# NEURAL NETWORK TRACKING CONTROLLER FOR PUMA 560 MANIPULATOR

**EVANGELOS-STAVROS TZIMOPOULOS**

**Submitted in partial fulfillment of the requirements of**
**The University of Reading for the degree of**
**Master of Science in Cybernetics**

**Department of Cybernetics**
**School of Systems Engineering**

**August 2005**

To my parents for
their unconditional
love and support
all these years

# Acknowledgments

I would like to thank my supervisor Dr. Victor M. Becerra for his support and guidance throughout the whole course. The discussions we had during last year, helped me shape the topic of my dissertation at first, and later go deep into the details of theory, design, and implementation of my project. He was there both as a supervisor and a tutor.

Best wishes to my MSc fellows and best luck for the future. It was a year to remember for all the good and bad times we had together. Kiran, I will always remember the last week we spent in the lab day and night.

This year was a great experience and I will never forget the people I met. Many thanks to my flatmates Antoni, Rodoulla and Maria for having to deal with me. We certainly had very good times in kitchen parties or Greek gatherings. Many thanks to all others for the fun we had throughout the year.

A special thank to Loukianos, a very close friend with whom we shared a lot, traveled and had unforgettable fun. I wish him the best for the future he's fighting for… He's strong and I have faith on him.

# Abstract

Neural Networks have been established as a valuable tool for research and industrial applications in many fields such as classification, pattern recognition and signal processing. This research project presents the framework proposed in [1], where Neural Networks are used for controller design in closed loop applications. As claimed, the feedback control topology and the weight tuning algorithm are such that guarantee closed loop stability and bounded weights. The resulting control scheme belongs to the class of model free controllers. The suggested control structure has been tested in simulations using both a 2 link planar elbow arm and a 6 DOF manipulator, validating its efficiency for closed loop control. Furthermore, the target applications of the above framework have been extended, by applying it to PUMA 560 manipulator for real time trajectory tracking

# Contents

# List of Tables

# List of Figures

# Chapter 1

# 1. Introduction

Biological systems have for long time been a valuable source of inspiration for researchers, due to their tremendous capabilities and mainly due to their properties, meaning autonomy, and robustness as the most important among numerous others.

Research on Artificial Intelligence has focused, besides other fields, in designing feedback control systems that exhibit properties similar to those of biological systems. Recently, the interest in model free controllers has been increased. Mimicking the functions of biological systems, these controllers do not need a mathematical model of the controlled plant. They learn instead, about the systems they are controlling on-line, thus automatically improving their performance.

Literature on Artificial Neural Networks is rich and researchers have demonstrated their effectiveness in many fields. Neural Networks(NNs) have achieved so far great success in the areas of classification , pattern recognition, and open-loop identification, thus becoming an important tool for research and applications in the relevant fields. Closed-loop applications is another class of NNs applications, in which, on the contrary with the previous ones, have not been established as a reliable method. This is due to the fact that designs and methods used were based on heuristics or ad hoc approaches with no repeatable design algorithms

or proofs of stability and guaranteed performance. Back-propagation was the most common weight tuning algorithm used in most of these approaches since tuning algorithms suitable for closed-loop applications were not available. Similar NNs designs have mimicked adaptive control techniques [2], [3]. In these cases, NNs have been used in feedback control topologies such as indirect identification-based control or inverse dynamics.

An extensive literature review regarding early NN approaches can be found in [4]. A brief overview will be given here, focusing on later approaches that paved the way of neural networks in system theory applications

When research on NNs closed loop applications focused on the fact that it is necessary to begin with the knowledge available about the system being controlled, progress on that field started. Narendra [5], [6] and other researchers [7], pioneered rigorous NN control applications by studying the dynamical behavior of NNs in closed-loop systems, including computation of the gradients needed for back-propagation tuning. Other groups provided rigorous techniques for closed-loop NN controllers, such as how to use radial basis function NN in feedback control , NN tuning algorithms based on dead-zone methods [8], [9], projection methods [10], NN controllers for discrete-time systems [11], and dynamic NN for feedback control [12].

The approach taken by Lewis in [1], [13] is adopted in this research project. The structure discussed is a multi-loop intelligent control structure with an inner NN feedback linearization loop and an outer tracking unity-gain PD loop. This configuration applies for all rigid Lagrangian systems, including robot arms, as well

as for unknown systems including certain important classes of nonlinear systems. Moreover, it is claimed that all major problems of NNs closed-loop applications have been addressed, demonstrating that neural networks do indeed fulfill the promise of providing model-free learning controllers for a class of nonlinear systems.

## 1.1 Research question

This dissertation deals with the application Neural Networks for the control of a certain class of nonlinear systems like are the rigid robot arms. The proposed framework provides the theoretical background for doing so, and presents rigorous results from the application of the controller into a simulated two-link planar arm. In addition, some practical applications, including force control and parallel-link robots, are presented. As claimed, this technique can be extended to robots with link flexibility, robots with actuator and gear dynamics, and general nonlinear processes that are feedback linearizable. The application of the proposed control scheme into a different platform as is the 6 DOF industrial Puma 560 manipulator is another step towards answering the research question regarding the suitability of NNs for closed loop applications:

> *Can Neural Networks provide a reliable framework for the control of manipulators and nonlinear systems in general?*

As mentioned before, Neural Networks are already established in fields such as classification, pattern recognition and open-loop identification. However, the same cannot be said for Neural Networks in system theory applications although a great deal of research has been done. It is generally considered to be an open research field,

and to the extend of the author's knowledge, most of the results presented so far have mainly focused on simulations rather than experimental work. In addition, very little or no experimental work has been done using a 6 DOF manipulator like Puma for neural network control.

## 1.2 Research objectives

This research project extends the target applications of the proposed framework by moving one step ahead. The control scheme is tested both on simulation and on real time experiments using the platform presented in [14].

The primary objectives upon which the achievements of this research will be measured are closely related to the challenges for neural network control as described in [1], with respect to the specific platform used for the experiments:

**1.** Repeatable design algorithm: The approach taken by this dissertation tries to assess whether the proposed algorithm qualifies as a repeatable design algorithm, by applying it to a different platform. Implementation must be exact and only modified to match the requirements of the specific platform.

**2.** NNs weight tuning: Selecting tuning algorithms to ensure system stability is a nonlinear adaptive control problem because it is nonlinear in the adjustable parameters. The weight tuning algorithm used, must demonstrate that weights remain bounded so that system can track the desired trajectory.

**3.** Trajectory tracking: The resulting system should demonstrate reliable trajectory tracking, meaning the error must remain bounded.

4. Simulations: Special care should be taken so that all parameters from simulated robot dynamics to possible disturbances or unmodeled dynamics are encountered properly in the simulations. The primary objective in simulations is to carry out experiments that achieve the desired performance and provide insight about the simulated system in order to proceed to the next step which is experiments on the real platform.

5. Real time control: The ultimate and final objective of this research project is to configure the system so the controller can respond in real time effectively allowing the PUMA manipulator to follow the desired trajectory

# Chapter 2

# 2. Methodology

The application of the proposed framework to a Puma 560 manipulator is a difficult and challenging process. On the one hand, the algorithm used is a very sophisticated control structure with many components. The target platform, on the other hand, is a 6 DOF manipulator with complex dynamics not always encountered in simulated models. The approach followed by this research project has been very systematic in order to facilitate modeling, implementation and testing of the proposed neural network controller to Puma. The stages involved include:

- controller design and testing using a simulated two link manipulator
- performance evaluation and comparison with a simple PD controller
- testing and tuning using a simulated Puma model
- real time experiments with Puma 560

The need for convenient and systematic simulations has dictated the design and implementation of a simulation platform that can provide the means for effective testing and debugging.

## 2.1 Simulation Platform

A minimal control scheme has been built, which serves as a platform for testing and simulations of the various components that will be used before incorporating them to Puma 560 for real time experiments. The simulation platform which consists of the

trajectory generator, the controller to be tested and the simulated robot model is shown in Figure 1. The facilities block allows for evaluation of the controller, by logging data, plotting crucial signals or generating metrics of the controller's performance.

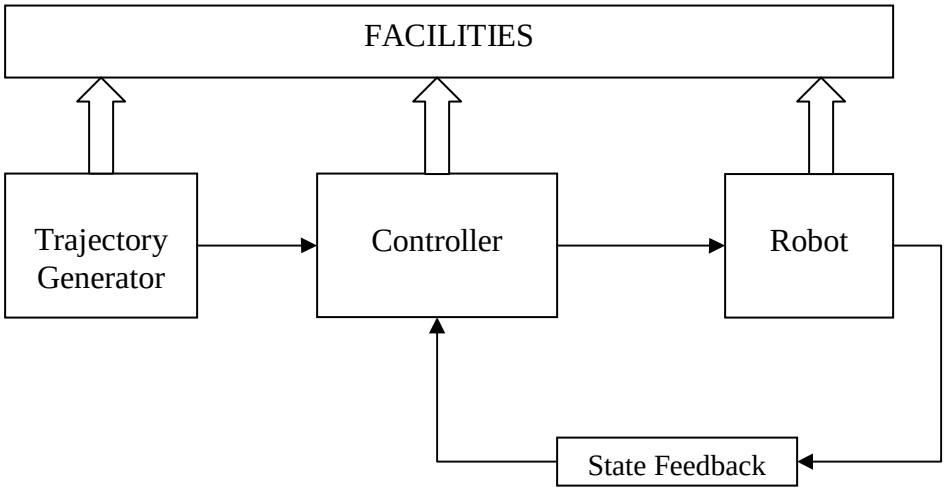



**Figure 1.** Platform used for simulations and testing

The main advantage of this control scheme is that its modularity allows separate controllers design which can be tested for different trajectory generators and different robot models. The latter is a desired property which is closely related to the evaluation of the first research objective which is the repeatable design algorithm. With respect to the above platform, the individual components have been developed separately in a toolbox like manner and used accordingly depending on the simulation configuration.

## 2.2 Controller design

This stage involves the design and implementation of the Neural Network rigid robot controller as described in [1]. Its structure is derived using a filtered error approach which is in standard use in robotics, in combination with some notions in Neural

Network theory. As a result, the controller guarantees bounded tracking errors as well as bounded NN weights. The weight tuning algorithm is based on back-propagation, with some extra features including novel correction terms to the delta rule and a robustifying signal. The multi-loop structure includes a PD controller that drives the filtered tracking error, the NN that is used as an estimator for robot dynamics and the robustifying signal that compensates for unmodeled dynamics. The structure of the controller along with its properties will be presented analytically in the following chapter. In this stage it is only necessary to give a brief description of the simulation platform components.

In addition to the above controller, a PD controller has been implemented which also uses filtered tracking to track the reference signal. The need for a simple controller with few components is obvious, since it allows to evaluate the robot model and the reference signals. Moreover, useful conclusions can be drawn regarding the performance of the complex NN-PD control scheme comparing to simple PD under same conditions, meaning the reference input and PD gains.

## 2.3 Robot models

The simulated robot dynamics of Puma 560 have been taken and used directly from Robotics Toolbox [15]. In addition, a planar two link elbow arm – RR model – has been implemented with its dynamics given in [4]. The robot parameters though, like mass and length, have been taken from [1], for being aligned with the original paper, thus having the opportunity to compare results. The RR model is simple enough to simulate conveniently, but still contains all nonlinear terms arising in a general n-link manipulators. As a result, it can be used for testing, debugging and evaluation of the

controller at the first stage of the simulations. Moreover, the reduced computational needs, due to its simplicity, allow for a lot experimentation with the controller parameters and different configurations.

## 2.4 Trajectory Generator

The main trajectory generator, which for convenience will be referenced as Gen1, uses a simple sin wave as the reference input signal of the joint desired trajectory, with cosines and sine waves as the joint desired velocity and acceleration:

$$\begin{aligned} q_d &= \sin(t) \\ \dot{q}_d &= \cos(t) \\ \ddot{q}_d &= -\sin(t) \end{aligned} \tag{1}$$

As an alternative, the standard trajectory generator – Gen2 – of the Robotics Toolbox [15] can be used, which uses interpolation to calculate the trajectory between an initial and final user defined joint positions. As an extra feature, a sine wave has been superimposed to the generated trajectory for any chosen link.

Finally a more complex trajectory of superimposed sine and cosine waves is available (Gen3) as will be discussed in the relative simulations:

$$\begin{aligned} q_d &= 0.9\sin(0.2t) + 0.2\cos(3t) \\ \dot{q}_d &= 0.18\cos(0.2t) - 0.6\sin(3t) \\ \ddot{q}_d &= -0.036\sin(0.2t) - 1.8\cos(3t) \end{aligned} \tag{2}$$

## 2.5 Simulations

Having described all the modular components that will be used throughout the simulations, herein will be introduced the methodology followed in order to validate the models, evaluate the controller and do comparative analysis. The approach taken

starts testing relative simple control schemes and the complexity is increased gradually in order to ensure functionality and reliability of the individual components.

The first stage involves simulations with the RR model as shown in Figure 2. The model has been tested and debugged using the simple PD controller. Switching from PD to NN control occurs when the model is validated and the gains of the PD controller have been chosen carefully for good tracking. These will be used as reference values for the gains of the PD controller of NN. The latter is being tested and the process repeats until reference values for the parameters of the NN and the robustifying signal can be found that provide efficient tracking. This stage of simulations is the most important as the RR model is relatively simple and thus the controller behavior can be studied under different reference input signals.

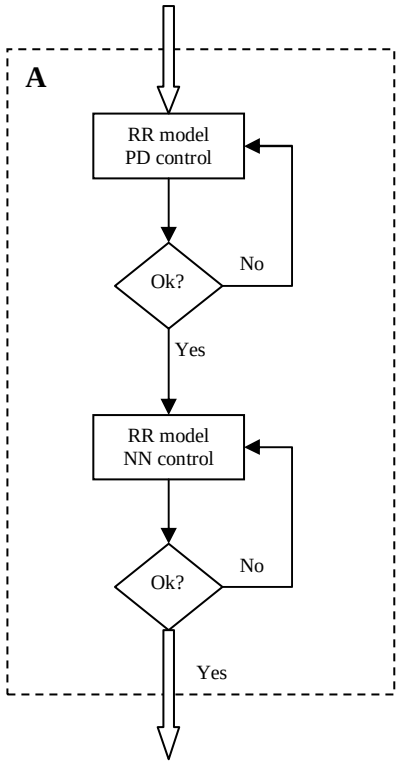



**Figure 2.** RR model

The second stage of the simulation process, reduces the complexity of the controller used, switching back to PD control, but increases the complexity of the model, since simulations with Puma model will take place in this stage. The steps

taken here are similar with those of previous stage. Figure 3 describes the simulations for Puma model while Figure 4 summarizes the whole simulation process with a flow diagram.

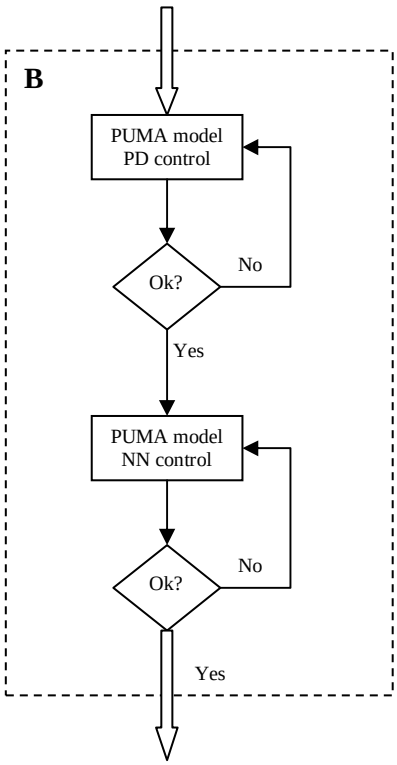



**Figure 3.** Puma model

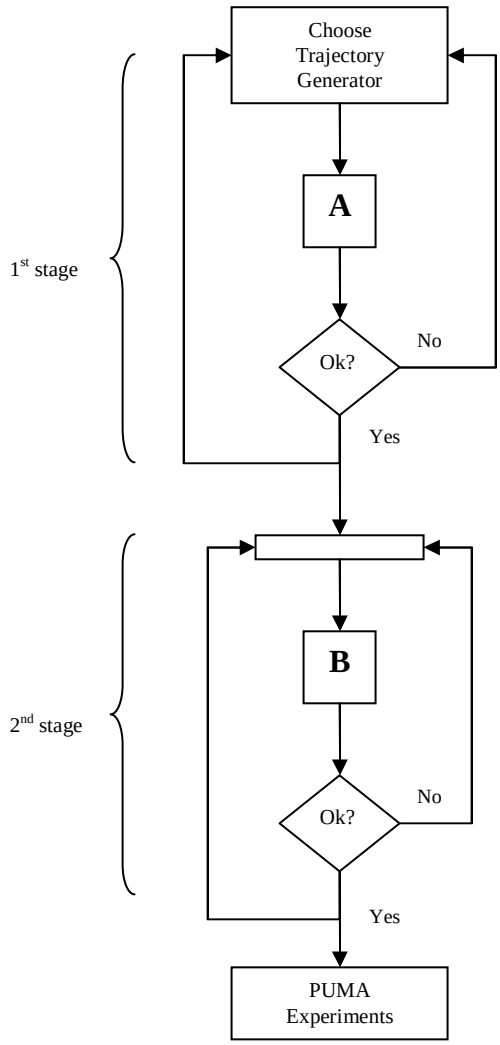



**Figure 4.** Simulation Process

Once the simulations are completed, the controller has been tested and evaluated extensively and a pool of base reference values of its parameters has been created. These values will be used in the experiments that follow the simulations.

## 2.6 Experiments

The final stage of this research project involves real time experiments with Unimation Puma 560 manipulator. The methodology followed in the experiments is similar to that of simulations:

- Choose trajectory generator
- Configure the system for PD tracking control
- Configure system for NN tracking control using the PD parameters from previous stage
- Repeat if necessary any of the steps

The insight of the controller features and behavior acquired in simulations is important in terms of providing some reference values from which one can start tuning the controller for real time experiments without completely unexpected results.

# Chapter 3

# 3. Neural Network Controller

This chapter discusses the details of the proposed framework which relies on some aspects of robot control and NN theory tied together. The approach taken, is not based on ad hoc techniques, on the contrary results directly from proofs based on ideas in robot control and a useful property of the neural networks that can provide estimation accuracy within given bounds. This neural network approach can be used for any rigid robot manipulator, meaning one without flexible links or high frequency joint and motor dynamics. Some characteristic properties of the neural network rigid robot controller that will be presented here are:

- The neural network weights are tuned online and no preliminary learning phase is needed
- The controller structure includes an outer tracking PD loop that ensures convenient performance until the neural network begins to learn
- The tracking performance is guaranteed in terms of bounded tracking errors and neural net weights

## 3.1. Neural Network background

A common NN structure is the feedforward three layer neural network presented in Figure 5. This structure has two layers of adjustable weights, the first to second layer interconnection weights $V_{jk}$ and the second to third layer weights $W_{ij}$.

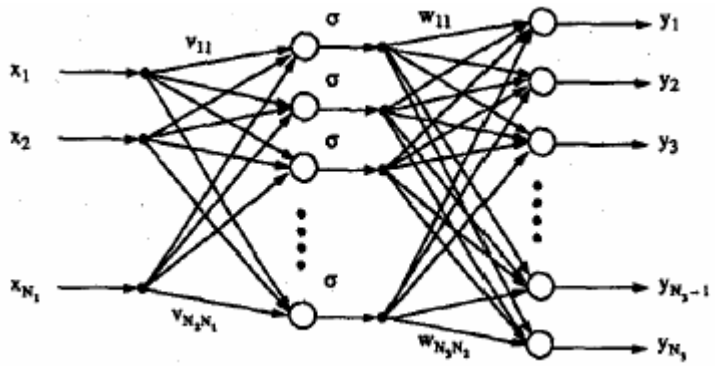



**Figure 5.** Feedforward three layer neural network

Given an input vector $\chi_{N1}$ the output $y$ is given by the following equation:

$$y_i = \sum_{j=1}^{N_2} \left[ w_{ij} \sigma \left[ \sum u_{jk} \chi_k + \vartheta_{uj} \right] + \vartheta_{wi} \right] \tag{3}$$

or in vector format:

$$y = W^T \sigma\left(V^T \chi\right) \tag{4}$$

where $N_r$ is the number of neurons in layer $r$ and $N_2$ is the number of hidden layer neurons. The $\theta_{uj}$ and $\theta_{wi}$ are the threshold offsets of the hidden and output layer respectively. The activation functions $\sigma(\ )$ are required to be smooth enough to have at least a first derivative.

### 3.1.1. Function approximation property

An important property of neural networks is the *function approximation property*. And that is, if $f(x)$ is a smooth function from $R^n$ to $R^m$ and $x$ is restricted to a compact

set $S$ of $R^n$, for some number of hidden layer neurons $N_h$, there are weights and thresholds such that

$$f(x) = W^T \sigma(V^T x) + \varepsilon \tag{5}$$

where $\varepsilon$ is the *neural network functional approximation error* and it generally decreases as the net size $N_h$ increases. This equation means that a neural network can approximate any smooth function on a compact set. The ideal weights $W$ and $V$ that best approximate a given nonlinear function $f(x)$ are difficult to determine. In fact, they might not be unique. However, this property ensures that for a specified value of $\varepsilon_N$ some ideal approximating neural network weights do exist. Then the estimate of $f(x)$ can be given by

$$\hat{f}(x) = \hat{W}^T \sigma(\hat{V}^T x) \tag{6}$$

where $\hat{W}$ and $\hat{V}$ are estimates of the ideal neural network weights provided by some on line weight tuning algorithms.

### 3.1.2. Weight tuning algorithms

In applications where weights change over time, weight tuning algorithms are used, which are usually based on a type of gradient algorithm. A common gradient based algorithm is Backpropagation, where the error to be back propagated in the training algorithm is defined as the desired neural network output minus the actual output $E$. The weight tuning law in this case is:

$$\begin{aligned} \dot{\hat{W}} &= F\sigma(\hat{V}^T x_d) E^T \\ \dot{\hat{V}} &= G x_d (\hat{\sigma}'^T W^T E^T) \end{aligned} \tag{7}$$

where $F$, $G$ are positive definite design parameters governing the algorithms speed of convergence. The hidden layer output gradient for the sigmoid activation function which is quite common is:

$$\hat{\sigma}' = diag\{\sigma(\hat{V}^T x_d)\}[I - diag\{\sigma(\hat{V}^T x_d)\}] \quad (8)$$

Backpropagation is considered to be a supervised training scheme, since the tuning occurs off line and requires specified training data ($x_d$,$y_d$) with $x_d$ the ideal neural network input that yields the desired neural network output $y_d$.

## 3.2. Robot arm dynamics

The dynamics of an n-link rigid robot manipulator can be expressed in the Lagrange form [1]:

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \quad (9)$$

where $q(t)$ is the joint variable vector, $M(q)$ the inertia matrix, $V_m$ the coriolis/centripetal matrix, $G$ the gravity vector, and $F$ the friction. $\tau_d$ denotes bounded unknown disturbances including unstructured unmodeled dynamics, and $\tau$ is the control input torque.

An important application in robot arm control is tracking a reference trajectory input signal. Given a desired trajectory $q_d(t)$ the tracking error is:

$$e(t) = q_d(t) - q(t) \quad (10)$$

In the robotics literature, a parameter of interest is the *filtered tracking error*:

$$r = \dot{e} + \Lambda e \quad (11)$$

where $\Lambda = \Lambda^T > 0$ is design parameter matrix, usually selected diagonal. Differentiating $r(t)$ and using (9), the arm dynamics can be written in terms of the filtered tracking error :

$$M\dot{r} = -V_m r - \tau + f + \tau_d \quad (12)$$

where the nonlinear robot function is:

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) + F(\dot{q}) \tag{13}$$

The vector required to compute *f(x)* can be selected as:

$$x = \begin{bmatrix} e^T & \dot{e}^T & q_d{}^T & \dot{q}_d{}^T & \ddot{q}_d{}^T \end{bmatrix} \tag{14}$$

Function *f(x)* contains all the robot parameters, including payload mass, link masses and lengths, and friction coefficients. Often in applications, these quantities are difficult to determine thus making *f* at least partially unknown.

A suitable control law in such cases is:

$$\tau = \hat{f} + K_v r - \nu \tag{15}$$

where $K_v = K_v^T > 0$, a diagonal gain matrix and *f* an estimate of the robot nonlinear function. The robustifying signal *v* is incorporated for disturbance rejection. This control law is based on *control-law partitioning* [16], where system parameters, such as coriolis and centripetal matrix, the gravity vector and the friction are separate in the design from the servo portion which is independent of these parameters. The application of the above control law to the closed loop system equation (12) yields:

$$M\dot{r} = -(K_v + V_m)r + \tilde{f} + \tau_d + \nu \tag{16}$$

where the functional estimation error is given by:

$$\tilde{f} = f - \hat{f} \tag{17}$$

This is an error system wherein the filtered tracking error is driven by the functional estimation error. The estimate of *f*, for the computation of the control signal, can be achieved by several techniques including adaptive control. This approach though will focus on the neural network approach as described in [1], [4] and [13] and which will be presented in the following subsection.

## 3.3. Multiloop neural network topology

The proposed Neural Network structure based on the control law discussed earlier is shown in Figure 6

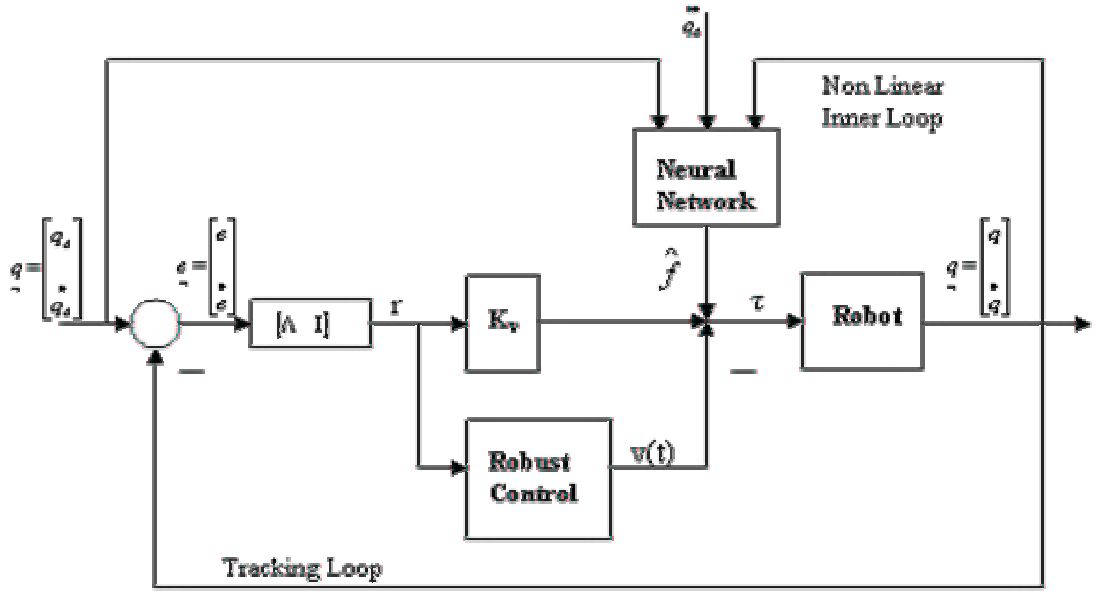



**Figure 6.** Neural Network control scheme

The chosen control law $\tau$ incorporates a proportional derivative PD outer loop in the term $K_v r=K(ed+\Lambda e)$, which keeps the system stable in the initial phase when NN has not started to learn. The neural network is used in an inner loop to provide the estimate for the robot function $f(x)$. PD control based on filtered tracking error is standard in the robotics literature and guarantees bounded tracking errors if the $K_v$ gain is large enough. The neural network approximation property confirms that there is always a neural network that can provide this estimate within a given accuracy.

The main issue of concern at this point is selecting a suitable NN tuning algorithm, signal $v(t)$ and gain $K_v$ that guarantee the stability of the filtered tracking error. Then, from system of equation (9), with $r(t)$ the input and $e(t)$ the output, standard techniques guarantee that $e(t)$ exhibits stable behavior. The standard back

propagation weight tuning algorithm does not suffice as addressed in [17] and thus modified weight tuning rules based on the delta rule with back propagation have been suggested that guarantee tracking and bounded weights. The following theorem is taken directly from [17] and presents the neural network rigid robot controller as proposed in [1],[4],[13],[17].

*Theorem 1:* Let the desired trajectory be bounded by

$$\left\| \begin{matrix} q_d \\ \dot{q}_d \\ \ddot{q}_d \end{matrix} \right\| \leq Q_d \tag{18}$$

and the ideal weights bounded by known positive value $Z_M$ so that

$$\|Z\|_F = \left\| \begin{matrix} W & 0 \\ 0 & V \end{matrix} \right\| \leq Z_M \tag{19}$$

Take the control input (15) with robustifying term

$$v(t) = -K_Z(\|\hat{Z}\|_F + Z_M)r \tag{20}$$

and gain

$$K_Z > C_2 \tag{21}$$

with $C_2$ known constant and let the neural network weight tuning be provided by

$$\begin{aligned} \dot{\hat{W}} &= F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^T x r^T - kF\|r\|\hat{W} \\ \dot{\hat{V}} &= Gx\left(\hat{\sigma}'^T\hat{W}r\right)^T - kG\|r\|\hat{V} \end{aligned} \tag{22}$$

with any constant matrices $F = F^T > 0$, $G = G^T > 0$ and scalar design parameter $k > 0$. Then, for large enough control gain $Kv$ the filtered tracking error $r(t)$ and NN weight estimates $W$, $V$ are *UUB*. Moreover the tracking error may be kept as small as desired by increasing the gains $Kv$.

The first terms in equation 22 are the same with the back propagation tuning algorithm. In this case thought, there is no need for input-output pairs $(x_d,y_d)$ for tuning. In fact the signal that needs to be propagated is the filtered tracking error $r(t)$. The hidden layer gradient can be easily computed in terms of the input vector and the current estimates of $V$. The last terms correspond to the $e$-modification in [18], while the second term in $W$ tuning is a novel term corresponding to a second order forward propagating wave in the back propagating tuning network [19].

# Chapter 4

# 4. Implementation

Following the analytical presentation of the proposed Neural Network structure, some important features of the controller will be presented. Aspects like the weights initialization, the preprocessing of neural network inputs and the complexity of the net will be discussed with respect to the simulation platform and its own individual properties. Moreover, issues concerning the 2 link RR model will be highlighted and Puma model specific limitations will be addressed. Finally, the transition from the simulation platform to Puma, entails some changes and modifications to facilitate the use of the experimental platform interfacing with the robot.

## 4.1. Neural Network Controller

The controller introduced in *Theorem 1* has been implemented with some slight modifications that will subsequently be discussed. The Simulink design of Neural Network Rigid Robot Controller Block appears in Appendix I.C. The weight tuning algorithm described by equation (22) represents a dynamic system with internal states, the estimates of weights $W$ and $V$. In each simulation step the output of the integrators stands for the current estimate of the ideal weights. For the activation function of the hidden layer neurons, a simple sigmoid has been chosen, although any NN activation function can be used as long as its first derivative is bounded and the approximation

property holds. The norms (2-vector and Frobenius matrix form) and the Jacobian σ' can easily be computed in terms of known signals according to equations. Finally, the values of bound $Z_M$ and gain $K_Z$ will be discussed at simulations, but generally large positive values suffice.

### 4.1.1. Neural network weights

In some neural network applications it is desirable that the weights and thresholds are adapted and tuned together in real time to provide a suitable performance of the net. A convenient way to do that is by augmenting the input vector $x$ by adding $x_0=1$ as its first element. This allows to include the threshold vector $[\theta_{u1}, \theta_{u2},\dots \theta_{uN2},]$ as the first column of $V^T$, so that $V^T$ contains both the weights and thresholds of the first to second layer interconnections. Then the output is given by equation (4). Similarly, by augmenting the $\sigma(V^T x)$ vector with one as its first element, allows one to incorporate the thresholds $\theta_{wi}$ as the first column of $W^T$. As a result, any tuning of $W$ and $V$ includes tuning of the thresholds as well.

### 4.1.2. Preprocessing neural network inputs

The input vector to neural network can be chosen differently than equation (14) with some simple signal processing. In this way, it becomes more informative as it can introduce some of the nonlinearities inherent to robot arm dynamics. Furthermore it can reduce the neural networks burden of expectation and reduce the functional approximation error $\varepsilon$ in equation (5). The suggested new input vector has been adopted from [1] and is:

$$x = \begin{bmatrix} \ddot{q}_d + \Lambda \dot{e} & \dot{q}_d + \Lambda e & \sin(q) & \cos(q) & \dot{q}^T \end{bmatrix} \quad (23)$$

Note that the first two terms correspond to the first two terms of the nonlinear robot function (13).

### 4.1.3. Weights initialization

A problem often encountered in neural networks is the initialization of neural network weights in order to achieve initial closed loop stability. This problem has lead to the need for extensive off line training schemes to estimate the plant dynamics. On the contrary, the neural network structure of *Theorem 1* needs no off line training. The weights are simply initialized to zero. Then the neural network learns on line and real time. This is due to the multiloop topology with the PD outer tracking loop keeping the system stable until the neural network adequately learns the nonlinear robot function. This on line learning feature is very important as, in contrast with standard backpropagation weight tuning, makes the controller work in unsupervised mode.

Alternatively, a suitable initialization of weights is more advantageous as shown in [20]. And that is to select *V(0)* randomly and *W(0)* equal to zero. Both approaches have been tested and the random initialization has been preferred.

### 4.1.4. Neural network complexity

The size of the neural network has an impact on the tracking performance of the neural network. A larger net with more neurons in the hidden layer increases the computational burden as for each weight an integrator is needed. On the other hand,

the larger the number of hidden layer neurons, the smaller the functional approximation error will be and thus the smaller the tracking error.

However, the properties of the neural network structure discussed here, allow for flexibility and relative freedom in choosing the design parameters. Even if a simplified NN with few neurons in the hidden layer is used, which yields larger tracking errors, by selecting larger values for $Kz$ or $\Lambda$ one can compensate for the errors. A fairly simple NN has been chosen with ten hidden neurons throughout the simulations and the experiments that will be described in the following chapters. This decision has been made after some early experimentation with RR and Puma models. The focus will thus be given on choosing the rest parameters for achieving good tracking.

## 4.2. Simulations

The simulation platform and all the components were implemented in Matlab/Simulink, since the experimental platform that interfaces with Puma 560 has been designed similarly [14]. More specifically, a toolbox was built for the needs of this project. *Lab560Toolbox* is presented in Appendix I with all the components designs that will be referenced throughout this text.

### 4.2.1. Getting started

The simulation platform appears in Appendix I.A as part of *Lab560Toolbox*. There are two versions available, one for the two link model and one for Puma, sharing though the same properties and capabilities. Depending on the model tested, the various components can be selected and plugged into the design. The initialization

script *initLab560Sim.m* appears in Appendix II and contains all parameters of the controller presented in *Theorem 1*. Initializing the simulation platform is as easy as selecting the desired parameter values, setting the variable for the model and running the script from Matlab command line.

### 4.2.2. RR model simulations

The model of the 2 link planar elbow arm has been implemented as an S-function Simulink block with its dynamics given by [4]. The robot parameters were taken as $m_1$=0.8Kg, $m_2$=2.3Kg, $l_1$=$l_2$=1m and the gravity constant as g=9.81m/s$^2$. The Simulink design of the simulation platform adjusted for RR simulations is presented in Appendix I.A1. Switching from PD to NN control for RR model can be done easily by selecting the relative controller from *Lab560toolbox*, plugging it into the design, and running the initialization script.

### 4.2.3. Puma model simulations

The robot model has been available through the robotics toolbox as mentioned before. The Simulink design of the simulation platform is presented in Appendix I.A2 and has been modified to facilitate realistic simulations of Puma. More specifically, an extra block has been added in order to control the actual position and velocities of the joints. This block is already implemented in the platform interfacing with Puma 560 and was included in the simulations for more smooth transition to real experiments. The *Limiter* block imposes the following restrictions to the range of position and velocities:

**Table 1.** Joint positions range

| LIMITER | |
|---|---|
| Link | Joint range (rad) |
| 1st | -2.95 < q1 < 2.95 |
| 2nd | -4.1078 < q2 < 0.9771 |
| 3rd | -1.0346 < q3 < 4.2428 |
| 4th | -3.141 < q4 < 3.141 |
| 5th | -1.8963 < q5 < 1.8963 |

The limit for joint velocities is:

$$-2.2 < \dot{q}_i < 2.2 \ \ rad/s \tag{24}$$

where the values are in radians.

These limits are virtual when running simulations, but were included in the simulations in order to configure and simulate a system that would be as close as possible to the physical system before trying real time experiments with Puma 560. The given range of values for position and velocities have thus been taken into account when tuning the system. The above limitations introduce changes at the design with respect to the two link model. For example, the input reference signal of the trajectory generator cannot be a sine wave with amplitude 1 for the second link. As a result Gen1 and Gen3 that were used in RR model simulations, have been modified. Gen 1 has been changed to:

$$\begin{aligned} q_1 &= \sin(t) \\ q_2 &= -1 + \sin(t) \\ q_3 &= 0.5 + \sin(t) \end{aligned} \tag{25}$$

and Gen3 to:

$$\begin{aligned} q_1 &= \sin(t) \\ q_2 &= -1 + 0.9\sin(0.2t) + 0.2\cos(3t) \\ q_3 &= 1 - 0.9\cos(0.2t) - 0.2\sin(3t) \end{aligned} \tag{26}$$

calculating the desired joint velocities and accelerations for each link respectively. Their designs appear in Appendix I.D as part of the toolbox. Note, simulations have focused in only the first three links as being representative enough to study the controllers behavior.

## 4.3. Experimental setup

The Simulink design of experimental platform [14] also appears in Appendix I.B, slightly modified to facilitate the needs of this research project. The transition from simulations to real time experiments introduces further changes with respect to the features of the experimental platform. Besides the restrictions imposed by the limiter, as discussed in Puma simulations, the robot initial position for performing experiments is fixed at [0 $-90^0$ $90^0$ 0 0 0] joint angle set, which should be taken into account when designing the input reference signal. As a result, Gen1 and Gen3 trajectory generators have been modified accordingly. More specifically Gen1 has changed to:

**Figure 7.** The Unimate PUMA 560 Manipulator in the 'ready' position

$$\begin{aligned} q_1 &= 0.8\sin(t) \\ q_2 &= -1.5708 + 0.5\sin(t) \\ q_3 &= 1.5708 \end{aligned} \tag{27}$$

and Gen3 to:

$$\begin{aligned} q_1 &= 0.5\sin(t) \\ q_2 &= -1.65 + 0.9\sin(0.2t) + 0.2\cos(3t) \\ q_3 &= 1.5708 + 0.3\sin(t) \end{aligned} \tag{28}$$

calculating the desired joint velocities and accelerations for each link respectively. The joint angles, velocities and acceleration of the rest links are hardwired to zero. The Simulink designs are shown in Appendix I.D as part of the Toolbox.

# Chapter 5

# 5. RR model simulations

Results from the first stage of the simulation process, as described in Chapter 2, will be presented and discussed herein. The simulations in this stage involve experimentation with the 2 link arm RR model in combination with the PD controller itself or along with the neural network. The simplicity of the model and the reduced computational needs, provide the opportunity for a lot of testing with many different configurations.

Simulations have been done using all three trajectory generators which are available in the *Lab560Toolbox*. For convenience though, focus has been given to the main trajectory generator – Gen1, for which a full set of configurations has been tested. This strategy allows one to demonstrate how the change of each parameter affects the controllers performance. Indeed, the results have proved to be very crucial, since the variety of simulations done with the controller parameters, reveal a distinct picture of its properties and capabilities. The other generators have been used for extra testing and mainly in order to validate the results and thus, the most representative configurations for each one have been presented. Analysis follows simulations at the end of this chapter before moving to the second stage of simulations.

## 5.1. Conventions

Some conventions regarding the presentation of the results will be highlighted in this section. The parameters of each simulation run are presented in a table and referenced as *Configuration set Sim5.x.x* where *x.x* is the section or subsection of the corresponding simulation. Plots have been used to evaluate controllers performance. Besides the controllers response with respect to the input reference signal, filtered tracking error, joint torques and some representative weights have also been plotted. The results presented throughout this chapter involve the configuration table and the above plots for each simulation.

## 5.2. Simulations with Gen1

Simulations in this section have been done using the main trajectory generator referenced as RR-Gen1. A configuration set of the controller parameters is given in Table 2, as being the most representative of the simulations done. Next follow the relative plots in Figures 8-11.

**Table 2.** Configuration set Sim5.2

| | |
|---|---|
| PD | $K_v = 50$ |
| | lamda = 2 |
| Robust | $K_z = 5$ |
| | $Z_M = 5$ |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |

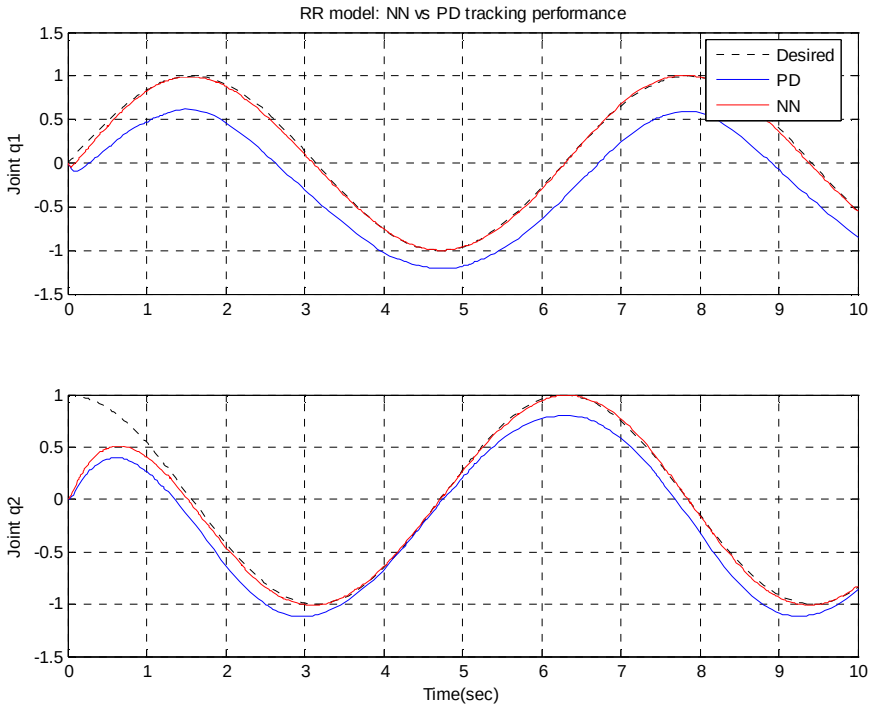



**Figure 8.** Tracking performance – Configuration set Sim5.2

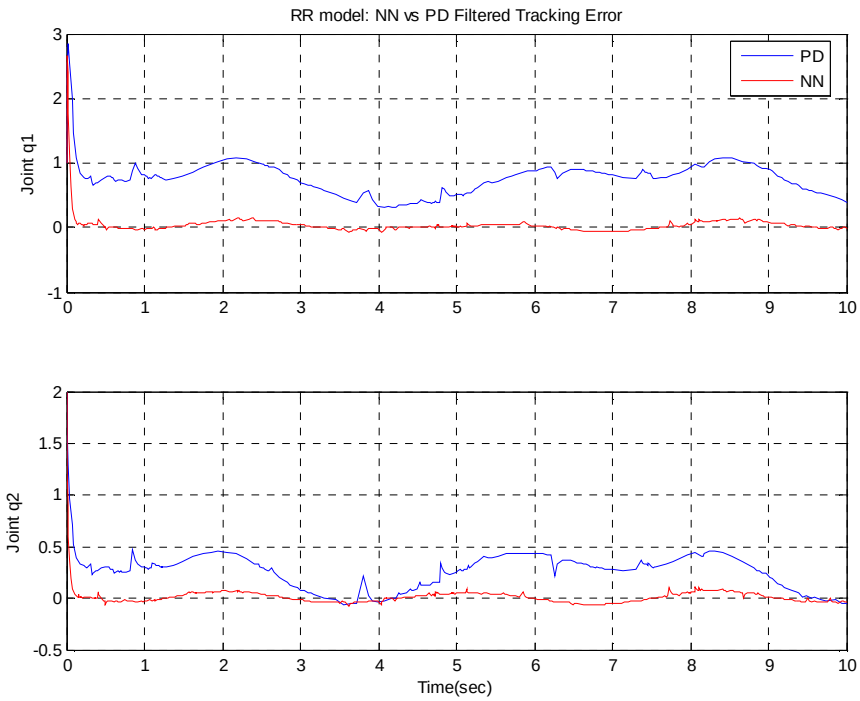



**Figure 9.** Filtered Tracking Error – Configuration set Sim5.2

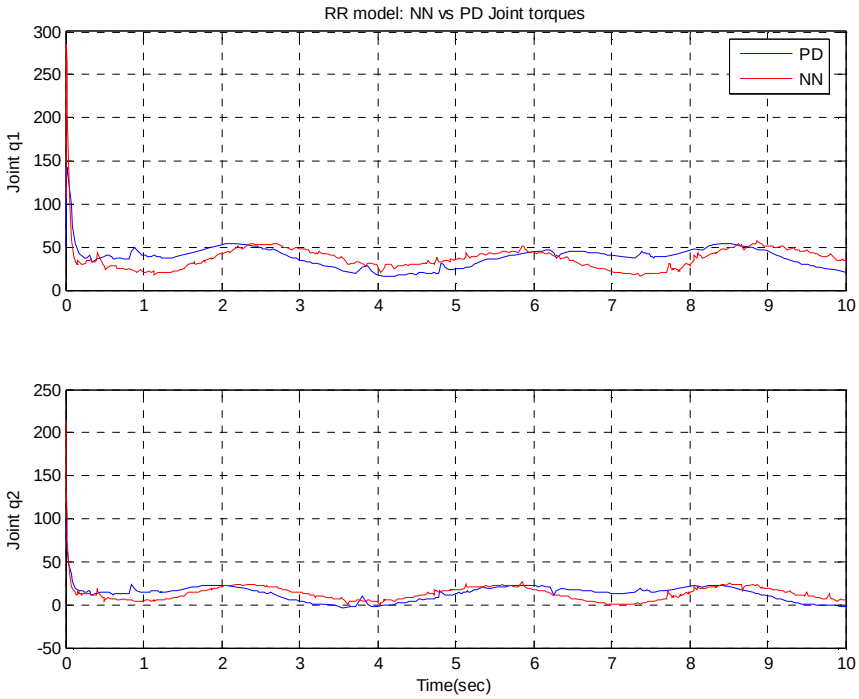



**Figure 10.** Joint torques – Configuration set Sim5.2

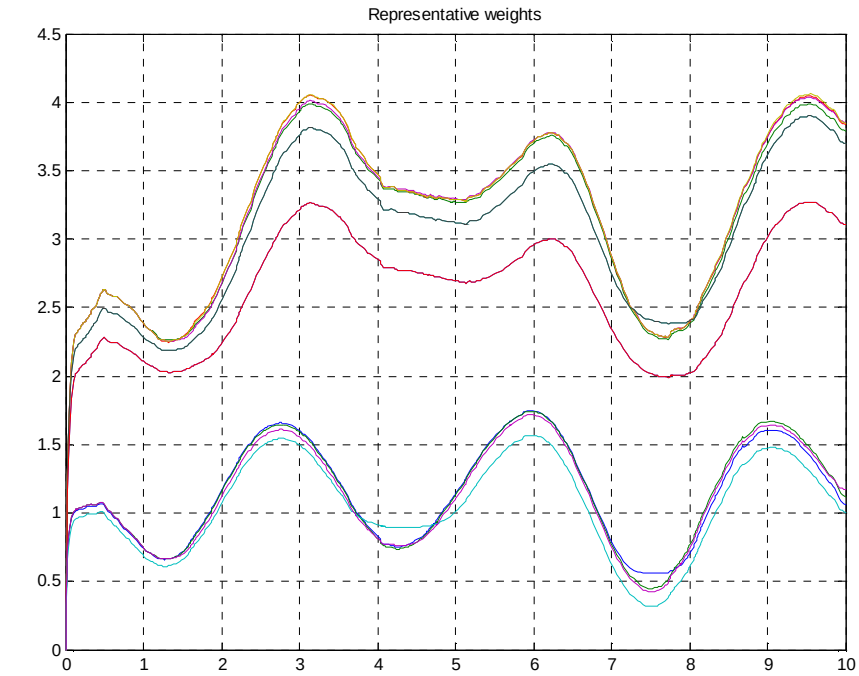



**Figure 11.** Representative weights – Configuration set Sim5.2

## 5.2.1. Changed PD parameters

Two simulations have been done in this section to demonstrate the effect of changing the parameters of PD controller. The gain $K_v$ is reduced to 30 for both simulations and *lamda* has been increased in the second. The configuration sets are summarized in Table 3 and the comparative results follow at Figures 12-14.

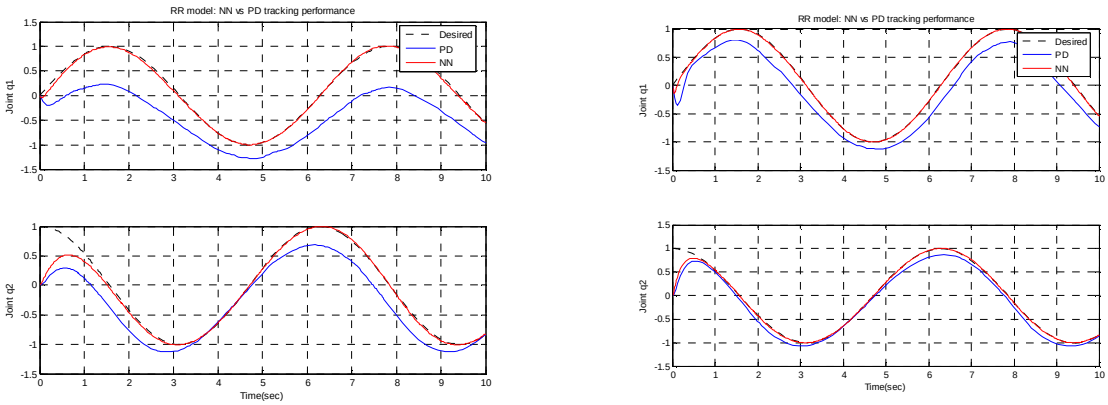
**Table 3.** Configuration sets Sim5.2.1.(a) and Sim5.2.1.(b)

(a)

| | |
|---|---|
| PD | $K_v$ = 30 |
| | ***lamda = 2*** |
| Robust | $K_z$ = 5 |
| | $Z_M$ = 5 |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |

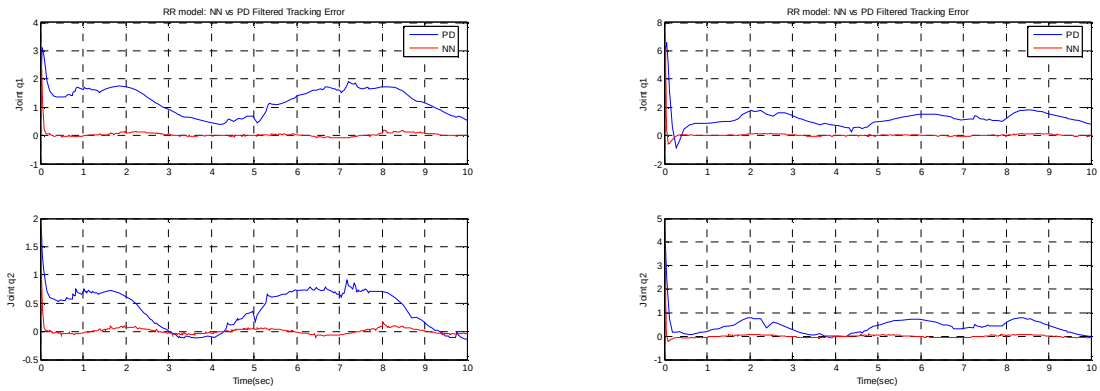
(b)

| | |
|---|---|
| PD | $K_v$ = 30 |
| | ***lamda = 5*** |
| Robust | $K_z$ = 5 |
| | $Z_M$ = 5 |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |



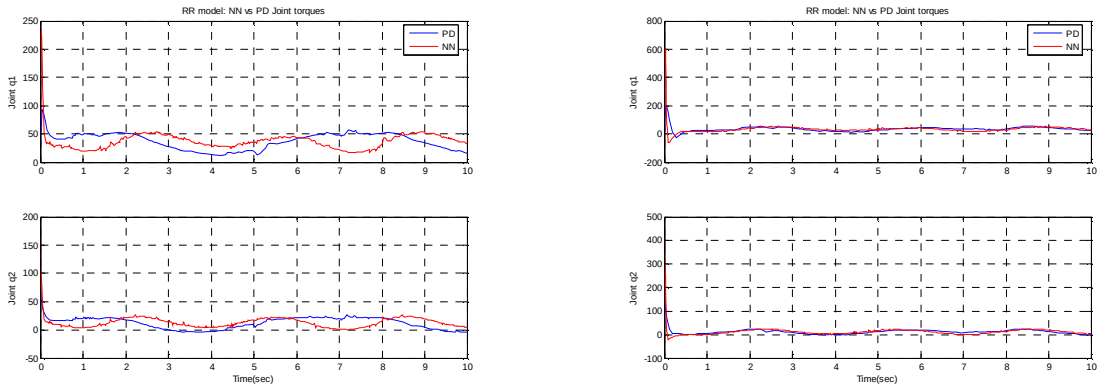

(a) (b)

**Figure 12.** Tracking Performance for (a)-Sim5.2.1.a and (b)-Sim5.2.1.b




(a) (b)

**Figure 13.** Filtered Tracking Error for (a)-Sim5.2.1.a and (b)-Sim5.2.1.b




(a) (b)

**Figure 14.** Joint Torques for (a)-Sim5.2.1.a and (b)-Sim5.2.1.b

## 5.2.2. Changed Robust parameters

The PD gains of the last configuration set – Sim5.2.1.b have been taken as a base reference for the simulations at this subsection. Larger values have been tested in order to investigate in what way the robust signal can improve tracking performance. Configuration sets follow in Table 4 highlighting the changes.

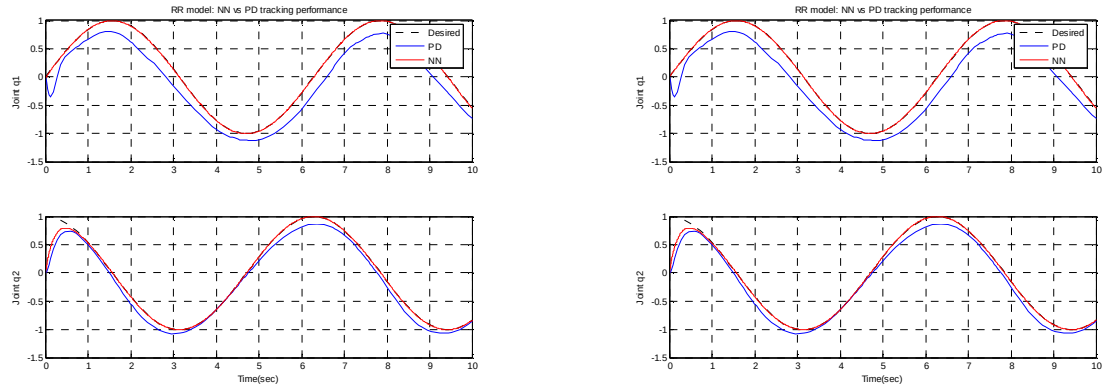
**Table 4.** Configuration sets Sim5.2.2.(a) and Sim5.2.2.(b)

(a)

| | |
|---|---|
| PD | $K_v$ = 30 |
| | lamda = 5 |
| Robust | ***$K_z$ = 50*** |
| | ***$Z_M$ = 15*** |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |

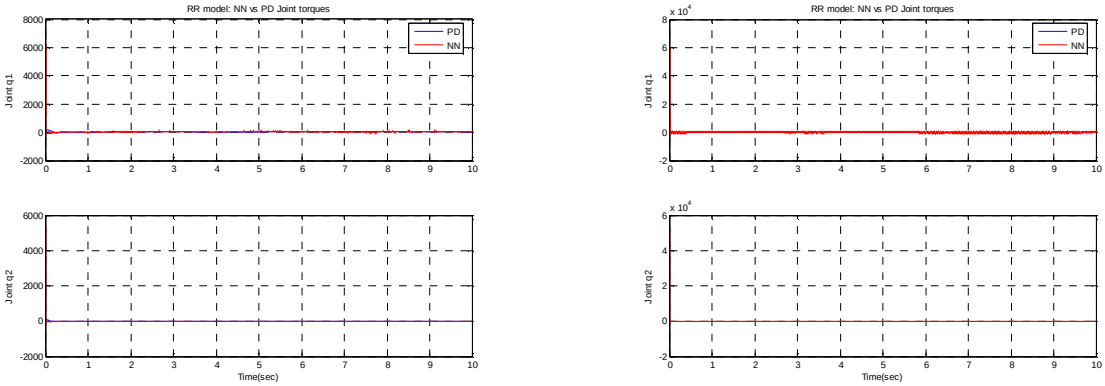
(b)

| | |
|---|---|
| PD | $K_v$ = 30 |
| | lamda = 5 |
| Robust | ***$K_z$ = 100*** |
| | ***$Z_M$ = 100*** |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |



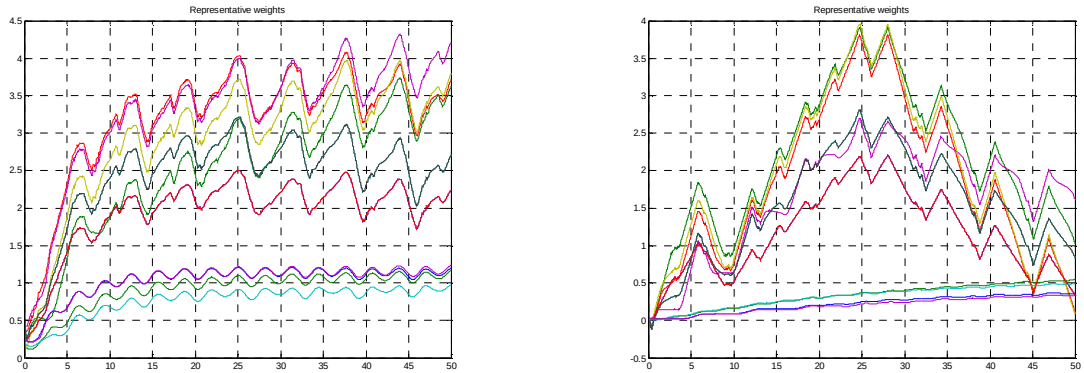

(a) (b)

**Figure 15.** Tracking Performance for (a)-Sim5.2.2.a and (b)-Sim5.2.2.b




**(a)** **(b)**

**Figure 16.** Joint torques for (a)-Sim5.2.2.a and (b)-Sim5.2.2.b




**(a)** **(b)**

**Figure 17.** Representative weights for (a)-Sim5.2.2.1 and (b)-Sim5.2.2 run for 50 sec

### 5.2.3. Changed NN parameters

In the simulations run at this subsection, the parameters of the NN have been changed with respect to *Sim5.2* configuration set. The learning rates *F*, *G* have been increased in the first two runs and *k* in the third.

**Table 5.** Configuration set Sim5.2.3.1

| | |
|---|---|
| PD | $K_v$ = 50 |
| | lamda = 2 |
| Robust | $K_z$ = 5 |
| | $Z_M$ = 5 |
| NN | F = 40 |
| | G = 50 |
| | ***k = 0.1*** |

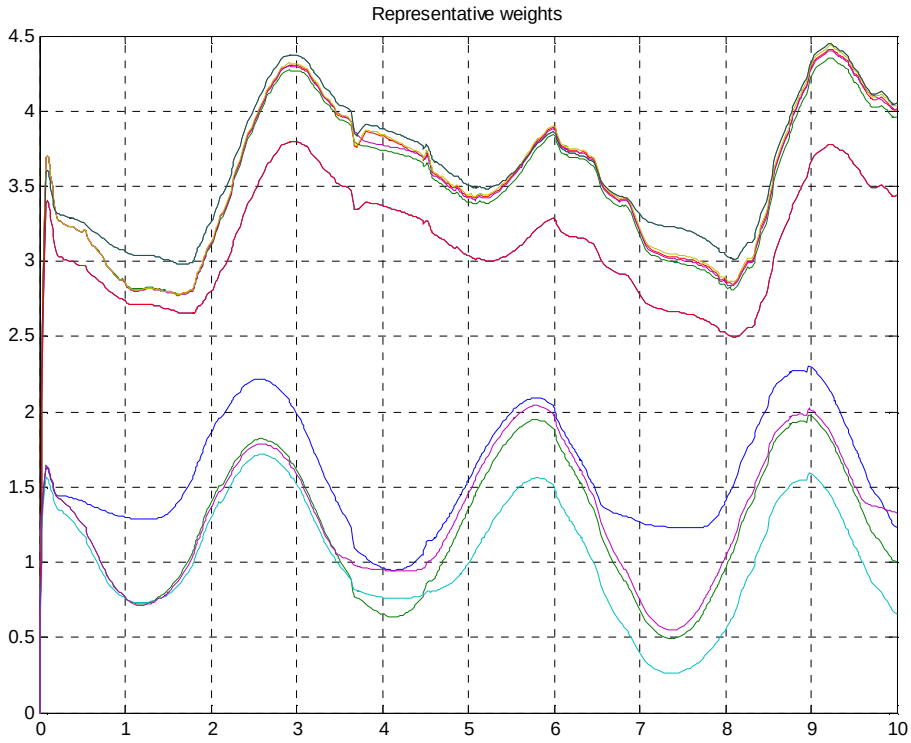



**Figure 18..** Representative weights – Configuration set Sim5.2.3.1

**Table 6.** Configuration set Sim5.2.3.2

| | |
|---|---|
| PD | $K_v = 50$ |
| | lamda = 2 |
| Robust | $K_z = 5$ |
| | $Z_M = 5$ |
| NN | F = 480 |
| | G = 500 |
| | k = 0.1 |

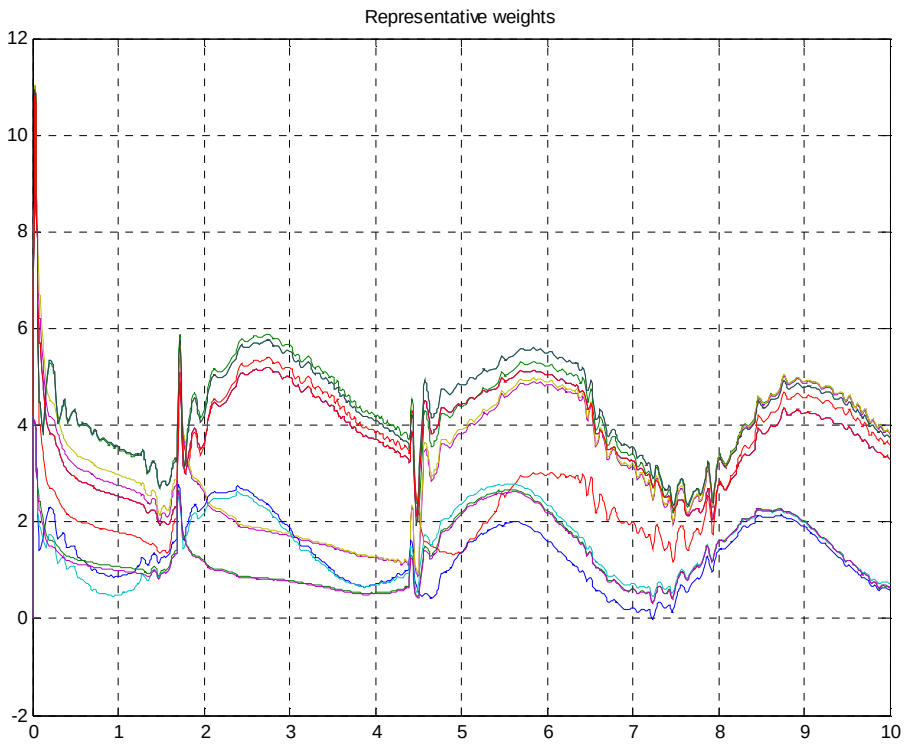



**Figure 19.** Representative weights – Configuration set Sim5.2.3.2

**Table 7.** Configuration set Sim5.2.3.3

| | |
|---|---|
| PD | $K_v = 50$ |
| | lamda = 2 |
| Robust | $K_z = 5$ |
| | $Z_M = 5$ |
| NN | F = 40 |
| | G = 50 |
| | ***k = 0.0001*** |

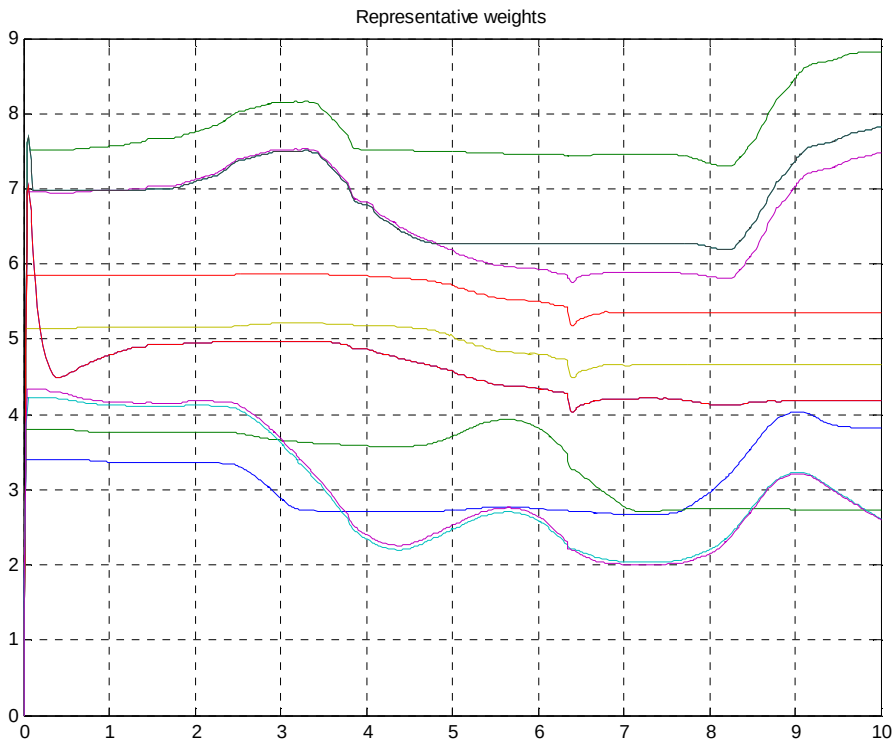



**Figure 20.** Representative weights – Configuration set Sim5.2.3.3

## 5.3. Simulations with Gen2

For the simulations done with the Robotics Toolbox default trajectory generator – RR-Gen2, a single representative run will be presented. The initial and final positions of robots interpolated trajectory were:

[0 0]➔[pi/4 –pi/2]

The configuration set and the relative plots are given in Table 8 and Figures 21-23 respectively

**Table 8.** Configuration set Sim5.3

| | |
|---|---|
| PD | $K_v$ = 50 |
| | lamda = 2 |
| Robust | $K_z$ = 5 |
| | $Z_M$ = 5 |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |

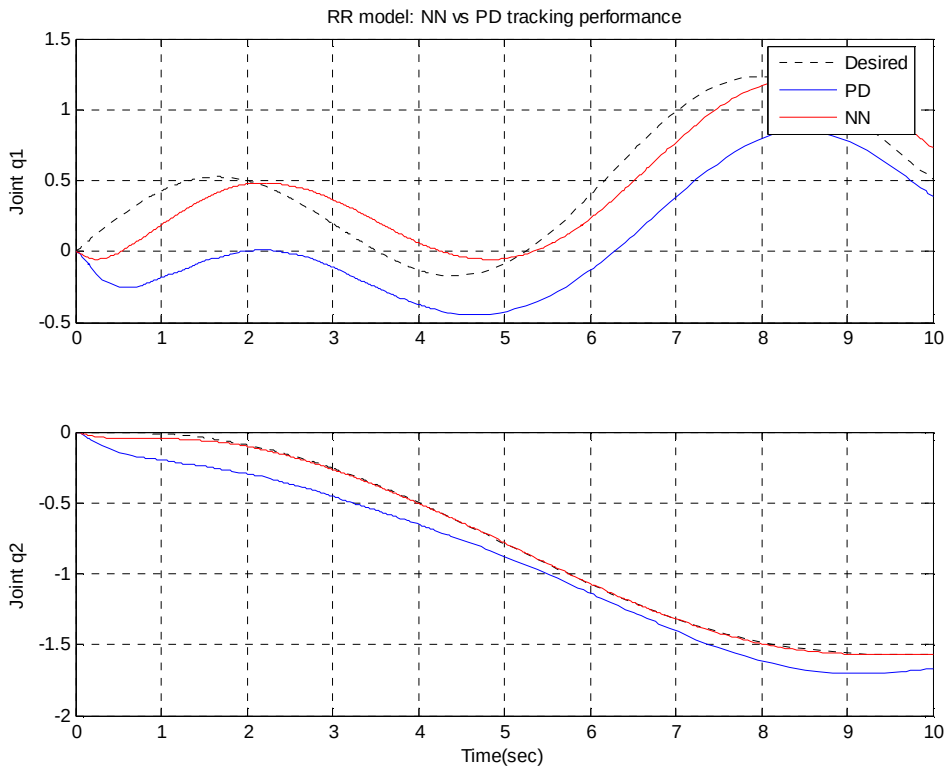



**Figure 21.** Tracking performance – Configuration set Sim5.3

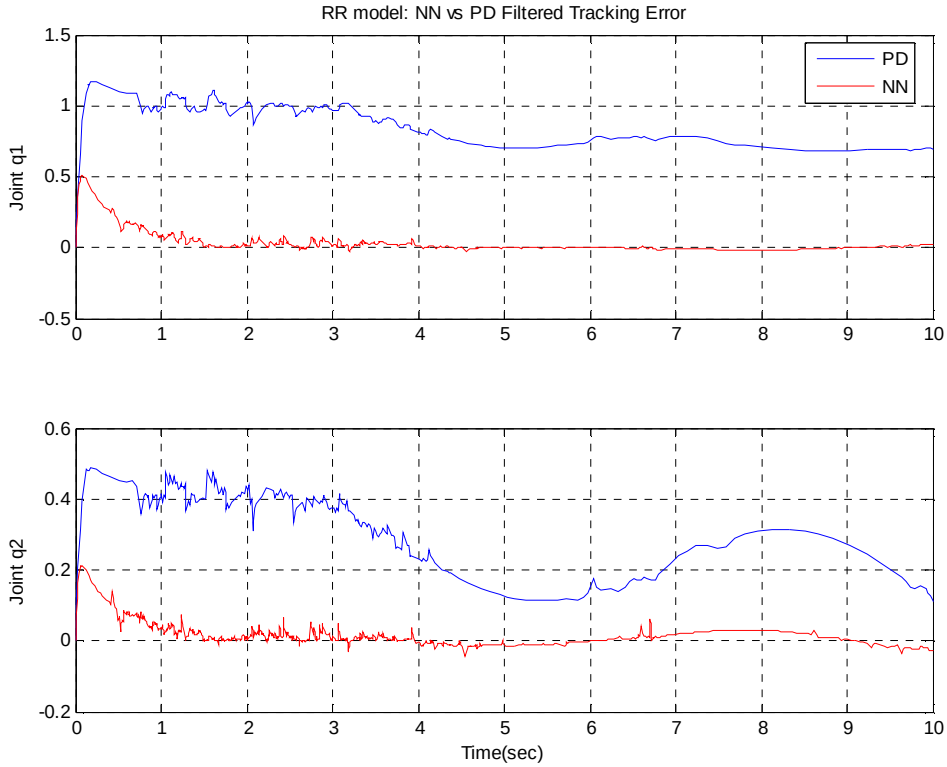
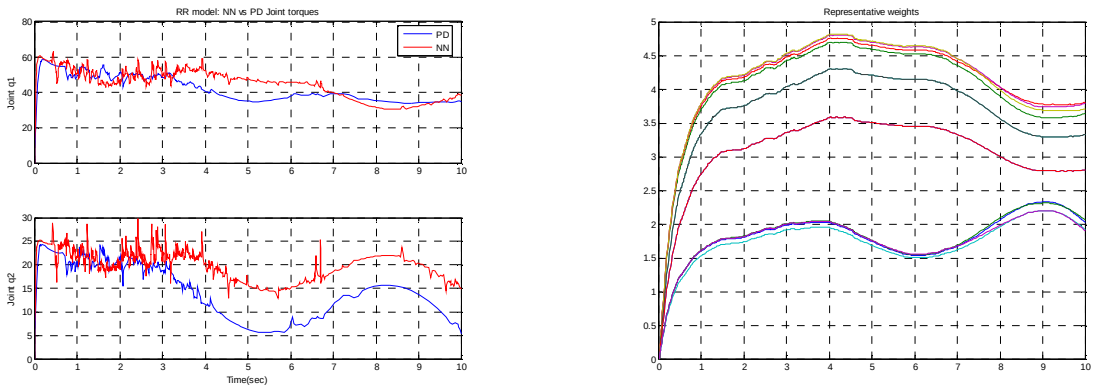



**Figure 22.** Filtered Tracking Error – Configuration set Sim5.3




(a) (b)

**Figure 23.** Joint torques (a) and representative weights (b) – Configuration set Sim5.3

## 5.4. Simulations with Gen3

Finally, Gen3 represents a more complex trajectory with superimposed sine and cosine waves as shown in Appendix I.D. The same configuration set with Gen1 has been tested in the simulations run here.

**Table 9.** Configuration set Sim5.4

| | |
|---|---|
| PD | $K_v = 50$ |
| | lamda = 2 |
| Robust | $K_z = 5$ |
| | $Z_M = 5$ |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |

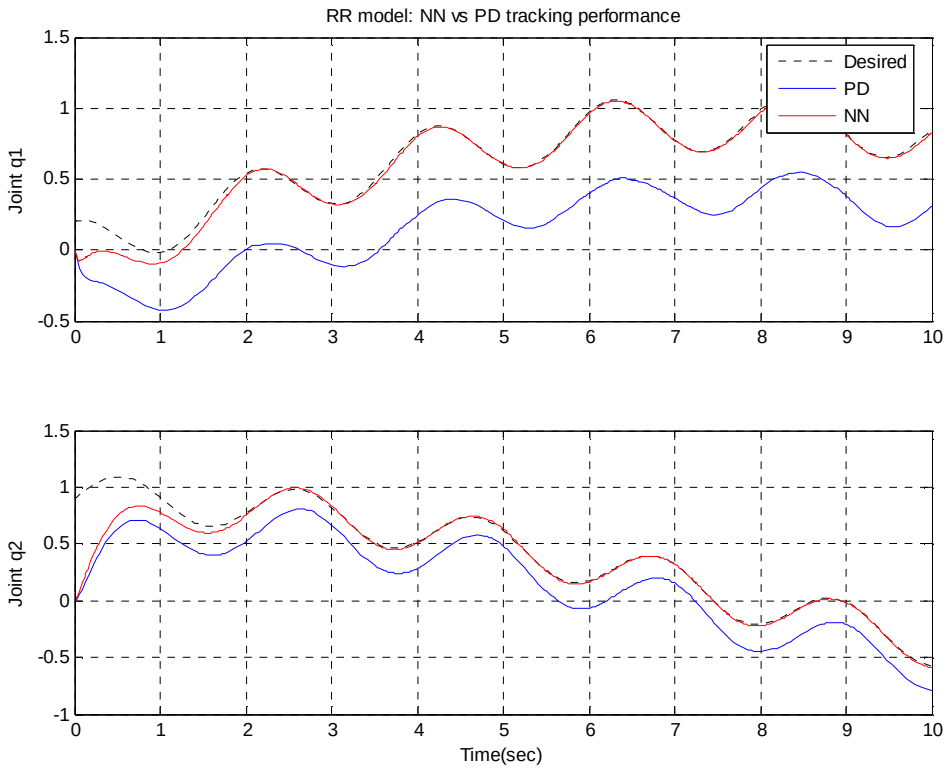



**Figure 24.** Tracking performance – Configuration set Sim5.4

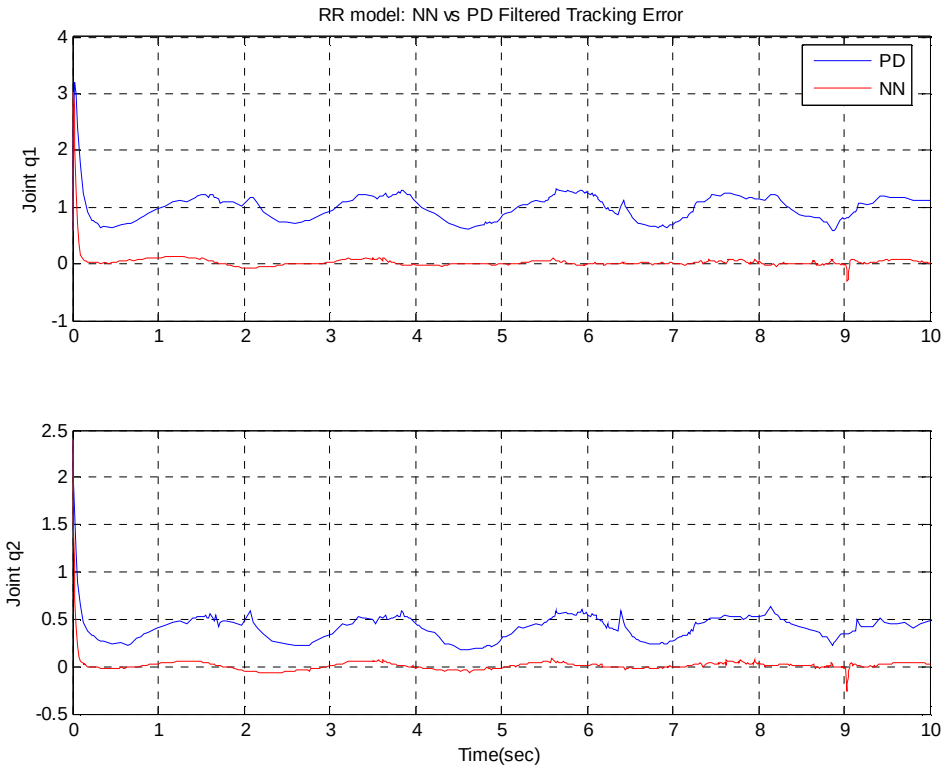
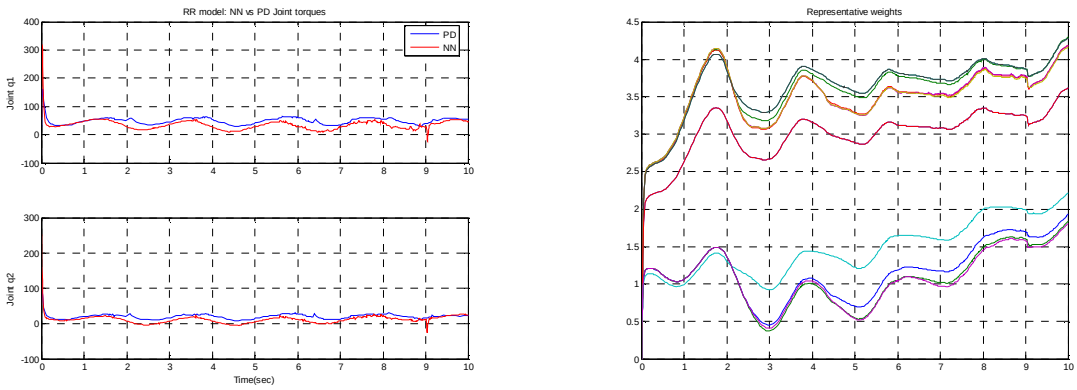



**Figure 25.** Filtered Tracking Error – Configuration set Sim5.4




(a) (b)

**Figure 26.** Joint torques (a) and representative weights (b) – Configuration set Sim5.4

## 5.5. Analysis

Before proceeding to Puma simulations, a discussion of the results presented for the two link model, will precede. Parameters sensitivity will be discussed with respect to specific simulation examples and conclusions will be drawn regarding their effect on controllers performance. The tracking efficiency of the NN control scheme will be compared to the PD controller, highlighting the contribution of NN. Finally, a summary of some distinct features will be given.

### 5.5.1. Sensitivity Analysis

The PD parameters of NN control structure have little effect on the overall tracking performance. Configuration sets *Sim5.2.1.a* and *Sim5.2.1.b* results show that despite changing the values of gain *Kv* and *lamda*, the NN manages to perform equally well with respect to *Sim5.2.1*. Besides the simulations presented here, many combinations have been tested, all indicating great robustness to these parameters. As discussed though in Chapter 3, the contribution of these parameters in systems overall performance is that they ensure initial closed loop stability.

The tracking performance of the representative configuration set is already very good. Increasing the gain *Kz* though, as shown in *Sim5.2.2.a* and *Sim5.2.2.b*, it is possible to achieve much better or almost perfect tracking. This is demonstrated in Figure 15. A side effect of this, is that joint torques are initially very big as shown in Figure 16, which might not be desirable in real time experiments. Moreover, the weights – although stable and bounded – seem to converge much later compared to *Sim5.2* set, without affecting the tracking performance. To validate that *Sim5.2.2.i* were run for 50 sec and the results confirm the initial guess – Figure 17.

The Neural Network parameters *F*, *G* and *k* affect the rate of convergence for the weights. The relative simulations show faster response initially followed by the learning phase which includes deviation within a certain region later, resulting in efficient tracking. Figures 18-20 show representative weights of *Sim5.2.3.1-3* respectively.

The simulations performed with trajectory generators RR-Gen2 and RR-Gen3 have shown the general capabilities of the Neural network in different reference input signals. The tracking performance in the case of Gen2 is not as good as expected. This though, is valid only for the first link where the reference signal is a sine wave superimposed to the interpolated trajectory. It might be the case that this combination does not yield a suitable reference signal. On the contrary, the neural network manages to track fairly easily the interpolated trajectory for the second link. Finally, the same configuration set *Sim5.2* was used for the simulations with Gen3. Despite the fact the reference signal in this case is more complex, the controllers performance under the same parameters remains outstanding.

### 5.5.2. Comparative Analysis

The results from the simulations run for the two link model, show that the tracking performance of the neural network control structure outperforms that of PD controller. This is apparent not only in figures comparing the tracking ability of each controller, but also in figures showing the filtered tracking error. The PD controller alone, produces a large tracking error. In fact, its magnitude is comparable to that of the reference input signal. Even when increasing the PD gains, as in *Sim5.2* or in further

simulations that have been done, it certainly performs better but cannot eliminate the steady state error. On the contrary, the NN manages to reduce the filtered tracking error relatively fast, eliminating it gradually as the learning phase goes on. The simulations done with Gen2 and Gen3 demonstrate the superior tracking efficiency of the NN compared to the PD controller.

### 5.5.3. Summary

The results presented in this chapter share some common characteristics independent of the configuration set used for the simulations. These are the features of the controller described in the theoretic framework in Chapter3 but are worth reviewing after having presented and discussed the results. Simulations for the two link model have demonstrated:

- Closed loop trajectory following
- Bounded tracking error
- Bounded weights
- Online learning feature

The conclusions drawn from this stage of simulations have provided an insight to the controller features and capabilities. Next, follows the second stage of simulations that involves experimentation with a Puma 560 model.

# Chapter 6

# 6. Puma 560 simulations

The results from simulations with the Puma 560 model will be presented in this Chapter. Simulations have mainly focused at the control of the first three links for the evaluation, testing and configuration of the controller parameters. A representative set of configurations and results will be presented herein, as a result of the simulations done in the previous stage.

## 6.1. Simulations with Gen1

The main trajectory generator Gen1, as has been modified for the needs of Puma model, has been used as the input reference signal for the simulations in this section. A representative configuration set with Puma-Gen1 is summarized in Table 10 while the plots follow at Figures 27-29.

**Table 10** Configuration set Sim6.1

| | |
|---|---|
| PD | $K_v = 50$ |
| | lamda = 2 |
| Robust | $K_z = 5$ |
| | $Z_M = 5$ |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |

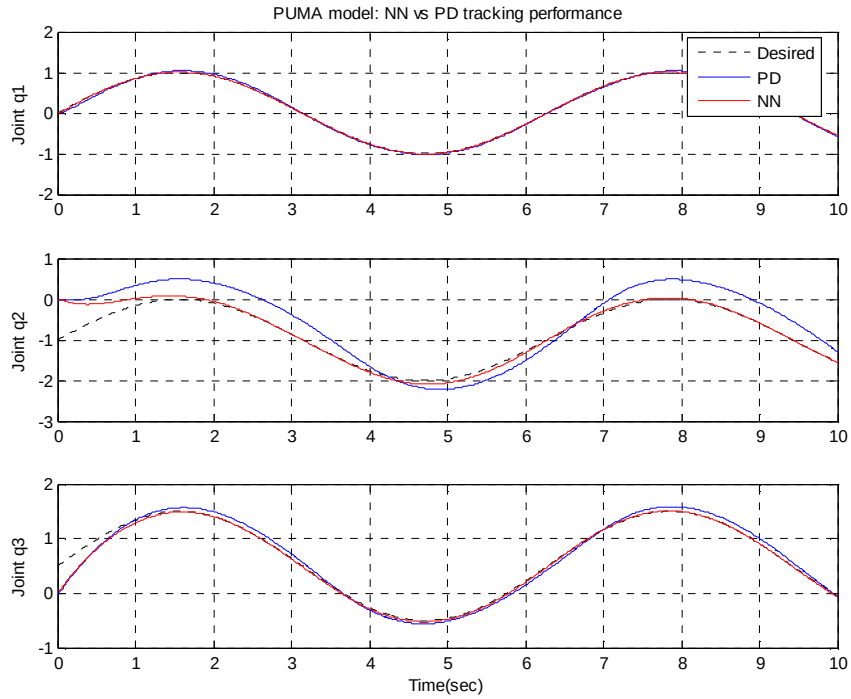



**Figure 27.** Tracking performance – Configuration set Sim6.1

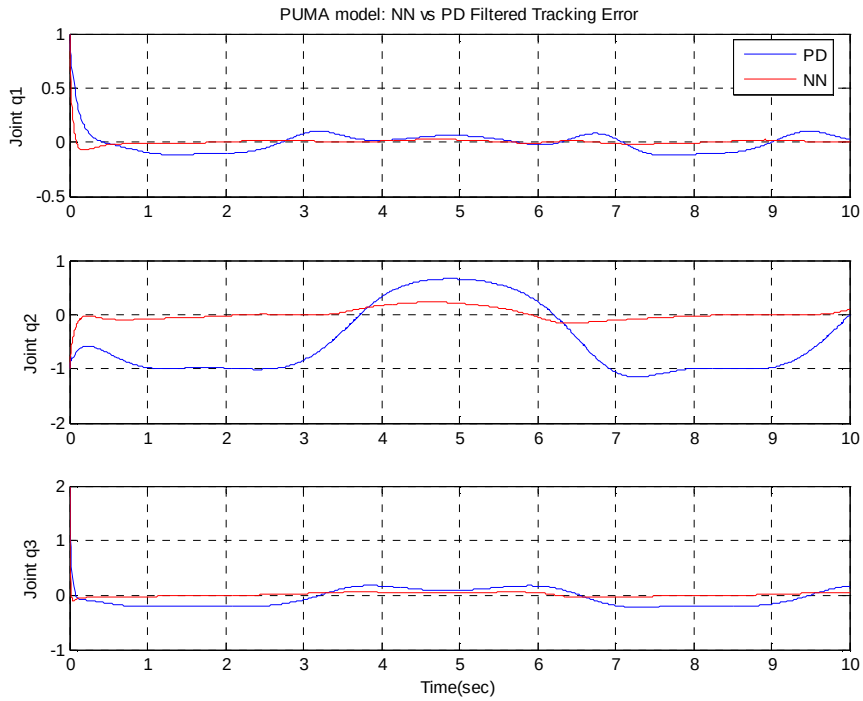
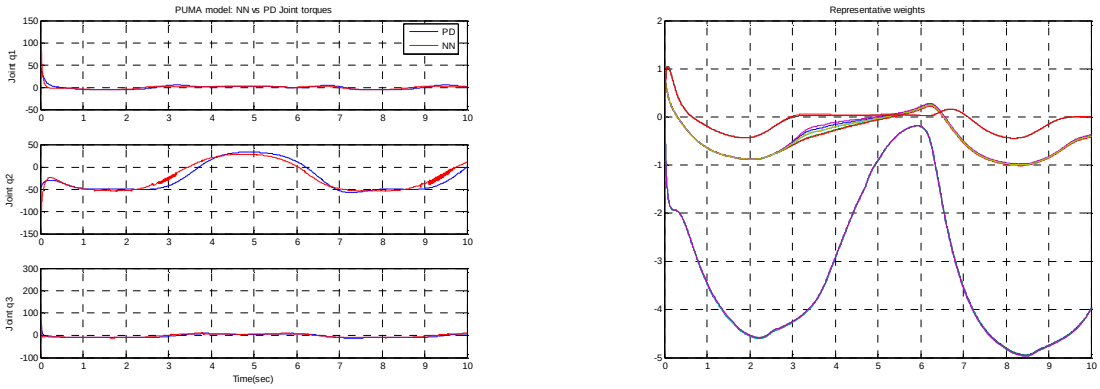



**Figure 28.** Filtered Tracking Error – Configuration set Sim6.1




(a) (b)

**Figure 29.** Joint torques (a) and representative weights (b) - Configuration Set Sim6.1

## 6.2. Simulations with Gen2

The default trajectory generator of the Robotics Toolbox – Puma-Gen2 has also been used for the simulations done in this section. The initial and final positions of robots interpolated trajectory were:

[0 0 0 0 0 0]➔[pi/4 –pi/2 pi/2 0 0 0]

The configuration set and the relative plots are given in Table 11 and Figures 30-32 respectively.

**Table 11.** Configuration set Sim6.2

| | |
|---|---|
| PD | $K_v$ = 50 |
| | lamda = 2 |
| Robust | $K_z$ = 5 |
| | $Z_M$ = 5 |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |

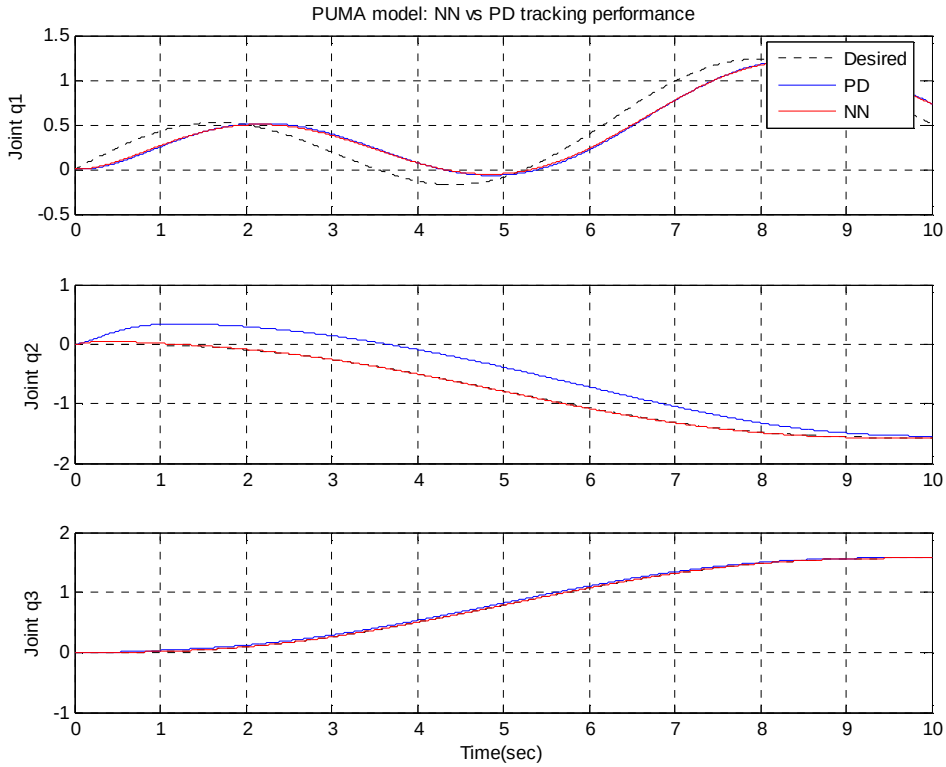



**Figure 30.** Tracking Performance – Configuration set Sim6.2

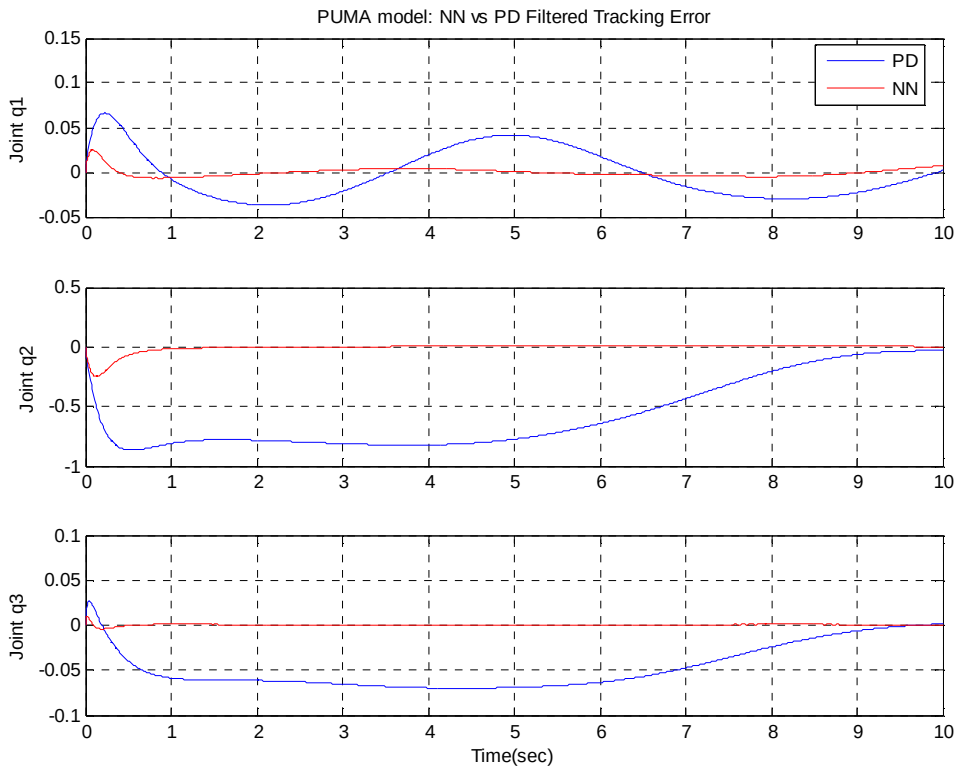
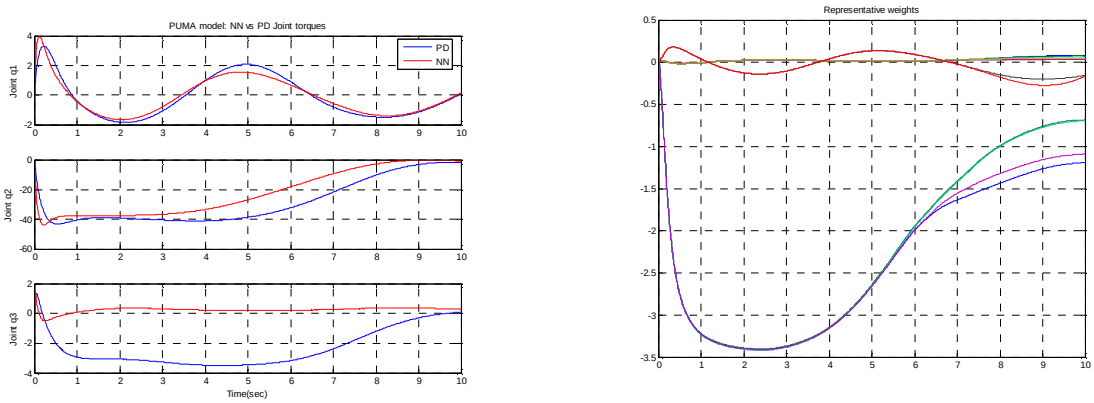



**Figure 31.** Filtered Tracking Error– Configuration set Sim6.2




(a) (b)

**Figure 32.** Joint torques (a) and representative weights (b) - Configuration Set Sim6.2

## 6.3. Simulations with Gen3

As the final part of the simulations with Puma model, Gen3 has been modified as described by equation (26), in order to evaluate controllers tracking performance to a complex reference input signal (Puma-Gen3). A representative set of configuration is given in Table 12, while the relative plots appear in Figures 33-35.

**Table 12.** Configuration Set Sim6.3

| | |
|---|---|
| PD | $K_v = 50$ |
| | lamda = 2 |
| Robust | $K_z = 5$ |
| | $Z_M = 5$ |
| NN | F = 20 |
| | G = 20 |
| | k = 0.1 |

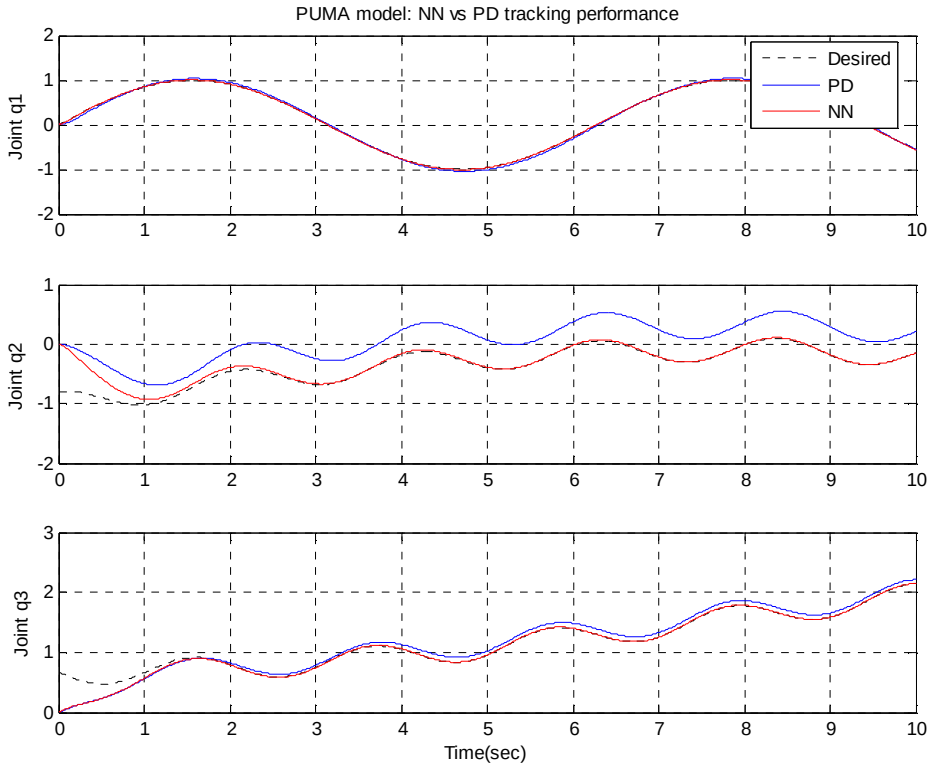



**Figure 33.** Tracking Performance – Configuration Set Sim6.3

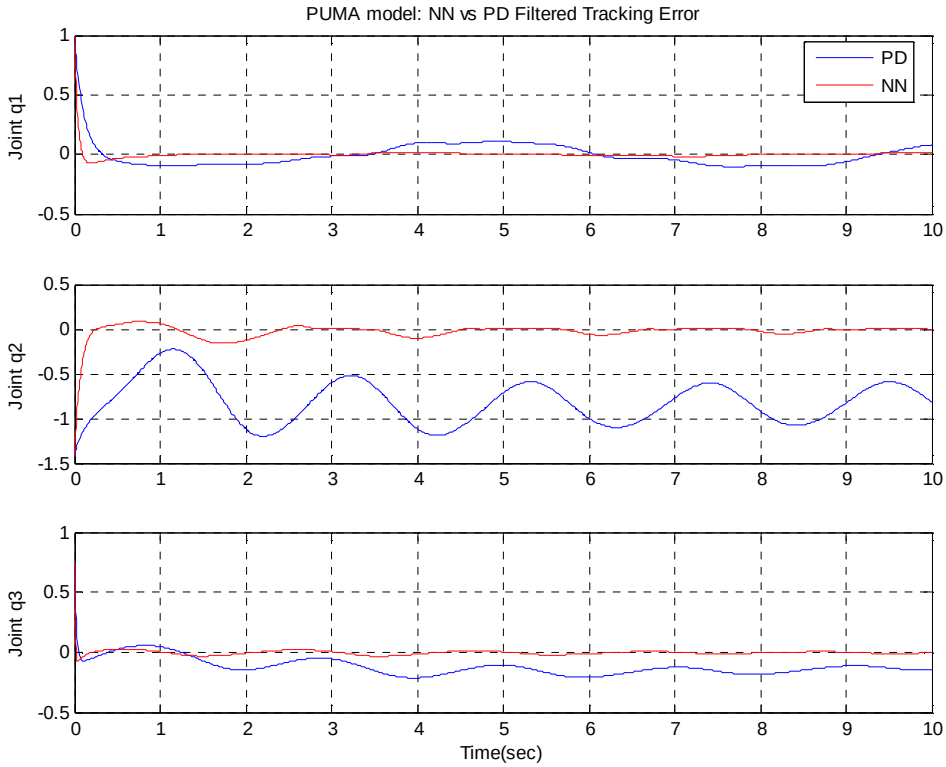
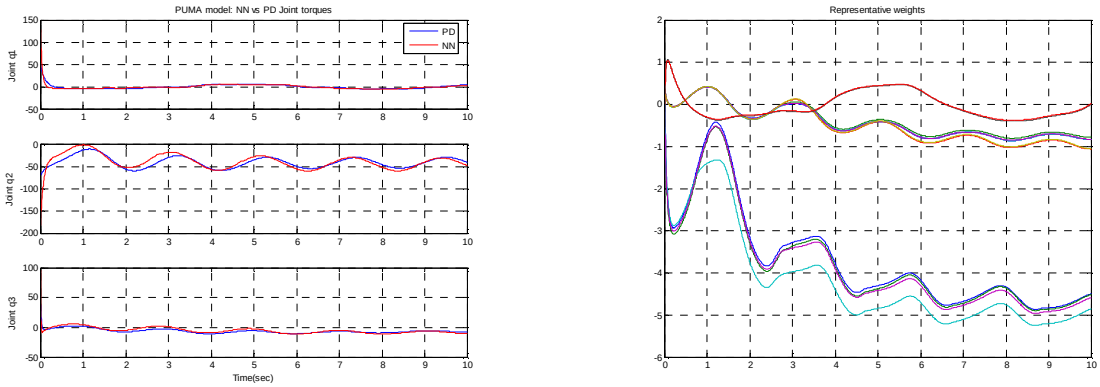



**Figure 34.** Filtered Tracking Error– Configuration Set Sim6.3




**(a)** **(b)**

**Figure 35.** Joint torques (a) and representative weights (b) - Configuration Set Sim6.3

## 6.4. Summary

The simulations have taken into account some issues that were not encountered in the two link model. Including the Limiter in the design, imposed restrictions with respect to the joint velocities and positions. Choosing a suitable trajectory generator that would satisfy these restrictions on the one hand and demonstrate the controllers tracking capabilities was crucial. The initial position of the robot model was [0 0 0 0 0 0]. Both Puma-Gen1 and Puma-Gen3 trajectory generators have been modified to provide a reference signal with an offset. Configuration of the PD parameters in this case was not as straightforward as in the two link model due to the velocity limits. More specifically, higher values for *Kv* and *lamda* may certainly achieve smaller tracking error but, will also increase the velocities of the links during the transient response in order to follow the reference trajectory. As a result, a balance between efficient tracking and smooth transient response to an offset from the initial position had to be found.

To conclude, the results of the simulations done with Puma model confirm the analysis made for the two link model regarding not only the controllers performance itself but also compared to the PD controller.

# Chapter 7

# 7. Puma 560 experiments

Following the simulations, experiments performed in Puma are presented herein. Experiments have focused on the first three links to evaluate controllers performance.

## 7.1. Experiments with Gen1

The results for the Neural Network and the PD controller are presented separately in the relative subsections. The configuration set of the parameters for Real Time-Gen1 is summarized in Table 13.

**Table 13.** Configuration set Exp7.1

| | |
|---|---|
| PD | $K_v = 10$ |
| | lamda = 2 |
| Robust | $K_z = 2$ |
| | $Z_M = 5$ |
| NN | F = 10 |
| | G = 10 |
| | k = 0.1 |

### 7.1.1. Neural Network Controller

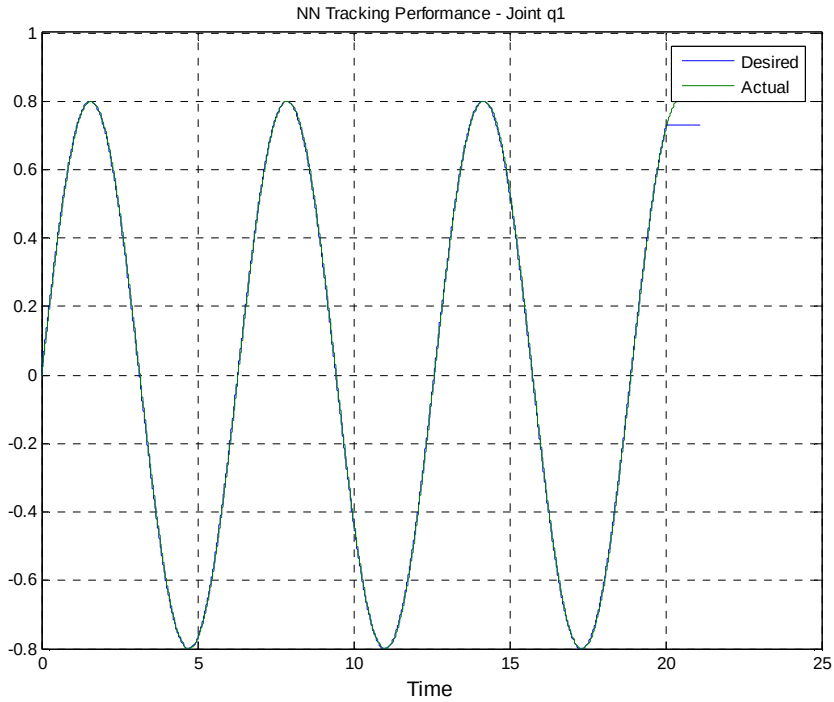



**Figure 36.** NN Tracking performance of Joint q1 for Exp7.1

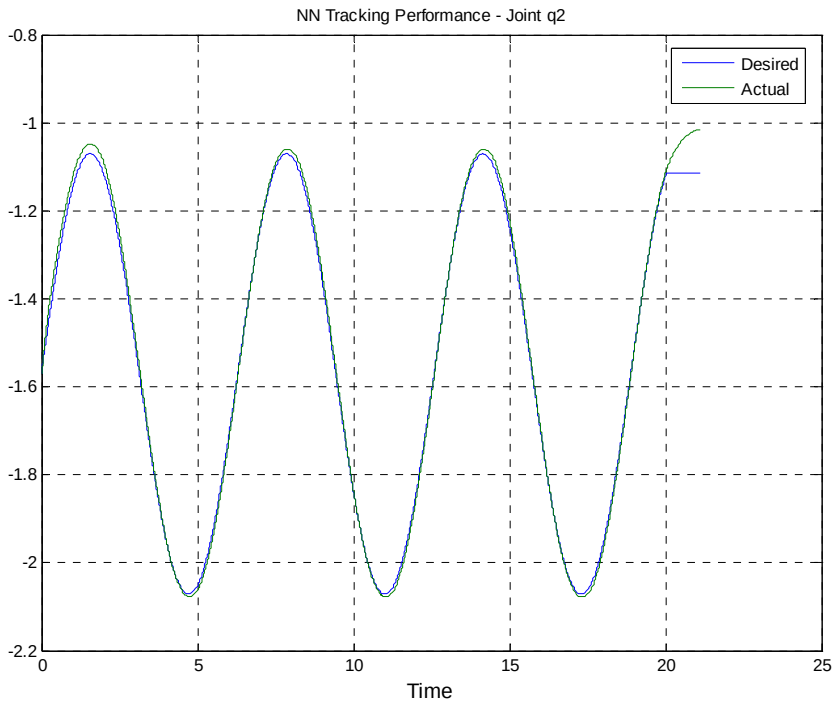



**Figure 37.** NN Tracking performance of Joint q2 for Exp7.1

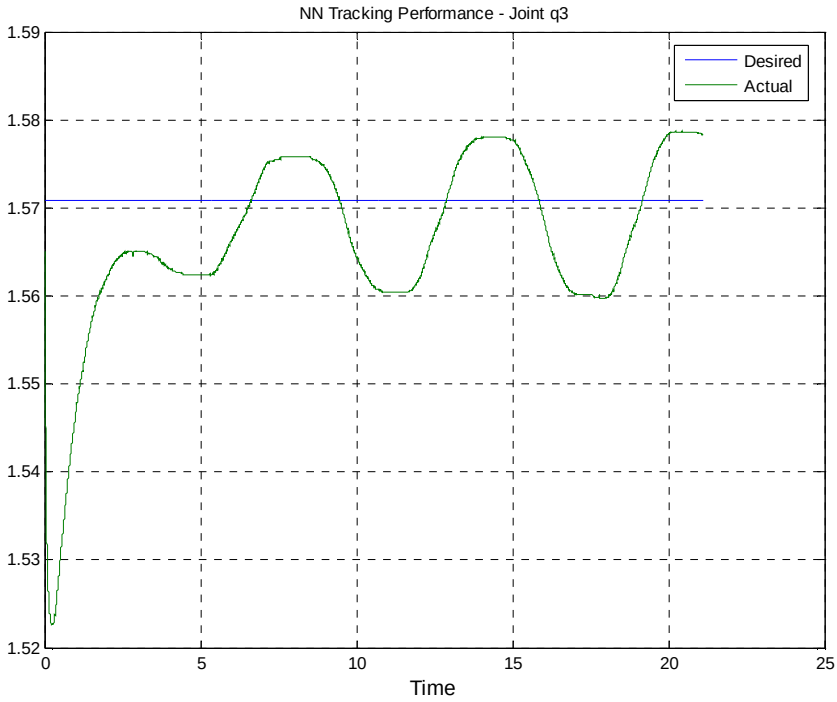



**Figure 38.** NN Tracking performance of Joint q3 for Exp7.1

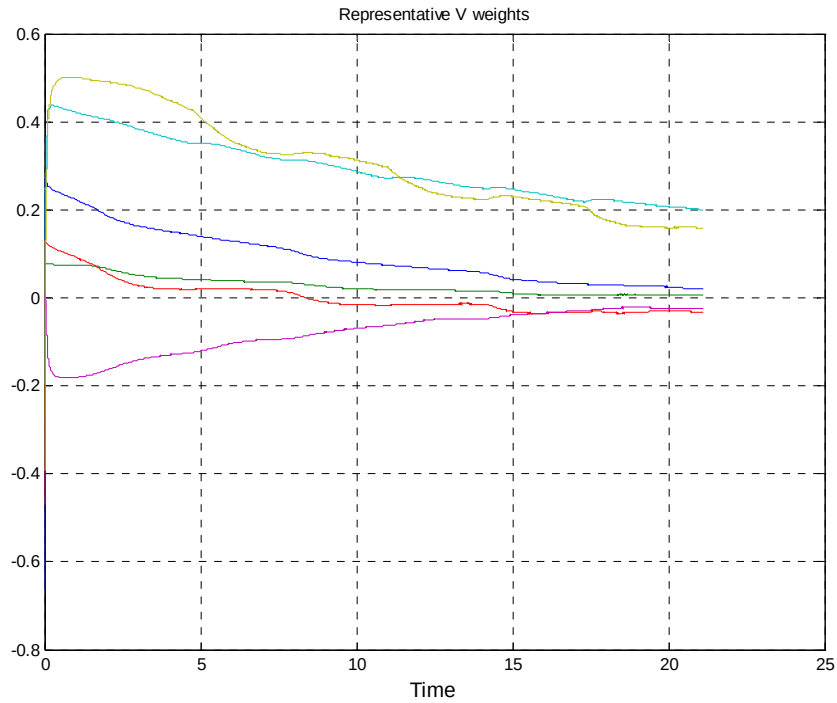



**Figure 39.** Representative V weights for Exp7.1

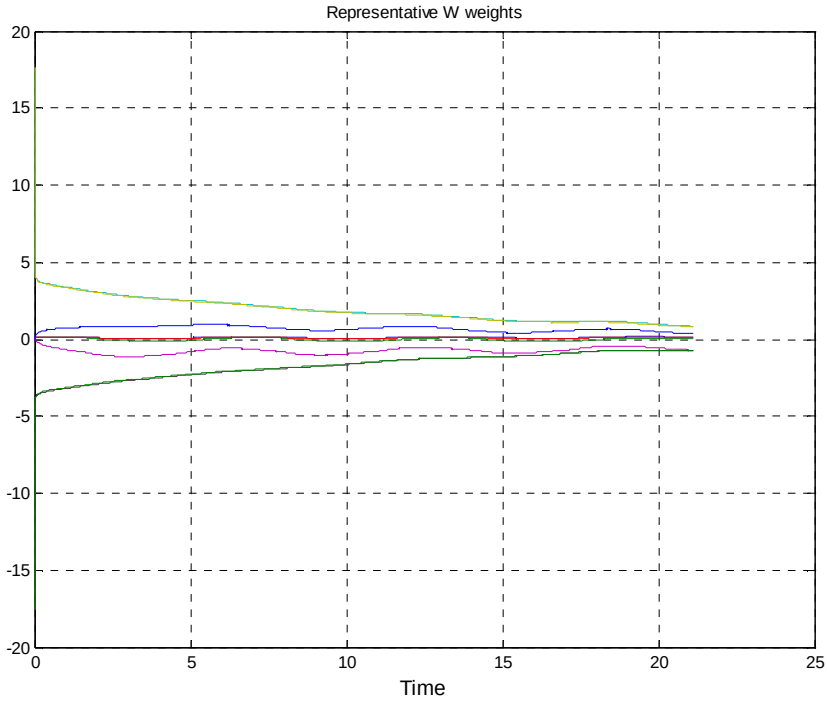
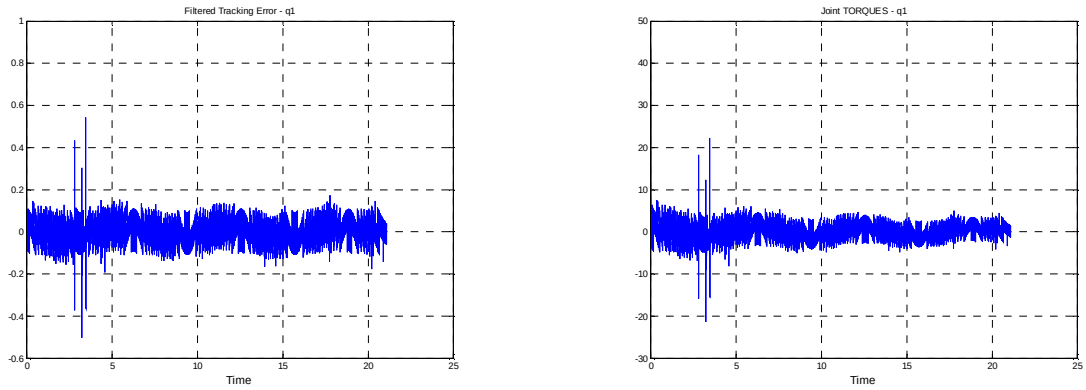
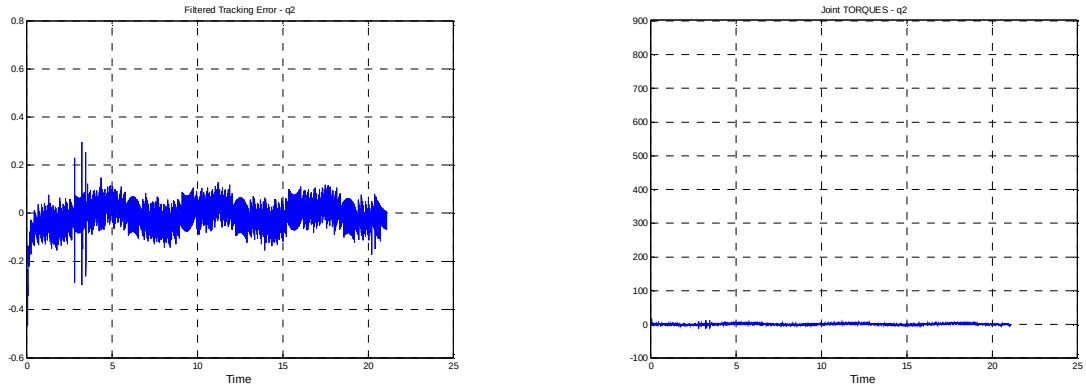



**Figure 40.** Representative W weights for Exp7.1



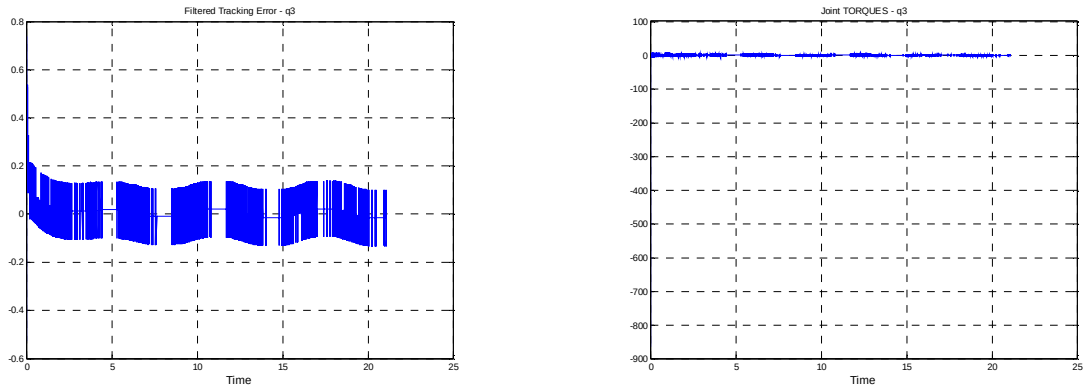

(a) (b)

**Figure 41.** NN Filtered Tracking Error (a) and Torque (b) for Joint q1 – Exp7.1




(a) (b)

**Figure 42.** NN Filtered Tracking Error (a) and Torque (b) for Joint q2 – Exp7.1




(a) (b)

**Figure 43.** NN Filtered Tracking Error (a) and Torque (b) for Joint q3 – Exp7.1

## 7.1.2. PD Controller

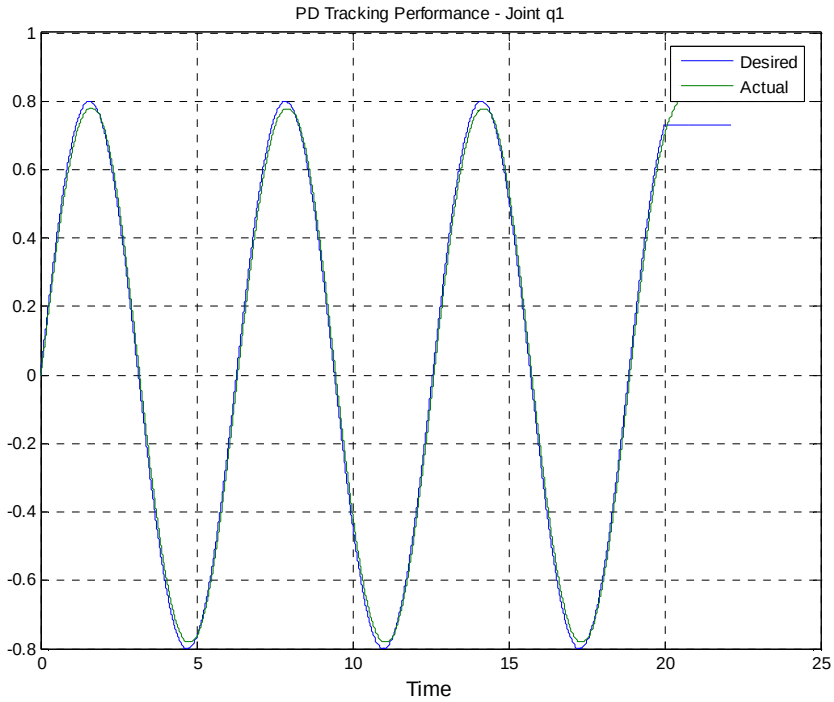



**Figure 44.** PD Tracking performance of Joint q1 for Exp7.1

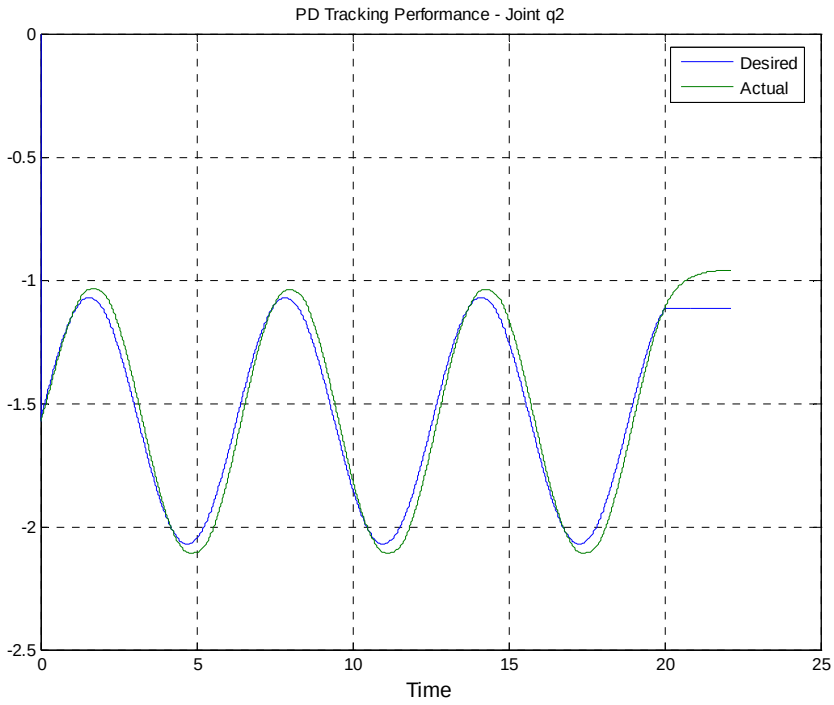



**Figure 45.** PD Tracking performance of Joint q2 for Exp7.1

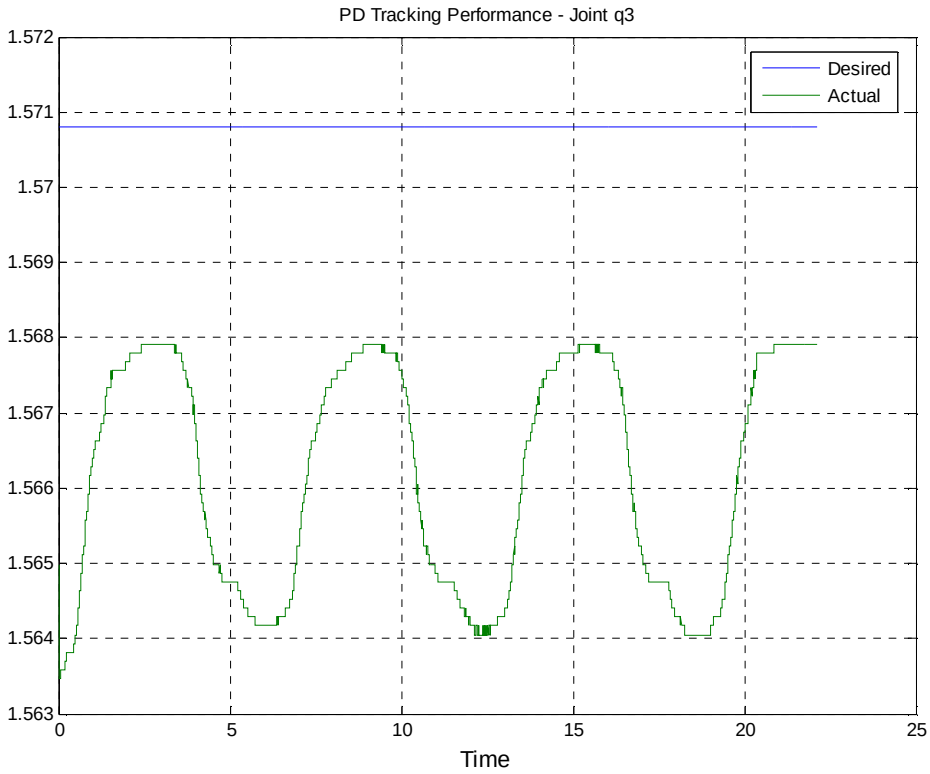
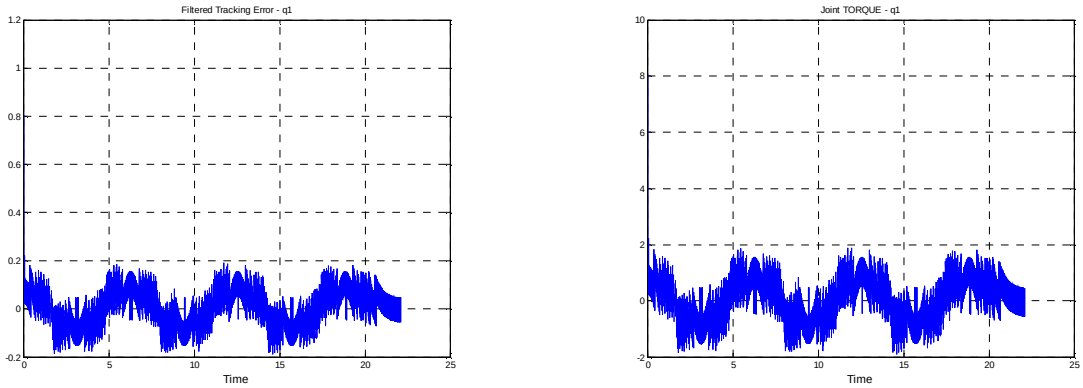
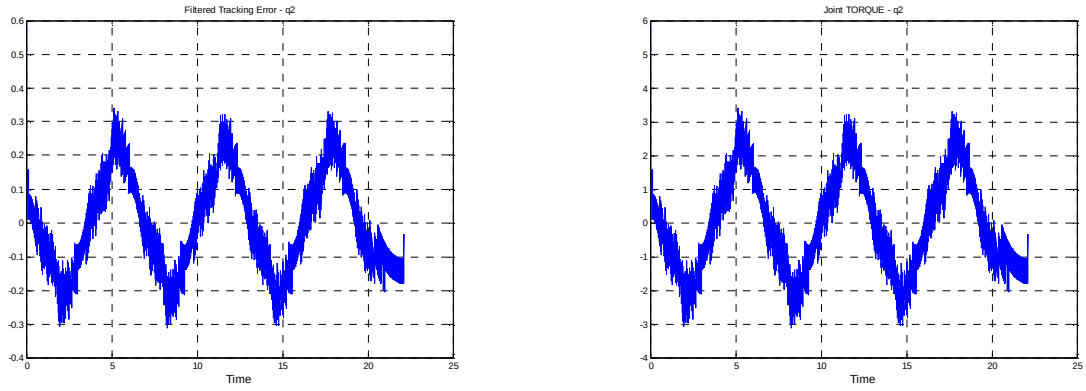



**Figure 46.** PD Tracking performance of Joint q3 for Exp7.1



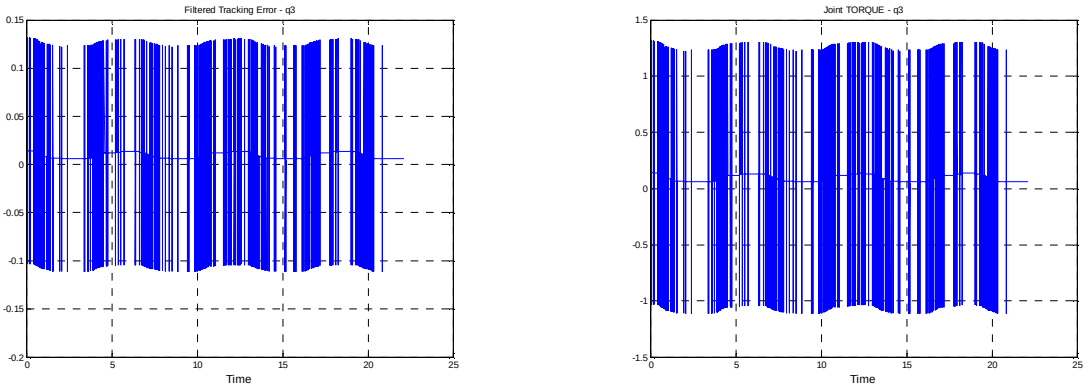

(a) (b)

**Figure 47.** PD Filtered Tracking Error (a) and Torque (b) for Joint q1 – Exp7.1




(a) (b)

**Figure 48.** PD Filtered Tracking Error (a) and Torque (b) for Joint q2 – Exp7.1




(a) (b)

**Figure 49.** PD Filtered Tracking Error (a) and Torque (b) for Joint q3 – Exp7.1

## 7.2. Experiments with Gen3

**Table 14.** Configuration set Exp7.2

| | |
|---|---|
| PD | $K_v$ = 10 |
| | lamda = 1.5 |
| Robust | $K_z$ = 1.5 |
| | $Z_M$ = 5 |
| NN | F = 10 |
| | G = 10 |
| | k = 0.1 |

### 7.2.1. Neural Network Controller

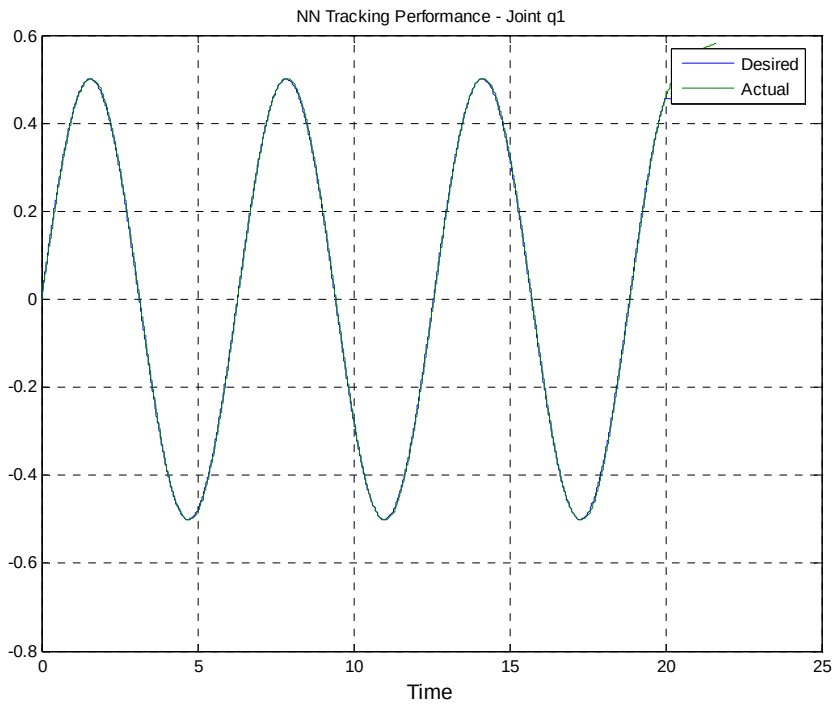



**Figure 50.** NN Tracking performance of Joint q1 for Exp7.2

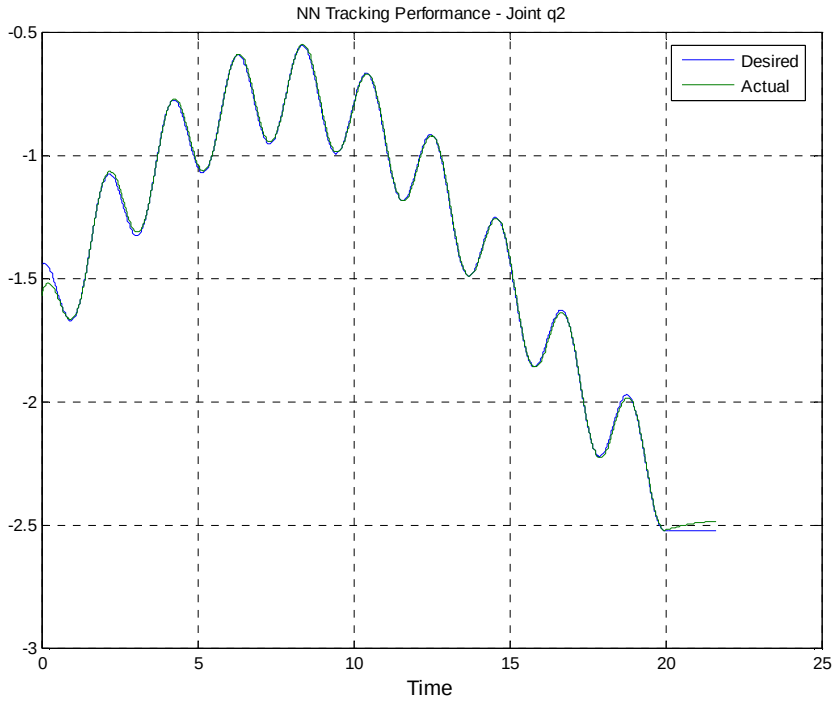



**Figure 51.** NN Tracking performance of Joint q2 for Exp7.2

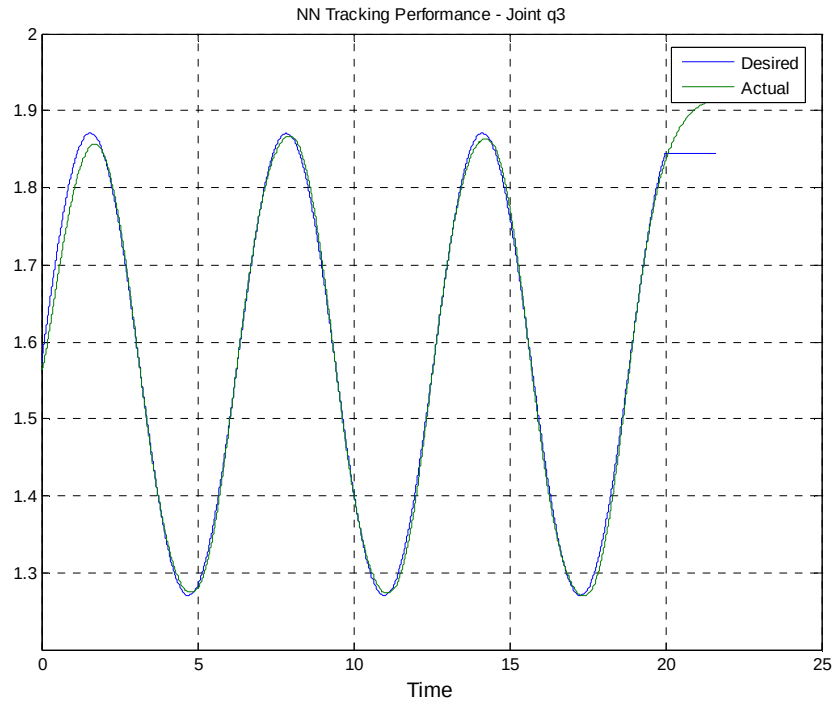



**Figure 52.** NN Tracking performance of Joint q3 for Exp7.2

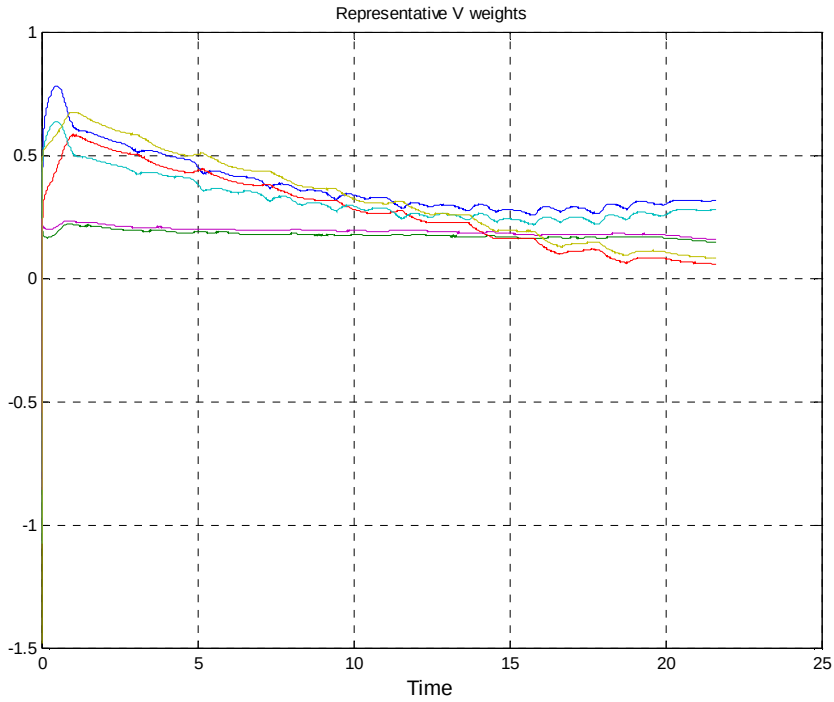



**Figure 53.** Representative V weights for Exp7.2

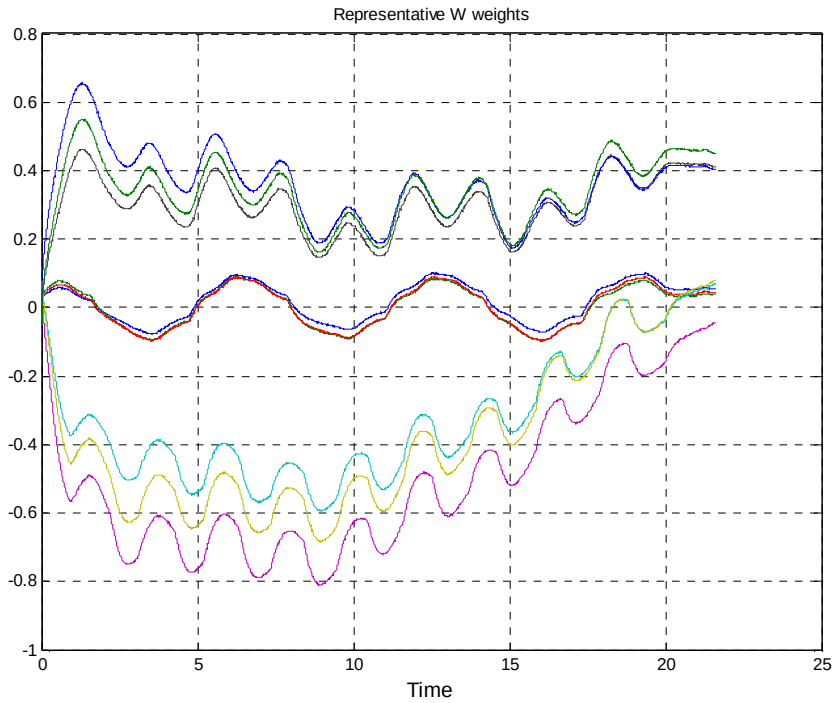
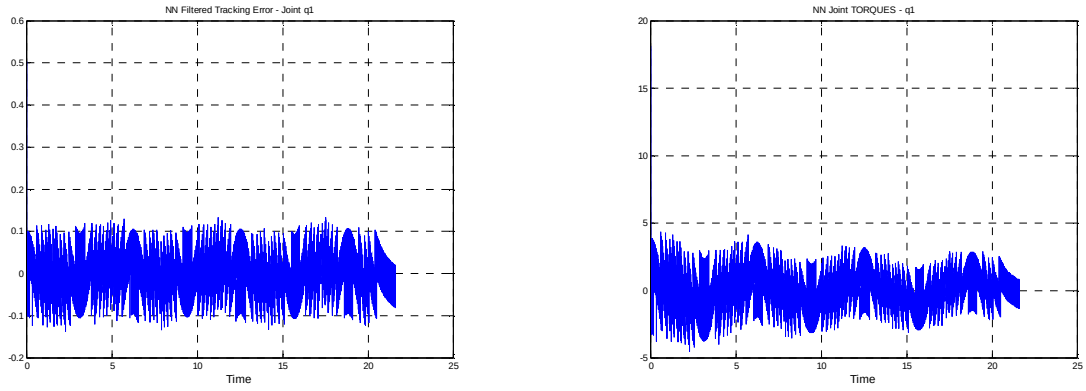
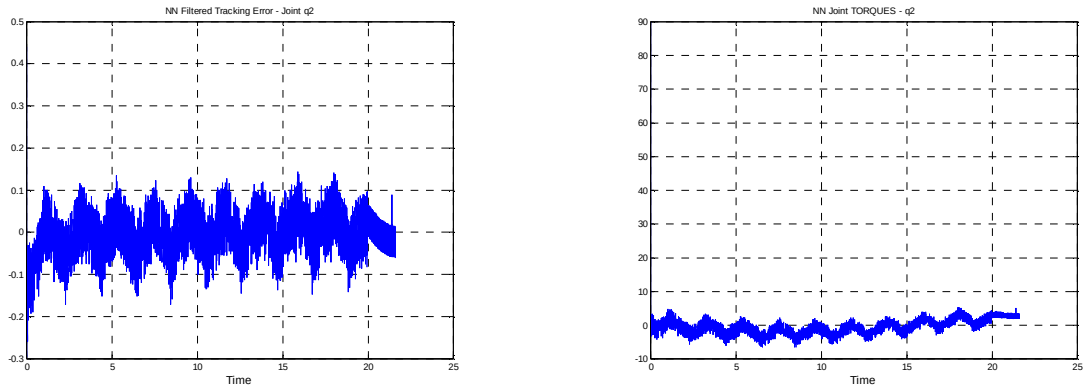



**Figure 54.** Representative W weights for Exp7.2



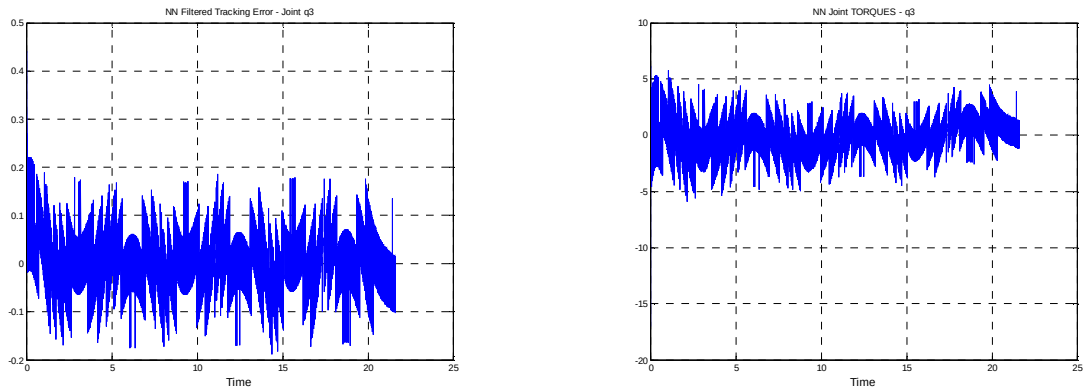

(a) (b)

**Figure 55.** NN Filtered Tracking Error (a) and Torque (b) for Joint q1 – Exp7.2




(a) (b)

**Figure 56.** NN Filtered Tracking Error (a) and Torque (b) for Joint q2 – Exp7.2




(a) (b)

**Figure 57.** NN Filtered Tracking Error (a) and Torque (b) for Joint q3 – Exp7.2

## 7.2.2. PD Controller

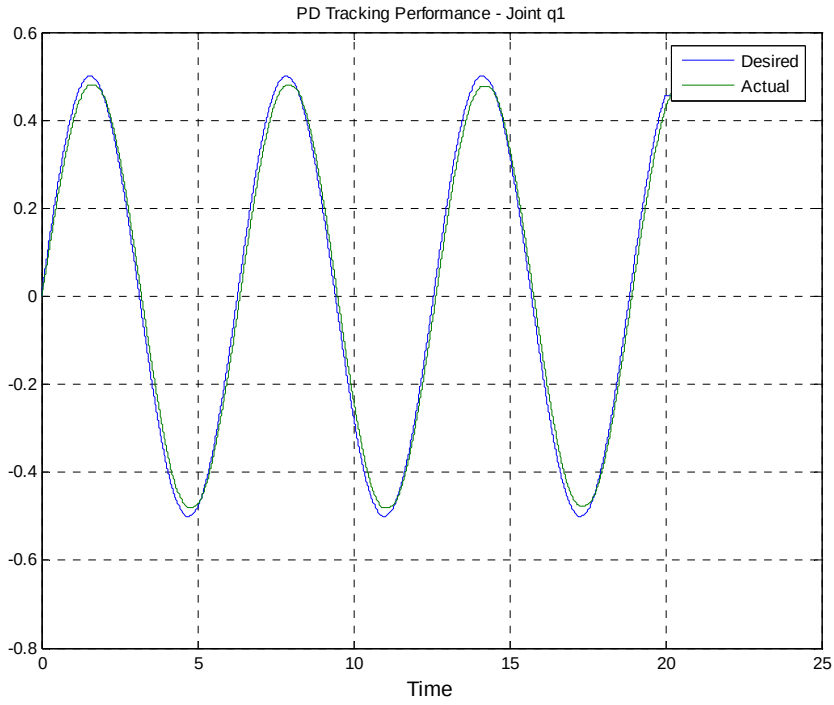



**Figure 58.** PD Tracking performance of Joint q1 for Exp7.2

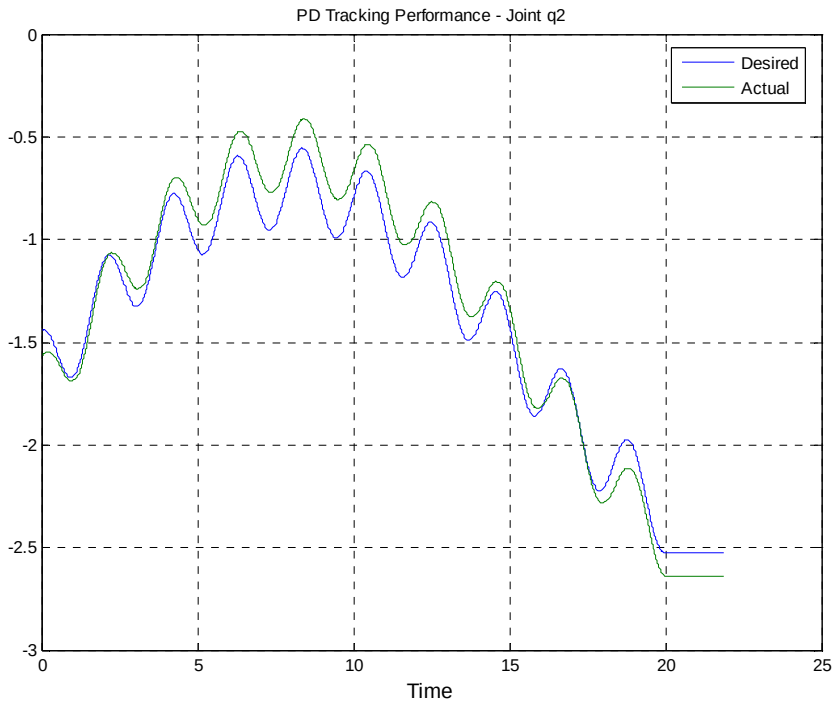



**Figure 59.** PD Tracking performance of Joint q2 for Exp7.2

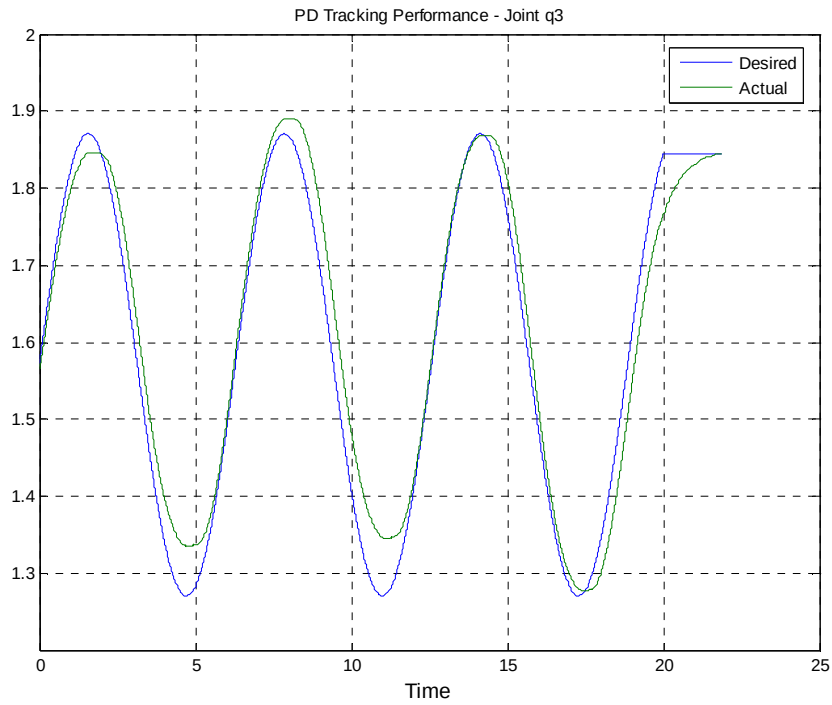
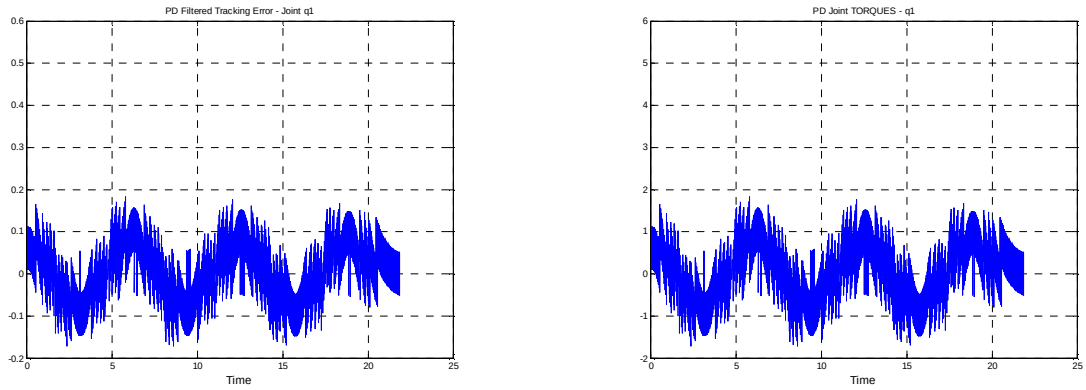
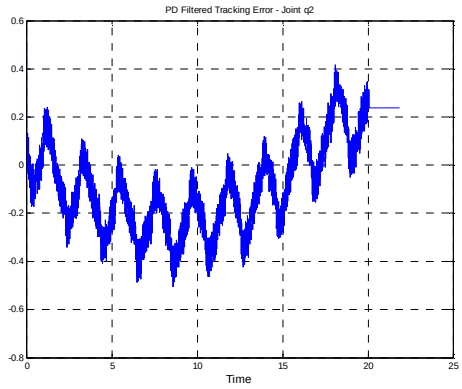



**Figure 60.** PD Tracking performance of Joint q3 for Exp7.2

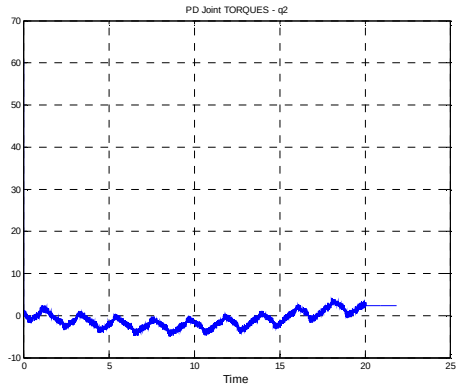



(a) (b)

**Figure 61.** PD Filtered Tracking Error (a) and Torque (b) for Joint q2 – Exp7.2

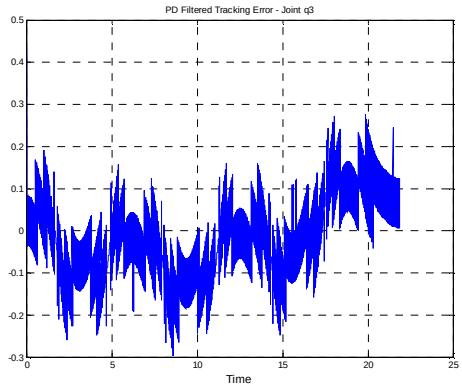



(a)

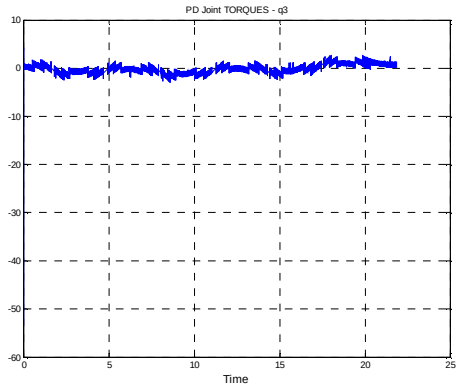



(b)

**Figure 62.** PD Filtered Tracking Error (a) and Torque (b) for Joint q2 – Exp7.2




(a)




(b)

**Figure 63.** PD Filtered Tracking Error (a) and Torque (b) for Joint q3 – Exp7.2

## 7.3. Summary

The results demonstrate that the controllers efficiency is similar to that revealed in the simulations. The effect of its parameters in the overall performance have been analyzed and only some experimental issues will be highlighted:

- The system's initial stability has been very sensitive to the parameters of the Robust signal and have thus been tuned accordingly in order to limit the initial spikes noticed at joint torques during transient response.

- Tracking the trajectory of the second link was more difficult due to the gravity vector. It is apparent that the PD controller cannot compensate effectively, while the NN responds very satisfactory achieving very good tracking.
- The filtered tracking error has been kept small even in the case of a more complex trajectory like Gen3

# Chapter 8

# 8. Conclusions

The Neural Network rigid robot controller proposed in [1] has been implemented and tested in simulations for a two link planar elbow arm and a simulated model of Puma. Furthermore, it has been used for real time experiments with Unimation Puma 560 manipulator. Its tracking performance has been evaluated and compared with the one of a simple PD controller both on simulations and experiments. With respect to the research objectives set by this project:

- The proposed Neural Network controller structure has demonstrated that it is a reliable algorithm that provides repeatability in the design. It has been tested in simulations with two different simulated robot models and in real time experiments with Puma 560
- The tuning algorithm has ensured that the weights remain bounded for trajectory tracking
- The filtered tracking error has remained bounded and tracking performance has been achieved
- Simulations have been used successfully to provide insight of the controller properties and features and configure the system before proceeding to real time experiments
- Real time experiments performed with Puma 560 manipulator have demonstrated very good tracking performance.

As a further work, more experiments need to be done in order to fine tune the controller with respect to Puma 560 platform in order to benefit from its capabilities and maximize its tracking efficiency in real time experiments. Finally, experiments including all six links are to be expected very soon.

# References


[1] F.L.Lewis, *Neural* Network Control of Robot Manipulators, *IEEE trans*, Vol 11, No.3, June 1996, pp.64-75.

[2] K.J.Astrom, B.Wittenmark, *Adaptive Control*, Addision Wesley, Reading, MA, 1989

[3] Y.D.Landau, *Adaptive Control*, Marcel Dekker, Basel, 1979

[4] F.L.Lewis, D.M.Dawson, C.T.Abdallat, *Robot Manipulator Control, Theory and Practice*, Marcel Dekker, 2004

[5] K.S.Narendra, "Adaptive Control Using Neural Networks", *Neural Networks for Control*, pp 115-142. ed. W.T.Miller, R.S.Sutton, P.J.Werbos, Cambridge: MIT press, 1991

[6] K.S.Narendra, K.Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks", *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 252-262, Mar 1991

[7] W.T.Miller, R.S.Sutton, P.J.Werbos, ed., *Neural Networks for Control*, Cambridge: MIT press, 1991

[8] F.C.Chen, H.K.Khalil, "Adaptive control of nonlinear systems using neural networks", *Int. J. Control*, vol. 55, no. 6, pp. 1299-1317, 1992

[9] F.C.Chen, C.C.Liu, "Adaptively controlling nonlinear continuous-time systems using multilayer neural networks", *IEEE Trans. Automat. Control*, vol. 39, no. 6, pp. 1306-1310, June 1994

[10] M.M.Polukarpou, P.A.Ioannou, “Identification and control using neural network models: design and stability analysis”, *Tech. Report 91-09-01*, Dept. Elect. Eng. Sys., Univ. S. Cal., Sept. 1991

[11] N.Sadegh, “A perceptron network for functional identification and control of non-linear systems”, *IEEE Trans. Neural Networks*, vol. 4, no. 6, pp. 982-988, Nov. 1993

[12] G.A.Rovithakis, M.A.Christodoulou, “Adaptive control of unknown plants using dynamical neural networks”, *IEEE Trans. Systems. Man, and Cybernetics*, vol. 24, no. 3, pp. 400-412, Mar. 1994

[13] F.L.Lewis, S.Jagannathan, and A.Yesilderik, *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor and Francis, London, 1999.

[14] V.M.Becerra, C.N.J.Cage, W.S.Harwin and P.M.Sharkey, “Hardware Retrofit and Computed Torque Control of a Puma 560 Robot”, *IEEE Control Systems Magazine*, Vol 24, No. 5, Okt 2004, pp. 78-82.

[15] P.I.Corke, *Robotics Toolbox for Matlab*, Release 6, April 2001.

[16] J.J.Craig, *Introduction to Robotics, Mechanics and Control*, Third edition,Pearson, Prenctice Hall, pp. 273-295.

[17] Frank L. Lewis, Aydin Yesildirek, and Kai Liu, “Multilayer Neural-Net Robot Controller with Guaranteed Tracking Performance”, *IEEE trans on Neural Networks*, Vol. 7, No2, March 1996, pp. 388-399

[18] K.S.Narendra and A.M.Annaswarny, “A new adaptive law for robust adaptation without persistent excitation”, *IEEE Trans. Automat. Control*, vol. AC-32, no. 2, pp. 134-145, Feb. 1987

[19] K.S.Narendra and K.Parthasarathy, “Identification and control of dynamical systems using neural networks”, *IEEE Trans. Neural networks*, vol. 1, pp. 4-27, Mar. 1990.

[20] L.B.Gutierrez, F.L.Lewis and J.A.Lowe, “Implementation of a neural network tracking controller for single flexible link: comparison with PD and PID controllers”, *IEEE Trans on Industrial Electronics*, vol. 45, No 2, April 1998, pp. 307-318

# Appendix I: Lab560 Toolbox

See attached documents

# Appendix II: Matlab code

```
% Initialization script


% Clear workspace
clear;
clc;

% Initializing Puma
PUMA560akb;
robot=p560m;

% Init Puma_Gui for Real time Trajectory tracking
% Comment next line if running simulations
Puma_Gui

% Hidden Layer Neurons
L = 10;

% PD parameters
Kv = 10;
lamda = 1.5;

% Robust Controller parameters
Kz = 1.5;
Zm = 5;

% Neural Network parameters
F = 10;
G = 10;
k = 0.1;

% Select model for Simulations:
% ----------------------------------------------
% outP = 2 for RR model(2 link Planar Elbow Arm)
% outP = 6 for Puma model
% ----------------------------------------------
outP = 6;

% Number of signals of NN input vector
num = 5

% Size of NN input vector
X = num*outP;

% Initialize the AUGMENTED NNs WEIGHT MATRICES
WHat_ini = zeros(L+1,outP);
temp = zeros(X+1,1);
VHat_ini = [temp rand(X+1,L)];
```