Noise analysis of robot manipulator using neural networks

Śahin Yildirim*, Ikbal Eski

Erciyes University, Faculty of Engineering, Mechanical Engineering Department, Kayseri 38039, Turkey

Abstract

Due to a lot of robot manipulators application in industry, low noise degree is very important criteria for robot manipulator’s joints. In this paper, joint noise problem of a robot manipulator with five joints is investigated both theoretically and experimentally. The investigation is consisted of two steps. First step is to analyze the noise of joints using a hardware and software. The hardware is a part of noise sensors. The second step; according to experimental results, some neural networks are employed for finding robust neural noise analyzer. Five types of neural networks are used to compare each other. From the results, it is noted that the proposed RBFNN gives the best results for analyzing joint noise of the robot manipulator.

1. Introduction

Neural network as a predictor has been widely used in the process and robotic handling industries because it is robust. Neural networks have been employed in various ways to construct nonlinear systems to handle a variety of control objectives. Typically, the key elements in their design procedure are the proper choice of the functions to be approximated and an appropriate control strategy that utilizes this approximation. However, these strategies usually require the neural network-based approximation to be sufficiently accurate.

Robot manipulators have become increasingly important in the field of flexible automation and manufacturing system in industrial applications such as material handling [1]. Programmability, high speed and high precision are some of the good qualities they possess which are desirable for flexible automation. Vibration analysis of robot manipulators has been investigated by several researchers. Cheong and Youm [2] have proposed the system mode analysis of horizontal vibration for 3-D two-link flexible manipulators. For the analysis, the authors have formulated and solved a set of partial differential equations which represent vibration mixed with bending and torsional moment. In their research, an examination and comparison between the two joint conditions have been performed. Numerical and experimental tests have showed the validity and effectiveness of the proposed analysis and modeling. Cheong et al. [3] have suggested the accessibility of horizontal vibration in a 3-D two-link flexible robot shows configuration-dependent nature. The analysis of horizontal vibration based on system mode approach takes an important role in examining the vibration accessibility. Both theoretical and numerical studies were presented to elucidate the meaning of the accessibility and the identifiability of horizontal vibration. In addition, the experimental results support the theoretical results.

An effect of flexibility on the trajectory of a planar two-link manipulator has studied using integrated computer-aided design/analysis (CAD/CAE) procedures [4]. Chu et al. [5] have proposed a new type of stepping piezoelectric micro-motor using wiggle vibration mode and three-dimensional contacts between the rotor and stator. A hybrid input shaping method to reduce the residual swing of a simply suspended object transported by a robot manipulator or the residual vibration of equivalent dynamic systems has been studied by Kapucu et al. [6].

A 3-PPR planar parallel manipulator, which consists of three active prismatic joints, three passive prismatic joints, and three passive rotational joints, has been proposed [7]. The analysis of the kinematics and the optimal design of the manipulator have also been discussed and an example using the optimal design was also presented.

A neural network has employed to analyze of inverse kinematics of PUMA 560 type robot [8]. Callegari et al. [9] have presented the inverse dynamics model of a novel translating parallel machine and proposes the structure of a force controller for the execution of tasks characterized by interaction with the environment. The effect of miniaturization on two dynamic and two quasi-static motion mechanisms has been experimentally evaluated using a miniature piezoceramic drive unit [10]. A new approach to the dynamic motion planning problems of mobile robots in uncertain dynamic environments based on the behavior dynamics from a control point of view was provided [11]. The
Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_j(t))</td>
<td>the output of competitive layer</td>
</tr>
<tr>
<td>(b_j)</td>
<td>the bias of the (j)th neuron in the hidden layer</td>
</tr>
<tr>
<td>(b_k)</td>
<td>the bias of the (k)th neuron in the output layer</td>
</tr>
<tr>
<td>(c)</td>
<td>the Coulomb friction constant</td>
</tr>
<tr>
<td>(E_1(t))</td>
<td>the error between theoretically and neural network output signals</td>
</tr>
<tr>
<td>(E_2(t))</td>
<td>the propagation error between hidden layer and output layer</td>
</tr>
<tr>
<td>(f)</td>
<td>the input vector</td>
</tr>
<tr>
<td>(F)</td>
<td>the force-torque vector</td>
</tr>
<tr>
<td>(g(\cdot))</td>
<td>activation function</td>
</tr>
<tr>
<td>(g_b)</td>
<td>the normalized output vector elements of the gating network</td>
</tr>
<tr>
<td>(G_0(\Theta))</td>
<td>the vector of gravity terms</td>
</tr>
<tr>
<td>(G_i(\Theta))</td>
<td>the vector of gravity terms in Cartesian space</td>
</tr>
<tr>
<td>(J(\Theta))</td>
<td>the Jacobian matrix</td>
</tr>
<tr>
<td>(K)</td>
<td>the number of local networks</td>
</tr>
<tr>
<td>(M(\Theta))</td>
<td>the mass matrix of the manipulator</td>
</tr>
<tr>
<td>(M_d(\Theta))</td>
<td>the Cartesian mass matrix</td>
</tr>
<tr>
<td>(N)</td>
<td>training numbers</td>
</tr>
<tr>
<td>(n)</td>
<td>number of joints</td>
</tr>
<tr>
<td>(N_f)</td>
<td>the experimentally obtained number of cycles to failure</td>
</tr>
<tr>
<td>(N_{fn})</td>
<td>the corresponding neural network</td>
</tr>
<tr>
<td>(n_i)</td>
<td>number of neurons in the input layer</td>
</tr>
<tr>
<td>(n_H)</td>
<td>number of neurons in the hidden layer</td>
</tr>
<tr>
<td>(n_O)</td>
<td>number of neurons in the output layer</td>
</tr>
<tr>
<td>(r_j)</td>
<td>the radius vector of the (j)th hidden unit</td>
</tr>
<tr>
<td>(T)</td>
<td>the target values</td>
</tr>
<tr>
<td>(\nu)</td>
<td>the viscous friction constant</td>
</tr>
<tr>
<td>(V(\Theta, \Theta))</td>
<td>the vector of Coriolis terms</td>
</tr>
<tr>
<td>(V_i(\Theta, \Theta))</td>
<td>the vector of velocity terms in Cartesian space</td>
</tr>
<tr>
<td>(y_j)</td>
<td>the output of the (j)th local network</td>
</tr>
<tr>
<td>(W_{ij})</td>
<td>the weights between (i)th neuron in the hidden layer and (k)th neuron in the output layer</td>
</tr>
<tr>
<td>(W_{ij})</td>
<td>the weights between (i)th neuron in the input layer and (j)th neuron in the hidden layer</td>
</tr>
<tr>
<td>(X)</td>
<td>the appropriate Cartesian vector representing position and orientation of the end-effector</td>
</tr>
<tr>
<td>(\eta)</td>
<td>learning rate</td>
</tr>
<tr>
<td>(\zeta)</td>
<td>momentum term</td>
</tr>
<tr>
<td>(\Delta W_{ij}(t))</td>
<td>the weight variations matrices from (i) layer to (j) layer after update</td>
</tr>
<tr>
<td>(\Delta W_{jk}(t))</td>
<td>the weight variations matrices from (j) layer to (k) layer after update</td>
</tr>
</tbody>
</table>

The purpose of this paper is to obtain a neural network vibration analyzer and develop a robust neural network model for robot manipulator’s end-effector under big payload conditions.

This paper is organized as follows: Section 2 describes the theory of robot manipulators. Section 3 gives some details about neural networks and the proposed neural network. The experimental results and discussions are given in Section 4. Finally, this paper is concluded with conclusions and discussion.

2. Robot manipulator

In this work, five degrees of freedom an industrial robot manipulator is employed for analyzing vibration parameters of joints are shown in Figs. 1 and 2. The robot manipulator’s joints are driven by servo motors for that reason; maximum speed of robot manipulator’s end-effector is 2200 mm/s (see Table 1). The positioning repetition accuracy of the robot manipulator is ±0.02 mm. The maximum reach ability of the robot manipulator’s end-effector is 410 mm. When the Newton–Euler equations are evaluated for five-link robot manipulator, dynamic equation can be written in the following form:

\[
\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) \tag{1}
\]

where \(M(\Theta)\) is the \(n \times n\) mass matrix of the manipulator, \(V(\Theta, \dot{\Theta})\) is the \(n \times 1\) vector of centrifugal and Coriolis terms, \(G(\Theta)\) is an \(n \times 1\) vector of gravity terms (\(n=5\)). Each element of \(M(\Theta)\) and \(G(\Theta)\) is a complex function which depends on \(\Theta\), the position of all the joints of the manipulator. Each element of \(V(\Theta, \dot{\Theta})\) is a complex function of both \(\Theta\) and \(\dot{\Theta}\).

The dynamics of the robot manipulator with respect to Cartesian variables can be expressed in the following form:

\[
F = M_d(\Theta)\dot{X} + V_d(\Theta, \dot{\Theta}) + G_d(\Theta) \tag{2}
\]

where \(F\) is a force-torque vector acting on the end-effector of the robot, and \(X\) is an appropriate Cartesian vector representing position and orientation of the end-effector. Analogous to the joint space quantities, \(M_d(\Theta)\) is the Cartesian mass matrix, \(V_d(\Theta, \dot{\Theta})\) is a vector of velocity terms in Cartesian space, and \(G_d(\Theta)\) is a vector of gravity terms in Cartesian space. Note that the fictitious forces acting on the end-effector, \(F\), could in fact be...
applied by the actuators at the joints using the relationship
\[ \tau = J'(\Theta)F \] (3)
where the Jacobian, \( J(\Theta) \), is written in the same frame as \( F \) and \( \dot{X} \), usually the tool frame, \( \{T\} \). The inverse of the Jacobian transpose to obtain
\[ J'^{-1}\tau = J'^{-1}M(\Theta)\dot{\Theta} + J'^{-1}V(\Theta, \dot{\Theta}) + J'^{-1}G(\Theta) \] (4)
or
\[ F = J'^{-1}M(\Theta)\dot{\Theta} + J'^{-1}V(\Theta, \dot{\Theta}) + J'^{-1}G(\Theta) \] (5)
The relationship between joint space and Cartesian acceleration has been developed and starting with the definition of the Jacobian,
\[ \ddot{X} = J\ddot{\Theta} \] (6)
and differentiating to obtain
\[ \ddot{X} = J\ddot{\Theta} \] (7)
Solving (7) for joint space acceleration leads to
\[ \dot{\Theta} = J^{-1}\ddot{X} - J^{-1}J\dot{\Theta} \] (8)
Substituting (8) into (5)
\[ F = J'^{-1}M(\Theta)J^{-1}\ddot{X} - J'^{-1}M(\Theta)J^{-1}J\dot{\Theta} + J'^{-1}V(\Theta, \dot{\Theta}) + J'^{-1}G(\Theta) \] (9)
from which the expressions for the terms in the Cartesian dynamics has been derived,
\[ M_x(\Theta) = J^{-1}(\Theta)M(\Theta)J^{-1}(\Theta) \]
\[ V_x(\Theta, \dot{\Theta}) = J^{-1}(\Theta)[V(\Theta, \dot{\Theta}) - M(\Theta)J^{-1}(\Theta)J(\Theta)\dot{\Theta}] \] (10)
\[ G_x(\Theta) = J^{-1}(\Theta)G(\Theta) \]
where \( c \) is Coulomb friction constant. The value of \( c \) is often taken at one value when \( \theta = 0 \), the static coefficient and at a lower value, the dynamic coefficient, when \( \theta \neq 0 \). Whether a joint of a particular manipulator exhibits viscous or Coulomb friction are a complicated issue of lubrication and other effects. A reasonable model is to include both, since both effects are likely:

\[
\tau_{\text{friction}} = c \text{sgn}(\dot{\theta}) + \nu \dot{\theta}
\]  

It turns out that in many manipulator joints, friction also displays a dependence on the joint position. A major cause of this effect might be gears which are not perfectly round (their eccentricity would cause friction to change according to joint position). So a fairly complex friction model would have the form

\[
\tau_{\text{friction}} = f(\theta, \dot{\theta})
\]  

These friction models are then added to the other dynamic terms derived from the rigid body model, yielding the more complete model

\[
\tau_n = M_n(\theta)\ddot{\theta} + V_n(\theta, \dot{\theta}) + G_n(\theta) + F_n(\theta, \dot{\theta})
\]  

2.1. Structure of the robot controller

Robot control process is the only process directly influencing the manipulator. It main task is to compute the set values for the five servo controllers. Currently the only influence on the dynamics of the system can be exerted by shaping the set values to the axis servo controllers. The set values can be computed from a prescribed trajectory expressed in Cartesian space, joint space or in motor increment space. In the case of Cartesian-Euler space, axially-symmetric tools are assumed, as the robot has only five DOF. The trajectories can also be computed from or modified according to virtual sensor readings. The experimental setup is depicted in Fig. 3.

The hardware of the system consists of a 32-bit Intel P-D microprocessor-based computer and five 8-bit Intel 8080 microprocessors. Each of the 8-bit microprocessor-based servo-drives MA-70 controls a DC electric motor actuating one of the five DOF manipulator axes (see Fig. 4). The P4 processor is interfaced directly to the servo-drives. It transmits position increments to each one at them and receives feedback signals from them. The P4

![Fig. 3. The setup used for the experiments and a schematic diagram.](image-url)
processors bus is connected by a 16-bit parallel 800 kbaund interface to the five servo motor controllers and I/O board 2A-RZ365/2A-RZ375. The controller hardware structure of the system is shown in Fig. 5.

3. Neural networks

Neural networks (NN) are made up of simple, highly interconnected processing units called neurons each of which performs two functions: aggregation of its inputs from other neurons or the external environment and generation of an output from aggregated inputs [13]. NN can be classified into two main categories based on their connection structures: feedforward and recurrent networks. Feedforward networks are the most commonly used type, mainly because of difficulty of training recurrent networks, although the last mentioned are more suitable for representing dynamical systems [14,15].

In this experimental investigation, feedforward neural network type is used to noise analyze in a robot manipulator.
where $j$ is the output of the $j$th neuron in the hidden layer, $W_0$ is the weight of the connection between the input layer neurons and the hidden layer neurons $b_j$ is the bias of the $j$th neuron in the hidden layer. $b_j$ can be regarded as the weight of the connection between a fixed input of unit value and neuron $j$ in the hidden layer. The function $g(\cdot)$ is called the activation function of the hidden layer. The output signal of the network can be expressed in the following form:

$$y_k = g \left( \sum_{j=1}^{10} \sum_{l=1}^{10} W_{jk} z_j + b_k \right)$$

where $W_{jk}$ are the weights between $j$th neurons hidden layer and $k$th neurons output layer and $b_k$ is the bias of the $k$th neurons in the output layer. Some supervised learning methods which are used to noise analyze in a robot manipulator, briefly described in the following subsections.

The application of NN typically comprises two phases: a learning phase and a testing phase. Learning is the process through which the parameters and structure of the network are adjusted to reflect the knowledge contained within the distributed network structure. A trained network subsequently represents a static knowledge base which can be recalled during its operation phase. There are three general learning schemes in neural networks:

- Supervised learning, for example, error back propagation which requires the correct output signal for each input vector to be specified.
- Unsupervised, competitive learning or self-organizing, in which the network self-adjusts its parameters and structure to capture the regularities of input vectors, without receiving explicit information from the external environment.
- Reinforcement or graded learning in which the network receives implicit scalar evaluations of previous inputs.

### 3.1. Back propagation neural network (BPNN)

The BPNN is a method of supervised neural network learning. During training, the network is presented with a large number of input patterns. The experimental outputs are then compared to the neural network output nodes. The error between the experimental and neural network response is used to update the weights of the network inter-connections. This update is performed after each pattern presentation. One run through the entire pattern set is termed an epoch. The training process continues for multiple epochs, until a satisfactorily small error is produced. The test phase uses a different set if input patterns. The neural network outputs are again compared to a set of experimental outputs. This error is used to evaluate the networks ability to generalize. Usually, the training set and/or architecture needs to be assessed at this point. The BPNN is the most commonly used to update the weights of the neural networks. The weights between input layer and the hidden layer are updated as follows:

$$\Delta W_{jk}(t) = -\eta \frac{\partial E_t(t)}{\partial W_{jk}(t)} + \alpha \Delta W_{jk}(t-1)$$

where $\eta$ is the learning rate, and $\alpha$ is the momentum term. $E_t(t)$ is the propagation error between hidden layer and output layer. $E_t(t)$ is the error between experimental and neural network output signals.

### 3.2. General regression neural network (GRNN)

GRNN are paradigms of the Probabilistic and Radial Basis Function used in functional approximation. To apply GRNN to analyze, an input vector $f$ is formed. The output $y$ is the weighted average of the target values $t_i$ of training cases $f$, close to a given input case $f$, as given by

$$y_k = \frac{\sum_{i=1}^{10} \frac{1}{\sum_{j=1}^{10} \sum_{l=1}^{10} W_{jk} z_j W_{ij}} W_{ik}}{\sum_{j=1}^{10} \sum_{l=1}^{10} W_{jk}}$$
where

\[ W_{jk} = \exp \left( -\frac{|f_j - f|^2}{2\sigma_j^2} \right) \]

The only weights that need to be learned are the smoothing parameters, \( h \) of the RBF units in Eq. (21), which are set using a simple grid search method. The distance between the computed value \( n \) and each value in the set of target values \( T \) is given by

\[ T = \{1, 2\} \quad (21) \]

The values 1 and 2 correspond to the training class and all other classes, respectively. The class corresponding to the target value with the minimum distance is chosen. As, GRNN exhibits a strong bias towards the target value nearest to the mean value \( \mu \) of \( T \) so we used target values 1 and 2 because both have the same absolute distance from \( \mu \).

3.3. Learning vector quantization neural network (LVQNN)

LVQNN is supervised neural network, which was developed by Kohonen and is based on the Self-Organizing Map (SOM) or Kohonen feature map, LVQNN methods are simple and effective adaptive learning techniques. They rely on the nearest neighbor classification model and are strongly related to the condensing methods, where only a reduced number of prototypes are kept from the whole set of samples. This condensed set of prototypes is then used to classify unknown samples using the nearest neighbor rule. LVQNN has a competitive and linear layer in the first and second layer, respectively. The competitive layer learns to classify the input vectors and the linear layer transforms the competitive layer’s classes into the target classes defined by the user. In the learning process, the weights of LVQNN are updated by the following Kohonen learning rule if the input vector belongs to the same category

\[ \Delta W_{ij} = \eta \delta_{ij}(y_{ij} - W_{ij}) \quad (22) \]

where \( \eta \) is the learning ratio and \( a_{ij} \) is the output of competitive layer.

3.4. Modular neural network (MNN)

MNN refer to the adaptive mixtures of local experts. The most attractive point of MNN architecture is that different experts handle different parts of the data. The problem of pattern interference can be alleviated by this type of architecture. Each local network receives the same input vector applied to the network. A gating network determines the contribution of each local network to the total output as well as what range of the input space each network should learn. The outputs of the local networks are combined to give the network output such that:

\[ y_k = \sum_{j=1}^{K} g_j y_j \quad (23) \]

where \( K \) is the number of local networks, \( y_j \) is the output of the \( j \)th local network and \( g_j \) are the normalized output vector elements of the gating network given by

\[ g_j = \frac{e^{um}}{\sum_{m=1}^{M} e^{um}} \quad (24) \]

where \( u_m \) is the weighted input received by the \( m \)th output unit of the gating network. In Eq. (23), \( \sum g_j = 1 \) and \( 0 \leq g_j \leq 1 \). The gating and local experts are trained to maximize the following
function using the learning rule:
\[ y = \ln \left[ \sum_{j=1}^{K} g_j e^{-(1/2)N_f j_N j} \right] \]  

\[(25)\]

where \( N_f \) is the experimentally obtained number of cycles to failure and \( N_j \) is the corresponding output given by the neural network. The neural network simultaneously adjusts the connection and threshold of the local networks as well as the connection weights and thresholds of the gating networks. The error for the \( j \)-th output of the gating network is calculated as:
\[ \frac{\partial y}{\partial y_j} = h_j - g_j \]  

\[(26)\]

where \( h_j \) is a posteriori probability that the \( j \)-th local network is responsible for the current output such that
\[ h_j = \frac{\sum_{j=1}^{K} g_j e^{-(1/2)N_f j_N j}}{\sum_{j=1}^{K} g_j e^{-(1/2)N_f j_N j}} \]  

\[(27)\]

while the error for the \( j \)-th local network is calculated as:
\[ \frac{\partial y}{\partial y_j} = (N_f - y_j) \frac{\partial y_j}{\partial y_j} \]  

\[(28)\]

The training process is terminated either when the mean square error between the observed data and the NN outcomes for all elements in the training set has reached a pre-specified threshold or after the completion of a pre-specified number of learning epochs.

### 3.5. Radial basis function neural network (RBFNN)

Traditionally, RBFNN which model functions \( y(x) \) mapping \( x \in \mathbb{R}^d \) to \( y \in \mathbb{R} \) have a single hidden layer so that the model,
\[ f(x) = \sum_{j=1}^{m} w_j h_j(x) \]  

\[(29)\]

is linear in the hidden layer to output weight \( (w_j)^m_{j=1} \). The characteristic feature of RBFNN is the radial nature of the hidden unit transfer functions, \( (h_j)^m_{j=1} \), which depend only on the distance between the input \( x \) and the centre \( c_j \) of each hidden unit, scaled by a metric \( R_j \),
\[ h_j(x) = \phi((x - c_j)^T R_j^{-1}(x - c_j)) \]

where \( \phi \) is some function which is monotonic for non negative numbers. Gaussian basis function so that the transfer functions can be written
\[ h_j(x) = \exp\left(-\sum_{k=1}^{n} \frac{(x_k - c_{jk})^2}{f_k^2}\right) \]  

\[(30)\]

Using the Gaussian approximation, the output network is approximately
\[ f(x) = \sum_{j=1}^{m} w_j \exp\left(-\sum_{k=1}^{n} \frac{(x_k - c_{jk})^2}{f_k^2}\right) \]  

\[(31)\]

where \( f_j \) is the radius vector of the \( j \)-th hidden unit. A direct approach to the model complexity issue is to select a subset of centres from a larger set which, if used in its entirety, would over fit the data.

### Table 2

<table>
<thead>
<tr>
<th>NN Type</th>
<th>Learning rule</th>
<th>( \eta )</th>
<th>( \alpha )</th>
<th>( N )</th>
<th>( n_1 )</th>
<th>( n_2 )</th>
<th>( n_0 )</th>
<th>( g(\cdot) )</th>
<th>RMSEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPNN</td>
<td>( \Delta W_i(t) = -\eta \frac{\partial \text{MSE}}{\partial W_i(t)} + \alpha \Delta W_i(t - 1) )</td>
<td>0.5</td>
<td>0.4</td>
<td>6,500,000</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>tanH</td>
<td>0.2989</td>
</tr>
<tr>
<td>GRNN</td>
<td>( W_i = \exp\left(-\frac{x_i^2}{\eta}\right) )</td>
<td>0.5</td>
<td>0.4</td>
<td>6,500,000</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>tanH</td>
<td>0.078</td>
</tr>
<tr>
<td>LVQNN</td>
<td>( \Delta W_i(j) = \eta \delta(t)(y(t) - W_i(t,j)) )</td>
<td>0.5</td>
<td>0.4</td>
<td>6,500,000</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>tanH</td>
<td>0.6582</td>
</tr>
<tr>
<td>MNN</td>
<td>( y = \ln \left[ \sum_{i=1}^{K} g_i e^{-(1/2)N_f i_N i} \right] )</td>
<td>0.5</td>
<td>0.4</td>
<td>6,500,000</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>tanH</td>
<td>0.6351</td>
</tr>
<tr>
<td>RBFNN</td>
<td>( f(x) = \sum_{j=1}^{m} w_j \exp\left(-\frac{1}{\eta} \sum_{k=1}^{n} (x_k - c_{jk})^2 \right) )</td>
<td>0.5</td>
<td>0.4</td>
<td>6,500,000</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>tanH</td>
<td>0.0053</td>
</tr>
</tbody>
</table>
4. Experimental approach

Fig. 7 shows the results of BP neural network for predicting noise of robot manipulator. From the Fig. 7, this neural network approach does not follow experimental noise vibration. Fig. 8 represents other neural network type noise analyze of robot manipulator. GRNN also does not follow experimental result. The result of LVQNN is not obtainable for following experimental approach (see Fig. 9). Commanding the end-effectors to track a prescribed trajectory of robot and the test performance of the MNN predictor scheme are shown in Fig. 10.

A propose neural network is employed to predict noise of robot manipulator. The proposed neural network result is shown in Fig. 11. It is clear that proposed neural network scheme is faster with better accuracy than BPNN, GRNN, LVQNN and MNN scheme tested in this study.

From the results obtained, it can be seen that the proposed neural network model system employing a proposed RBFNN produced the best performance while the BPNN, GRNN, LVQNN and MNN predictors yielded poor analyze. A reason for strong performance of the RBFNN model system was the inclusion of gauss activation functions in the neurons for the hidden and the output layer (see Table 2).

5. Conclusions

The objective of this work was to prove the effectiveness of neural network predictor as an alternative to noise analyze in a robot manipulator. Methods and neural network model were developed to analyze noise faults in case of carrying load for robot manipulator. The neural network-based noise fault detection of robot manipulator’s joints was done using noise sensors with a hardware and software. The different types of the neural networks were used for comparison. The use of RBFNN was given the best results in nearly the same experimental approach. These results were principally proofed experimentally on robot manipulator.

The applied methodology proves to the very flexible and powerful and is suitable for the solution of other prediction problems.

Acknowledgements

This research results consisted of part of project number DPT-05-06. The authors would like to express their thanks to Erciyes University and State Planning Organization for supporting this project.

References