapunov equation
M	Inertia matrix in euler-lagrangian equation
C	Centripetal-coriolis matrix in euler-lagrangian equation
F	Friction term in euler-lagrangian equation
D_z	Excess torque for deadzone compensation
τ_d	Bounded external disturbance

Chapter 1

Introduction

Robot manipulators are complicated multi-input–multi-output (MIMO) system, with strongly coupled nonlinear characteristics. Given structured uncertainties, such as inertia, Coriolis, gravity, and friction torque, the position tracking accuracy of a robot and the accuracy of joint torque information are seriously affected. So there is need for fast and reliable controllers that can handle the nonlinearities and give good steady-state and transient response. The dynamics of the robot are modelled by euler-lagrange equation. Several controllers have been designed for uncertain euler-lagrange systems like classical adaptive control [1], Composite adaptive control [8], neuroadaptive control [6]. Neural network possesses an nonlinear structure which is suitable for estimation and control of unknown nonlinear systems and thus has been used widely for systems with deadzone [3]. In [5] state-feedback control is used to control robot and the deadzone effect is approximated by a Radial Basis Function Neural Network (RBFNN). In [11] neural network is used to control manipulators with input deadzone and output constraints. In [2] multilayer neural network is trained to identify and control the system using a modified back-propogation algorithm. In [10] a nonlinear compensator using neural networks is proposed for manipulator control. In this paper a multilayer neuroadaptive controller is simulated on a two link manipulator with input deadzone. Most of the existing papers deal with single layer neuroadaptive controllers but this paper considers its variant multilayer neuroadaptive controller with sigmoidal activation function, MN is shown to be better in terms of reducing error than single layer adaptive controller, PID, and classical adaptive controller. Novelty of the work lies in the fact that rather than estimating deadzone by neural network, a much simpler approach is shown. The deadzone along with inverse deadzone is modelled as disturbance at the input. All the simulations were done on Matlab. Tracking performance of the manipulator are shown and error norms are calculated for comparison.

1.1 Problem Statement

Many controllers are already designed for manipulator robots. These controllers will also work on our system but the performance would be worse when the direction of the motor changes. This is because at this point the effect of deadzone is more. So the aim of this project is to model the robot include the deadzone and then accordingly design neural network controller to work on it.

We have used various controllers in this project. Intially theses controllers were tested on one-link manipulator which were

- PID
- Classical Adaptive Controller
- State Feedback Controller

Then after analysing the results we proceeded to test multiple adaptive controllers on two-link manipulator which were

- PID
- Classical Adaptive Controller
- Single Layer Neuroadaptive Controller
- Multi Layer Neuroadaptive Controller

1.2 Methodology

First of all the simple dynamics of a loaded motor with input deadzone is used to test the adaptive algorithm. Based on it's performance the same or better result is tried to be obtained in two-link manipulator.

The dynamics of rigid planar two-link manipulater is modelled with euler-lagrange equation. The deadzone at the input is of varying positive and negative widths. The upper limit of the deadzone width is known and is used to design inverse deadzone which adds an fixed offset greater than or equal to the deadzone width to the control input . The change is incorporated as disturbance at the input. The parameters of the robot are not known. For neuroadaptive controller the dynamics of the robot is modelled in neural network and its weights are estimated online. The control objective is to track the trajectory by links of the robot.

1.3 Organization of Thesis

Next chapter introduces the robots, their dynamics along with some properties of euler-langrangian systems. The controller will be designed based on this information.

Third chapter is on deadzone modeling which will then be included in the robot dynamics. This is then considered as deadzone compensated dynamics.

Fourth and fifth chapter has controller design and stability proofs of various adaptive controllers implemented on single and double link manipulator. Results are then compared.

Sixth chapter concludes the result obtained and suggests future work that could be made in the controller for improvement.

The appendix has the Simulink blocks along with matlab codes of the various controllers implemented on the model.

Chapter 2

System Description

2.1 Introduction

For designing any controller for a system, first the system need to be modeled mathematically. Relation between input and ouput should be defined based on which a good controller is derived. In this project Euler-Lagrangian equations are used to define the dynamics of the manipulator. The dynamics is then shown as a combination of regressor and unknown parameters. This type of model is used for adaptive controller design.

2.2 Model Description

The general Euler-Lagrange dynamics is of the form

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) \tag{2.1}$$

where $\tau(t) \in \mathbb{R}^n$ is input torque, $q(t) \in \mathbb{R}^n$, represents the generalised coordinates of the system. $M(q) \in \mathbb{R}^{n\times n}$ symmetric positive definite inertial matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is centripetal-Coriolis matrix, $G(q) \in \mathbb{R}^n$ is the gravity vector, and $F(\dot{q}) \in \mathbb{R}^n$ is represents friction vector. The initial conditions are defined as $q\left(t_0\right) = q_0$ and $\dot{q}\left(t_0\right) = \dot{q}_0$. The states $q(t)$ and $\dot{q}(t)$ are measurable and $M(q)$, $C(q,\dot{q})$, $F(\dot{q})$, $G(q)$ are locally Lipschitz in their respective arguments and contain uncertain unknown parameters. Following are some important properties of the above dynamics, which will be used in control design and stability proofs [4], [9].

Property 2.1: $M(q) > 0 \ \forall q$ and is bounded by the following inequality

$$\mu_1 I \leq M(q) \leq \mu_2 I\,, \quad \mu_1\,, \mu_2 \in \mathbb{R}_{>0} \tag{2.2}$$

Property 2.2: There exists a skew symmetric relationship

$$y^{\mathrm{T}}\left(\dot{M} - 2C\right)y = 0, \quad \forall y \in \mathbb{R}^n \tag{2.3}$$

Property 2.3: The centripetal-Coriolis matrix, gravity vector, and the friction vector satisfy the following bounds.

$$\|C(q,\dot{q})\| \leq c\|\dot{q}\| \tag{2.4}$$

$$\|G(q)\| \leq \bar{g} \tag{2.5}$$

$$\|F(\dot{q})\| \leq f_1 + f_2\|\dot{q}\| \tag{2.6}$$

where c, $\bar{g}$, f_1, f_2 are positive constants.

Property 2.4: The EL dynamics in (2.1) is linear in parameter, so, it can be expressed as $\Phi(q,\dot{q},\ddot{q})\theta$, where Φ is the regressor matrix and θ is the unknown parameter vector.

2.3 Single-Link Manipulator

Initially a single-link manipulator is modeled to test the controller. Single link manipulator is basically a loaded motor and can be described by the dynamics:

$$\tau = J\ddot{q} + B\dot{q} \tag{2.7}$$

Here q being the angular position of the link, J is the moment of inertia and B is the viscous friction. Torque τ is the input given to the system. This could also be written in the form of first order transfer function

$$\frac{\omega(s)}{\tau(s)} = \frac{K}{Ts+1} \tag{2.8}$$

Here ω is angular velocity of the link, $K = 1/B$ and $T = J/B$.

2.4 Two-Link Manipulator

For experiment, a planar 2-DOF RR (revolute joints) manipulator is used. The euler-lagrangian dynamics for the robot is

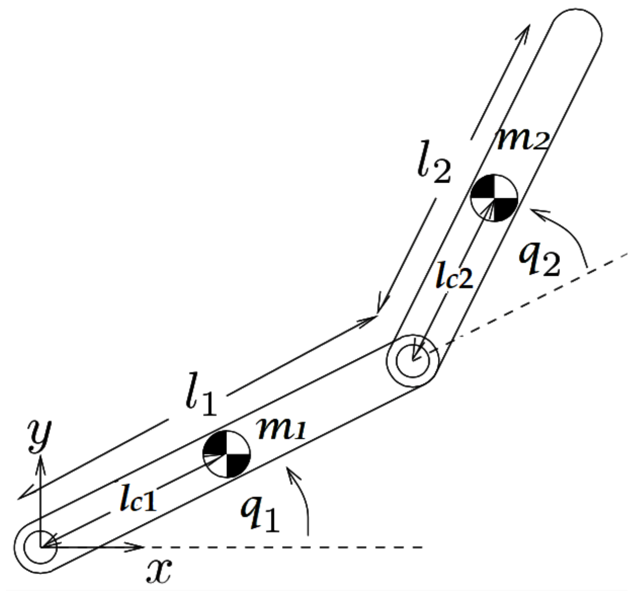

Figure 2.1: Rigid planar two-link manipulator

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \underbrace{\begin{bmatrix}p_1+2p_2c_2 & p_3+p_2c_2\\ p_3+p_2c_2 & p_3\end{bmatrix}}_{M(q)}\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\end{bmatrix} + \underbrace{\begin{bmatrix}-p_2s_2\dot{q}_2 & -p_2s_2(\dot{q}_1+\dot{q}_2)\\ p_2s_2\dot{q}_1 & 0\end{bmatrix}}_{C(q,\dot{q})}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix}$$
$$+\underbrace{\underbrace{\begin{bmatrix}f_{v1} & 0\\ 0 & f_{v2}\end{bmatrix}}_{F_v}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix}+\underbrace{\begin{bmatrix}d_{z1}\\ d_{z2}\end{bmatrix}}_{D_z}}_{F(\dot{q})} \tag{2.9}$$

Where the parameters are

$$\begin{aligned} p_1 &= I_1 + I_2 + m_2 l_1^2\\ p_2 &= m_2 l_1 l_{c2}\\ p_3 &= I_2\\ c_2 &= cos(q_2),\ s_2 = sin(q2) \end{aligned} \tag{2.10}$$

Here I_1, I_2 are the moment of inertia of the links about the axis of rotation and m_1, m_2 are the masses of the links. The mass of first link also includes the mass of motor placed on it. l_1, l_2 are the length of the links and L_{c1}, L_{c2} are the position of center of mass of the links as shown in Fig.2.1. There is no $G(q)$ term as it is a planar manipulator. $F_v \in \mathbb{R}^{2\times2}$ is viscous friction coefficient matrix. D_z is excess torque for deadzone compensation that is discussed in section 3.2. D_z and F_v together constitute the friction vector $F(\dot{q})$.

The equation (2.9) is linear in parameter and thus can be written as

$$\tau = \Phi^{\mathrm{T}}(\ddot{q}, \dot{q}, q)\theta \tag{2.11}$$

Where Φ is the known regressor matrix and θ is the unknown parameter vector:

$$\theta^{\mathrm{T}} = \begin{bmatrix} p_1, & p_2, & p_3, & f_{v1}, & f_{v2}, & d_{z1}, & d_{z2} \end{bmatrix} \tag{2.12}$$

2.5 Conclusion

In this chapter, Euler-Lagrangian model of two link and single link manipulator were derived from the general Euler-Lagrangian equations. Important properties of the dynamics are mentioned which will be later used for deriving adaptive laws.

The dynamics was written in the form of regressor matrix and unknown parameter vector.

Chapter 3

Deadzone Modeling

3.1 Introduction

Like all mechanical robots with actuators, the manipulator in discussion also has deadzone at input. This deadzone need to be modeled so that the controller compensated for it and takes the extra variable into consideration. This will make for a better controlling action.

The regressor matrix of the dynamics has acceleration term which is not available. So dynamics filtering is done to remove that term. This will also remove noise and the need to calculate acceleration by differentiation.

3.2 Deadzone Compensation

In mechanical systems, there is always deadzone present due to friction forces. Considering the deadzone width changes within a range and is incorporated in model for accurate control action. Let the input deadzone be (3.1)

$$\tau_{in} = \begin{cases} u - d^{+} & \text{for} \quad u > d^{+} \\ 0 & \text{for} \quad d^{-} \le u \le d^{+} \\ u - d^{-} & \text{for} \quad u < d^{-} \end{cases} \tag{3.1}$$

Where u is the control input and τ_{in} is the actual torque input that affects the system.

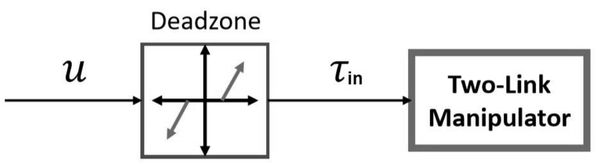

Figure 3.1: Deadzone at the input

To remove the effect of deadzone we add an inverse deadzone in the controller. The deadzone widths are not strictly constant and vary slightly within a range. The inverse deadzone is constructed with the upper limit of the deadzone widths $\overline{d}$. Now after this modification, let τ be the control input, then

$$u = \begin{cases} \tau + \overline{d^{+}} & \text{for} \quad \tau > 0 \\ 0 & \text{for} \quad \tau = 0 \\ \tau + \overline{d^{-}} & \text{for} \quad \tau < 0 \end{cases} \tag{3.2}$$

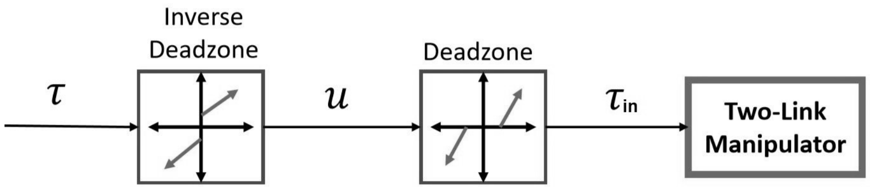

Figure 3.2: Inverse deadzone to compensate the input deadzone

Let, $D_z = d - \bar{d}$. D_z is negaitve as $\bar{d} \geq d$. So the actual torque input being sent to the actuators after applying inverse deadzone is

$$\begin{aligned} \tau_{in} &= \tau + \bar{d} - d \\ \implies \quad \tau &= \tau_{in} - (\bar{d} - d) \\ \implies \quad \tau &= M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + \underbrace{F_v(\dot{q}) + D_z}_{F(\dot{q})} \end{aligned} \tag{3.3}$$

Similarly the equivalent deadzone compensated 1-DOF manipulator dynamics becomes

$$\tau = J\ddot{q} + B\dot{q} + D_z \tag{3.4}$$

3.3 Dynamic Filtering

Regressor Φ in (2.11) depends on the value of $\ddot{q}$. Acceleration can be computed by differentiating velocity but doing so would introduce noise. To remove the dependency on acceleration data, regressor Φ and torque τ are passed through a low pass filter.

$$\dot{\Phi}_F = -p_F \Phi_F + \Phi \qquad \Phi_F(t_0) = 0 \tag{3.5}$$

$$\dot{\tau}_F = -p_F \tau_F + \tau \qquad \tau_F(t_0) = 0 \tag{3.6}$$

where $p_F > 0$ is a scalar filter gain. The above filter equations can be solved for getting the filtered regressor $\Phi_F(t, q, \dot{q}) \in \mathbb{R}^{n \times p}$ and the filtered torque $\tau_F \in \mathbb{R}^n$ as shown below

$$\Phi_F(t) = \exp\{-p_F t\} \int_{t_0}^{t} \exp\{p_F s\} \Phi(s) ds \tag{3.7}$$

$$\tau_F(t) = \exp\{-p_F t\} \int_{t_0}^{t} \exp\{p_F s\} \tau(s) ds \tag{3.8}$$

Using (2.11) in (3.8) the following relation is deduced.

$$\Phi_F^{\mathrm{T}}(t, q, \dot{q})\theta = \tau_F \tag{3.9}$$

Although $\tau_F(t)$ can be obtained directly by solving (3.6), $\Phi_F(t, q, \dot{q})$ cannot be solved from (3.5) because $\Phi(q, \dot{q}, \ddot{q})$ is dependent on $\ddot{q}$ which is not known. However, $\Phi(q, \dot{q}, \ddot{q})$ can be split into measurable and unmeasurable terms as

$$\Phi^T\theta = \Phi_1^T(q, \ddot{q})\theta + \Phi_2^T(q, \dot{q})\theta \tag{3.10}$$

where Φ_1 and Φ_2 are the following regressors

$$\Phi_1(q, \ddot{q})\theta = M(q)\ddot{q} \tag{3.11}$$

$$\Phi_2(q, \dot{q})\theta = C(q, \dot{q})\dot{q} + G(q) + F(\dot{q}) \tag{3.12}$$

Since $\Phi_2(q, \dot{q})$ is known, the differential equation below can be solved online.

$$\dot{\Phi}_{F2} = -\rho_F \Phi_{F2} + \Phi_2, \quad \Phi_{F2}(t_0) = 0 \tag{3.13}$$

Further, considering the following differential equation.

$$\dot{\Phi}_{F1} = -\rho_F \Phi_{F1} + \Phi_1, \quad \Phi_{F1}(t_0) = 0 \tag{3.14}$$

The solution of (3.14) is given by

$$\Phi_{F1}(t) = e^{-kt} \int_{t_0}^{t} e^{ks} \Phi_1(s) ds \tag{3.15}$$

Structure $M(q)\ddot{q}$, is such that the elements of $\Phi_1(q, \ddot{q})$ are of the form $f(q_k)\ddot{q}_l$ for some $l, k \in \{1, 2, \ldots, n\}$, and q_i represents the i^{th} scalar component of the vector q. Thus, by using integration by parts on the elements of Φ_{F1} we get

$$e^{-\rho_F t} \int_{t_0}^{t} e^{\rho_F s} f(q_k(s)) \ddot{q}_l(s) ds = f(q_k(t)) \dot{q}_l(t) - e^{-p_F(t-t_0)} f(q_k(t_0)) \dot{q}_l(t_0) - h(t) \tag{3.16}$$

where $h(t)$ is given by the differential equation:

$$\dot{h} = -\rho_F h + \dot{f}\left(q_k(t)\right)\dot{q}_l(t) + \rho_F f\left(q_k(t)\right)\dot{q}_l(t), h\left(t_0\right) = 0 \tag{3.17}$$

Using (3.14), (3.16) and (3.17), $\Phi_{F1}(t)$ can also be computed online, and therefore $\Phi_F = \Phi_{F1} + \Phi_{F2}$ can be obtained online (refer [8] for further details).

3.4 Conclusion

Deadzone was modeled and compensated by adding an offset at the input. Deadzone is not fixed but changing. So the extra offset term is the included in the dynamics for better control action.

A first order low pass filter was applied on the dynamics which removed the acceleration term. The final filtered dynamics will be used henceforth for designing the controller.

Chapter 4

Controller Design for 1-DOF Manipulator

4.1 Introduction

Single-Link or 1-DOF manipulator is simply a motor with load. First adaptive controller is tested on this basic robot which gives insight into the control system. Based on the performance of different controller, controller is designed for more complex robot in the next chapter.

The objective is to design an adaptive state-feedback controller to track a desired trajectory $q_d(t) \in \mathbb{R}$ in the presence of parametric uncertainties in J, B and D_z. The tracking error is defined as

$$e(t) \triangleq q(t) - q_d(t) \tag{4.1}$$

It is assumed that the desired trajectory $q_d(t)$ is continuous and differentiable and $q_d(t), \dot{q}_d(t) \in \mathcal{L}_\infty$.

4.2 PID Controller

Most of the industrial controllers are PID controller due to its ease of design and implementation. The generalized equation of PID controller in time domain is given by

$$U(t) = K_P e(t) + K_D \frac{de(t)}{dt} + K_I \int e(t)dt \tag{4.2}$$

where $U(t)$ is control input, K_P is proportional gain, K_D is derivative gain, K_I is integral gain. In s-domain the equation of PID controller is given as

$$U(s) = K_P E(s) + K_D sE(s) + \frac{K_I E(s)}{s} \tag{4.3}$$

The benefit of PID controller is that it neither needs plant parameters nor deadzone widths. The integral action of PID controller automatically compensates for the nonlinearity i.e. deadzone. PID controller requires the gain to be tuned according to the user requirements. We have tuned the PID gains by trail and error method.

The PID gains are $k_p = 4, k_d = 12, k_I = 3$

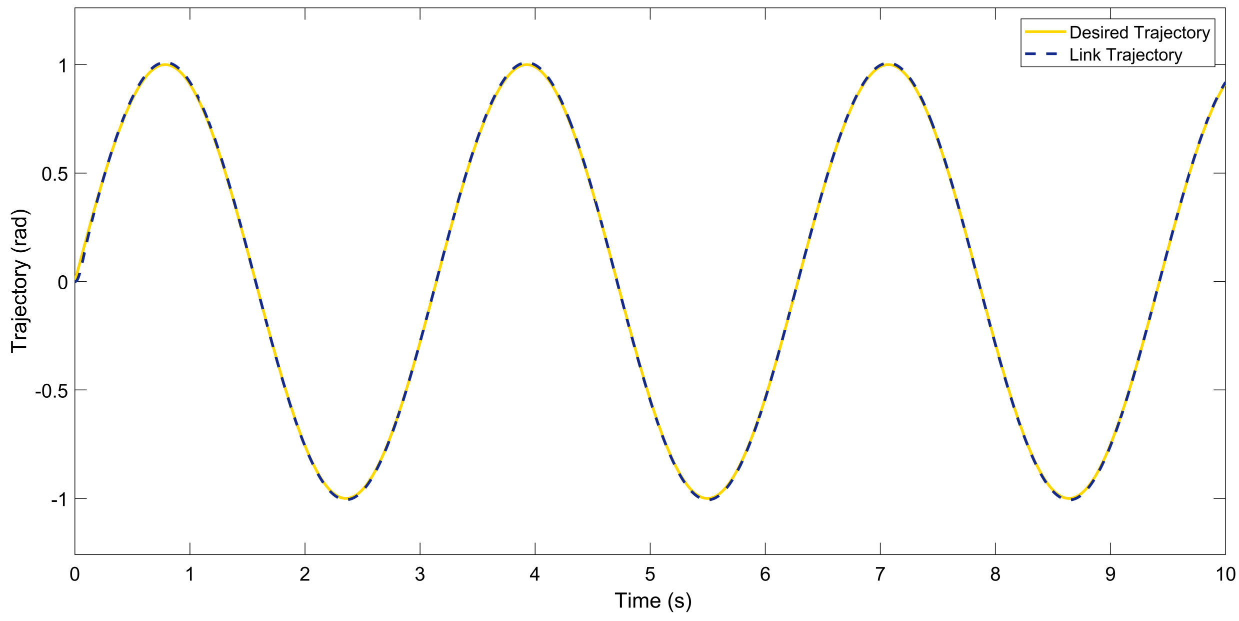

Figure 4.1: PID Tracking Control of Single Link

4.3 State Feedback Controller

The state space of the system is

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{4.4}$$

where $x = \begin{pmatrix} q \\ \omega \end{pmatrix}$, $A = \begin{pmatrix} 0 & 1 \\ 0 & -B/J \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ 1/J \end{pmatrix}$, and $C = \begin{pmatrix} 1 & 0 \end{pmatrix}$

The feedback controller is

$$u = -Kx \tag{4.5}$$

where $K = (K_1 \quad K_2)$ is controller gain, which is given by placing the pole to desired location.

The close loop system becomes

$$\dot{x} = (A - BK)x \tag{4.6}$$

The characteristics equation becomes

$$\det(sI - A + BK) = 0 \tag{4.7}$$

We can place poles to desired location to get the gain matrix K. For this controller we need complete system knowledge i.e. J, B and deadzone widths for using pre-compensator as used in previous part.

$$\begin{aligned} &\dot{x} = Ax + Bu \\ &y = Cx \\ &A = \begin{pmatrix} 0 & 1 \\ 0 & -0.33 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 33.13 \end{pmatrix}, C = \begin{pmatrix} 1 & 0 \end{pmatrix} \end{aligned} \tag{4.8}$$

To ensure same control effort we will directly use matrix K as

$$K = \begin{bmatrix} 1 & 0.05 \end{bmatrix}$$

With this K the close loop poles of system are at $-0.185 \pm j0.869$

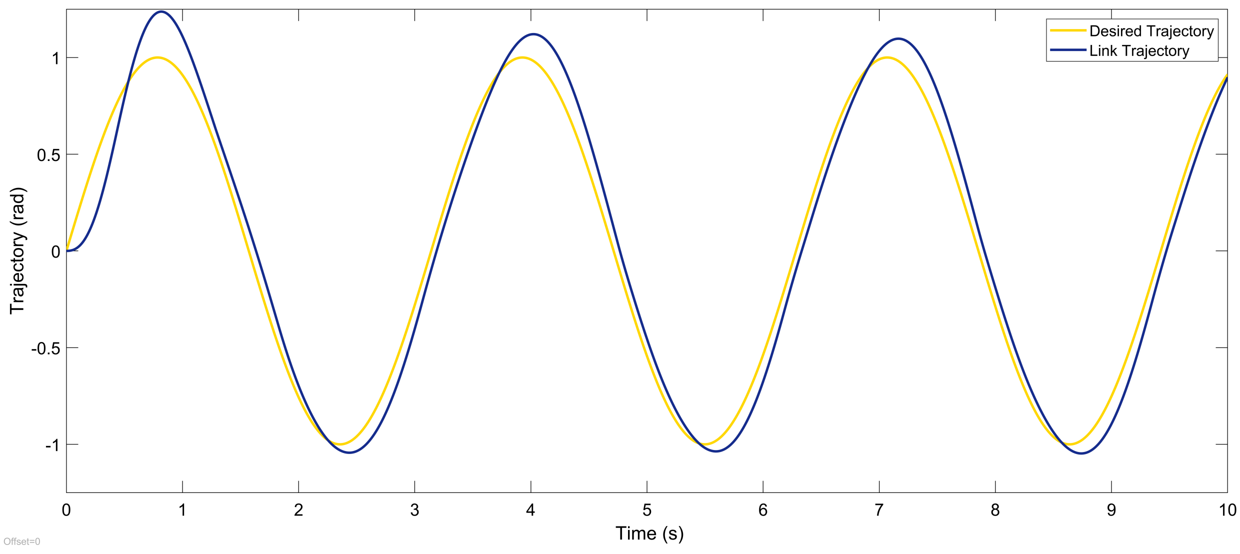

Figure 4.2: State Feedback Tracking Control of Single Link

4.4 Adaptive Controller

Let

$$[\tau(s) + \tau_{off}(s)] \frac{K_p}{s(J_p + B_p)} = q(s) \tag{4.9}$$

$$\tag{4.10}$$

$$\implies \tau(s) + \tau_{off}(s) = \frac{J_p}{K_p} s^2 q(s) + \frac{B_p}{K_p} sq(s) \tag{4.11}$$

Taking the laplace inverse

$$\implies \tau(t) + \tau_{off}(t) = \frac{J_p}{K_p} \ddot{(q)} + \frac{B_p}{K_p} \dot{q} \tag{4.12}$$

We know that error $e(t) = q(t) - q_d(t)$. Define filtered tracking error:

$$r = \dot{e} + \alpha e \qquad \alpha > 0 \tag{4.13}$$

Differentiate and multiply $\frac{J_p}{K_p}$ on both sides we get

$$\frac{J_p}{K_p} \dot{r} = \frac{J_p}{K_p} \ddot{e} + \alpha \frac{J_p}{K_p} \dot{e} \tag{4.14}$$

$$= \frac{J_p}{K_p} \ddot{q} - \frac{J_p}{K_p} \ddot{q}_d + \alpha \frac{J_p}{K_p} \dot{e} \tag{4.15}$$

Substituting the value from

$$= - \frac{B_p}{K_p} \dot{q} + \tau_{off} + \tau - \frac{J_p}{K_p} \ddot{q}_d + \alpha \frac{J_p}{K_p} \dot{e} \tag{4.16}$$

$$= \underbrace{\begin{bmatrix} -\dot{q} & -\ddot{q}_d + \alpha \dot{e} \end{bmatrix}}_{Y(e,\dot{e},t)} \underbrace{\begin{bmatrix} \frac{B_p}{K_p} \\ \frac{J_p}{K_p} \end{bmatrix}}_{\Phi} + \tau_{off} + \tau \tag{4.17}$$

Writing in the form of regressor and parameter matrix we get

$$\frac{J_p}{K_p} \dot{r} = Y\Phi + \tau_{off} + \tau \tag{4.18}$$

Let the control input be $\tau = -Y\hat{\Phi} - Kr$ where $K > 0$ and $\hat{\Phi}$ is the estimated Φ. Substituting this torque in the above equation we closed-loop error system

$$\frac{J_p}{K_p} \dot{r} = Y\tilde{\Phi} - Kr + \tau_{off} \tag{4.19}$$

Here $\tilde{\Phi} = \Phi - \hat{\Phi}$.

Choose Lyapunov function candidate

$$V(r, \tilde{\Phi}) = \frac{1}{2}\frac{J_p}{K_p}r^2 + \frac{1}{2}\tilde{\Phi}^T\Gamma^{-1}\tilde{\Phi} \tag{4.20}$$

Differentiating and substituting the values we get

$$\dot{V} = r(Y\tilde{\Phi} - Kr + \tau_{off}) - \tilde{\Phi}^T\Gamma^{-1}\dot{\hat{\Phi}} \tag{4.21}$$

$$= -Kr^2 + \tau_{off}r + rY\tilde{\Phi} - \tilde{\Phi}^T\Gamma^{-1}\dot{\hat{\Phi}} \tag{4.22}$$

Consider the adaptive law with sigma modification:

$$\dot{\hat{\Phi}} = \Gamma Y^T r - \Gamma\sigma\hat{\Phi} \tag{4.23}$$

Substituting this in the above equation

$$\dot{V} = -Kr^2 + \tau_{off}r + \sigma\tilde{\Phi}^T\hat{\Phi} \tag{4.24}$$

Substituting $\hat{\Phi} = \Phi - \tilde{\Phi}$

$$\dot{V} = -Kr^2 + \tau_{off}r + \sigma\tilde{\Phi}^T\Phi - \sigma\tilde{\Phi}^T\tilde{\Phi} \tag{4.25}$$

Taking upper bound $\tau_{off} \leq \bar{\tau_{off}}$ and norms

$$\leq -K\left\|r\right\|^2 + \bar{\tau_{off}}\left\|r\right\| + \sigma\left\|\tilde{\Phi}^T\right\|\left\|\Phi\right\| - \sigma\left\|\tilde{\Phi}\right\|^2 \tag{4.26}$$

Using Young's Inequality

$$\dot{V} \leq -K\left\|r\right\|^2 + \bar{\tau_{off}}\left\|r\right\| - \frac{\sigma}{2}\left\|\tilde{\Phi}\right\|^2 + -\frac{\sigma}{2}\left\|\Phi\right\|^2 \tag{4.27}$$

This can be written as

$$\dot{V} \leq \alpha_v V + C \tag{4.28}$$

where α_v and C are some constants. This controller is UUB (Uniformly Ultimately Bounded) Stable.

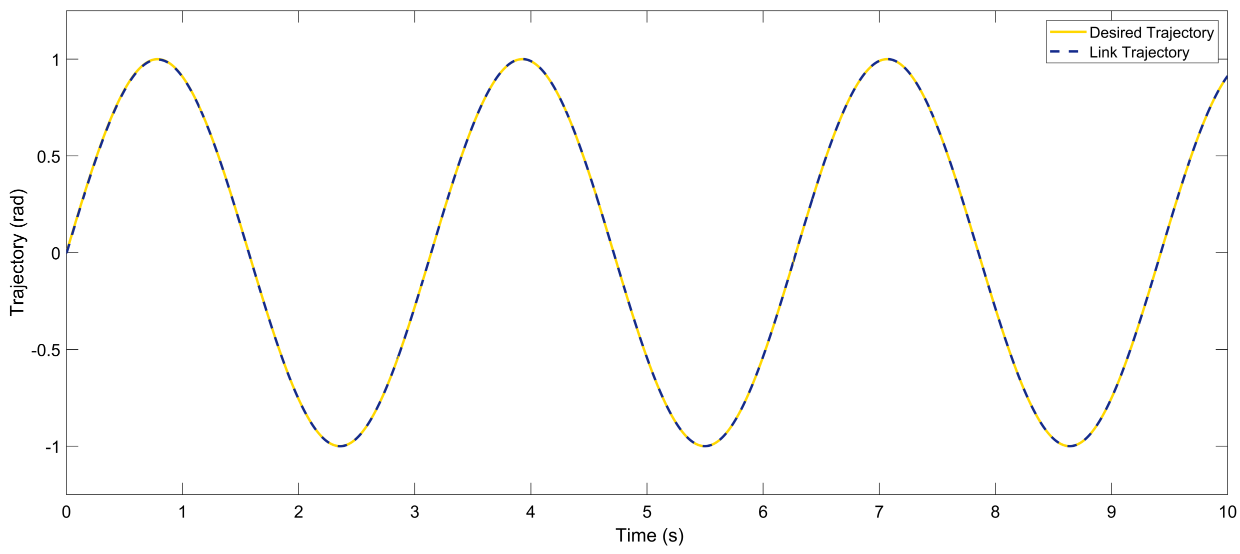

Figure 4.3: Adaptive Tracking Control of Single Link

4.5 Performance Comparison

Table 4.1 shows the RMS values of error, filtered error and the estimation error. The formula used for calculating RMS is

$$U_{rms} = \sqrt{\frac{\sum_{n=0}^{N-1} u^2(n)}{N}} \tag{4.29}$$

Here, N is the total samples taken in the period of 10 seconds which comes out to be 10,0000 samples (sampling time = 0.0001 s).

Table 4.1: RMS measures of control system performance of Single Link Manipulator

Controller	$\|\|e\|\|$
PID	6.523
State Feedback	120.1
Classical Adaptive	1.097

4.6 Conclusion

As seen in the plots, and the Table 4.1, confirms that out of all the controllers, adaptive controller performed the best in reducing the tracking error but the state feedback controller performed worst due to effect of changing deadzone width.

Chapter 5

Controller Design for Two Link Manipulator

5.1 Introduction

Two-Link horizontal planar manipulator is typically used as industrial robot for welding, painting, pick-and-place,etc. It is basically emulating the arm of humans. The robot has coupled dynamics and deadzone at input which makes the designing of the controller difficult.

In this chapter various adaptive controllers are tested on the robot and conclusion is drawn based on their performance.

The objective is to design an adaptive state-feedback controller to track a desired trajectory $q_d(t) \in \mathbb{R}^n$ in the presence of parametric uncertainties in $M(q), C(q, \dot{q})$ and $F(\dot{q})$. The tracking error is defined as

$$e(t) \triangleq q(t) - q_d(t) \tag{5.1}$$

It is assumed that the desired trajectory $q_d(t)$ is continuous and twice differentiable and $q_d(t), \dot{q}_d(t), \ddot{q}_d(t) \in \mathcal{L}_\infty$.

5.2 Classical PD Adaptive Controller

The adaptive controller consists of a PD feedback part and a full dynamics feedforward compensation part, with the unknown manipulator parameters being estimated online [7].

5.2.1 Controller Design

Consider filtered tracking error which is but the PD feedback,

$$r(t) \triangleq \dot{e} + \alpha e \tag{5.2}$$

where $\alpha > 0$ is a positive scalar. Using (2.1), (5.1), and (5.2) we get the dynamics

$$M\dot{r} = -C\dot{q} - G - F - M\ddot{q}_d + M\alpha\dot{e} + \tau \tag{5.3}$$

Using Property 4.3 the above dynamics can be represented as

$$M\dot{r} = Y\left(q, \dot{q}, q_d, \dot{q}_d\right)\theta + \tau - Cr \tag{5.4}$$

where $\theta \in \mathbb{R}^p$ is the unknown parameter vector, p being the number of parameters and $Y \in \mathbb{R}^{n\times p}$ is the known regressor matrix. Here, $Y\theta$ is given by

$$Y\theta = -C\dot{q} - G - F - M\ddot{q}_d + M\alpha\dot{e} + Cr \tag{5.5}$$

Consider the following torque control input design

$$\tau = -Y\hat{\theta} - kr \tag{5.6}$$

where $k > 0$ is a scalar feedback-gain, and $\hat{\theta}(t) \in \mathbb{R}^p$ is an online estimate of the unknown system parameter vector θ, updated using the following adaptive law.

$$\dot{\hat{\theta}} = \Gamma_\theta Y^{\mathrm{T}} r \tag{5.7}$$

where Γ_θ is a PD adaptive gain matrix.

5.2.2 Stability Analysis

Substituting the torque control input (5.6) the dynamics (5.4), the closed-loop error dynamics becomes

$$M\dot{r} = Y\tilde{\theta} - kr - Cr \tag{5.8}$$

Consider the following PD, radially unbounded, and decrescent Lyapunov candidate

$$V = \frac{1}{2}r^{\mathrm{T}}Mr + \frac{1}{2}\tilde{\theta}^{\mathrm{T}}\Gamma_\theta^{-1}\tilde{\theta} \tag{5.9}$$

Taking the time-derivative of (5.9) and substituting the error dynamics from (5.8) yields

$$\dot{V} = r^T\left(Y\tilde{\theta} - kr - Cr\right) + \frac{1}{2}r^T\dot{M}r - \tilde{\theta}^{\mathrm{T}}\Gamma_\theta^{-1}\dot{\hat{\theta}} \tag{5.10}$$

Canceling the like terms, using Property 2.2, and the adaptive law (5.7) the above equation further simplifies to

$$\dot{V} = -r^{\mathrm{T}}kr \leq 0 \tag{5.11}$$

implying $r(t) \to 0$ as $t \to \infty$. The same cannot be proved about $\tilde{\theta}$, however, we can say that it will be bounded. Thus the adaptive controller is globally asymptotically stable and guarantees zero steady-state error for joint positions.

5.3 Single Layer Neuroadaptive Controller

A single layer neural network(SLNN) with sigmoidal activation function for estimation of θ is implemented [5,3]. For SLNN we can approximate $Y\theta$ as

$$Y\theta = W^T\gamma\left(V^Tx\right) + \epsilon\left(x\right) \tag{5.12}$$

$$Y\theta = W^T\Phi\left(x\right) + \epsilon\left(x\right) \tag{5.13}$$

Here W is the output weight matrix, $\Phi\left(x\right) = \gamma\left(V^Tx\right)$ is the activation function applied on the input layer, V is the input to hidden layer neuron weight matrix, and $\gamma = \left[1 \;\; \gamma_1 \;\; \gamma_2 \; ... \; \gamma_L\right]^T$, L being the number of neurons is the activation function vector as shown in the Fig. 3.

$$y_i = \Sigma_{j=1}^{L} w_{ij}\sigma\left(\Sigma_{k=1}^{n} v_{jk}X_k + \theta_{vi}\right) + \theta_{wi} \text{ for } j, i = 1, \cdots, m$$

In Matrix form

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = W^T\sigma(V^Tx)$$

$$V^TX = \underbrace{\begin{bmatrix} \theta_{v_1} & v_{11} & v_{12} & \dots & V_{1n} \\ \theta_{v_2} & v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & & & \\ \theta_{v_2} & v_{21} & v_{12} & \cdots & v_{2n} \end{bmatrix}_{L\times(n+1)}}_{V^T} \underbrace{\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}}_{X}$$

$$W^T = \begin{bmatrix} \theta_{w1} & w_{11} & \dots & w_{1L} \\ \theta_{w2} & w_{21} & \dots & w_{2L} \\ \vdots & \vdots & \dots & \vdots \\ \theta_{wm} & w_{m1} & \dots & w_{mL} \end{bmatrix}_{m\times(L+1)}$$

$$\sigma = \begin{bmatrix} 1 \\ \sigma_1 \\ \vdots \\ \sigma_n \end{bmatrix}_{L+1\times1}$$

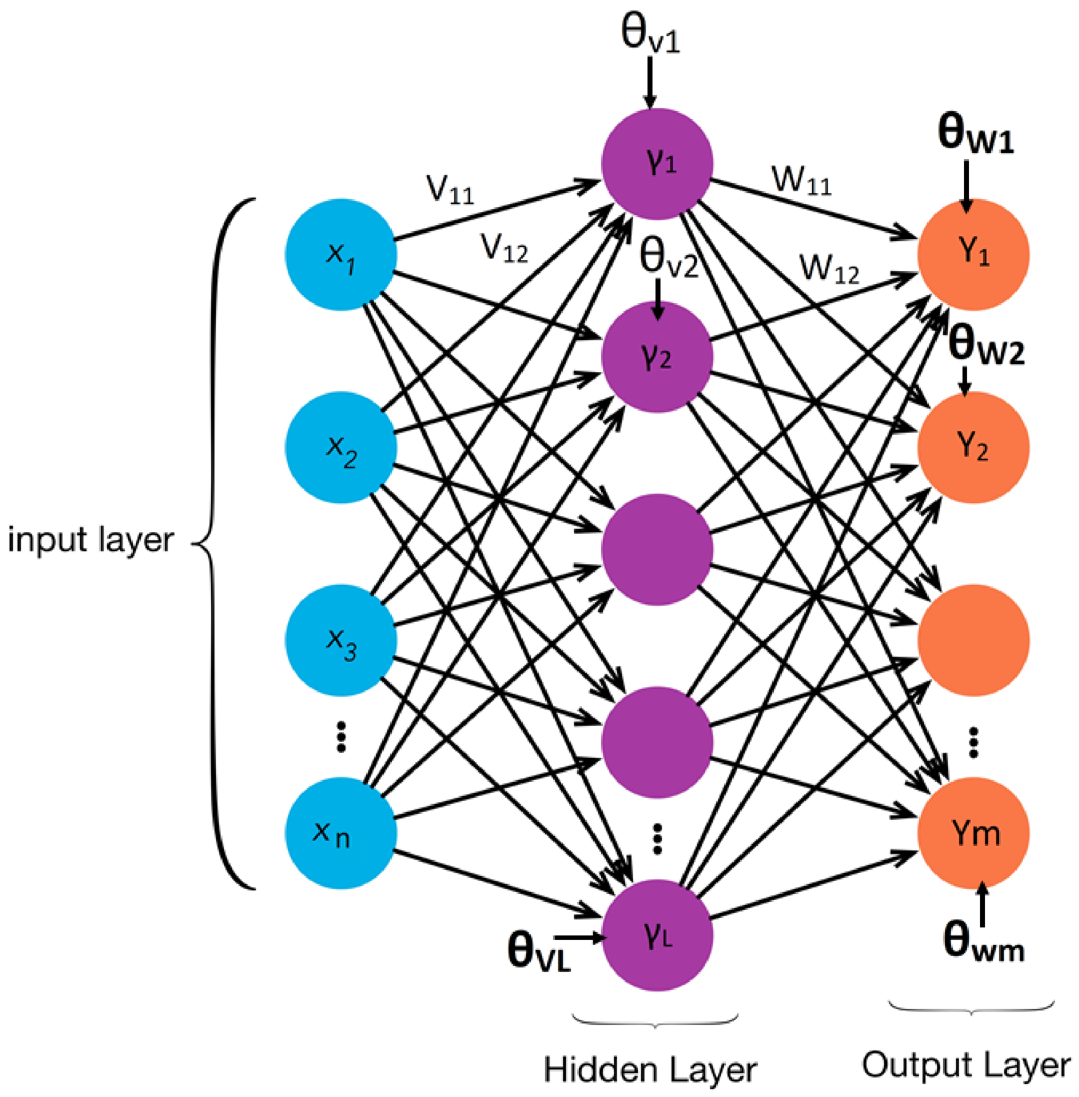

Figure 5.1: Neural Network Architecture

The θ terms are just the bias(offset) in the network weights. $\epsilon(x)$ is the function approximation error. x is the input to neurons which in this case

$$x = [1 \quad q \quad \dot{q} \quad q_d \quad \dot{q}_d \quad \ddot{q}_d]^T \tag{5.14}$$

So $x_1 = 1, x_2 = q ... x_6 = \ddot{q}_d$. For single layer neural network $V = [V_{ij}]$ and $V_{ij} = 1 \; \forall \; i, j$. So the input to all the neurons is same with $\gamma = \sum_{i\,=\,1}^{6} x_i$. From Eq. (5.4) and (5.13) we get

$$M\dot{r} = W^{\mathrm{T}}\Phi(x) + \epsilon(x) + \tau - Cr \tag{5.15}$$

The control input equation considered is

$$\tau = -\widehat{W}^{\mathrm{T}}\Phi(x) - Kr \tag{5.16}$$

Kr is a positive stabilizing term. Since there is no training data we use adaptive method to train the network. The adaptive law used is

$$\dot{\hat{W}} = \Gamma_w \Phi^{\mathrm{T}}(x)\, r - \Gamma_w \sigma \|e\| \hat{W} \tag{5.17}$$

Stability Consider the lypunov function

$$V = \frac{1}{2} r^T M r + \frac{1}{2} tr(\tilde{W}^T \Gamma_w^{-1} \tilde{W}) \tag{5.18}$$

Differentiating w.r.t time

$$\dot{V} = r^T M \dot{r} - tr(\tilde{W}^T \Gamma_w^{-1} \dot{\hat{W}}) + \frac{1}{2} r^T \dot{M} r \tag{5.19}$$

Substituting values we get

$$\begin{aligned}\dot{V} &= r^T(\tilde{W}^T\Phi + \epsilon(x) - Kr - V_m r) - tr(\tilde{W}^T\Gamma_w^{-1}\dot{\hat{W}}) \\ &= r^T K r + r^T\epsilon(x) + r^T\tilde{W}^T\Phi(x) - tr(\tilde{W}^T\Gamma_w^{-1}\dot{\hat{W}})\end{aligned}$$

Using $\dot{\hat{W}} = \Gamma_w \Phi^T(x) r$

$$\dot{V} = -r^T K r + r^T \epsilon(x) \tag{5.20}$$

This is similar to robust disturbance rejection. Using sigma modification

$$\dot{\hat{W}} = \Gamma_w \Phi^T(x) r - \Gamma_w \sigma \tilde{W}$$

$$\begin{aligned}
\dot{V} &= r^T K r + r^T \epsilon - tr(\tilde{W}^T \sigma \hat{W}) \\
&= r^T K r + r^T \epsilon - \sigma tr(\tilde{W}^T (W - \tilde{W})) \\
&= r^T K r + r^T \epsilon + \sigma tr(\tilde{W}^T W) - \sigma tr(\tilde{W}^T \tilde{W})
\end{aligned}$$

We know that $tr(A^T B) \le \|A\|_F \|B\|_F$ where $\|\|_F$ denotes Frobenius norm.

$$\begin{aligned}
\dot{V} &\le -\lambda_{min} K \|r\|^2 + \sigma \left\|\tilde{W}\right\|_F \|W\|_F - \sigma \left\|\tilde{W}\right\|_F^2 + r^T \epsilon \\
&\le \frac{\lambda_{min} K \|r\|^2}{2} - \frac{\sigma}{2} \left\|\tilde{W}\right\|_F^2 + \sigma \left\|\tilde{W}\right\|_F \|W\|_F - \frac{\sigma}{2} \left\|\tilde{W}\right\|_F^2 + \bar{\epsilon} \|r\| - \frac{\lambda_{min} K \|r\|^2}{2} \\
&\le \frac{\lambda_{min} K \|r\|^2}{2} - \frac{\sigma}{2} \left\|\tilde{W}\right\|_F^2 + \overbrace{\frac{\sigma}{2} \|W\|_F^2 + \frac{\sigma}{2} \left\|\tilde{W}\right\|_F^2}^{\textbf{Multiplication Inequality}} - \frac{\sigma}{2} \sigma \left\|\tilde{W}\right\|_F^2 \\
&\quad \underbrace{- \left[\frac{\lambda_{min} K \|r\|^2}{2} - \bar{\epsilon} \|r\| + \frac{\bar{\epsilon}^2}{2 \lambda_{min} K} \right]}_{\textbf{Square Term}} + \frac{\epsilon^2}{2 \lambda_{min} K} \\
&\le \frac{\lambda_{min} K \|r\|^2}{2} - \frac{\sigma}{2} \left\|\tilde{W}\right\|_F^2 + \underbrace{\frac{\sigma}{2} \|W\|_F^2 + \frac{\epsilon^2}{2 \lambda_{min} K}}_{C}
\end{aligned}$$

The above equation can be approximated as

$$\dot{V} \le -\alpha V + C \tag{5.21}$$

Thus the control system in UUB stable. To decrease C, increase K in $\frac{\epsilon^2}{2\lambda_{min} K}$ or decrease γ, but this also decreases the term $\left[-\frac{\gamma}{2} \left\|\tilde{W}\right\|_F^2 \right]$ i.e.

$$V(t) \le e^{-\alpha t} V(t) + \frac{C}{\alpha} \left[1 - e^{-\alpha t} \right] \tag{5.22}$$

5.4 Multilayer Neuroadaptive Controller

The equation of multilayer Neuroadaptive controller is similar to single layer Neuroadaptive controller [1]. The only difference is the extra hidden layer in the neural network. So $\Phi(x) = \gamma \left(V^T x \right)$.Following the above procedure the Eq. (5.15) is now

$$M \dot{r} = W^{\mathrm{T}} \gamma \left(V^T x \right) + \epsilon(x) + \tau - C r \tag{5.23}$$

And the control input equation becomes

$$\tau = -\widehat{W^{\mathrm{T}}} \Phi(x) - K r \; \breve{} e \tag{5.24}$$

Hence

$$M\dot{r} = -Cr + W^T\gamma(V^Tc) - \hat{W}^T\gamma(\hat{V}^Tx) - Kr + \epsilon - e \tag{5.25}$$

By using Taylor series expansion about $\hat{V}^Tx$:

$$\gamma(V^Tx) = \gamma(\hat{V}^Tx) + \frac{\partial\gamma}{\partial V^Tx}\|_{\hat{V}^Tx}\tilde{V}^Tx + \underbrace{O(\tilde{V}^Tx)^2}_{\text{Higher order terms that are neglected}} \tag{5.26}$$

Let

$$\gamma \triangleq \gamma(V^Tx) \tag{5.27}$$

$$\hat{\gamma} \triangleq \gamma(\hat{V}^Tx) \tag{5.28}$$

$$\gamma' \triangleq \frac{\partial\gamma}{\partial V^Tx} \tag{5.29}$$

Thus

$$\gamma = \hat{\gamma} + \hat{\gamma}'\tilde{V}^Tx + O(\tilde{V}^Tx)^2 \tag{5.30}$$

Substituting it in the equation

$$\begin{aligned}
M\dot{r} &= -Cr + W^T\left(\hat{\gamma} + \hat{\gamma}'\tilde{V}^Tx + O(\tilde{V}^Tx)^2\right) - \hat{W}^T\hat{\gamma} - Kr + \epsilon - e && (5.31)\\
&= -Cr + \tilde{W}^T\hat{\gamma} + W^T\hat{\gamma}'\tilde{V}^Tx + W^TO - Kr + \epsilon - e && (5.32)\\
&= -Cr + \tilde{W}^T\hat{\gamma} + W^T\hat{\gamma}'V^Tx - W^T\hat{\gamma}'\hat{V}^Tx + \hat{W}^T\hat{\gamma}'\tilde{V}^Tx + W^TO - Kr + \epsilon - e && (5.33)\\
&= -Cr + \tilde{W}^T\left(\hat{\gamma} - \hat{\gamma}'\hat{V}^Tx\right) + \hat{W}^T\hat{\gamma}'\tilde{V}^Tx - Kr - e + X && (5.34)
\end{aligned}$$

where $X \triangleq \tilde{W}^T\hat{\gamma}'V^Tx + W^TO + \epsilon$

Stability Consider the Lyapunov function

$$V = \frac{1}{2}r^TMr + \frac{1}{2}e^Te + \frac{1}{2}tr(\tilde{W}^T\Gamma_w^{-1}\tilde{W}) + \frac{1}{2}tr(\tilde{V}^T\Gamma_v^{-1}\tilde{V}) \tag{5.35}$$

Differentiating w.r.t

$$\begin{aligned}
\dot{V} =& \frac{1}{2}r^TM\dot{r} + \frac{1}{2}r^T\dot{M}r + e^Te - tr(\tilde{W}^T\Gamma_w^{-1}\dot{\hat{W}}) - tr(\tilde{V}^T\Gamma_v^{-1}\dot{\hat{V}}) && (5.36)\\
=& r^T\left(-Cr + \tilde{W}^T\left(\hat{\gamma} - \hat{\gamma}'\hat{V}^Tx\right) + \hat{W}^T\hat{\gamma}'\tilde{V}^Tx - Kr - e + X\right) + \frac{1}{2}r^T\dot{M}r + e^T(r - \alpha e)\\
&- tr(\tilde{W}^T\Gamma_w^{-1}\dot{\hat{W}}) - tr(\tilde{V}^T\Gamma_v^{-1}\dot{\hat{V}})
\end{aligned}$$

Let the adaptive laws be

$$\dot{\hat{W}} = \Gamma_w \left(\hat{\gamma} - \hat{\gamma}'\hat{V}^T x\right) r^T \tag{5.37}$$

$$\dot{\hat{V}} = \Gamma_v x r^T \hat{W}^T \hat{\gamma}' \tag{5.38}$$

Substituting the adaptive laws we the the simplified equation

$$\dot{V} = -r^T K r - \alpha e^T e + r^T X \tag{5.39}$$

We need to upperbound X. Now

$$X = \tilde{W}^T \hat{\gamma}' V^T x + W^T O + \epsilon$$
$$O = \gamma - \hat{\gamma} - \hat{\gamma}'\tilde{V}^T x$$
$$\|O\| \leq C_{11} + C_{12} \left\|\tilde{V}\right\|_F \|x\|$$

Also

$$\|x\| \leq C_1 + C_2 \|Z\|$$

Here $Z \triangleq \left[e^T r^T\right]^T$ Assuming

$$\|W\|_F \leq \bar{W} \tag{5.40}$$

$$\|V\|_F \leq \bar{V} \tag{5.41}$$

$$\epsilon \leq \bar{\epsilon} \tag{5.42}$$

Substituting all the constants

$$\|X\| \leq \bar{C_{11}} + \bar{C_{12}} \left\|\tilde{V}\right\|_F + \bar{C_{13}} \left\|\tilde{W}\right\|_F + \bar{C_{14}} \left\|\tilde{V}\right\|_F + \bar{C_{15}} \left\|\tilde{W}\right\|_F \|Z\| \tag{5.43}$$

Adaptive update law with σ-mod

$$\dot{\hat{W}} = \Gamma_w \left(\hat{\gamma} - \hat{\gamma}'\hat{V}^T x\right) r^T - \Gamma_w \sigma_w \hat{W} \tag{5.44}$$

$$\dot{\hat{V}} = \Gamma_v x r^T \hat{W}^T \hat{\gamma}' - \Gamma_v \sigma_v \hat{V} \tag{5.45}$$

Substituting the above modified adaptive laws

$$\dot{V} = -r^T K r - \alpha e^T e + r^T X - \sigma_w \left\|\tilde{W}\right\|_F^2 - \sigma_v \left\|\tilde{V}\right\|_F^2 + \sigma_w \left\|\tilde{W}\right\|_F \|W\|_F + \Gamma_v \left\|\tilde{V}\right\|_F \|V\|_F$$
$$\dot{V} \leq -r^T K r - \alpha e^T e + \|r\| \|X\| - \sigma_w \left\|\tilde{W}\right\|_F^2 - \sigma_v \left\|\tilde{V}\right\|_F^2 + \sigma_w \left\|\tilde{W}\right\|_F \|W\|_F + \Gamma_v \left\|\tilde{V}\right\|_F \|V\|_F$$

Substituting the value of $\|X\|$ we get

$$\begin{aligned}\dot{V} &= -r^TKr - \alpha e^Te + \|r\|\left[\bar{C_{11}} + \bar{C_{12}}\left\|\tilde{V}\right\|_F + \bar{C_{13}}\left\|\tilde{W}\right\|_F + \bar{C_{14}}\left\|\tilde{V}\right\|_F + \bar{C_{15}}\left\|\tilde{W}\right\|_F\|Z\|\right] \\ &- \sigma_w\left\|\tilde{W}\right\|_F^2 - \sigma_v\left\|\tilde{V}\right\|_F^2 + \sigma_w\left\|\tilde{W}\right\|_F\|W\|_F + \Gamma_v\left\|\tilde{V}\right\|_F\|V\|_F\end{aligned}$$

From Young's Inequality $\sigma_w\left\|\tilde{W}\right\|_F\|W\|_F \leq \frac{\sigma_w}{2}\left\|\tilde{W}\right\|_F^2 + \frac{\sigma_w}{2}\|W\|_F^2$

$$\begin{aligned}\dot{V} &= -r^TKr - \alpha e^Te + \|r\|\left[\bar{C_{11}} + \bar{C_{12}}\left\|\tilde{V}\right\|_F + \bar{C_{13}}\left\|\tilde{W}\right\|_F + \bar{C_{14}}\left\|\tilde{V}\right\|_F + \bar{C_{15}}\left\|\tilde{W}\right\|_F\|Z\|\right] \\ &- \frac{\sigma_w}{2}\left\|\tilde{W}\right\|_F^2 - \frac{\sigma_v}{2}\left\|\tilde{V}\right\|_F^2 + \frac{\sigma_v}{2}\|V\|_F^2 + \frac{\sigma_w}{2}\|W\|_F^2 + \frac{\sigma_v}{V}{}^2_F\end{aligned}$$

Further simplifying

$$\begin{aligned}\dot{V} &\leq -K\|r\|^2 - \alpha\|e\|^2 \cdots + \frac{\bar{C_{11}}^2}{2} + \frac{\|r\|^2}{2} + \|r\|\left[\cdots\right] \\ &\leq -(-K - \frac{1}{2})\|r\|^2 - \alpha\|e\|^2 \cdots \frac{\bar{C_{11}}^2}{2} + \|r\|\left[\cdots\right]\end{aligned}$$

$$\begin{aligned}\dot{V} \leq &-\left(K - \frac{1}{2} - \frac{\bar{C_{12}}}{2} - \frac{\bar{C_{13}}}{2}\right)\|r\|^2 - \alpha\|e\|^2 + \frac{\bar{C_{11}}^2}{2} - \left(\frac{\sigma_w}{2} - \frac{\bar{C_{13}}}{2}\right)\left\|\tilde{W}\right\|_F^2 - \left(\frac{\sigma_v}{2} - \frac{\bar{C_{12}}}{2}\right)\left\|\tilde{V}\right\|_F^2 \\ &+ \frac{\sigma_w}{2}\|W\|_F^2 + \frac{\sigma_v}{2}\|V\|_F^2 + \|r\|\left[\bar{C_{14}}\left\|\tilde{V}\right\|_F\|Z\| + \bar{C_{15}}\left\|\tilde{W}\right\|_F\|Z\|\right]\end{aligned}$$

Again

$$\begin{aligned}\bar{C_{14}}\left\|\tilde{V}\right\|_F^2\|r\|\,\|Z\| &\leq \frac{\bar{C_{14}}}{2}\left\|\tilde{V}\right\|_F^2\|r\|^2 + \frac{\bar{C_{14}}}{2}\|Z\|^2 \\ \bar{C_{15}}\left\|\tilde{W}\right\|_F^2\|r\|\,\|Z\| &\leq \frac{\bar{C_{15}}}{2}\left\|\tilde{W}\right\|_F^2\|r\|^2 + \frac{\bar{C_{15}}}{2}\|Z\|^2\end{aligned}$$

Gain conditions

$$\begin{aligned}K &> \frac{1}{2} + \frac{\bar{C_{12}}}{2} + \frac{\bar{C_{13}}}{2} \\ \sigma_w &> \bar{C_{13}} \\ \sigma_v &> \bar{C_{12}} \\ K_\alpha &> \frac{\bar{C_{14}}}{2} + \frac{\bar{C_{15}}}{2}\end{aligned}$$

Now

$$\left\|\tilde{V}\right\|_F^2 = \left\|V - \hat{V}\right\|_F^2 \leq \left(\bar{V} + \left\|\hat{V}\right\|_F\right)^2$$

This makes

$$\bar{C_{14}} \left\|\tilde{V}\right\|_F \|r\| \|Z\| \leq \frac{\bar{C_{14}}}{2} \left(\bar{V} + \left\|\hat{V}\right\|_F\right)^2 \|r\|^2 + \frac{\bar{C_{14}}}{2} \|Z\|^2 \tag{5.46}$$

$$\bar{C_{15}} \left\|\tilde{W}\right\|_F \|r\| \|Z\| \leq \frac{\bar{C_{15}}}{2} \left(\bar{W} + \left\|\hat{W}\right\|_F\right)^2 \|r\|^2 + \frac{\bar{C_{15}}}{2} \|Z\|^2 \tag{5.47}$$

Let

$$\tau = \hat{W}^T \hat{\gamma} - Kr - e - \frac{\bar{C_{14}}}{2} \left(\bar{V} + \left\|\hat{V}\right\|_F\right)^2 \|r\|^2 - \frac{\bar{C_{15}}}{2} \left(\bar{W} + \left\|\hat{W}\right\|_F\right)^2 \|r\|^2 \tag{5.48}$$

Then

$$\dot{V} \leq -\left(K_\alpha - \frac{\bar{C_{14}}}{2} - \frac{\bar{C_{15}}}{2}\right) \|Z\|^2 - \left(\frac{\sigma_w}{2} - \frac{\bar{C_{13}}}{2}\right) \left\|\tilde{W}\right\|_F^2 - \left(\frac{\sigma_v}{2} - \frac{\bar{C_{12}}}{2}\right) \left\|\tilde{V}\right\|_F^2 + \bar{C}$$

Which can be further simplified as

$$\dot{V} \leq -\beta V + \bar{C} \tag{5.49}$$

Thus the control system is locally UUB stable.

5.5 Experimental Results

All the control algorithms were implemented on the rigid two link planar manipulator (refer Section 2.4 for the dynamics). Due to presence of unmodeled disturbance $\sigma - mod$ was used with all the adaptive controllers.

For adaptive controller (section 5.2), the gains were chosen as $\alpha = 2$, $k = 5$, $\Gamma_\theta = 10$, and $\sigma = 0.1$. All the common gains for the different adaptive controllers were chosen same for comparing their performance. The desired trajectory is

$$q_d(t) = [2.5 + 1.2sin(t), \quad 2.5 + 1.2sin(t)] \quad \text{radians} \tag{5.50}$$

The controller gains were chosen by trail and error method (refer section 3.2 of [8]). Initially, classical adaptive controller was implemented which was tuned by first keeping all the gains high and then bringing the values down till a good performance is achieved. Once these gains were determined the common gains for the single layer neuroadaptive controller as well as the multilayer neuroadaptive controller were chosen the same.

Fig. 5.2 - 5.5 shows all the controller performance.

In addition to these, a PID controller was also implemented with the gains $K_p = 12$, $K_i = 3$, $K_d = 10$ for link-1, and $K_p = 6$, $K_i = 3$, $K_d = 12$ for link-2. The values were tuned

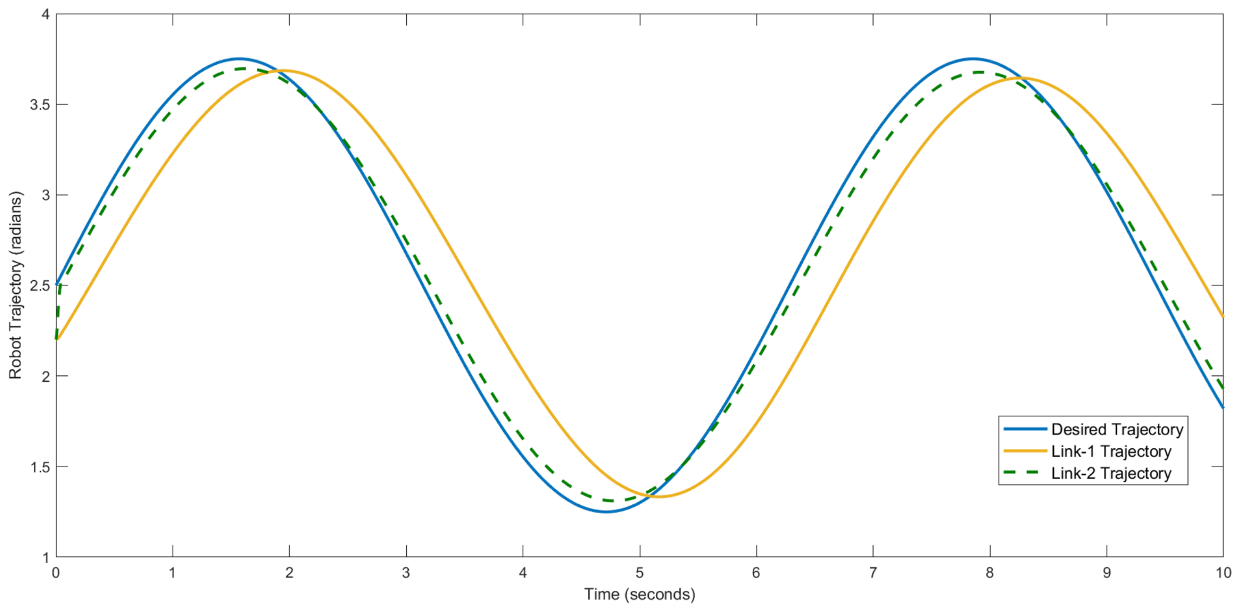

Figure 5.2: PID Tracking Control

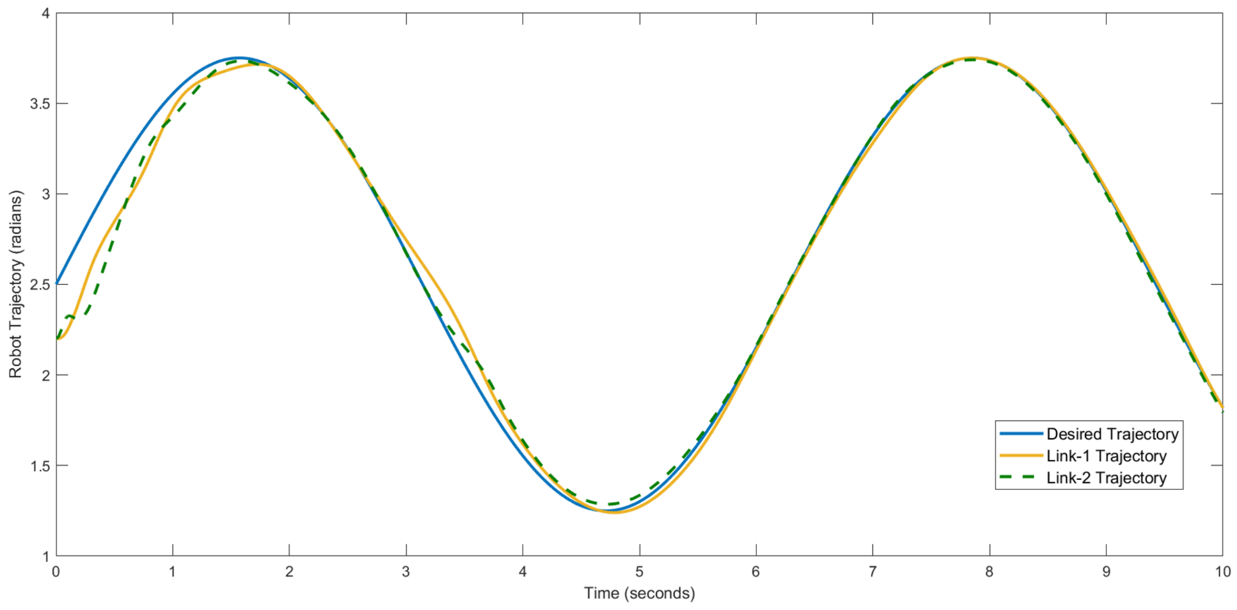

Figure 5.3: Adaptive Tracking Control

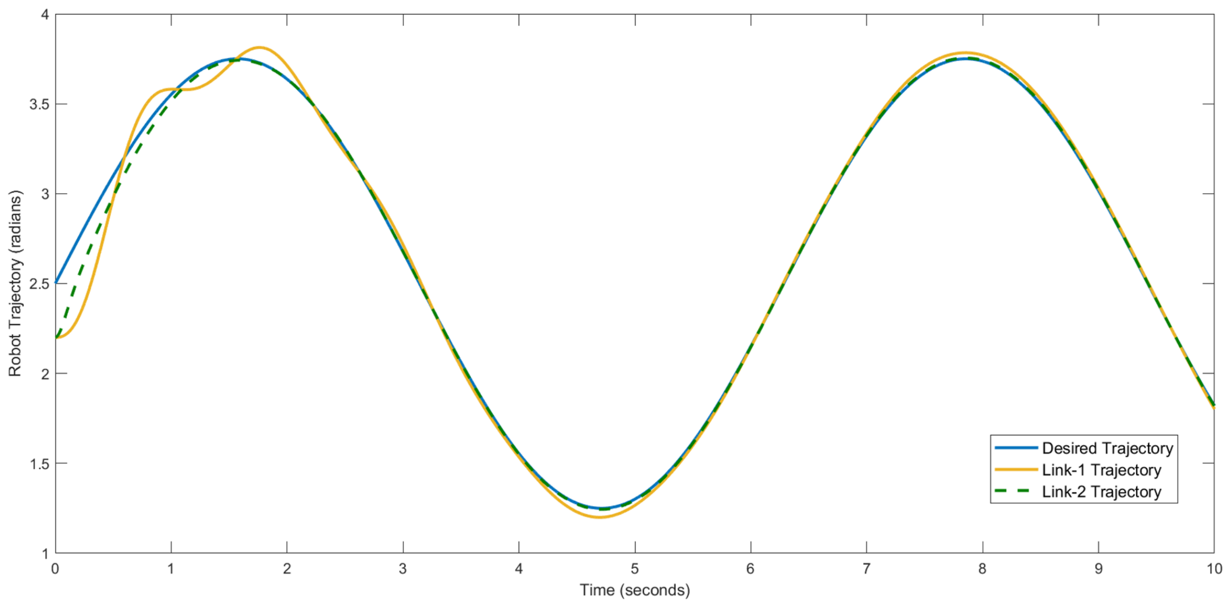

Figure 5.4: Single Layer Neuroadaptive Tracking Control

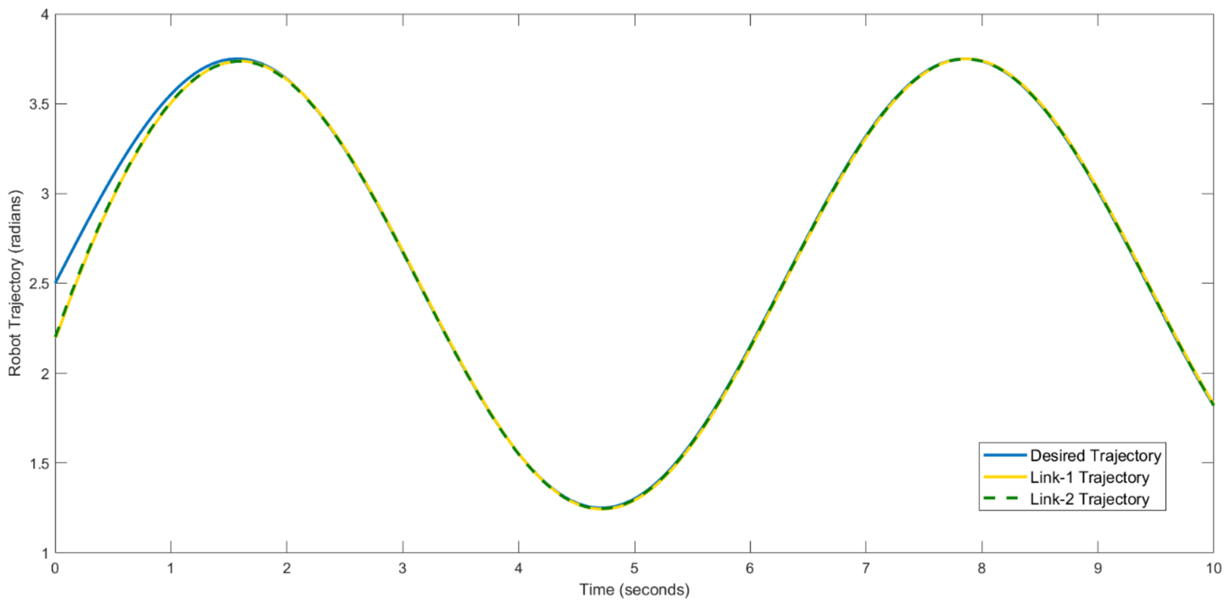

Figure 5.5: Multilayer Neuroadaptive Tracking Control

by trail and error method. First proportional gain K_p was tuned while keeping integral gain K_i and derivative gain K_d to zero. K_p was increased till the system response became fast and steady state error was minimum. While tuning the values were chosen within the saturation limit. Integral gain was then tuned to remove the steady-state error. Further, derivative gain was tuned to minimize the overshoot and to speed up the response.

Due to $\sigma-$modification, the parameters are not guaranteed to converge to the true values. Even in those conditions it is observed that the multilayer neuroadaptive controller gives the least estimation error.

5.6 Performance Comparison

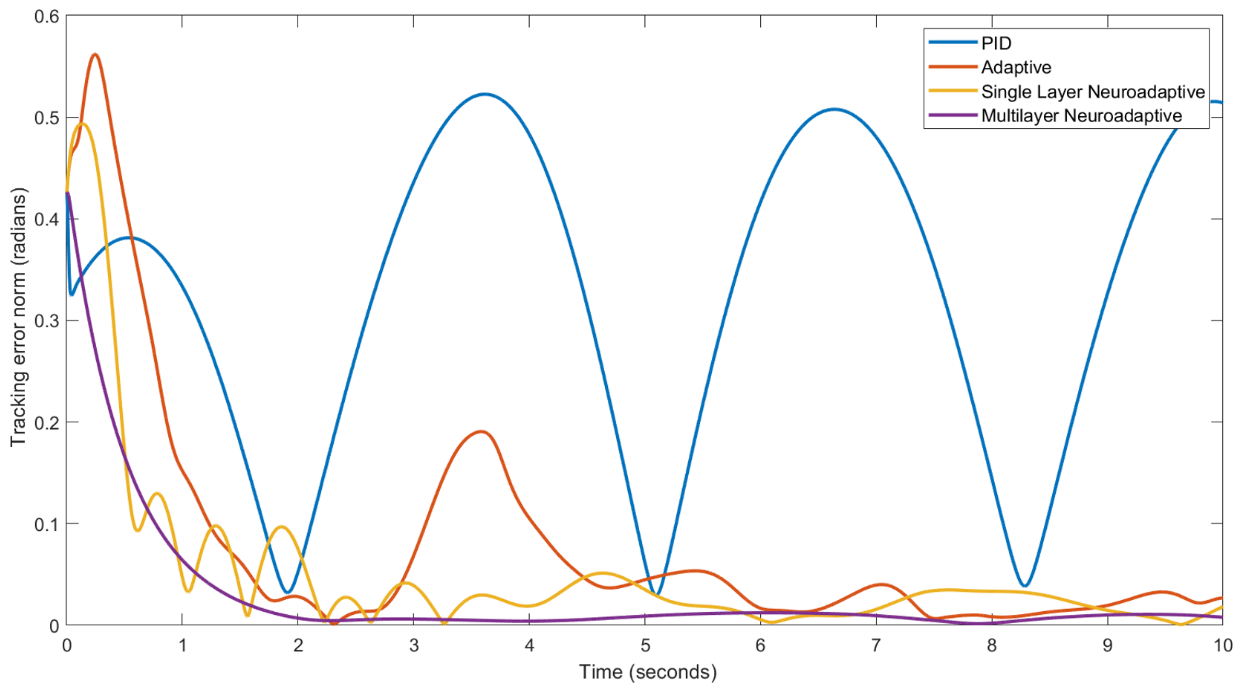

Figure 5.6: Plot of tracking error norm of different controllers

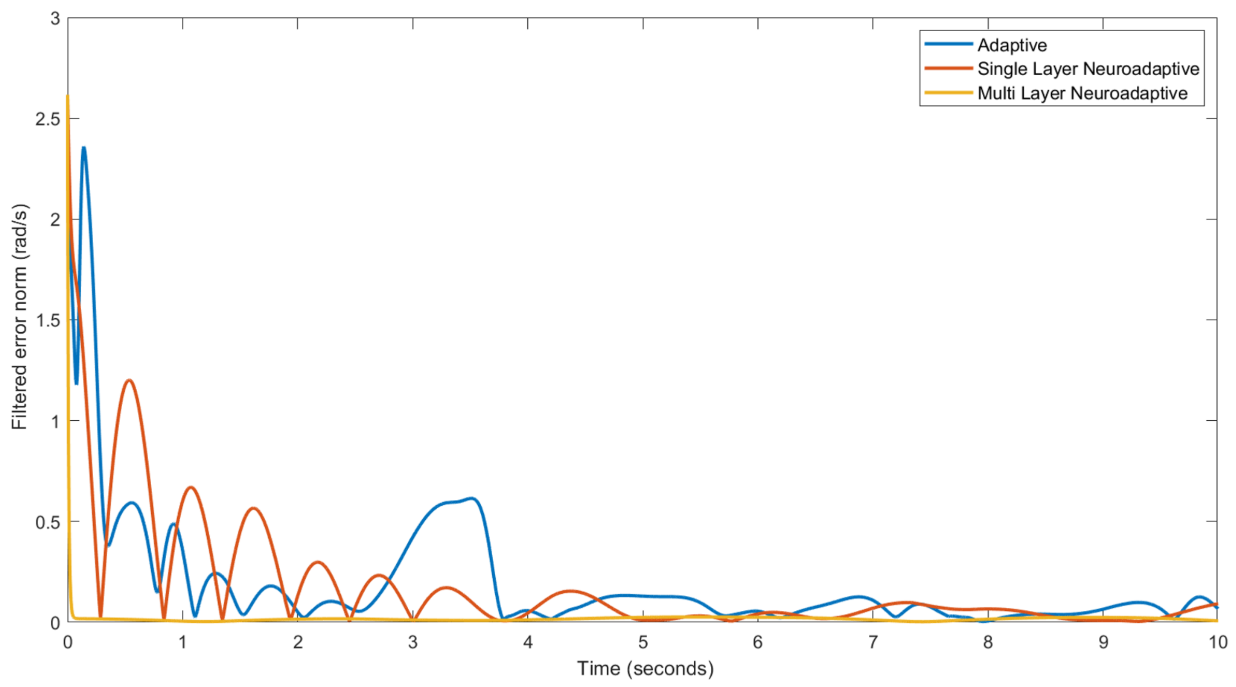

Figure 5.7: Plot of filtered tracking error norm of different controllers

Fig. 5.6 and Fig. 5.7 shows the norm tracking error and filtered error norm of all the controllers used. It can be seen that Multilayer Neuroadaptive controller performed the best in both the categories.

In all the controllers the plant paramteres, initial condition, desired trajectory and controller parameters were kept same. Integral Squared Error and Integral Time Absoulte Error (ITAE) were calculated for a fixed period of 10 seconds of the tracking error.

$$ISE = \int_0^{10} e^2 dt$$

$$ITAE = \int_0^{10} t|e| dt$$

The results are shown in Tab. 5.1 for the sum of the errors in tracking by link-1 and link-2.

Table 5.1: ISE and ITAE measure of different controllers

Controller	**ISE**	**ITAE**
PID	12716.0	196830.0
Adaptive	2042.4	25355.0
Single Layer Neuroadaptive	1036.6	14006.0
Multilayer Neuroadaptive	500.0	6088.1

5.7 Conclusion

As seen in the plots, and the Table 5.1, confirms that out of all the adaptive controllers, multilayer neuroadaptive controller performed the best in reducing the tracking error as filtered tracking error.

Neural Network based adaptive controller performed better because they are more accurate in estimating the unknown parameters. Adaptive training combined with neural architecture worked best in such robots.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Initially controllers- PID, Adaptive, and State Feedback were tested on a single link manipulator. The gains were kept the same and performance was compared. The Adaptive controller performed best with and without deadzone compensation. So for two link manipulator adaptive controller was used to get to the best tracking performance.

Deadzone nonlinearity was handled by passing the control input through inverse deadzone which just adds an offset slightly above the deadzone width. Without the deadzone compensation the adaptive controller though worked, but the tracking error was comparatively bad. The errors were particularly large at the instants where input changed sign.

It was observed that the varying deadzone width leads to huge tracking error and drift in the parameters. This was solved by adding $\sigma-$modification term. The additional term makes the controller UUB stable, so error convergence to zero cannot be guaranteed.

Multilayer neural adaptive controller with sigmoidal activation function was simulated on a two link manipulator with varying deadzone at input. Classical adaptive controller, PID single layer and multilayer nurodaptive controller tracking performance was compared. In term of integral absolute error and integral time absolute error the MN controller performed best. Also the plot of error norms and filtered error norms were shown which was least in the case of MN controller.

6.2 Future Work

In extension to this work, controller can be designed for other nonlinearities like deadband, delay, and backlash. Compensating for these nonlinearities will give more precise tracking control.

Rather than tuning the gains by trail and error method a more better procedure can be used to set the gains like in [8] (section 3.2). Similar methods should be extended for neuroadaptive controller.

Appendix A

Simulink Diagrams

A.1 Classical PD Adaptive controller

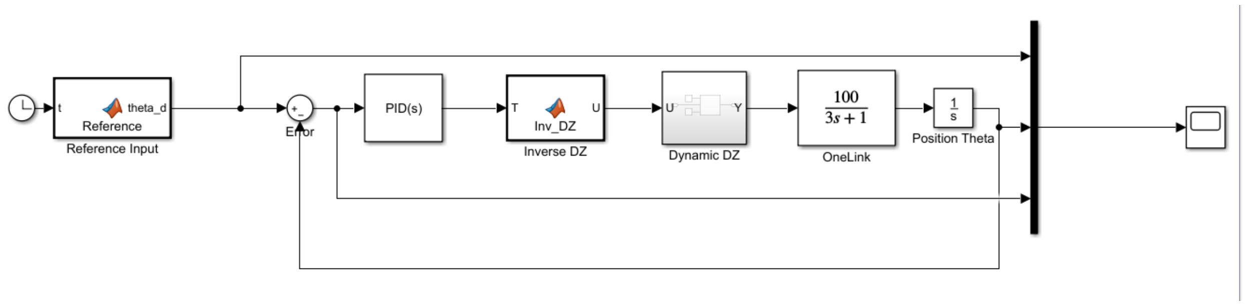

Figure A.1: Simulink model of PID controller

Inverse DZ Block

```matlab
function U = Inv_DZ(T)
dp = 0.4;
dm = 0.18;
n = (T > 0);
U = T + n*dp + (n-1)*dm;
if(T == 0)
U = 0;
end
```

A.2 State Feedback Controller

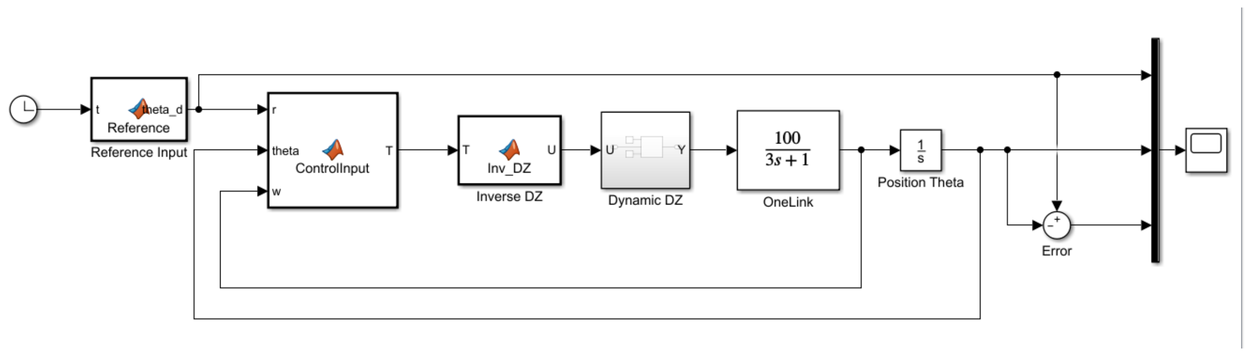

Figure A.2: Simulink model of State Feedback controller

Control Input Block

```
function T = ControlInput(r,theta,w)
x = [theta; w];
T = -[1, 0.05]*x + r;
```

A.3 Single Link Adaptive Controller

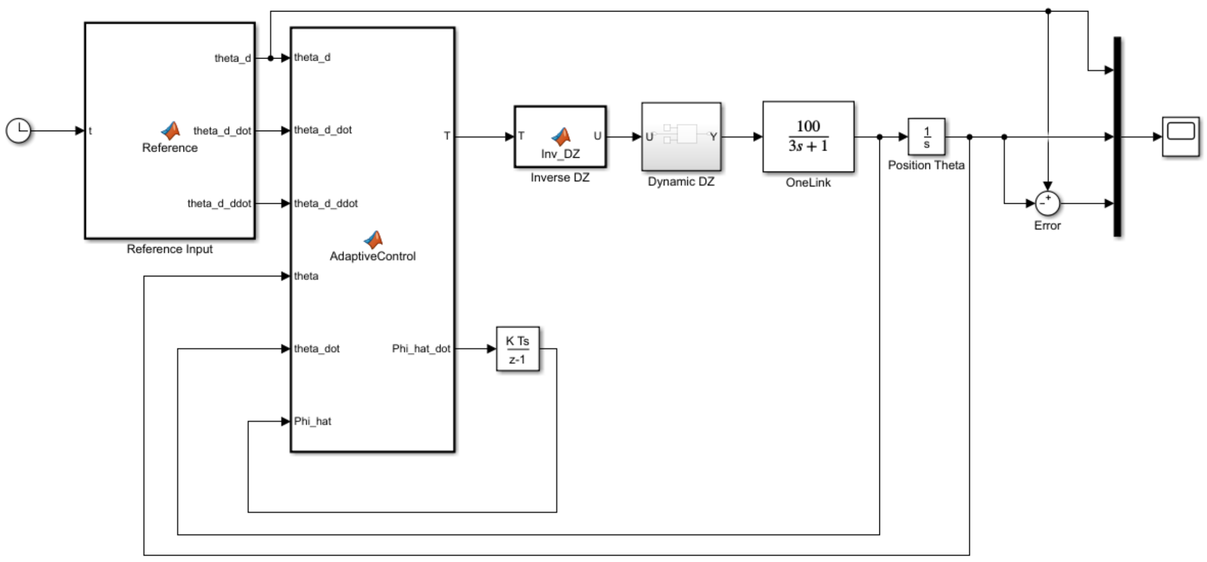

Figure A.3: Simulink model of Single Link Adaptive controller

AdaptiveControl Block

```
function [T, Phi_hat_dot] = AdaptiveControl(theta_d, theta_d_dot, ...
    theta_d_ddot, theta, theta_dot, Phi_hat)
e = theta - theta_d;
e_dot = theta_dot - theta_d_dot;

alpha = 2;
r = e_dot + alpha*e;
%% Adaptive Law
Y = [-theta_dot , -theta_d_ddot + alpha*e_dot];
Gamma = 10;
Phi_hat_dot = Gamma*Y'*r - Gamma*0.1*Phi_hat;
%% Control Input
K = 10;
T = -Y*Phi_hat - K*r;
```

A.4 Two LinkPID Controller

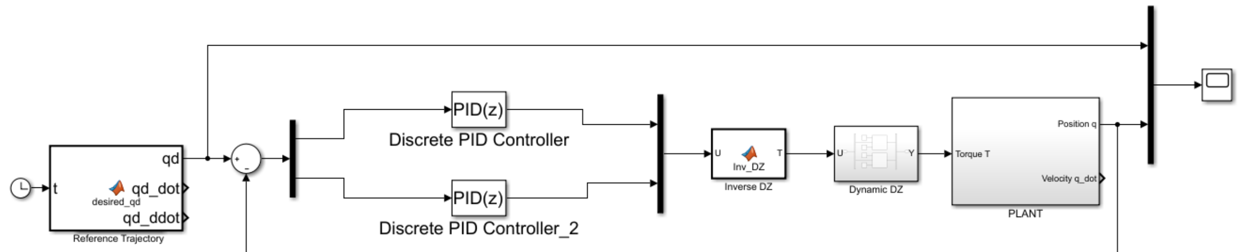

Figure A.4: Simulink model of Two Link PID controller

A.5 Two Link Manipulator

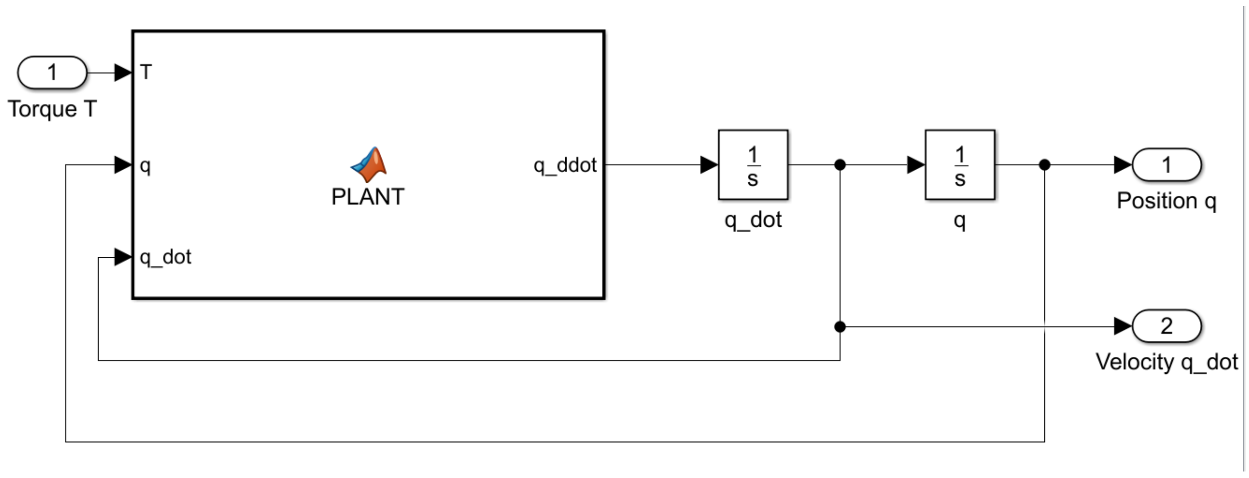

Figure A.5: Simulink model of Two Link Manipulator

Plant Block

```
function q_ddot  = PLANT(T,q,q_dot)
q2 = q(2);
q1_dot = q_dot(1);
q2_dot = q_dot(2);

M = [(3.473+2*0.242*cos(q2)) , (0.196+0.242*cos(q2)); ...
    (0.196+0.242*cos(q2)) , 0.196];
Vm = [-0.242*sin(q2)*q2_dot , -0.242*sin(q2)*(q1_dot+q2_dot); ...
    0.242*sin(q2)*q1_dot , 0];
Fd = [5.3 0;0 1.1];

q_ddot = M\(T-Vm*q_dot-Fd*q_dot);
end
```

A.6 Two Link Adaptive controller

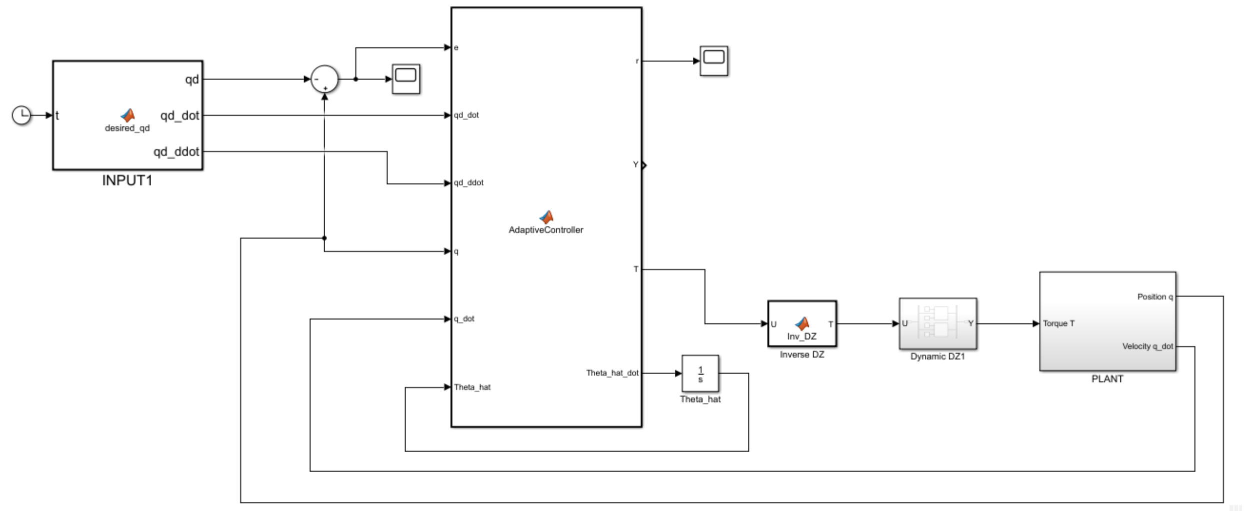

Figure A.6: Simulink model of Two Link Adaptive controller

Adaptive Controller Block

```
function [r,Y,T,Theta_hat_dot] = ...
    AdaptiveController(e,qd_dot,qd_ddot,q,q_dot,Theta_hat)

e_dot = q_dot - qd_dot;
e1_dot = e_dot(1);
e2_dot = e_dot(2);

qd1_dot = qd_dot(1);
qd2_dot = qd_dot(2);

qd1_ddot = qd_ddot(1);
qd2_ddot = qd_ddot(2);

q1_dot = q_dot(1);
q2_dot = q_dot(2);

q2 = q(2);

C2 = cos(q2);
S2 = sin(q2);

a = 2;

r = e_dot + a*e;
r1 = r(1);
```

```
r2 = r(2);
%%
Y11 = a*e1_dot - qd1_ddot;
Y12 = -S2*q1_dot*r2 - C2*(2*qd1_ddot + qd2_ddot) + a*C2*(2*e1_dot + ...
    e2_dot) + S2*q2_dot*(2*q1_dot + q2_dot - r1 - r2);
Y13 = a*e2_dot - qd2_ddot;
Y14 = -q1_dot;
Y15 = 0;
Y16 = -1;%sign(q1_dot);
Y17 = 0;
Y21 = 0;
Y22 = S2*q1_dot*(r1 - q1_dot) + C2*(a*e1_dot - qd1_ddot);
Y23 = a*(e1_dot + e2_dot) - (qd1_ddot + qd2_ddot);
Y24 = 0;
Y25 = -q2_dot;
Y26 = 0;
Y27 = -1;%sign(q2_dot);
Y = [Y11 Y12 Y13 Y14 Y15 Y16 Y17; Y21 Y22 Y23 Y24 Y25 Y26 Y27]';
%%
sigma = 0.1;
Gamma = 10;
Theta_hat_dot = Gamma*Y*r - Gamma*sigma*norm(e)*Theta_hat;
K = 5;
T1 = -Y'*Theta_hat;
T2 = - K*r;
T = T1 + T2 ;
```

A.7 Single Layer Neuroadaptive Controller

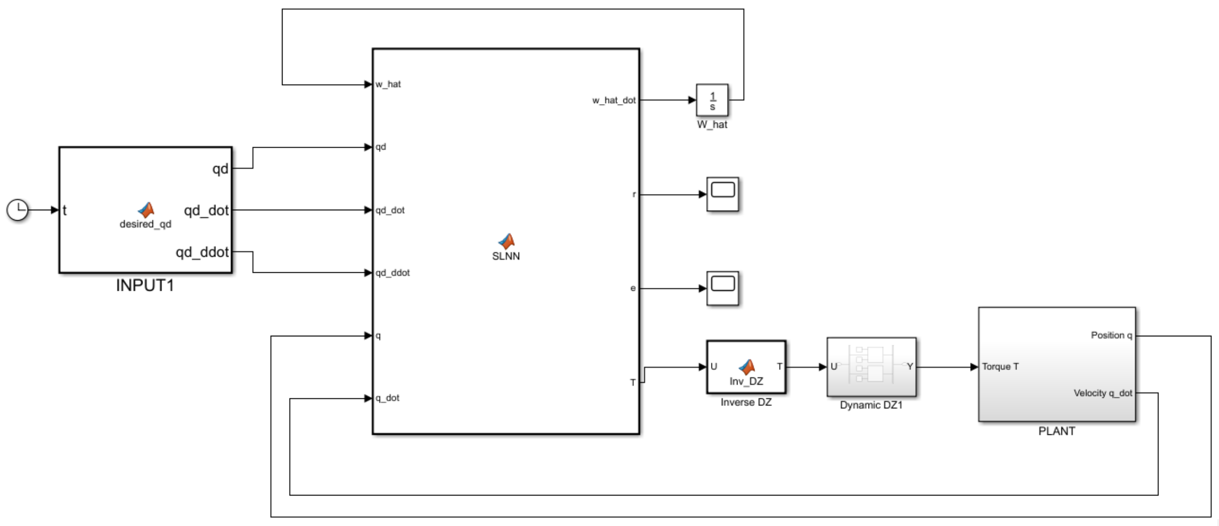

Figure A.7: Simulink model of single layer neuroadaptive controller

SLNN block

```
function [w_hat_dot,r,e,T] = SLNN(w_hat,qd,qd_dot,qd_ddot,q,q_dot)

e = q - qd;
e_dot = q_dot - qd_dot;

X = [1 ; q ; q_dot ; qd ; qd_dot ; qd_ddot];
sigma_input=0;
L = 10;
for i=1:size(X,1)
sigma_input=sigma_input+X(i);
end

sigma = [1 ;zeros(L,1)];

for i= 2:L
sigma(i)= 1/(1+exp(-sigma_input));
end

Tw = 10;
a = 2;

r = e_dot+a*e;

phi = sigma;

w_hat_dot = Tw*phi*r'- Tw*0.1*norm(e)*w_hat;
K = 5;
T = -(w_hat)'*phi-K*r;
end
```

A.8 Multilayer Neuroadaptive Controller

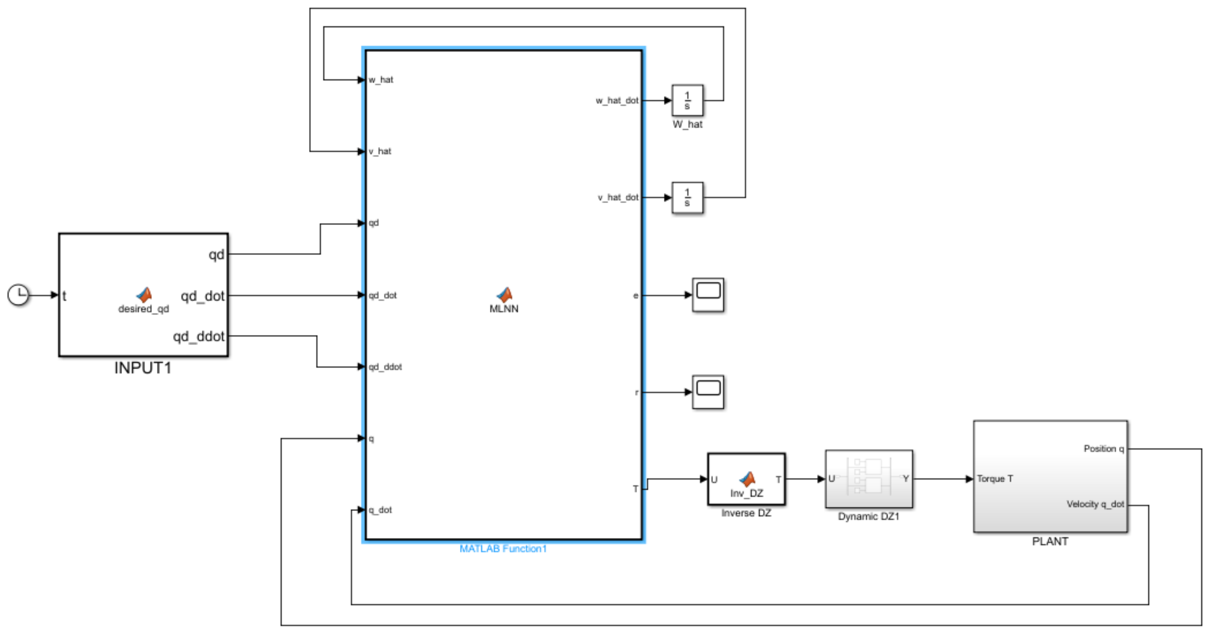

Figure A.8: Simulink model of multilayer neuroadaptive controller

MLNN block

```
function [w_hat_dot,v_hat_dot,e,r,T] = ...
    MLNN(w_hat,v_hat,qd,qd_dot,qd_ddot,q,q_dot)

e = q - qd;
e_dot = q_dot - qd_dot;

L = 10;
X = [1 ; q ; q_dot ; qd ; qd_dot ; qd_ddot];
sigma = [1;zeros(10,1)];
sigma_hatd = zeros(11,10);

Tw = 10*eye(11,11);
Tv = 10*eye(11,11);

A=(v_hat)'*X;
for i=1:L
sigma(i+1)=1/(1+exp(-A(i)));
end
for j=1:L
sigma_hatd(j+1,j) = 0.5*(1-(sigma(j+1))^2);
end

```

```
a = 2;
r = e_dot+a*e;

w_hat_dot = Tw*(sigma-sigma_hatd*A)*r';
v_hat_dot = Tv*X*r'*(w_hat)'*(sigma_hatd);

K = 5;
T = -(w_hat)'*sigma -K*r - e ...
    -5*((2+norm(v_hat))^2)*r-5*((3+norm(w_hat))^2)*r;
end
```

Bibliography

[1] B. X. A. Behal, W. Dixon and D. Dawson. *Lyapunov-based control of robotic systems.* CRC Press, 2010.

[2] R. Ahmed, K. Rattan, and O. Abdallah. Adaptive neural network for identification and tracking control of a robotic manipulator. In *Proceedings of the IEEE 1995 National Aerospace and Electronics Conference. NAECON 1995*, volume 2, pages 601–609 vol.2, 1995.

[3] A. S. D. Psaltis and A. A. Yamamura. A multilayered neural network controller. *IEEE Control Systems Magazine*, 8(2):17–21, 1988.

[4] D. D. F.L. Lewis and C. Abdallah. *Robot manipulator control: theory and practice.* CRC Press, 2003.

[5] S. S. Ge, W. He, and S. Xiao. Adaptive neural network control for a robotic manipulator with unknown deadzone. In *Proceedings of the 32nd Chinese Control Conference*, pages 2997–3002, 2013.

[6] B. Rahmani and M. Belkheiri. Robust adaptive control of robotic manipulators using neural networks: Application to a two link planar robot. *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*, pages 839–844, 2016.

[7] J.-J. E. Slotine and W. Li. On the adaptive control of robot manipulators. *The International Journal of Robotics Research*, 6(3):49–59, 1987.

[8] J.-J. E. Slotine and W. Li. Composite adaptive control of robot manipulators. *Automatica*, 25(4):509–519, 1989.

[9] M. W. Spong and M. Vidyasagar. *Robot dynamics and control.* John Wiley & Sons, 2008.

[10] T. F. S. O. T. Ozaki, T. Suzuki and Y. Uchikawa. Trajectory control of robotic manipulators using neural networks. *IEEE Transactions on Industrial Electronics*, 38(3):195–202, 1991.

[11] Z. Y. W. He, A. O. David and C. Sun. Neural network control of a robotic manipulator with input deadzone and output constraint. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6):759–770, 2016.