Neural Network Based Tuning Algorithm for MPID Control

Tamer Mansour, Atsushi Konno and Masaru Uchiyama
Tohoku University
Japan

1. Introduction

One of the biggest problems for space manipulators are to cope with flexibility. If manipulator links undergo deflection during the course of operation, it may prove difficult to reach a desired position or to avoid obstacles. Furthermore, once the manipulator has reached a set point, the residual vibration may degrade positioning accuracy and may cause a delay in task execution. At the same time the flexible manipulators has the advantages of high payload to weight ratio, which make them superior in the space exploration and orbital operation. The high payload to weight ration is not the only merits of using flexible manipulators in space application. Lower power consumption, smaller actuators and speedy operation make the flexible manipulators the optimum choice for space manipulators. Since Cannon et al. (Cannon & Schmitz, 1984) started initial experiments using the linear quadratic approach methods to control flexible link manipulators, much researches on the usage of flexible manipulator had been developed.
Using the approach of enhancement the measurements of the vibration variables was studied by (Ge et al., 1999; Sun et al., 2005) while Etxebarria et al. (Etxebarria et al., 2005) gives attention to the algorithms used in controlling the flexible manipulators. The enhancement of the traditional PD controller by adding a vibration control term is one of the most effective methods for the flexible manipulators. Lee et al. proposed PDS (proportional-derivative strain) control for vibration suppression of multi-flexible-link manipulators and analysed the Liapunov stability of the PDS control (Lee et al., 1988). Maruyama et al. (Maruyama et al., 2006) developed a golf robot whose swing simulates human motion. They presented model accounting for golf club flexibility with all parameters identified in experiments and generated and implemented trajectories for different criterion such as minimizing total consumed work, minimizing summation of the squared derivative of active torque and maximizing impact speed. Matsuno and Hayashi applied the PDS control to a cooperative task of two one-link flexible arms (Matsuno & Hayashi, 2000). They aimed to accomplish the desired grasping force for a common rigid object and the vibration absorption of the flexible arms.
A neural network is a data modelling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural Networks resemble the human brain in the following two ways:

- A neural network acquires knowledge through learning.
- A neural network knowledge is stored within inter neuron connection strengths known as weights.

The true power and advantages of neural networks lies in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modelled. Traditional linear models are simply inadequate when it comes to modelling data that contains non-linear characteristics. Some researchers tried to use the neural network (herein after abbreviated as NN) as a main controller like (Talebi et al., 1998). In their research the controllers are designed by utilizing the modified output re-definition approach. The modified output re-definition approach requires only a priori knowledge about the linear model of the system but does not require a priori knowledge about the payload mass. Various NN schemes have been proposed so far such as a modified version of the "feedback error-learning" approach to learn the inverse dynamics of the flexible manipulator (Kawato et al., 1987). On the other proposed NN structure the controller is designed based on tracking the reference joint angle while controlling the elastic deflection at the tip. Isogai et al. (Isogi et al., 1999) proposed a fault-tolerant system using inverse dynamics constructed by NN for sensor fault detection and NN adaptive control for the actuator fault to reconfigure control to compensate for parameter changes due to actuator faults.

Other researches like Lianfang (Lianfang et al., 2004) use the neural network as a correction for the main control signal coming from the main feed-back controller. In his research the neural network approach is presented for the motion control of constrained flexible manipulators where both the contact forces exerted by the flexible manipulator and the position of the end-effectors contacting with a surface are controlled. Cheng and Patel in (Cheng & Patel, 2003) tried to made stable tracking control of a flexible macro-micro manipulator utilizing two layer neural network to approximate the non-linear robot dynamic behavior. A learning algorithm for the neural network using Lyapunov stability is derived. Yazdizadeh et al. proposed two neuro-dynamic identifiers to identify the input-output relationship of two-link flexible manipulator. They provided in (Yazdizadeh et al., 2000) a selection criterion for specifying the fixed structural parameters as well as the adaptation laws for updating the adjustable parameters of the networks.

A Modified PID control (MPID) is proposed to control a single-link flexible manipulator by Mansour et al. (Mansour et al., 2008). The MPID control depends mainly on vibration feedback to improve the response of the flexible arm without the massive need of measurements. The advantage of the MPID is that it is not affected by residual strain due to material defect and/or static deformation. The residual strain and material defect may lead to inaccurate movement. The difficulty with the MPID is that it includes nonlinear terms and so the standard gain tuning method can not be used for the controller. The motivation for this research is to find a fast and simple way to tune the MPID controller, which is able to achieve final accurate tip position for the flexible arm and at the same time reduce resulting vibration. The NN is used to solve this problem. In this research a NN is used to find an optimum vibration gain of MPID controller. The main advantage of the NN approach to tune the vibration control gain of the MPID control is the considerable low computational cost to find an optimal tuned gain with different tip payload.

The neural network is used to estimate a result of the dynamic simulation when the simulation condition is given. As a result of the dynamic simulation, integral of the squared tip deflection weighted by exponential function

$$\text{Criterion function} = \int_0^{t_s} \delta^2 e^t dt, \qquad t_s : \text{settling time} \tag{1}$$

is considered in this work. Therefore, the input to the neural network is the simulation condition, while the output is the criterion function defined in 1. The mapping from the input to the output is many-to-one. In order to train the neural network, the results of a dynamic simulator for a given condition are used as teacher signals. In this shadow the feed-forward neural network can be used as a mapping between the simulation conditions and the output response all over the time span which is represented by $\int_0^{t_s} \delta^2 e^t dt$.
The powerful ability of the neural network to model nonlinear model is utilized to map the relation between the vibration control gain of the MPID and the output response represented by the criterion function. Once this relation is drawn, the optimum value of the vibration control gain is corresponding to the minimum value of the criterion function.
The sequence of finding the optimum value for the vibration control gain for the single link flexible manipulator is summarized in Fig. 1. This chapter is organized as follows: An introduction to the control of flexible manipulator and using NN in the control process is highlighted in section 1. The mathematical model of the flexible manipulator is shown in section 2. The detailed of the controller structure and the simulatiom model are presented in sections 3 and 4. In section 5 the NN algorithm used in this research is explained, the structure of the NN is also shown and the optimal vibration control gain finding procedure are highlighted. The learning and training process of the NN is shown in section 6. The response results for the flexible manipulator with the tuned gain using the NN is shown in section 7. Finally, section 8 concludes this chapter with some remarks.

2. Mathematical Model

Before discussing the NN based gain tuning method, the MPID controller (Mansour et al., 2008) is briefly introduced in sections 2 and 3. From the analysis of the single-link flexible arm shown in Fig. 2, the flexible link is approximated by a continuous clamped-free beam. The flexible arm is rotating in the horizontal plane with a rotational angle $\theta(t)$ and the effect of gravity is not taken into consideration. Frame *O-XY* is the fixed base frame and frame *O-xy* is the local frame rotating with the hub. The tip deflection $\delta(L,t)$ is the difference between the actual tip position and the rotating frame *O-xy*. The deflection $\delta(x,t)$ is assumed to be small compared to the length of the arm. Let $p(x,t)$ represents the tangential position of a point on the flexible arm with respect to frame *O-xy*. From the assumption of the deflection of the flexible arm, the tangential position is expressed as:

$$p(x,t) = x\theta(t) + \delta(x,t). \tag{2}$$

The flexible arm is treated as Euler-Bernoulli beam with uniform cross-sectional area and constant characteristics. Then, the Euler-Bernoulli equation for the link is given as follows :

$$EI\frac{\partial^4 p(x,t)}{\partial x^4} + \rho\frac{\partial^2 p(x,t)}{\partial t^2} = 0, \tag{3}$$

where ρ is the sectional density, E is the Young (elastic) modulus, and I is the second moment of area. Substituting (2) into (3) the following equation is obtained :

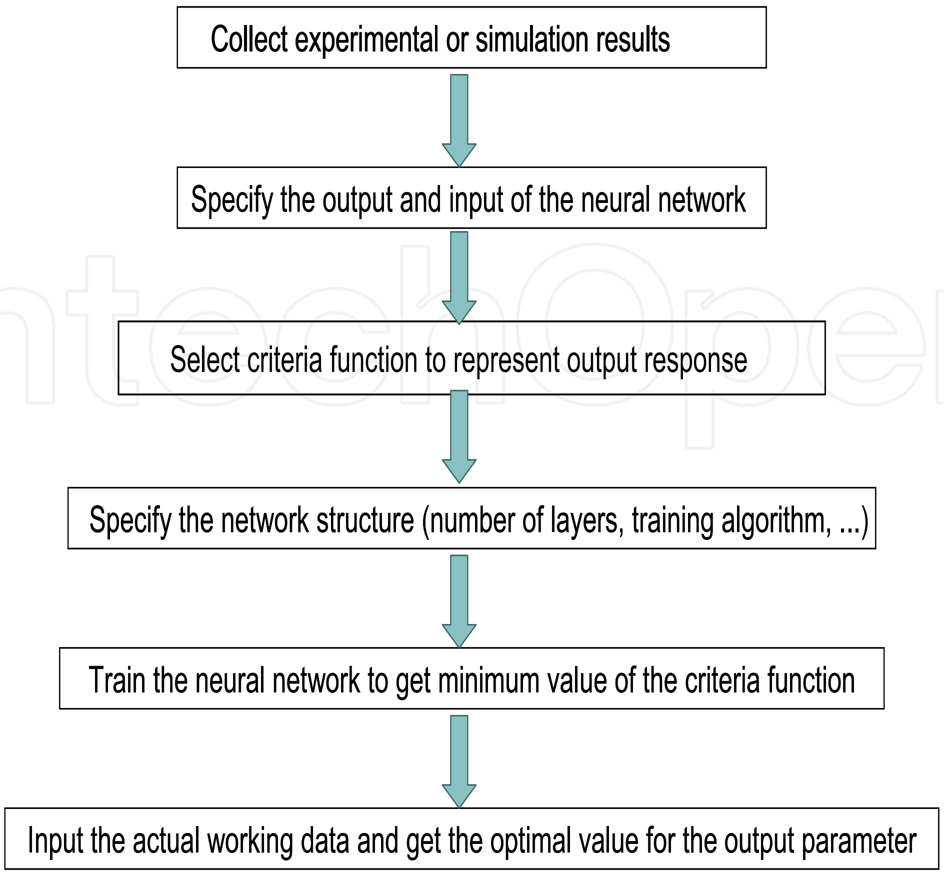

Collect experimental or simulation results
Specify the output and input of the neural network
Select criteria function to represent output response
Specify the network structure (number of layers, training algorithm, ...)
Train the neural network to get minimum value of the criteria function
Input the actual working data and get the optimal value for the output parameter

Fig. 1. Neural network using sequence.

$$EI\frac{\partial^4\delta(x,t)}{\partial x^4}+\rho\frac{\partial^2\delta(x,t)}{\partial t^2}=-\rho x\ddot{\theta}(t). \tag{4}$$

The flexible arm is clamped at its base, so both the deflection and slope of the deflection curve must be zero at the clamped end. Bending moment at the free end also equals zero. Making force balance at the tip obtains the following boundary conditions:

$$\delta(x,t)|_{x=0} = 0, \tag{5}$$

$$\left.\frac{\partial\delta(x,t)}{\partial x}\right|_{x=0} = 0, \tag{6}$$

$$EI\left.\frac{\partial^2\delta(x,t)}{\partial x^2}\right|_{x=L} = 0, \tag{7}$$

$$EI\left.\frac{\partial^3\delta(x,t)}{\partial x^3}\right|_{x=L} = m_t\left[x\ddot{\theta}(t)+\frac{\partial^2\delta(x,t)}{\partial t^2}\right]_{x=L}, \tag{8}$$

where L is the arm length. The dynamic equation describing the system presented in (**?**) is written as follows:

$$\begin{aligned} T(t) &= \left(I_h+\frac{1}{3}\rho L^3\right)\ddot{\theta}(t)+\rho\int_0^L x\ddot{\delta}(x,t)dx \\ &\quad +m_tL\left(L\ddot{\theta}(t)+\ddot{\delta}(L,t)\right). \end{aligned} \tag{9}$$

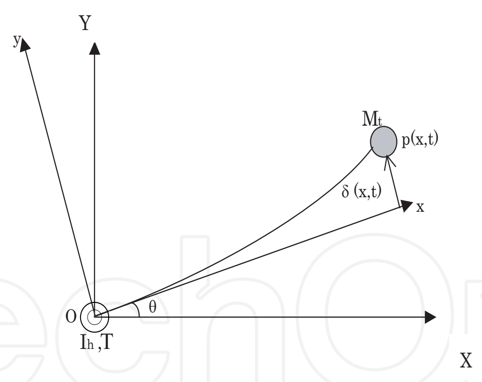


Fig. 2. Single-link flexible manipulator.

A flexible manipulator simulator is built in MATLAB Simulink software using the mathematical model shown before to study the performance of the MPID control with different loading and gains conditions.

3. Controller

A Modified PID controller (MPID) is proposed for controlling the tip position of the single-link flexible manipulator (Mansour et al., 2008). This controller used three measurements to generate the control signal, the hub rotational angle $\theta(t)$, the tip deflection $\delta(L,t)$, and the velocity of the hub $\dot{\theta}(t)$. If we choose the tip position as the output from the system then the error includes two components. The first component $e_j(t)$ is a result of the joint motion and is equal to $L(\theta_{ref} - \theta(t))$ which is identical with the rigid arm error where θ_{ref} is the reference joint angle. The second one is much more important and is due to the flexibility of the arm and equals $\delta(L,t)$. These two error components are coupled to each other. The Modified PID (MPID) controller replaces the classical integral term of a PID controller with a vibration feedback term to affect the flexible modes of the beam in the generated control signal. The MPID controller is formed as follows (Mansour et al., 2008):

$$\begin{aligned} u(t) \quad &= \quad K_p e_j(t) + K_d \dot{e}_j(t) \\ & \quad\quad +K_{vc} g(t) \ \mathrm{sgn}(\dot{e}_j(t)) \int_0^t g(t) dt, \end{aligned} \tag{10}$$

where $u(t)$ is the control signal, K_p, K_d are the proportional and derivative gains for the joint control,respectively, K_{vc} is the vibration control feedback gain, $e_j(t)$ is the tangential position error and $g(t)$ is a vibration variable such as strain, deflection, shear force or acceleration under a single condition that the vibration variable value equal zero when the flexible manipulator is static and under goes no deformation. The stability of the proposed controller had been studied previously in (Mansour et al., 2008). It was proved that the system is stable as long as $K_d \geq 0$. The flexible manipulator simulator is used to validate the MPID controller given by (10) and the results are shown in Figs. 3 and 4. We found from the simulation results that the response of the flexible manipulator is sensitive to the change of the controller gains. In addition to that, the change in the tip payload have a noticeable influence on the response of the flexible manipulator end effector. If the controller gain is not tuned well, the response with the new loading condition will suffer a performance degradation. As shown in Fig. 3, a

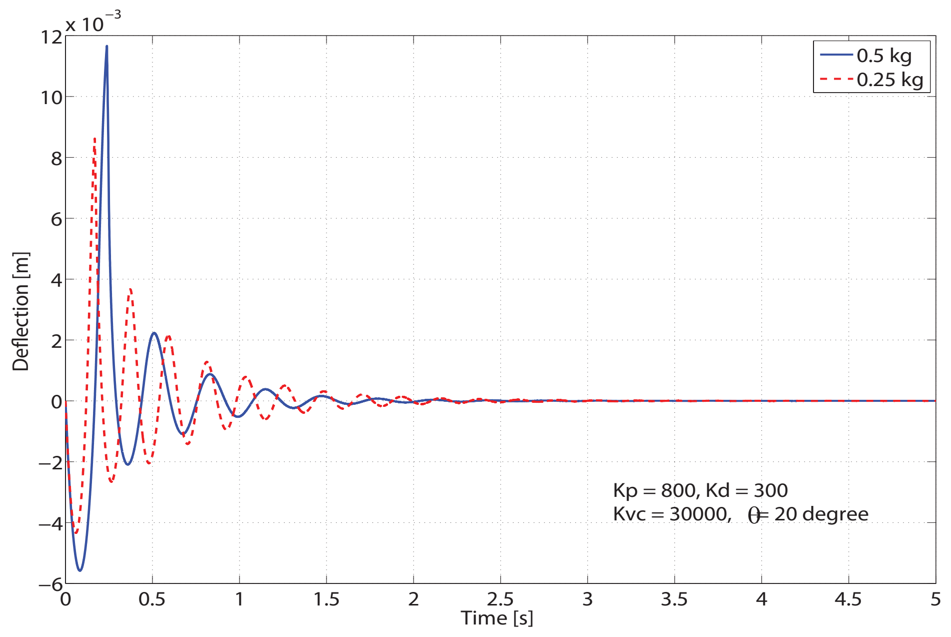


Fig. 3. Effect of changing tip load.

change in the tip load of the flexible manipulator has an undesirable effect for the vibration of the end effector.

Not only the change of the environment parameters like the tip payload causes an undesirable effect on the response as shown in Fig. 3, but also changing the system configuration like joint angle causes the same effect. Unlike industrial manipulators, both the environment parameter (i.e. tip payload) and the system configuration (i.e. joint angle) are always changeable in the case of space manipulators. This highlights the importance of optimization the gain used with this controller.

Another important point is that the change of the vibration control gain K_{vc} has seriously affects on the response of the single-link flexible manipulator. This is completely noticeable from the results in Fig. 4. This fact is the main motivation to find out a good way for tuning K_{vc} that brings the minimum vibration for the tip as well as a fast response for the joint position. It is predicted from Fig. 4 that the damping effect becomes stronger as the vibration control gain K_{vc} increases to a certain limit. However if the gain K_{vc} exceeds the limits it start to create an overshoot in the joint response.

The most difficulty of using the MPID controller is the adjustment of the vibration control gain. Ge et al. tried to use the genetic algorithm optimization process to find the suitable gain for the controller (Ge et al., 1996). In their research they consider the fixed tip payload of the flexible manipulator and generate a set of gains for this configuration using the genetic algorithm. However in general, the tip payloads and the joint angle are not the same in each operation but it varies from one task to another. Hence the tuning of the vibration control gain K_{vc} becomes the most importance issue to achieve the required position with a minimum vibration. To overcome the lake of consideration with the changing of tip payload and joint angle in the tuning of the MPID we proposed to use the NN in the tuning of the MPID.

In this research the vibration control gain K_{vc} for the MPID controller given by equation (10) is tuned using the NN for the environment parameter (i.e. tip payload), the system configuration

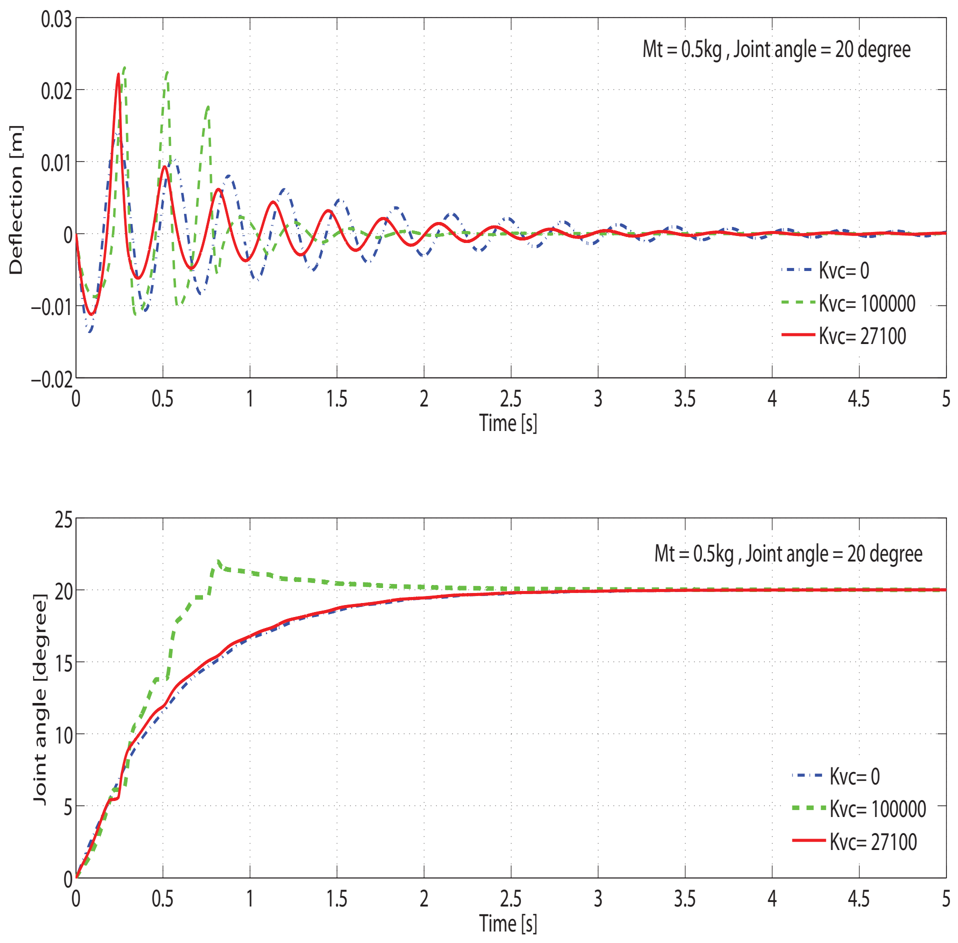


Fig. 4. Effect of changing vibration control gain.

(i.e. joint angle) and for both the other controller gains (i.e. K_p, K_d). By this way the controller gives the best response with respect to all the parameters related to the flexible manipulator.

4. Simulation Model

In this section, a highlight to the simulation which is used to simulate the flexible manipulator is given. A model of the flexible robot control system is simulated in Matlab-Simulink software. The development of the Matlab-Simulink model allows control algorithms to be evaluated before we use the neural network. Also, the simulation program is used to provide information about the behave of the system. The result we get from the simulation will be used in selecting the criterion function, which we will train the neural network on it. In this simulation, we used the mathematical equation derived on section (2) for the flexible manipulator. The block of the simulation model which had been used is shown in Fig. 5. We wish to give attention to some point we consider in make the simulation and important simulation parameters. In the simulation, we use the variable step solvers not the fixed step solvers. The Variable step solvers vary the step size during the simulation, reducing the step size to

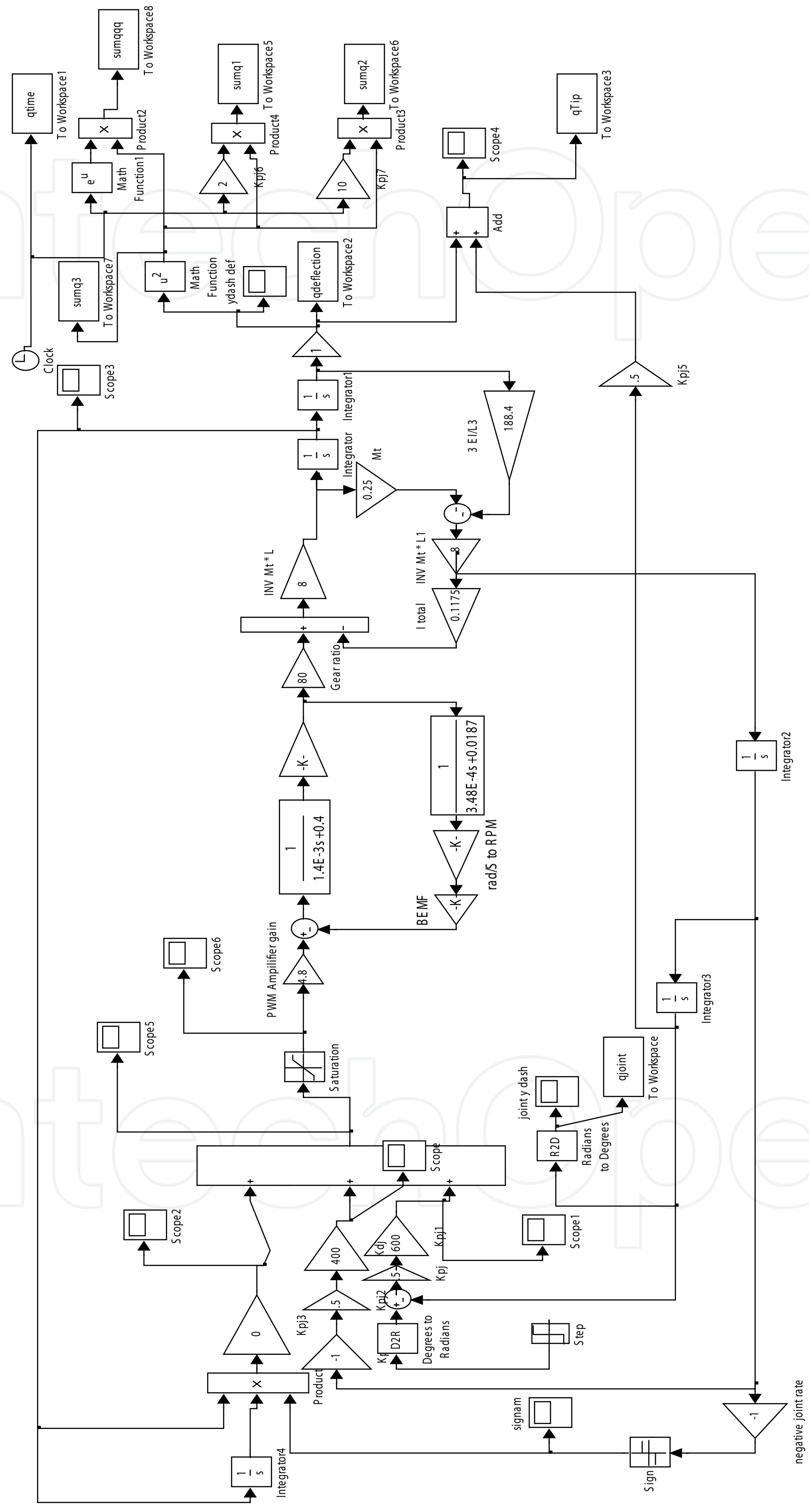


Fig. 5. Simulation model for the flexible manipulator.

increase accuracy when a model’s states are changing rapidly and increasing the step size to avoid taking unnecessary steps when the model’s states are changing slowly. Computing the step size adds to the computational overhead at each step but can reduce the total number of steps, and hence simulation time, required to maintain a specified level of accuracy for models with rapidly changing or piecewise continuous states. Also for the numerical integration techniques for solving the ordinary differential equations (ODEŠs) that represent the continuous states of dynamic systems. Simulink provides an extensive set of fixed-step and variable-step continuous solvers, each implementing a specific ODE solution method. There are many solver types in the solution methods. First we identifying the optimal solver for the model, optimal means acceptable accuracy with the shortest simulation. We use in the solution of the numerical integration for the ordinary differential equations the ode45 (Dormand-Prince). ODE45 is based on an explicit Runge-Kutta (4,5) formula. It is a one-step solver; that is, in computing $y(t_n)$, it needs only the solution at the immediately preceding time point, $y(t_{n-1})$. In general, ode45 is the best solver to apply if the problem is not stiff. It is also the default solver used by Simulink for models with continuous states.

5. Neural Network

There is no precise agreed definition among researchers as to what a neural network is, but most would agree that it involves a network of simple processing elements (neurons), which can exhibit complex global behavior, determined by the connections between the processing elements and element parameters. The original inspiration for the technique was from examination of the central nervous system and the neurons (and their axons, dendrites and synapses) which constitute one of its most significant information processing elements. In a neural network model, simple nodes (called variously “neurons," “neurodes," or “PEs- processing elements") are connected together to form a network of nodes Ů hence the term “neural network." While a neural network does not have to be adaptive itself, its practical use comes with algorithms designed to alter the strength (weights) of the connections in the network to produce a desired signal flow. These networks are also similar to the biological neural networks in the sense that functions are performed collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which various units are assigned. Currently, the term Artificial Neural Network (ANN) tends to refer mostly to neural network models employed in statistics, cognitive psychology and artificial intelligence.
In modern software implementations of artificial neural networks, the approach inspired by biology has more or less been abandoned for a more practical approach based on statistics and signal processing. In some of these systems, neural networks or parts of neural networks (such as artificial neurons) are used as components in larger systems that combine both adaptive and non-adaptive elements. While the more general approach of such adaptive systems is more suitable for real-world problem solving, it has far less to do with the traditional artificial intelligence connection models. What they do however have in common is the principle of non-linear, distributed, parallel and local processing and adaptation. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an “expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer “what if" questions. Other advantages include:

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
- Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

A simple representation of neural network is shown in Fig. 6. The Input to the neural network is presented by $X_1, X_2, \ldots.., X_R$ where R is the number of inputs in the input layer, S is the number of neuron in the hidden layer and w is the weight. The output from the neural network Y is given by

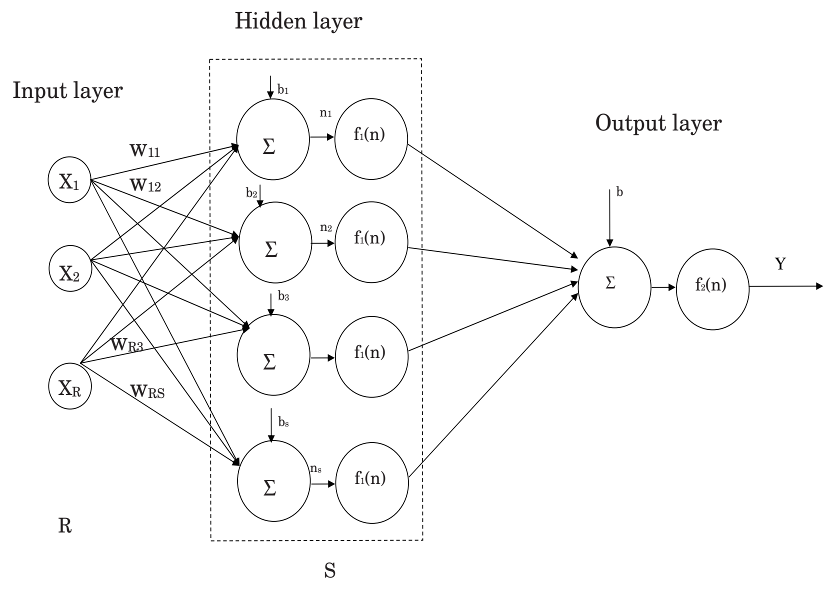


Fig. 6. Simple presentation of neural network.

$$Y = f_2\left(\sum_{j=1}^{j=S} f_1(n_j) + b\right) \tag{11}$$

$$n_j = \sum_{j=1}^{j=S} \sum_{i=1}^{i=R} X_i w_{ij} + b_j \tag{12}$$

where $i = 1, 2, \ldots, R$, $\quad j = 1, 2, \ldots, S$,
f_1 and f_2 represents transfer functions.

To overcome the problem of tuning the vibration control gain K_{vc} due to the changing in the manipulator configuration, environment parameter or the other controller gains the neural network is proposed. The main task of the neural network is to get the optimum vibration control gain which can achieve the vibration suppression while reaching the desired position for the flexible manipulator.

So the function of the neural network is to receive the desired position θ_{ref} and the manipulator tip payload M_t with the classical PD controller gains K_p, K_d. The neural network will give out the relation between the vibration control gain K_{vc} and the criterion function at a certain inputs $\theta_{ref}, M_t, K_p, K_d$. From this relation the value of the value of optimum vibration control gain K_{vc} is corresponding to the minimum criterion function.
A flow chart for the training process of the neural network with the parameters of the manipulator and gains of the controller is shown in Fig. 7. The details of the learning algorithm and how is the weight in changed will be discussed later in the training of the neural network.

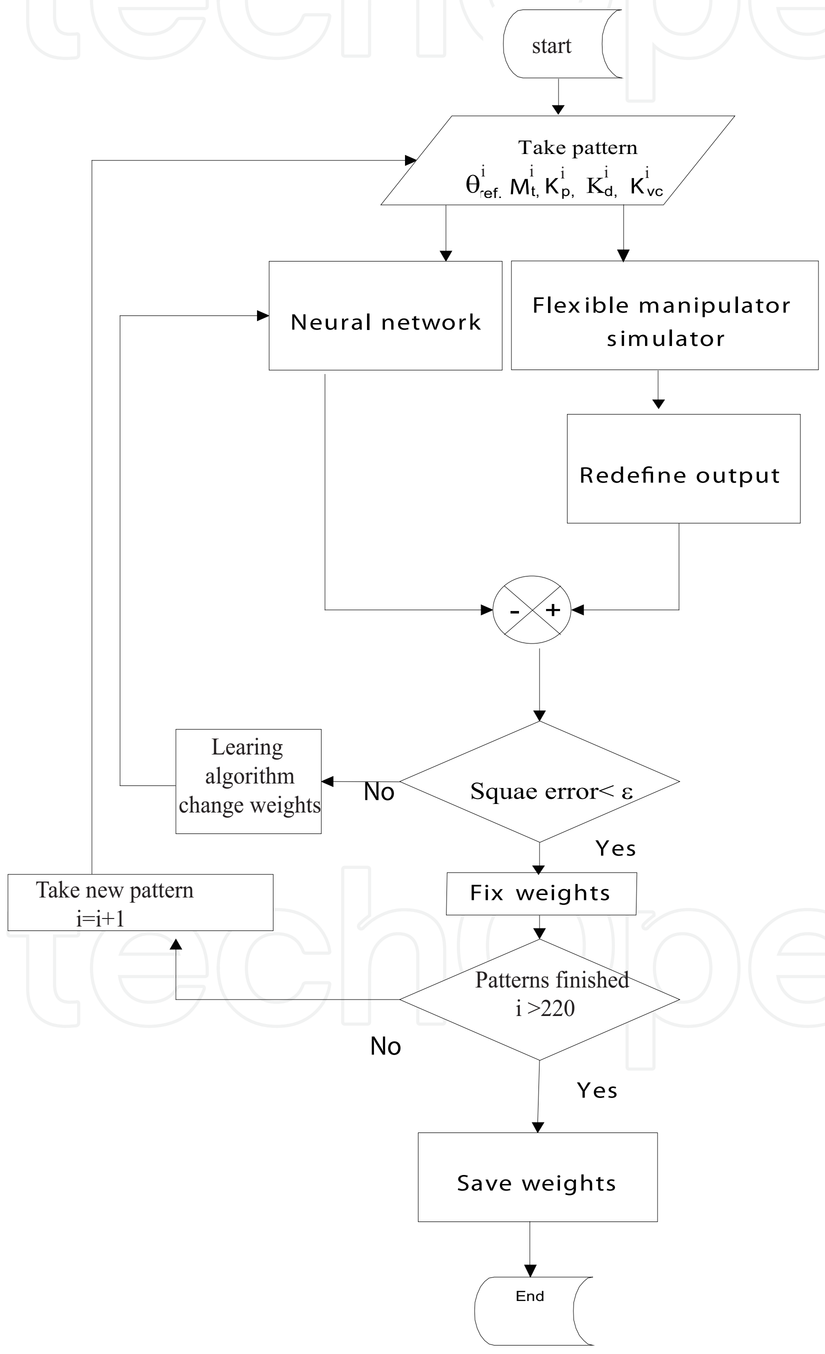


Fig. 7. Flow chart for the training of the neural network.

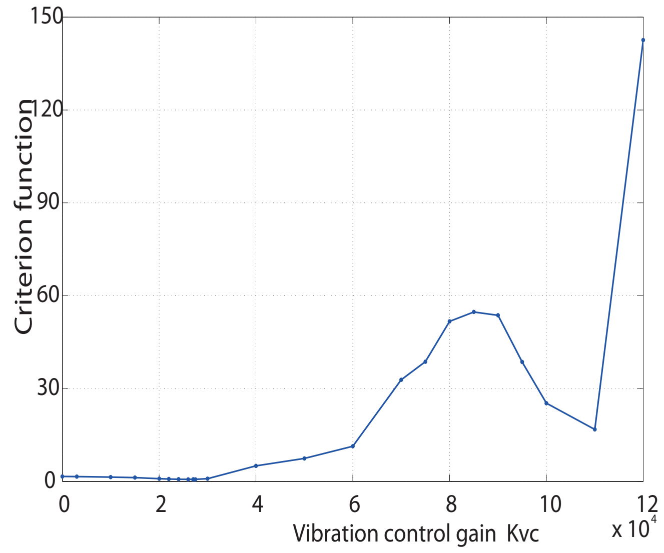


Fig. 8. Relation between vibration control gain and criterion function.

By trying many criterion function to select one of them as a measurement for the output response from the simulation. We put in mind when selecting the criterion function to include two parameters. The first one is the amplitude of the defection of the end effector and the second one is the corresponding time. A set of criterion function like $\int_0^{t_s} t\delta^2 dt$, $\int_0^{t_s} 10t\delta^2 dt$, $\int_0^{t_s} \delta^2 e^t dt$ is tried and a comparison between the behave for all of them and the vibration control gain K_{vc} is done. The value of t_s here represent the time for simulation and on this research we take it as 10 seconds. The criterion function $\int_0^{t_s} \delta^2 e^t dt$ is selected as its value is always minimal when the optimum vibration control gain is used. The term optimum vibration control gain K_{vc} pointed here to the value of K_{vc} which give a minimum criterion function $\int_0^{t_s} \delta^2 e^t dt$ and on the same time keep stability of the system.

The neural network is trained on the results from the simulation with different $\theta_{ref}, M_t, K_p, K_d, K_{vc}$. The neural network is trying to find how the error in the response from the system (represented by the criterion function $\int_0^{t_s} \delta^2 e^t dt$ is changed with the manipulator parameter (tip payload, joint angle) i.e. M_t, θ_{ref} and also how it changes with the other controller parameters K_p, K_d, K_{vc}. The relation between the vibration control gain of the controller, K_{vc} which will be optimized using the neural network and the criterion function, $\int_0^{t_s} \delta^2 e^t dt$ which represent a measurement for the output response from the simulation is shown in Fig. 8. After the input and output of the neural network is specified, the structure of the neural network have to been built. In the next section the structure of the neural network used to optimize the vibration control gain K_{vc} will be explained.

5.1 Design

The neural network structure mainly consists of input layer, output layer and it also may contain a hidden layer or layers. Depending on the application whether it is a classification, prediction or modelling and the complexity of the problem the number of hidden layer is decided. One of the most important characteristics of the neural network is the number of neurons in the hidden layer(s). If an inadequate number of neurons are used, the network will be unable to model complex data, and the resulting fit will be poor. If too many neurons

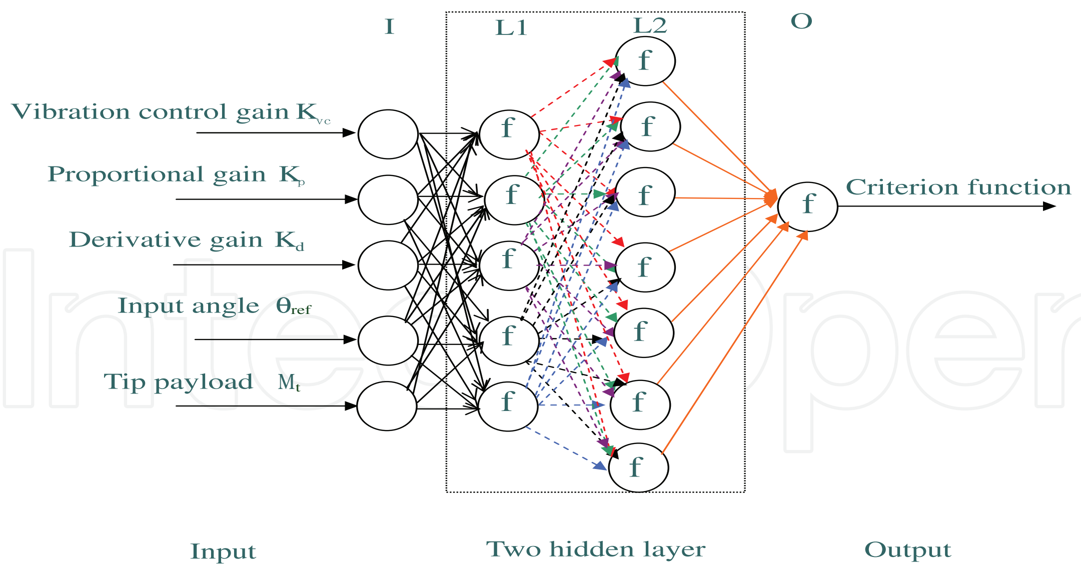


Fig. 9. NN structure.

are used, the training time may become excessively long, and, worse, the network may over fit the data. When over fitting occurs, the network will begin to model random noise in the data. The result is that the model fits the training data extremely well, but it generalizes poorly to new, unseen data.
Validation must be used to test for this. There are no reliable guidelines for deciding the number of neurons in a hidden layer or how many hidden layers to use. As a result, the number of hidden neurons and hidden layers were decided by a trial and error method based on the system itself (Principe et al., 2000). Networks with more than two hidden layers are rare, mainly due to the difficulty and time of training them. The best architecture to be used is problem specific.
A proposed neural network structure is shown in Fig. 9. A neural network with one input layer and one output layer and two hidden layers is proposed. In the proposed neural network the input layer contains five inputs, $\theta_{ref}, M_t, K_p, K_d, K_{vc}$. Those inputs represent the manipulator configuration, environment variable and controller gains. The output layer is consists of one output which is the criterion function and a bias transfer function on the neuron of this layer. The first one of the two hidden layers is consists of 5 neuron and the second one is consists of 7 neurons. For the transfer function used in the neuron of the two hidden layer first we use the sigmoid function described by 13 to train the neural network.

$$f(x_i, w_i) = \frac{1}{1 + \exp(-x_i^{bias})}, \tag{13}$$

where $x_i^{bias} = x_i + w_i$.
The progress of the training of the neural network is shown when using sigmoid transfer function in Fig. 10. As we notice that no good progress in the training we propose to use the tanh as a transfer function for the neuron for both of the two layers. Tanh applies a biased tanh function to each neuron/processing element in the layer. This will squash the range of each neuron in the layer to between -1 and 1. Such non-linear elements provide a network with the ability to make soft decisions. The mathematical equation of the tanh function is give

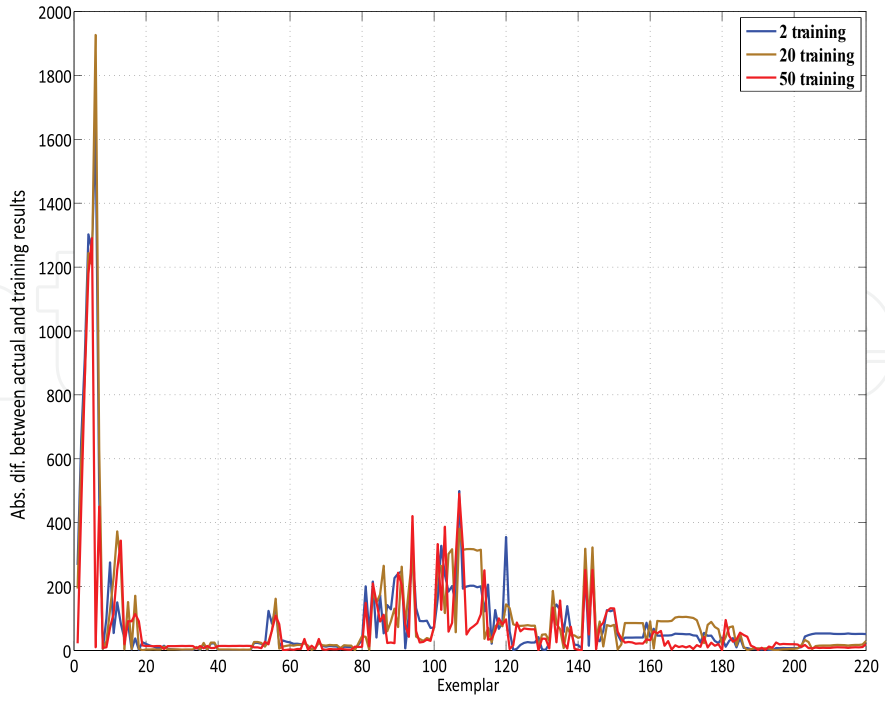


Fig. 10. Progress in training using sigmoid function.

by 14.

$$f(x_i, w_i) = \frac{2}{1 + \exp(-2x_i^{bias})} - 1, \tag{14}$$

where $x_i^{bias} = x_i + w_i$. Also the progress in the training of the neural network using the tanh function is shown in Fig. 11.

5.2 Optimal Vibration Control Gain Finding Procedure

The MPID controller includes non-linear terms such as $\text{sgn}(\dot{e}_j(t))$, therefore standard gain tuning method like Ziegler-Nichols method can not be used for the controller. For the optimal control methods like pole placement, it involves specifying closed loop performance in terms of the closed-loop poles positions.
However such theory assumes a linear model and a controller. Therefore it can not be directly applied to the MPID controller.
In this research we propose a NN based gain tuning method for the MPID controller to control flexible manipulators. The true power and advantages of NN lies in its ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modelled. Traditional linear models are simply inadequate when it comes to modelling data that contains non-linear characteristics. The basic idea to find the optimal gain K_{vc} is illustrated in Fig. 12 (a). The procedure is summarized as follows.

1. A task, i.e. the tip payload M_t and reference angle θ_{ref}, is given.
2. The joint angle control gains K_p and K_d are appropriately tuned without considering the flexibility of the manipulator.
3. Initial K_{vc} is given.

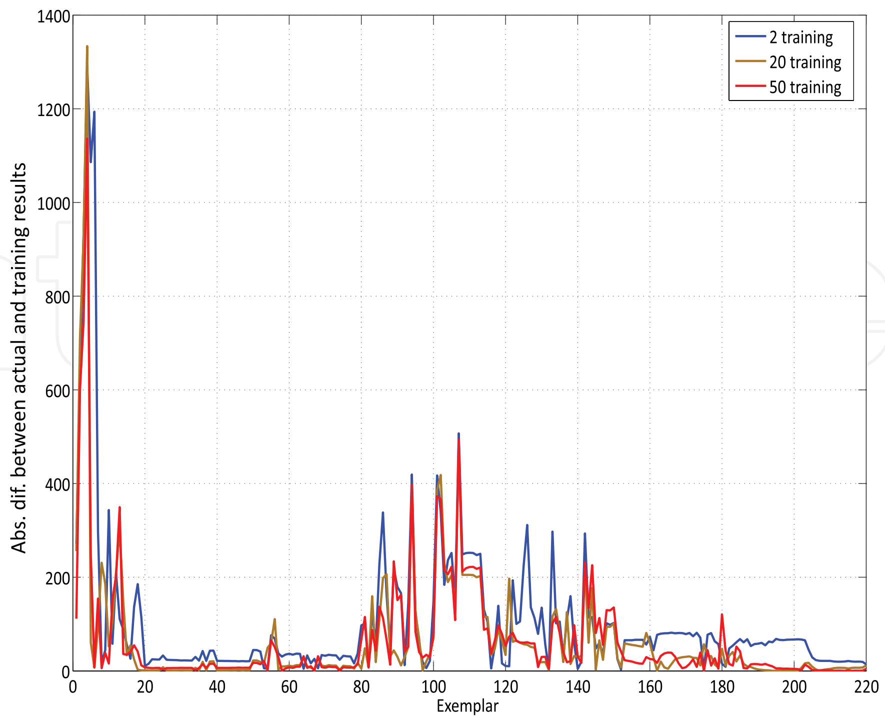


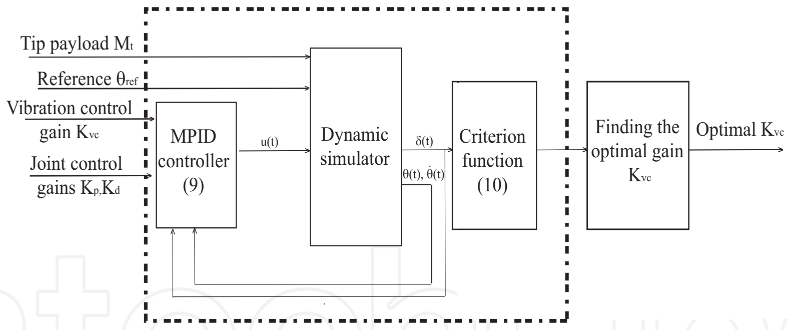
Fig. 11. Progress in training using tanh function.

4. The control input $u(t)$ is calculated with given K_p, K_d, K_{vc}, θ_{ref} and θ_t using (10).
5. Dynamic simulation is performed with given tip payload M_t and the control input $u(t)$
6. 4 and 5 are iterated when $t \leq t_s$ (t_s: given settling time).
7. Criterion function is calculated using (15).
8. 4 $\sim$ 7 are iterated for another K_{vc}.
9. Based on the obtained criterion function for various K_{vc}, an optimal gain K_{vc} is found

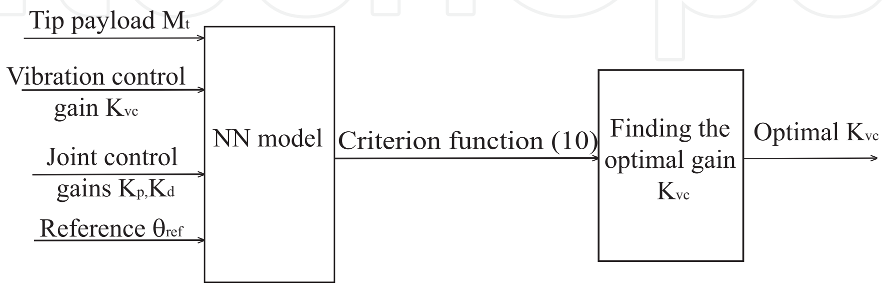
As the criterion function $C(M_t, \theta_{ref}, K_p, K_d, K_{vc})$, the integral of the squared tip deflection weighted by exponential function is considered as:

$$C(M_t, \theta_{ref}, K_p, K_d, K_{vc}) = \int_0^{t_s} \delta^2(t) e^t dt, \tag{15}$$

where t_s is a given settling time and $\delta(t)$ is one of the output of the dynamic simulator (see Fig. 12 (a)).
The NN replaces the MPID control and dynamic simulator and bring out the relation between the input to the simulator, control gains and the criterion function. Based on this relation we can get the optimal vibration gain K_{vc} for any combination of simulator input and PD joint gains K_p, K_d.
However the procedure 5 (dynamic simulation) requires high computational cost and procedure 5 is iterated plenty of times. Consequently it is difficult to find an optimal gain K_{vc} on-line.
Therefore we propose to replace the blocks enclosed by a dashed rectangle in Fig. 12 (a) by a NN model illustrated in Fig. 12 (b). By this way the input to the NN is the simulation



(a) Concept behind finding optimal gain K_{vc}.


Tip payload Mt
Vibration control gain Kvc
Joint control gains Kp,Kd
Reference θref
NN model
Criterion function (10)
Finding the optimal gain Kvc
Optimal Kvc

(b) Finding optimal gain K_{vc} using a NN model.

Fig. 12. Finding optimal gain K_{vc}.

condition, $\theta_{ref}, M_t, K_p, K_d, K_{vc}$ while the output is the criterion function defined in (15). The mapping from the input to the output is many-to-one.

5.3 A NN Model to Simulate Dynamic of A Flexible Manipulator

The NN structure generally consists of input layer, output layer and hidden layer(s). The number of hidden layer is depending on the application such as classification, prediction or modelling and on the complexity of the problem. One of the most important problems of the NN is the determination of the number of neurons in the hidden layer(s). If an inadequate number of neurons are used, the network will be unable to model complex function, and the resulting fit will not be satisfactory. If too many neurons are used, the training time may become excessively long, and, if the worst comes, the network may over fit the data. When over fitting occurs, the network will begin to model random noise in the data. The result of the over fitting is that the model fits the training data well, but it is failed to be generalized for new and untrained data. The over fitting should be examined (Principe et al., 2000). The proposed NN structure is shown in Fig. 9. The NN includes one input layer, one output layer and two hidden layers. In the designed NN the input layer contains five inputs: $\theta_{ref}, M_t, K_p, K_d, K_{vc}$ (see also Fig. 12). Those inputs represent the manipulator configuration, environment variable and controller gains. The output layer consists of one output which is the criterion function, $\Sigma\delta^2 e^t$ and a bias transfer function on the neuron of this layer. The first hidden layer consists of five neurons and the second hidden layer consists of seven neurons. For the transfer function used in the neurons of the two hidden layers a tanh function is used.
The mathematical equation of the tanh function is give by:

$$f(x_i, w_i) = \frac{2}{1 + \exp(-2x_i^{bias})} - 1, \tag{16}$$

where x_i is the ith input to the neuron, w_i is the weight for the input x_i and $x_i^{bias} = x_i + w_i$. After the NN is structured, it is trained using a various examples to generate the correct weights to be used in producing the data in the operating stage.
The main task of the NN is to represent the relation between the input parameters to the simulator, MPID gains and the criterion function.

6. Learning and Training

The training for the NN is analogous to the learning process of the human. As human starts in the learning process to find the relationship between the input and outputs. The NN does the same activity in the training phase.
The block diagram which represents the system during the training process is shown in Fig. 13.

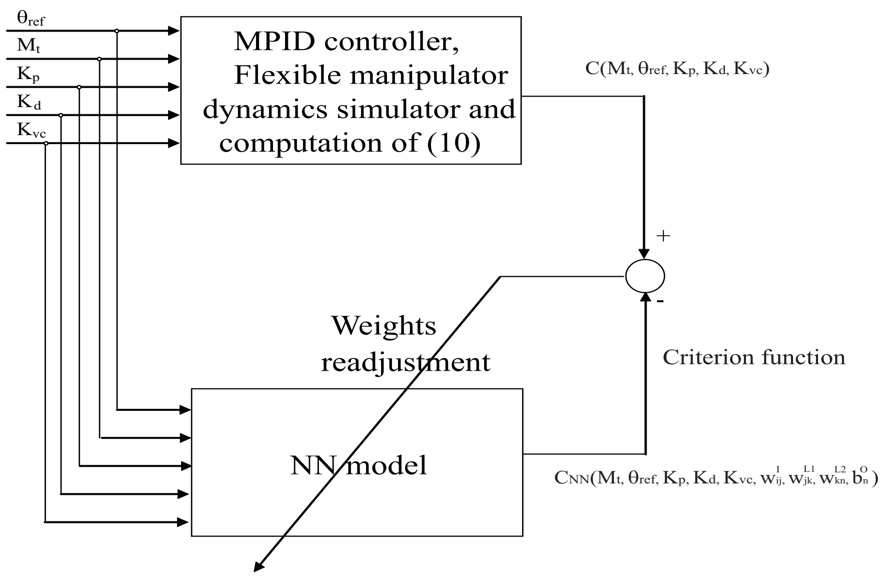

θref
Mt
Kp
Kd
Kvc
MPID controller, Flexible manipulator dynamics simulator and computation of (10)
C(Mt, θref, Kp, Kd, Kvc)
+
-
Weights readjustment
Criterion function
NN model
CNN(Mt, θref, Kp, Kd, Kvc, wIij, wL1jk, wL2kn, bOn)

Fig. 13. Block diagram for the training the NN.

After the NN is constructed by choosing the number of layers, the number of neurons in each layer and the shape of transfer function in each neuron, the actual learning of NN starts by giving the NN teacher signals. In order to train the NN, the results of the dynamic simulator for given conditions are used as teacher signals. In this shadow the feed-forward NN can be used as a mapping between $\theta_{ref}, M_t, K_p, K_d, K_{vc}$ and the output response all over the time span which is calculated by (15).
For the NN illustrated in Fig. 9, the output can be written as

$$\text{Output} = C_{NN}(M_t, \theta_{ref}, K_p, K_d, K_{vc}, w_{ij}^I, w_{jk}^{L1}, w_{k1}^{L2}, b_1^O), \tag{17}$$

where w_{ij}^I is the weight from element i $(i = 1 \sim 5)$ in input layer (I) to element j $(j = 1 \sim 5)$in next layer $(L1)$. w_{jk}^{L1} is the weight from element j $(j = 1 \sim 5)$ in first hidden layer $(L1)$ to element k $(k = 1 \sim 7)$ in next layer $(L2)$. w_{k1}^{L2} is the weight from element k $(k = 1 \sim 7)$ in second hidden layer $(L2)$ to element n in output layer (O). b_1^O is the bias of the output layer. The NN begins to adjust the weights is each layer to achieve the desired output.

Herein, the performance surface $E(w)$ is defined as follows:

$$E(w) = (C(M_t, \theta_{ref}, K_p, K_d, K_{vc}) - C_{NN}(M_t, \theta_{ref}, K_p, K_d, K_{vc}))^2. \tag{18}$$

The conjugate gradient method is applied to readjustment of the weights in NN. The principle of the conjugate gradient method is shown in Fig. 14.

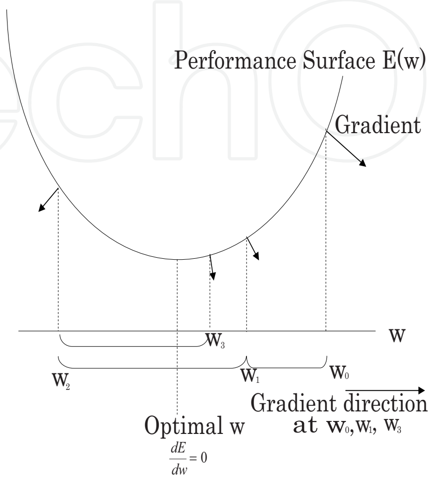


Fig. 14. Conjugate gradient for minimizing error.

By always updating the weights in a direction that is conjugate to all past movements in the gradient, all of the zigzagging of 1st order gradient descent methods can be avoided. At each step, a new conjugate direction is determined and then move to the minimum error along this direction. Then a new conjugate direction is computed and so on. If the performance surface is quadratic, information from the Hessian can determine the exact position of the minimum along each direction, but for non quadratic surfaces, a line search is typically used. The equations which represent the conjugate gradient method are:

$$\Delta \boldsymbol{w} = \alpha(n)\boldsymbol{p}(n), \tag{19}$$

$$\boldsymbol{p}(n+1) = -\boldsymbol{G}(n+1) + \beta(n)\boldsymbol{p}(n), \tag{20}$$

$$\beta(n) = \frac{\boldsymbol{G}^T(n+1)\boldsymbol{G}(n+1)}{\boldsymbol{G}^T(n)\boldsymbol{G}(n)}, \tag{21}$$

where $\boldsymbol{w}$ is a weight, $\boldsymbol{p}$ is the current direction of weight movement, α is the step size, $\boldsymbol{G}$ is the gradient (back propagation information) and β is a parameter that determines how much of the past direction is mixed with the gradient to form the new conjugate direction. And as a start for the searching we put $\boldsymbol{p}(0) = -\boldsymbol{G}(0)$. The equation for α in case of line search to find the minimum mean squared error (MSE) along the direction $\boldsymbol{p}$ is given by:

$$\alpha = \frac{-\boldsymbol{G}^T(n)\boldsymbol{p}(n)}{\boldsymbol{p}^T(n)\boldsymbol{H}(n)\boldsymbol{p}(n)}, \tag{22}$$

where $\boldsymbol{H}$ is the Hessian matrix. The line search in the conjugate gradient method is critical for finding the right direction to move next. If the line search is inaccurate, then the algorithm

may become brittle. This means that we may have to spend up to 30 iterations to find the appropriate step size.
The scaled conjugate is more appropriate for NN implementations. One of the main advantages of the scaled conjugate gradient (SCG) algorithm is that it has no real parameters. The algorithm is based on computing $\boldsymbol{H}\boldsymbol{d}$ where $\boldsymbol{d}$ is a vector. It uses equation (22) and avoids the problem of non-quadratic surfaces by manipulating the Hessian so as to guarantee positive definiteness, which is accomplished by $\boldsymbol{H} + \lambda \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. In this case α is computed by:

$$\alpha = \frac{-\boldsymbol{G}^T(n)\boldsymbol{p}(n)}{\boldsymbol{p}^T(n)\boldsymbol{H}(n)\boldsymbol{p}(n) + \lambda \mid \boldsymbol{p}(n) \mid^2}, \tag{23}$$

instead of using (22). The optimization function in the NN learning process is used in the mapping between the input to the simulator and the output criterion function not in the optimization of the vibration gain.

6.1 Training result

The SCG is chosen as the learning algorithm for the NN. Once the algorithm for the learning process is selected, the NN is trained on the patterns. The result of the learning process is shown in this subsection. The teacher signals (training data set) are generated by the simulation system illustrated in Fig. 12 (a). The examples of the training data set are listed in Table 1. 220 data sets are used for the training. The data is put in a scattered order to allow the NN to get the relation in a correct manner.

Pattern	θ_{ref}	M_t	K_p	K_d	K_{vc}	$\Sigma\delta^2 e^t$
1	5	0.5	300	100	20000	0.0129
2	15	0.25	800	300	80000	7.242
3	10	0.25	600	200	0	1.21
4	25	0.5	600	200	10000	0.1825
5	25	0.5	600	200	10000	0.1825
6	15	0.25	600	150	70000	4.56
...	...	...	...	...	...	...

Table 1. Sample of NN training patterns.

As shown in Fig. 15, two curves are drawn relating the value of the normalized criterion for each example used in the training. The normalized the criterion function $C(M_t, \theta_{ref}, K_p, K_d, K_{vc}$ obtained from the simulation is plotted in circles while the normalized criterion function $C_{NN}(M_t, \theta_{ref}, K_p, K_d, K_{vc})$ generated by the NN in the training process is plotted in cross marks. The results of Fig. 15 show that training of the NN enhance its ability to follow up the output from the simulation. A performance measure is used to evaluate whether the training of the NN is completed. In this measurement, the normalized mean squared error (NMSE) between the two datasets (i. e. the dataset the NN trained on and the dataset the NN generate) is calculated. For this case NMSE is 0.0054. Another performance index is also used which is the correlation coefficient r between the two datasets. The correlation coefficient r is 0.9973. When a test is done for the trained NN upon a complete new set of data the NMSE is 0.0956 and r is 0.9664.

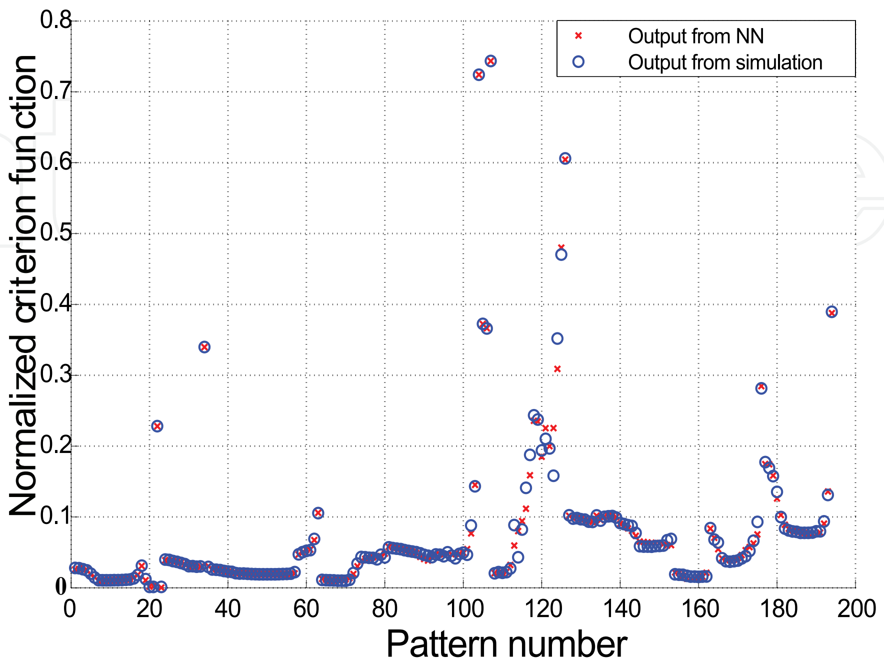


Fig. 15. NN training.

7. Optimization result

In this section, the results obtained using the simulation are compared with the results obtained using the NN. The criterion function C computed by (15) and the output of NN, C_{NN}, for the vibration control gain K_{vc} are plotted in Fig. 16. Comparing the results obtaind using the NN for the criterion function with the results obtained using dynamic simulator in Fig. 16. shows good coincidence. This means that the NN network can successfully replace the dynamic simulator to find how the criterion function changes with the changing of the system parameters.
Form Fig. 16 the optimum gain K_{vc} can be easily found. One of the main advantages of using the NN to find the optimal gain for the MPID control is the computional speed. To generate the data of the simulation curve, which is indicated by the triangles in Fig. 16, 1738 seconds is needed while only 6 seconds are needed to generate the data using the NN, which is indicated by the circles. The minimum values of the criterion function occurs when the value of the vibration control gain K_{vc} equals 22500 V s/m^2.

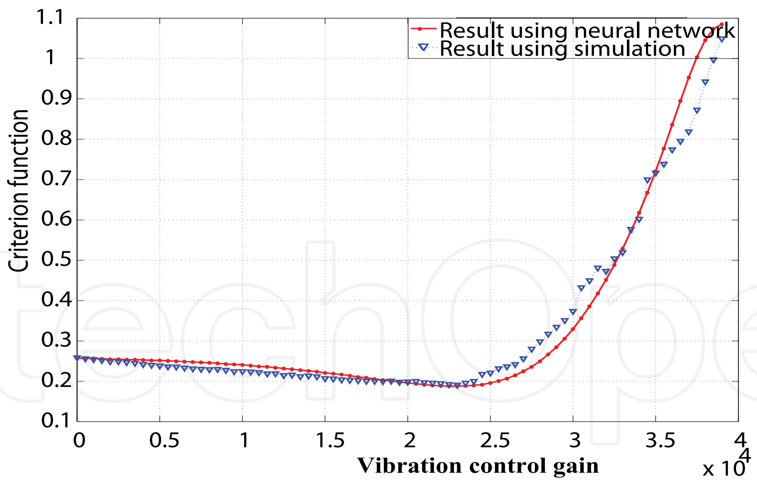


Fig. 16. Vibration control gain vs. criterion function.

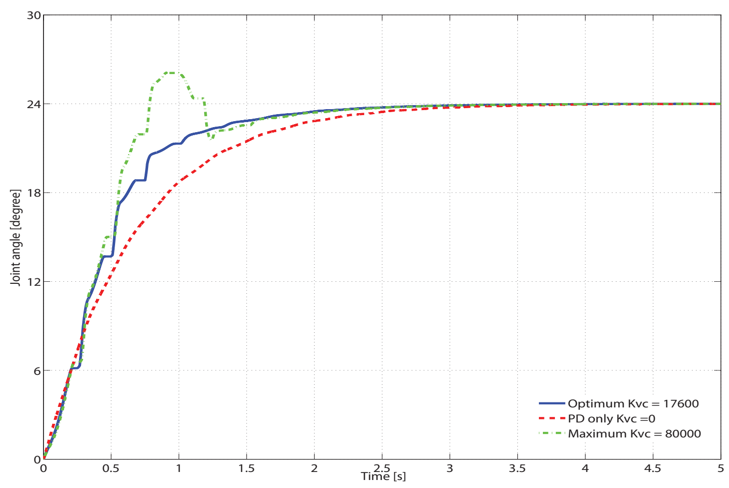


Fig. 17. Response using optimum gain.

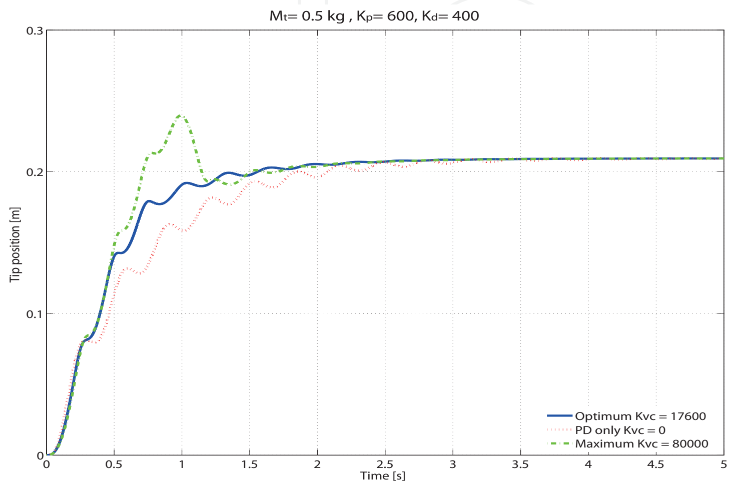


Fig. 18. Response using optimum gain.

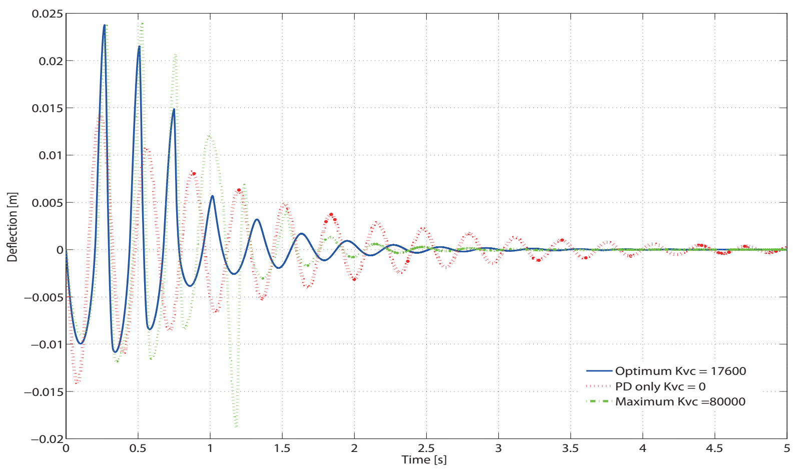


Fig. 19. Response using optimum gain.

The response of the flexible manipulator using the optimal gain K_{vc} is shown in Fig. 17, Fig. 18 and Fig. 19. 0.5 kg is used as a tip payload M_t with 24 degree for the joint reference angle θ_{ref}. For the controller described by equation (10), the values of K_p and K_d are set at 600 V rad/m and 400 V s rad/m respectively. The response with different vibration gains K_{vc} is plotted. In the beginning the response with PD control only (i.e. $K_{vc} = 0$) is plotted in dash line while the response with the maximum K_{vc} which is 80000 V s/m^2 is plotted in a dash-dot line. The response with the optimum K_{vc} -which was tuned using NN- appears in a continous line. The value of the optimum vibration control gain K_{vc} is 17600 V s/m^2. Increasing the vibration control gain K_{vc} leads the system to have fast response for the joint position as shown in Fig. 17 but more increasing in the value of the vibration control gain leads to an undesirable overshoot as shown in Fig. 18 with a dash-dot line. To focus on the effect of the vibration gain on the end-effector vibration Fig. 19 is plotted. It is clear from the figure that the optimum vibration control gain for the MPID succeed to suppress the vibration at the end of the flexible manipulator.

8. Conclusions

This chapter discusses a NN based gain tuning method for the vibration control PID (MPID) controller of a single-link flexible manipulator. The NN is trained to simulate the dynamics of the single-link flexible manipulator and to produce the integral of the squared tip deflection weighted by exponential function. A dynamic simulator is used to produce the teacher signals.
The main advantage of using NN to find an optimal gain is the computational speed. The NN based method is approximately 290 times faster than the dynamic simulation based method. Simulation results with the obtained optimal gain validate the proposed method.

9. References

Cannon, R. H. Jr. & Schmitz, E. (1984). Initial Experiments on the End Point Control of a Flexible One-Link Robot. *International Journal of Robotics Research*, Vol. 3, No. 4, pp. 62–75.

Ge, S. S.; Lee, T. H. & Gong, J. Q. (1999). A Robust Distributed Controller of a Single-Link SCARA /Cartesian Smart Materials Robot, *Mechatronics*, Vol. 9, No. 1, 1999, pp. 65–93.

Sun, D.; Shan J.; Su, Y.; Liu H. & Lam, C. (2005). Hybrid Control of a Rotational Flexible Beam Using Enhanced PD Feedback with a Non-Linear Differentiator and PZT Actuators, *Smart Mater. Struct.*, Vol. 14, pp 69–78.

Etxebarria, V.; Sanz, A. & Lizarraga, I. (2005). Control of a Lightweight Flexible Robotic Arm Using Sliding Modes, *International Journal of Advanced Robotic Systems*, Vol. 2, No. 2, pp. 103–110.

Lee,H. G.; Arimoto, S., & Miyazaki, F. (1988). Liapunov Stability Analysis for PDS Control of Flexible Multi-link Manipulators, *Proceeding of the Conference on Decision and Control, Austin*, pp. 75–80.

Maruyama, T.; Xu, C.; Ming, A. & Shimojo, M. (2006). Motion Control of Ultra-High-Speed Manipulator with a Flexible Link Based on Dynamically Coupled Driving, *Joural of Robotics and Mechatronics*, Vol. 18, No. 5, pp. 598–607.

Matsuno, F. & Hayashi, A. (2000). PDS Cooperative Control of Two One-link Flexible Arms, *Proceeding of the 2000 IEEE International Conference on Robotics and Automation, San Francisco*, pp. 1490–1495.

Talebi, H. A.; Khorasani, K.,& Patel, R. V. (1998).Neural Network Based Control Schemes for Flexible Link Manipulators: Simulations and Experiments, *Neural Networks*, Vol. 11, pp. 1357–1377.

Kawato, M.; Furukawa, K. & Suzuki, R. (1987). A Hierarchical Neural Network Model for Control and Learning of Voluntary Movement, *Biological Cybernetics*, Vol. 57, pp. 169–185.

Isogai, M.; Arai, F. & Fukuda, T. (1999). Intelligent Sensor Fault Detection of Vibration Control for Flexible Structures, *Joural of Robotics and Mechatronics*, Vol. 11, No. 6, pp. 524–530.

Lianfang, T.; Wang, J., & Mao, Z. (2004). Constrained Motion Control of Flexible Robot Manipulators Based on Recurrent Neural Networks, *IEEE Transactions On Systems, Man, And Cybernetics Part B: Cybernetics*, Vol. 34, No. 3, pp. 1541–1552.

Cheng, X. P. & Patel, R. V. (2003). Neural Network Based Tracking Control of a Flexible MacroŰMicro Manipulator System, *Neural Networks*, Vol. 16, pp. 271–286.

Yazdizadeh, A.; Khorasani, K. & Patel, R.V. (2000). Identification of a Two-Link Flexible Manipulator Using Adaptive Time Delay Neural Networks, *IEEE Transactions On Systems, Man, And Cybernetics Part B: Cybernetics*, Vol. 30, No. 1, pp. 165–172.

Ge, S. S.; Lee, T. H. & Zhu, G. (1996). Genetic Algorithm Tuning of Lyapunov-Based Controllers" An Application to a Single-Link Flexible Robot System, *IEEE Transactions On Industrial Electronics*, Vol. 43, No. 5, pp. 567–573.

Principe, J.; Euliano, N. & Lefebvre, W. (2000). Neural and Adaptive Systems: Fundamentals Through Simulations, John Wiley and Sons, New York, pp. 100–172.

Mansour, T.; Konno, A. & Uchiyama, M. (2008). Modified PID Control of a Single-Link Flexible Robot, *Advanced Robotics*, Vol. 22, pp. 433–449.