Published in: *Robotica*, **20**, pp. 93-109, 2002.

Dynamic modelling of a single-link flexible manipulator: parametric and non-parametric approaches

M. H. Shaheed[1] and M. O. Tokhi[2]

[1]Department of Engineering
Queen Mary, University of London, London, UK.

[2]Department of Automatic Control and Systems Engineering,
The University of Sheffield, Sheffield, UK.

ABSTRACT

This paper presents an investigation into the development of parametric and non-parametric approaches for dynamic modelling of a flexible manipulator system. The least mean squares, recursive least squares and genetic algorithms are used to obtain linear parametric models of the system. Moreover, non-parametric models of the system are developed using a non-linear AutoRegressive process with eXogeneous input model structure with multi-layered perceptron and radial basis function neural networks. The system is in each case modelled from the input torque to hub-angle, hub-velocity and end-point acceleration outputs. The models are validated using several validation tests. Finally, a comparative assessment of the approaches used is presented and discussed in terms of accuracy, efficiency and estimation of the vibration modes of the system.

KEYWORDS: Backpropagation, dynamic modelling, flexible manipulator, genetic algorithm, LMS algorithm, multi-layered perceptron, NARX model, neural networks, radial basis function, RLS algorithm.

1 INTRODUCTION

The increased utilization of robotic manipulators in various applications has been motivated by the requirements of industrial automation in recent years. Among the two types of manipulators, namely, rigid and flexible, attention is focused more on flexible manipulators,

as they are capable of being operated at high speeds and handling of larger payloads as compared to rigid manipulators with the same actuator capabilities. Moreover, flexible manipulators offer several other advantages. These include: light weight, lower energy consumption, safer operation due to reduced inertia, smaller actuator requirement, low mounting strength requirement, low rigidity requirement and less bulky design.[1] Due to such advantages flexible manipulators are extensively used in various applications, including space exploration and nuclear plants.

System identification is extensively used as a fundamental requirement in many engineering and scientific applications. The objective of system identification is to find exact or approximate models of dynamic systems based on observed inputs and outputs. Once a model of a physical system is obtained, it can be used for solving various problems; for example to control the physical system or to predict its behaviour under different operating conditions.[2, 3] A number of techniques have been devised by researchers to determine models that best describe input-output behaviour of a system. Parametric and non-parametric identification are two major classes of system modelling techniques.

Parametric identification of a system comprises two main steps. The first step is qualitative operation, which defines the structure of the system for example, type and order of the (differential/difference) equation relating the input to the output. This is known as *characterization*, which means selection of a suitable model structure, for example, AutoRegressive with eXogeneous inputs (ARX), AutoRegressive Moving Average with eXogeneous inputs (ARMAX), Box- Jenkins etc. The second step namely *identification,* consists of determination of the numerical values of the structural parameters which minimize the distance between the system to be identified and its model. Common estimation methods are least mean squares (LMS), recursive least squares (RLS), instrumental-variables, maximum-likelihood and prediction-error. In simple terms, this is a curve fitting exercise. Recently, genetic algorithm (GA) based parametric identification techniques have been utilised in many applications. Kristinsson and Dumont used a GA for purposes of system identification and control.[4] The GA has been used for non-linear model term selection by Fonseca et al.,[5] where, the GA has been employed as an alternative to orthogonal least-square

regression to find a smaller set of non-linear model terms from a broader set of possible terms. Caponetto et al.[6] have utilised GAs to determine the parameters of the Chua's oscillator. However, it is evident from the literature that little has been reported on the use of GA-based optimisation in modelling flexible robot arms.

In the case of (non-linear) non-parametric models, neural networks (NNs) and fuzzy logic are commonly utilised. Neural networks possess various attractive features such as massive parallelism, distributed representation and computation, generalization ability, adaptability and inherent contextual information processing.[7] Due to the efficient nature of their working principles and other attractive characteristics, attempts are now made to use neural networks extensively in various identification and control applications, including robotics.

Among the various types of NNs, the multi-layered perceptron (MLP) and radial basis function (RBF) are commonly utilised in identification and control of dynamic systems. Narendra and Parthasarathy have addressed system identification using the globally approximating characteristics of NNs, and have suggested a number of identification structures using NNs for the adaptive control of unknown non-linear dynamic systems.[8] A comparative study of four NN learning methods for dynamic system identification has been reported by Qin et al.[9] Backpropagation through adjoints for the identification of non-linear dynamic systems using recurrent neural models has been addressed by Srinivasan et al.[10] They re-investigated the backpropagation for an efficient evaluation of the gradient with arbitrary interconnections of recurrent systems. They also proposed the accelerated backpropagation to eliminate the delay in obtaining the gradient. Training methodologies of recurrent NNs for dynamical process modelling have been addressed by Nerrand et al.[11] Olurotimi has addressed recurrent network training methodologies with feedforward complexity to learn Bessel's difference equation, thereby generating Bessel functions within as well as outside the training set.[12] Hagan and Menhaj have proposed modification of the backpropagation algorithm with the Marquardt algorithm to train feedforward networks. [13]

The successful application of RBF networks for modelling dynamical systems has also been widely addressed in the literature.[14-16] Chen et al.[17] proposed the orthogonal least

squares (OLS) learning algorithm for RBF networks to model non-linear dynamic systems. Elanayar and Yung[18] have addressed the use of RBF networks to approximate the dynamic and state equations of stochastic systems and to estimate state variables. Identification of robotic manipulators with NNs has also been reported in the literature.[19] However, the literature is very limited in the application of NNs to the identification of flexible manipulator systems.

In this investigation dynamic modelling of a single-link flexible manipulator is considered using parametric and non-parametric identification techniques. The resulting models are subjected to several validation methods, and a comparative assessment of the results is presented and discussed.

2 THE FLEXIBLE MANIPULATOR SYSTEM

A schematic diagram of the experimental manipulator rig used in this study is shown in Figure 1. The main parts of the rig include: the flexible arm, the driving motor with amplifier, measuring devices and a computer with interfacing system. The measuring devices used to record the various responses of the manipulator are shaft encoder, tachometer and accelerometer.[20] The shaft encoder is used for measuring the hub angle of the manipulator. A tachometer is used for measurement of the hub velocity. An accelerometer is located at the end-point of the flexible arm measuring the end-point acceleration. An IBM compatible PC based on 486DX2-50MHz CPU, with 20 Mbytes of dynamic and 540 Mbytes of static memory is used with this experimental rig. Data acquisition and control are accomplished through an RTI-815 I/O board. This board can provide a direct interface between the microcomputer, the actuator and transducers, through AM9513A counter/timer chip for a variety of data acquisition, analog output, digital I/O and time related digital I/O applications.

3 PARAMETRIC IDENTIFICATION TECHNIQUES

The characteristics of flexible structure systems are generally of distributed nature, amounting to resonance modes of vibration. The primary interest in this work lies in locating frequencies of these *resonance modes*, which ultimately dictate the behaviour of the system.

With a view to find an accurate model of the system a comparative assessment of the LMS, RLS and GA methods is carried out in the parametric dynamic modelling of the flexible manipulator.

3.1 Least mean squares algorithm

The LMS algorithm is based on the steepest descent method.[21] The computational procedure of the LMS algorithm can be summarized as follows:

Initially, set each weight $w_i(k)$; $i = 0,1,...,N-1$, to an arbitrary fixed value, such as 0. For each subsequent sampling instants $k = 1, 2, \cdots,$ execute the relations below in the order given.

$$\hat{y}(k) = \sum_{i=0}^{N-1} w_i(k) x(k-1) \tag{1}$$

$$e(k) = y(k) - \hat{y}(k) \tag{2}$$

$$w_i(k+1) = w_i(k) + 2\mu e(k) x(k-1) \tag{3}$$

where $x(k)$ is the observation vector at time step k, $e(k)$ is the error between the actual output $y(k)$ and the predicted output $\hat{y}(k)$ and μ represents the learning rate and is a constant.

3.2 Recursive least squares algorithm

The standard RLS estimation process at a time step *k* is described by[22]

$$\varepsilon(k) = \Theta(k)\Psi(k) - y(k) \tag{4}$$

$$\Theta(k) = \Theta(k-1) - P(k-1)\Psi^T(k)\left[1 + \Psi(k)P(k-1)\Psi^T(k)\right]^{-1}\varepsilon(k) \tag{5}$$

$$P(k) = \lambda^{-1}P(k-1) - \frac{\lambda^{-1}P(k-1)\Psi^T(k)\Psi(k)P(k-1)}{\lambda + \Psi(k)P(k-1)\Psi^T(k)} \tag{6}$$

where, $\Psi(k)$ represents the observation matrix, $y(k)$ is the system output, $\Theta(k)$ is the model parameter vector, $P(k)$ is the covariance matrix and λ represents the forgetting factor. Thus,

the RLS estimation process is to implement and execute the relations in equations (4)-(6) in the order given. The convergence of the algorithm is determined by the magnitude of the modeling error $\varepsilon(k)$ reaching a minimum or by the estimated set of parameters reaching a steady level. The purpose with using forgetting factor is to help the algorithm converge to the global minimum. However, the use of a forgetting factor could cause the predicted values of parameters tend to fluctuate rather than to converge to a certain value. The level of fluctuation depends on the value of λ ; the smaller the value of λ the larger the fluctuation in the parameter values.

3.3 Genetic algorithms

Genetic algorithms form one of the prominent members of the broader class of evolutionary algorithms, inspired by the mechanism of natural biological evolution, i.e., the principles of survival of the fittest.[23] From an operational perspective, a GA comprises two basic elements- a set of individuals, i.e., potential solutions (the population) and a set of biologically inspired operators active over the population. A new set of approximations/solutions is created at each generation, by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using the operators. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.[24] In fact, individuals or current approximations are encoded as strings (typically represented in binary), chromosomes, and then the most promising strings are manipulated using the GA operators for better and better approximation to a solution. The operating mechanism of a GA can be described through the stages shown in Figure 2. These comprise the following:

1. Creation of initial set of potential solutions (population) as strings: An initial population of potential solutions is created. Each element of the population is mapped onto a set of strings (the chromosome) to be manipulated by the genetic operators.
2. Evaluation of each solution and selection of the best ones: The performance of each member of the population is assessed through an objective function imposed by the

problem. This establishes the basis for selection of pairs of individuals that will mate during reproduction. For reproduction, each individual is assigned a fitness value derived from its raw performance measure, given by the objective function. This value is used in the selection to bias towards more fit individuals. Highly fit individuals, relative to the whole population, have a high probability of being selected for mating, whereas less fit individuals have a correspondingly low probability of being selected.[25]

3. Genetic manipulation to create new population: In this phase, genetic operators such as crossover and mutation are used to produce a new population of individuals (offspring) by manipulating the "genetic information" usually called genes, possessed by the members (parents) of the current population. The crossover operator is used to exchange genetic information between pairs, or larger groups, of individuals. Mutation is generally considered to be a background operator, which ensures that the search process is not trapped at a local minimum, by introducing new genetic structures.

After manipulation by the crossover and mutation operators, the individual strings are then, if necessary, decoded, the objective function evaluated, a fitness value assigned to each individual and individuals selected for mating according to their fitness, and so the process continues through subsequent generations. In this way, the average performance of individuals in a population is expected to increase, as good individuals are preserved and breed with one another and the less fit individuals die out. The GA is terminated when some criteria are satisfied, e.g., a certain number of generations completed or when a particular point in the search space is reached.

In this investigation, randomly selected parameters are optimized for different, arbitrarily chosen order to fit to the system by applying the working mechanism of GAs as described above. The fitness function utilized is the sum-squared error between the actual output, $y(n)$, of the system and the predicted output, $\hat{y}(n)$;

$$f(e) = \sum_{i=1}^{n} \left(\left| y(n) - \hat{y}(n) \right| \right)^2 \tag{7}$$

where n is the number of input/output samples. With the fitness function given above, the global search technique of the GA is utilised to obtain the best set of parameters among all the attempted orders for the system.

4 NON-PARAMETRIC IDENTIFICATION TECHNIQUES

Various modelling techniques can be used with NNs to identify non-linear dynamical systems. These include state-output model, recurrent state model and Non-linear AutoRegressive Moving Average process with eXogeneous input (NARMAX) model. It is evident from the literature that if the input and output data of the plant are available, the NARMAX model is a suitable choice for modelling nonlinear systems with suitable neuro-learning algorithms. Mathematically the model is given by [26]

$$\begin{aligned}\hat{y}(t) = f[&(y(t-1), y(t-2), \cdots, y(t-n_y),\\ &u(t-1), u(t-2), \cdots, u(t-n_u),\\ &e(t-1), \mathrm{e}(t-2), \cdots, e(t-n_e)] + e(t)\end{aligned} \tag{8}$$

where $\hat{y}(t)$ is the output vector determined by the past values of the system input vector, output vector and noise with maximum lags n_y, n_u and n_e respectively, $f(\cdot)$ is the system mapping constructed through MLP or RBF neural networks with an appropriate learning algorithm. The model is also known as NARMAX equation error model. However, if the model is good enough to identify the system without incorporating the noise term or considering the noise as additive at the output the model can be represented in a NARX form as [16, 26]

$$\begin{aligned}\hat{y}(t) = f[&(y(t-1), y(t-2), \cdots, y(t-n_y),\\ &u(t-1), u(t-2), \cdots, u(t-n_u)] + e(t)\end{aligned} \tag{9}$$

This is described in Figure 3.

4.1 Multi-layered perceptron neural networks and the backpropagation learning algorithm

Multi-layered perceptron NNs are extensively used in numerous applications including pattern recognition, function approximation, system identification, prediction and control,

speech and natural language processing. An MLP-NN is capable of forming arbitrary decision boundaries and representing Boolean functions. [27] The network can be made up of any number of layers with reasonable number of neurons in each layer, based on the nature of the application. The layer to which the input data is supplied, is called the input layer and the layer from which the output is taken is known as the output layer. All other intermediate layers are called hidden layers. The layers are fully interconnected which means that each processing unit (neuron) is connected to every neuron in the previous and succeeding layers. However, the neurons in the same layer are not connected to each other. A neuron performs two functions, namely, combining and activation. Different types of function such as threshold, piecewise linear, sigmoid, tansigmoid and Gaussian are used for activation.

The backpropagation learning algorithm is commonly used with MLP neural networks. According to the backpropagation algorithm the connection weights between the layers of the multilayered NN are adapted in accordance with the following rules:

$$\Delta w_{kj} = \eta \delta_k O_j \tag{10}$$

$$\Delta w_{ji} = \eta \delta_j O_i \tag{11}$$

where, for tansigmoid function,

$$\delta_k = O_k(1 - O_k^2)(\tau_k - O_k) \tag{12}$$

$$\delta_j = O_j(1 - O_j^2)\sum_k \delta_k w_{kj} \tag{13}$$

O_k, $O_{j,}$ and O_i are output values at the output, hidden and input layers respectively. w_{kj} is a connection weight from neuron j in the hidden layer to neuron k in the output layer. Similarly, w_{ji} is a connection weight from neuron i at the input layer to neuron j in the hidden layer. Derivation of the algorithm can be found in a number of books.[26, 28]

The NN training may get stuck in a shallow local minimum with standard backpropagation. In order to avoid entering the local minimum, the learning parameters, number of hidden neurons, or initial values of the connecting weights could be changed. The learning rate η is a proportionality constant. The larger this constant, the larger the changes

in the connection weights. Usually, a learning rate η is selected as large as possible without leading to oscillations. It is noted that increasing the learning rate could solve the problem of shallow local minimum associated with the standard backpropagation algorithm. In order to increase the learning rate, without leading to oscillation in the output response, equations (10) and (11) can be modified to include a momentum term. This can be accomplished by the rule [28]

$$\Delta w_{kj}(t+1) = \eta \delta_k O_j + \alpha \Delta w_{kj}(t) \tag{14}$$

$$\Delta w_{ji}(t+1) = \eta \delta_j O_i + \alpha \Delta w_{ji}(t) \tag{15}$$

where t indexes the presentation time, η is the learning rate, and α is a constant which determines the effect of past connection weight changes on the current direction of movement in the connection weights space.

The shallow local minimum problem associated with standard backpropagation can alternatively be solved by using the Marquardt-Levenberg[29] modified version of the backpropagation. While backpropagation is a steepest descent algorithm, the Marquardt-Levenberg algorithm is an approximation to Newton's method.[30]

4.2 Radial basis function neural networks

The RBF-NN is a special class of multilayer feed-forward networks, widely studied and applied with supervised learning to solve engineering problems. The hidden neurons of this network provide a set of "functions" that constitute an arbitrary "basis" for the input vectors when they are expanded into the hidden-neuron space. These functions are called "radial basis" functions. The distance between the input vector and a prototype vector (centre vector) determines the activation of the hidden neuron. In contrast to the MLP network, the RBF network has one hidden layer. The input layer has neurons with a linear function that simply feeds the input signals to the hidden layer. Moreover, the connections between the input layer and the hidden layer are not weighted, that is, each hidden neuron receives each corresponding input value unaltered. The hidden neurons are processing units that perform the RBF. The transfer function of the hidden neurons in the RBF network can be local or

global. Local RBFs that are widely studied include Gaussian and inverse multiquardric. Among the global RBFs, thin plate spline, linear and multiquardric functions are widely used.

Consider an RBF neural network with the input at time t denoted by $\mathbf{u}(t) = [u_1(t),\ u_2(t), \cdots, u_n(t)]^T$ and let the connection vector (centre) of each hidden neuron be denoted by $\mathbf{c}_j$ (for $j = 1, 2, \cdots, n_h$), where n_h represents the number of neurons in the hidden layer. Then, the output of each neuron in the hidden layer is given as

$$h_j(t) = f_j(\left\|\mathbf{u}(t) - \mathbf{c}_j\right\|) \quad (j = 1, 2, \cdots, n_h) \tag{16}$$

The connection between the output layer and the hidden layer are weighted. Each neuron of the output layer has a linear input-output relationship so that they perform simple summations; that is, the output of the k th neuron in the output layer at time t is

$$\hat{y}_k(t) = \sum_{j=1}^{n_h} w_{jk} h_j(t) = \sum_{j=1}^{n_h} w_{jk} f_j(\left\|\mathbf{u}(t) - \mathbf{c}_j\right\|) \ (k = 1, 2, \cdots, M) \tag{17}$$

where M represents the number of neurons in the output layer and w_{jk} is the connection weight between the j th neuron in the hidden layer and k th neuron in the output layer. Including the bias parameter w_{k0} in the linear sum, equation (17) becomes,

$$\hat{y}_k(t) = w_{k0} + \sum_{j=1}^{n_h} w_{jk} h_j(t) = w_{ko} + \sum_{j=1}^{n_h} w_{jk} f_j(\left\|\mathbf{u}(t) - \mathbf{c}_j\right\|) \tag{18}$$

The bias parameter compensates for the difference between the average value over the data set of the basis function activation and the corresponding average value of the targets.

It follows from equations (16) and (18) that, in general an RBF-NN is specified by two sets of parameters: the connection weights and the vector of centres. These parameters can in principle be determined from the available sample vectors (training data) by solving the optimisation problem,[26]

$$E = \sum_{t=1}^{S} \left\|\mathbf{y}(t) - \hat{\mathbf{y}}(t)\right\|^2 \tag{19}$$

where S is the number of available sample vectors, $\hat{\mathbf{y}}(t) = [\hat{y}_1(t), \hat{y}_2(t), \cdots, \hat{y}_M t)]^T$ is the output vector computed from the sample input vector using equations (16) and (18), and $\mathbf{y}(t)$ is the corresponding desired output vector.

Since an RBF-NN has only one layer of weighted connections and the output neurons are simple summation units, equation (19) will become a linear least-squares problem once the vector of centres and the scaling parameters have been determined. That is,

$$\min_{\mathbf{w}} \sum_{t=1}^{S} \left\| \mathbf{y}(t) - \hat{\mathbf{y}}(t) \right\|^2 = \min_{\mathbf{w}} \left\| \mathbf{wF} - \hat{\mathbf{y}} \right\|^2, \tag{20}$$

where, $\mathbf{w} = \{w_{ij}\}$ is an $M \times n_h$ matrix of connection weights, $\mathbf{F}$ is an $n_h \times S$ matrix consisting of the outputs of the hidden neurons and whose elements are computed with

$$F_{jt} = f_j(\left\| \mathbf{u}(t) - \mathbf{c}_j \right\|) \quad (j = 1, 2, \cdots, n_h;\ t = 1, 2, \cdots, S) \tag{21}$$

and $\hat{\mathbf{y}} = [\hat{\mathbf{y}}(1), \hat{\mathbf{y}}(2), \cdots, \hat{\mathbf{y}}(S)]$ is the $M \times S$ matrix of desired outputs. The connection weight matrix $\mathbf{w}$ in equation (16) can be found in an explicit form as

$$\mathbf{w} = \hat{\mathbf{y}}\mathbf{F}^{+} \tag{22}$$

where $\mathbf{F}^{+}$ is the pseudo-inverse of $\mathbf{F}$.[26]

The task of a learning algorithm (optimisation scheme) for the RBF network is to select the centres and find a set of weights that makes the network perform the desired mapping. A number of learning algorithms are commonly used for this purpose including,

- Non-linear optimisation of all the parameters (centres and output weights, or other free parameters).
- Random centre selection and a least square algorithm.
- Clustering and a least square algorithm.
- The OLS algorithm.

Among these the OLS algorithm[17] is commonly used, and adopted in this work.

5 MODEL VALIDATION

Once a model of the system is obtained, it is required to validate whether the model is good enough to represent the system. A number of such validation tests are available in the literature, some of which are described below:[31]

A common measure of predictive accuracy used in control and system identification is to compute the one step-ahead (OSA) prediction of the system output. This is expressed as

$$\hat{y}(t) = f(u(t), u(t-1), ..., u(t-n_u), y(t-1), ..., y(t-n_y)) \tag{23}$$

where $f(\cdot)$ is a non-linear function, u and y are the inputs and outputs respectively. The residual or prediction is given by

$$\varepsilon(t) = y(t) - \hat{y}(t) \tag{24}$$

Often $\hat{y}(t)$ will be a relatively good prediction of $y(t)$ over the estimation set, even if the model is biased, because the model was estimated by minimising the prediction errors.

Another method to evaluate the predictive capability of the fitted model is to compute the model predicted output (MPO). This is defined by

$$\hat{y}_d(t) = f(u(t), u(t-1), ..., u(t-n_u), \hat{y}_d(t-1), ..., \hat{y}_d(t-n_y)) \tag{25}$$

and the deterministic error or deterministic residual is

$$\varepsilon_d(t) = y(t) - \hat{y}_d(t) \tag{26}$$

If only lagged inputs are used to assign network input nodes, then

$$\hat{y}(t) = \hat{y}_d(t) \tag{27}$$

If the fitted model behaves well for the OSA and MPO, this does not necessarily imply that the model is unbiased. The prediction over a different set of data often reveals that the model could be significantly biased. One way to overcome this problem is by splitting the data set into two sets, estimation set and test set (prediction set). Normally, the data set is divided into two halves. The first half is used to train the NN and the output computed. The NN usually tracks the system output well and converges to a suitable error minimum. New

inputs are presented to the trained NN and the predicted output is observed. If the fitted model is correct, i.e., correct assignment of lagged *u's* and *y's* then the network will predict well for the prediction set.

A more convincing method of model validation is to use correlation tests. If a model of a system is adequate, then the residuals or prediction errors $\varepsilon(t)$ should be unpredictable from all linear and non-linear combinations of past inputs and outputs. The derivation of simple tests, which can detect these conditions is complex, but it can be shown that the following conditions should hold [32].

$$\begin{aligned}
\phi_{\varepsilon\varepsilon}(\tau) &= E[\varepsilon(t-\tau)\varepsilon(t)] = \delta(\tau) \\
\phi_{u\varepsilon}(\tau) &= E[u(t-\tau)\varepsilon(t)] = 0 \qquad \forall\tau \\
\phi_{u^2\varepsilon}(\tau) &= E[(u^2(t-\tau) - \overline{u}^2(t))\varepsilon(t)] = 0 \qquad \forall\tau \\
\phi_{u^2\varepsilon^2}(\tau) &= E[(u^2(t-\tau) - \overline{u}^2(t))\varepsilon^2(t)] = 0 \qquad \forall\tau \\
\phi_{\varepsilon(\varepsilon u)}(\tau) &= E[\varepsilon(t)\varepsilon(t-1-\tau)u(t-1-\tau)] = 0 \qquad \tau \geq 0
\end{aligned} \tag{28}$$

where $\phi_{u\varepsilon}(\tau)$ indicates the cross-correlation function between $u(t)$ and $\varepsilon(t)$, $\varepsilon u(t) = \varepsilon(t+1)u(t+1)$, and $\delta(\tau)$ is an impulse function.

Ideally the model validity tests should detect all the deficiencies in the performance of the NN including bias due to internal noise. In practice normalised correlations are computed. The sampled correlation function between two sequences $\psi_1(t)$ and $\psi_2(t)$ is given by,

$$\hat{\phi}_{\psi_1\psi_2}(\tau) = \frac{\sum_{t=1}^{N-\tau}\psi_1(t)\psi_2(t+\tau)}{\sqrt{\sum_{t=1}^{N}\psi_1{}^2(t)\sum_{t=1}^{N}\psi_2{}^2(t)}} \tag{29}$$

Normalisation ensures that all the correlation functions lie in the range $-1 \leq \hat{\phi}_{\psi_1\psi_2}(\tau) \leq 1$ irrespective of the signal strengths. The correlations will never be exactly zero for all lags and the 95% confidence bands defined as $1.96/\sqrt{N}$ are used to indicate if the estimated correlations are significant or not, where *N* is the data length. Therefore, if the correlation functions are within the confidence intervals the model is regarded as adequate.

The OSA prediction and MPO tests are normally used to determine the model validity in the case of non-linear modelling. Estimation set and test set could be used in the cases of both linear and non-linear modelling. Among the five correlation tests, the first two in equation (28) are generally used to determine model validity in the case of linear modelling whereas all five are used in the case of non-linear modelling.

6 DATA PRE-PROCESSING

It is nearly always advantageous to apply pre-processing transformations to the input data before it is presented to the model. It is required to reduce the difference of magnitude of input variables used to fit to the model. This leads to faster convergence. One of the widely used form of pre-processing is linear rescaling of the input variables. Naturally, input variables presented to the model might have values, which differ by several orders of magnitude. Linear transformation is applied to arrange for all the inputs to have similar values. This is usually done by treating each of the input variables independently and calculating the mean $\bar{x}_i$ and variance σ_i^2 for each variable with respect to the presented data set;[33]

$$\bar{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_i^n \tag{30}$$

$$\sigma_i^2 = \frac{1}{N-1}\sum_{n=1}^{N} (x_i^n - \bar{x}_i)^2 \tag{31}$$

where $n = 1, \cdots, N$ labels the patterns. Thus, a set of re-scaled variables is defined by

$$\tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{\sigma_i} \tag{32}$$

The transformed variables given by the $\tilde{x}_i^n$ have zero mean and unit standard deviation over the transformed data set.

7 EXPERIMENTATION AND RESULTS

7.1 Parametric modelling

For identification with the LMS and RLS algorithms the ARMAX structure was considered. This is given as:

$$y(t)+a_1y(t-1)+.....+a_{n_a}y(t-n_a)=b_1u(t-1)+....+b_{n_b}u(t-n_b)+ \\ e(t)+c_1e(t-1)+...+c_{n_c}e(t-n_c) \tag{33}$$

where, a_i, b_i, c_i, are the parameters to be identified. This accounts for both the true system and noise models.

For the input to be persistently exciting, a combination of bang-bang and pseudo-random binary sequence (PRBS) signal was utilised to excite the system in the range of 0-100 Hz covering the rigid body motion and the first three resonance modes. The level of the bang-bang torque was chosen as ± 0.065 Nm, which is enough to deliver the required amount of energy to excite the rigid-body mode of the system without driving it beyond its maximum angular displacement. A PRBS signal with a level of ± 0.0325 Nm was used to excite the plant at higher modes. Figure 4 shows the combined bang-bang and PRBS signal, referred to as the composite PRBS signal. The system response was observed over a duration of 7.5 seconds with no payload. It was noted with the hub-angle, hub-velocity and end-point acceleration responses that the composite PRBS sufficiently excited the first three vibration modes of the system. The first three resonance modes found from these results are shown in Table 1.

The manipulator was modelled with the LMS algorithm from torque input to the hub-angle, hub-velocity and end-point acceleration output with model orders 8, 7 and 6 respectively. The simulated outputs of hub-angle, hub-velocity and end-point acceleration models with the actual system output are shown in Figure 5. It was noted with the corresponding correlation tests of the hub-angle model that none of the correlation functions were within the 95% confidence interval indicating that the model was inadequate to represent the system, and as noted all the three cases the first mode was not identified.

In modelling the system with the RLS algorithm model orders of 8, 7 and 6 were chosen for the hub-angle, hub-velocity and end-point acceleration respectively. Figure 6 shows the responses of the hub-angle, hub-velocity and end-point acceleration models. It was noted with the corresponding correlation tests of the hub-angle model that all the correlation functions were within the 95% confidence interval indicating an adequate model fit.

In modelling the manipulator with the GA different initial values and operator rates were investigated, and satisfactory results were achieved with the following set of parameters:

Generation gap	: 0.9
Crossover rate	: 0.7
Mutation rate (hub angle model)	: 0.00313
Mutation rate (hub velocity model)	: 0.00357
Mutation rate (end-point acceleration model)	: 0.00417

The GA was designed with 100 individuals in each generation. The maximum number of generations was set to 500. Different model orders were investigated, and the best results were achieved with orders 8, 7 and 6 with the sum-squared error levels of 0.000163, 0.10482 and 0.18944 in the 500th generation for the hub-angle, hub-velocity and end-point acceleration models respectively. It was noted that a reasonable level of accuracy was achieved with as smaller number as 50 generations. Figure 7 shows the responses of the hub-angle, hub-velocity and end-point acceleration models with the best parameter set resulting in the 500th generation. It was noted with the corresponding correlation tests of the hub-angle model that all the correlation functions were within the 95% confidence interval indicating an adequate model fit.

7.2 Non-parametric modelling

The composite PRBS signal described earlier has four points in its amplitude, which might not be sufficient to capture the nonlinearity present in the system. In the experiments to follow a uniformly distributed white noise signal in the range of 0-100 Hz, which covers the first three vibration modes of the system, was used to excite the system. The level of the

signal was chosen within ± 0.3 Nm. Figure 8 shows the noise input in the time and frequency domains thus used. The corresponding hub-angle, hub-velocity, end-point acceleration responses of the system are shown in Figure 9. The first three vibration modes of the system found from these results are listed in Table 2. It is noted that the value of the first resonance frequency is consistent with the three responses. This is also reflected in a single resonance peak in each as noted in Figure 9. The discrepancy in the frequency of each of the second and third modes is due to multiple resonance peaks appearing about these modes as noted in Figure 9. In obtaining the frequencies in Table 2 for the second and third modes, an average of the frequencies corresponding to these peaks was taken in each case.

In this section results of modelling the manipulator from input torque to hub-angle, hub-velocity and end-point acceleration outputs with MLP and RBF NNs are presented. In these investigations both the OSA prediction and the MPO models were obtained. The data set, in each case, comprised 1500 data points. This was divided into two sets of 750 data points each. The first set was used to train the network and the whole 1500 points, including the 750 points that were not used in the training process, were used to test the model.

7.2.1 Modelling with MLP-NN

Figure 10 shows the OSA prediction of hub-angle of the flexible manipulator using a three layered MLP-NN with 5 neurons in the hidden layer and $n_u = n_y = 8$. The model reached a sum-squared error level of 0.0025 with 100 training passes. It is noted with the corresponding correlation tests performed on the predicted data set that all the tests were largely within the 95% confidence interval indicating acceptable performance of the model in characterising the hub-angle behaviour of the manipulator.

For the OSA prediction hub-velocity model an NN with 3 neurons in the input layer and 15 neurons in the hidden layer was constructed. Figure 11 shows the performance of the model with the training set and the test set. It is evident from the point to point error graph that the model performed very well in approximating the hub-velocity. The model reached a sum-squared error level of 0.023 with 50 training passes. It was also noted that all the

correlation functions were within the 95% confidence interval indicating adequate performance of the model.

A model constructed with 3 neurons in the input layer and 15 in the hidden layer was also used for the MPO hub-velocity model. The results are presented in Figure 12, which show that an adequate level of performance was achieved. However, the performance of the model is not as good as in the case of OSA prediction. The model reached a sum-squared error level of 0.015 with 1300 training passes.

A neuro-model constructed with 5 neurons in the input layer and 15 in the hidden layer was realized fit for the OSA prediction end-point acceleration model of the manipulator. The model reached a sum-squared error level of 0.0135 with 500 training passes. Figure 13 shows the performance of the model. It was noted in the corresponding correlation tests that all the test results were within the 95% confidence interval indicating good performance of the model.

The model predicted end-point acceleration output with an MLP network, consisting of 5 neurons in the input layer and 15 neurons in the hidden layer, is shown in Figure 14. The model reached a sum-squared error level of 0.0097 with 5000 training passes. The results of all the corresponding correlation tests were largely within the 95% confidence interval indicating acceptable performance of the model.

7.2.2 Modelling with RBF-NN

Figure 15 shows the results of OSA hub-angle prediction of the manipulator using 5 RBF terms with a spread constant of unity and lag of 8. As noted the network has predicted the system output very well. The model reached a sum squared error level of 0.0034 within only 4 training passes. It was noted that the corresponding correlation test functions were within the 95% confidence interval indicating an adequate hub-angle model fit.

An OSA hub-velocity prediction model of the manipulator was constructed with an RBF network incorporating 200 RBF terms and a spread constant of unity. Figure 16 shows the results of the OSA prediction achieved. The model reached a sum-squared error level of 0.014 after 199 training passes. It is evident from point to point error curve that the error is

comparatively larger for the data points that were not used for training the network. However, The error as a whole is not large. Moreover, it was noted that the corresponding correlation test functions were all within the 95% confidence interval indicating adequate performance of the model.

The model predicted hub velocity output with an RBF-NN comprising 200 RBF terms is shown in Figure 17. The model reached a sum-squared error level of 0.02 after 199 training passes.

An NN designed with 250 RBF terms, spread constant of unity and $n_u = n_y = 6$ was found fit for the OSA prediction of end-point acceleration. The model reached a sum-squared error level of 0.0115 with 249 training passes, as shown in Figure 18. It was noted that the corresponding correlation test results were all within the 95% confidence interval indicating adequate performance of the model.

Figure 19 shows the training and test set validation of the model predicted end-point acceleration output of the network constructed with 250 RBF terms, $n_u = n_y = 6$ and spread constant of unity. The model reached a sum squared error level of 0.0115 with 249 training passes. It was noted that the corresponding correlation test results were all within the 95% confidence interval indicating an adequate model fit.

8 COMPARATIVE ASSESSMENT

Table 3 shows a comparative assessment of the three parametric modelling approaches in terms of estimation of the resonance modes of the system. It is noted that the performances with the GA and RLS-based modelling approaches were better than that with the LMS based modelling. The LMS failed to detect the first vibration mode of the system in all the three modelling domains namely hub-angle, hub-velocity and end-point acceleration. Among the GA and RLS based modelling the performance was better with the GA, as the GA uses a global search process in finding the parameters. It was also noted in the correlation tests results that the GA and RLS performed better than the LMS. However, a major advantage of the LMS is that the algorithm is very simple. The LMS is faster as compared to the RLS and

the GA. The GA is extremely slower than the LMS and the RLS and thus, with the current computing technology it is not easy to utilise a GA based identification process in real-time.

It is noted from the input/output mappings presented above that the NN-based models have performed well. All the models were validated through the steps of training and test. It was noted from the results of the correlation tests that the NN-based modelling techniques considered in this study performed adequately well.

Comparative performance of the NN-based modelling of the manipulator is summarised in Table 4. It is noted that with all the techniques the vibration modes of the system were successfully determined with minor or no error except in very few cases. It was also noted that the RBF NNs were faster in convergence as compared to MLP networks. Comparing the results between OSA prediction and MPO, it is noted that the OSA prediction performed well in approximating the system responses. This agrees with the theory as the OSA prediction uses actual data for prediction whereas the MPO uses the predicted value for the same purpose. However, the MPO is more reliable as compared to the OSA prediction and in this investigation the results of the MPO were good enough to be relied upon.

9 CONCLUSION

Parametric and non-parametric modelling of a flexible manipulator have been carried out in this investigation. Among the parametric modelling approaches, the LMS is extensively used in many real-time applications. However, its performance in this case was worse as compared to the RLS and the GA. These algorithms are suitable to different applications, but a closer match depends on the nature of the application. In this particular application the RLS and the GA performed very well in detecting the vibration modes of the system and their performances were comparable.

Investigations in the case of non-parametric modelling of the manipulator reveal that both MLP and RBF NNs are suitable for modelling such systems. Results of non-linear modelling techniques adopted have been validated through various tests including input/output mapping, training and test validation and correlation tests. It has been noted that all the neuro-modelling techniques have performed well in approximating the system

response; where in the time-domain the developed models have predicted the system output closely and in the frequency-domain the resonance frequencies of the model have matched those of the actual system very well.

REFERENCES

1. C. -H. Menq and J. -S. Chen, "Dynamic model and payload-adaptive control of a flexible manipulator", *Proceedings of the IEEE Conference on Robotics and Automation*, Philadelphia, USA, (1998) pp. 448-453.
2. M. Tabrizi, H. Jamaluddin, S. A. Billings and R. Skaggs, "Use of identification techniques to develop a water-table prediction model", *Transaction of the ASME*, **33**(6), 1913-1918 (1990).
3. K. Worden, P. Stansby, G. Tomlinson, and S. A. Billings, "Identification of non-linear-wave forces", *Journal of Fluids and Structures*, **8** (1), 19-71 (1994).
4. K. Kristinsson, and G. Dumont, "System identification and control using genetic algorithms", The *IEEE Transactions on Systems, Man and Cybernetics*, **22** (5), 1033-1046 (1992).
5. C. M. Fonseca, E. M. Mendes, P. J. Fleming and S. A. Billings, "Non-linear model term selection with genetic algorithms", *Workshop on natural algorithms in signal processing,* Chelmdford, Essex (1993) pp. 27/1-27/8.
6. R. C. Caponetto, L. Fortuna, G. Manganaro and M. G. Xibilia, "Chaotic system identification via genetic algorithm", The *1st IEE/IEEE International Conference on GAs in Engineering Systems: Innovations and Applications*, Sheffield, UK (1995) pp. 170-174.
7. Jain, A. K. and Mohiuddin K. M. (1996). Artificial Neural Networks: A Tutorial. *Computer: The IEEE Computer Society Magazine,* **29**, (3), pp. 31-44.
8. Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *The IEEE Transactions on Neural Networks*, **1**, (1), pp. 4-27.
9. Qin, S.-Z., Su, H.-T., and McAvoy, T. J. (1992). Comparison of four neural net learning methods for dynamic system identification. The *IEEE Transaction on Neural Networks*, **3**, (1), pp. 122-130.
10. Srinivasan, B., Prasad, U. R., and Rao, N. J. (1994). Backpropagation through adjoints for the identification of non-linear dynamic systems using recurrent neural models. The *IEEE Transactions on Neural Networks,* **5**, (2), pp. 213-228.

11. Nerrand, O., Roussel-Ragot, P., Urbani, D., Personnaz, L. and Dreyfus, G. (1994). Training recurrent neural networks: why and how? An illustration in dynamical process modeling. The *IEEE Transactions on Neural Networks*, **5**, (2), pp. 178-184.
12. Olurotimi, O. (1994). Recurrent neural network training with feedforward complexity. The *IEEE Transactions on Neural Networks*, **5**, (2), pp. 185-197.
13. Hagan, M. T., Menhaj, M. B. (1994). Training feedforward networks with Marquardt algorithm. The *IEEE Transactions on Neural Networks*, **5**, (6), pp. 989-993.
14. Casdagli, M. (1989). Non-linear prediction of chaotic time series. *Physics D*, **35**, pp. 335-356.
15. He, X. and Lapedes, A. (1993). Non-linear modelling and prediction by successive approximation using radial basis functions, *Physics D*, **70**, pp. 289-301.
16. Sze, T. L. (1995). *System identification using radial basis neural networks*, PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK.
17. Chen, S., Cowan, C. F. N. and Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. The *IEEE Transactions on Neural Networks*, **2**, (2), pp. 302-309.
18. Elanayar, S. V. T., and Yung, C. S., (1994). Radial basis function neural network for approximation and estimation of non-linear stochastic dynamic systems, The *IEEE Transaction on Neural Networks*, **5**, (4), pp. 594-603.
19. Karakasoglu, A., Sudharsanan, S. I. And Sundareshan, M. K. (1993). Identification and decentralised adaptive control using dynamical neural networks with application to robotic manipulators. The *IEEE Transactions on Neural Networks*, **4**, (6), pp. 919-930.
20. A. K. M. Azad, *Analysis and design of control mechanisms for flexible manipulator systems*, PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, UK (1994).
21. B. Widrow, J. R. Glover, J. M. Mc Cool J. Kaunitz , C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong and R. C. Goodlin, "Adaptive noise cancelling: priciples and applications", *Proceedingo of the IEEE*, 63, 1662-76 (1975).
22. M. O. Tokhi, and R. R. Leitch, "Design and implementation of self tuning active noise control system", *IEE Proceedings-D*, 138, 283-292 (1991).
23. J. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, USA, (1975).
24. A. J. Chipperfield and P. J. Fleming, *Parallel Genetic Algorithms: A Survey*, Research report no. 518, Department of Automatic Control and Systems Engineering, The University of Sheffield, UK (1994).

25. A. J. Chipperfield, P. J. Fleming, H. Pohlheim and C. Fonseca, "A genetic algorithm toolbox for MATLAB", *Proceedings of the International Conference on Systems Engineering*, Coventry, UK (September, 1994).
26. Luo, F-L. and Unbehauen, R. (1997). *Applied neural networks for signal processing*. Cambridge University Press, Cambridge, New York.
27. Minsky, M. and Papert, S. (1969). *Perceptrons: An introduction to computational geometry*, MIT Press, Cambridge.
28. Omatu, S., Khalid, M. and Yusof, R. (1996). *Neuro-control and its applications,* Springer, London.
29. Marquardt, D. (1963). An algorithm for least squares estimation of non-linear parameters. *J, Soc. Ind. Appl. Math*., pp. 431-441.
30. Battiti, R. (1992). First and second order methods for learning: between steepest descent and Newton's method. *Neural computation*, **4**, (2), pp. 141-166.
31. M. H. Shaheed, *Neural and genetic modelling, control and real-time finite element simulation of flexible manipulators*, PhD Thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK (2000).
32. S. A. Billings and W. S. F. Voon, "Correlation based model validity tests for non-linear systems", *International Journal of Control*, **15** (6), 601-615 (1986).
33. C. M Bishop, *Neural networks for pattern recognition*. Clarendon Press, Oxford (1995).

LIST OF TABLES AND FIGURES

Table 1: Modes found from system responses to the composite PRBS input

Response	1st mode (Hz)	2nd mode (Hz)	3rd mode (Hz)
Hub-angle	11.112	34.26	63.89
Hub-velocity	11.112	34.26	63.89
End-point acceleration	11.112	36.11	63.89

Table 2: Resonance frequencies found from the responses to the white noise input.

Response	1st mode (Hz)	2nd mode (Hz)	3rd mode (Hz)
Hub-angle	11.112	34.03	64.58
Hub-velocity	11.112	36.11	64.58
End-point acceleration	11.112	36.11	62.5

Table 3. Performance of parametric models in characterising the system.

Modelling Inputs	Modelling domains	1st mode error (%)	2nd mode error (%)	3rd mode error (%)
C-PRBS input	LMS-based hub-angle model	*	5.40	0.93
	LMS-based hub-velocity model	*	5.40	0
	LMS-based end-point acceleration model	*	0	0
	RLS-based hub-angle model	0.93	0	0
	RLS-based hub-velocity model	0	5.40	0
	RLS-based end-point acceleration model	7.99	0	0
	GA-based hub-angle model	0	0	0
	GA-based hub-velocity model	0	5.40	0
	GA-based end-point acceleration model	0	0	0

* Mode did not appear

Table.4: Performance of non-parametric models in characterising the system.

Model type	Number of epochs	Sum-squared error	% error in first mode	% error in second mode	% error in third mode
MLP-NN one-step-ahead hub-angle prediction	100	0.0025	0	0.68	4
MLP-NN one-step-ahead hub-velocity prediction	50	0.023	0	2.56	0.36
MLP-NN model predicted hub-velocity output	1300	0.015	0	2.56	0.36
MLP-NN one-step-ahead end-point acceleration prediction	500	0.0135	0	2.56	3.70
MLP-NN model predicted end-point acceleration output	5000	0.0097	0	2.56	3.70
RBF-NN one-step-ahead hub-angle prediction	4	0.0034	0	0.68	0.36
RBF-NN one-step-ahead hub-velocity prediction	199	0.014	0	2.56	0.36
RBF-NN model predicted hub-velocity output	199	0.02	0	2.56	0.36
RBF-NN one-step-ahead end-point acceleration prediction	249	0.0115	0	2.56	3.70
RBF-NN model predicted end-point acceleration output	249	0.0115	0	2.56	3.70

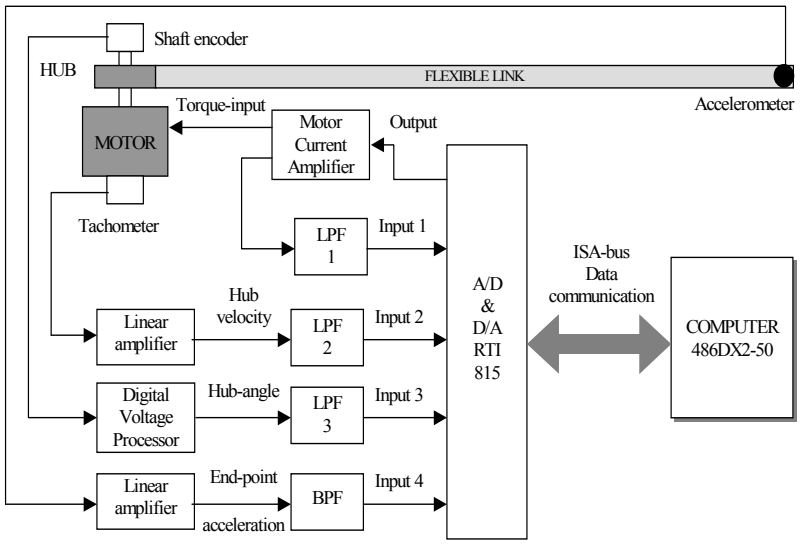


Figure 1: The flexible manipulator experimental rig.

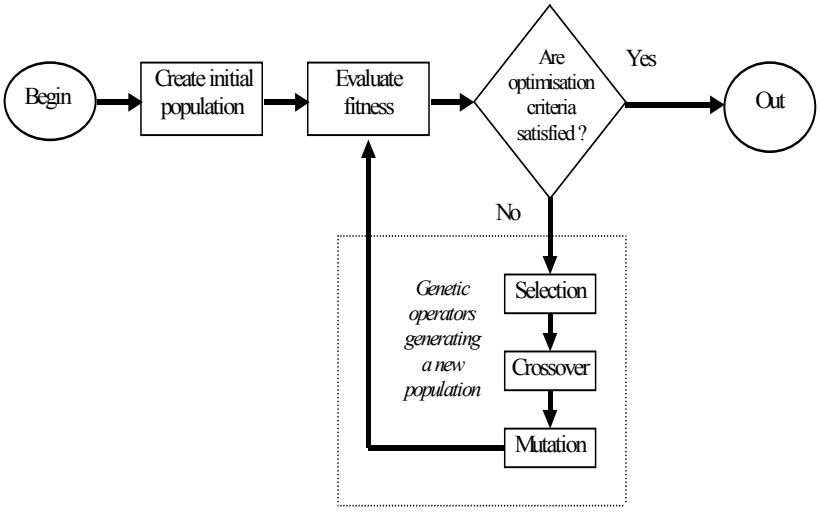


Figure 2: Genetic algorithm-simple working principles.

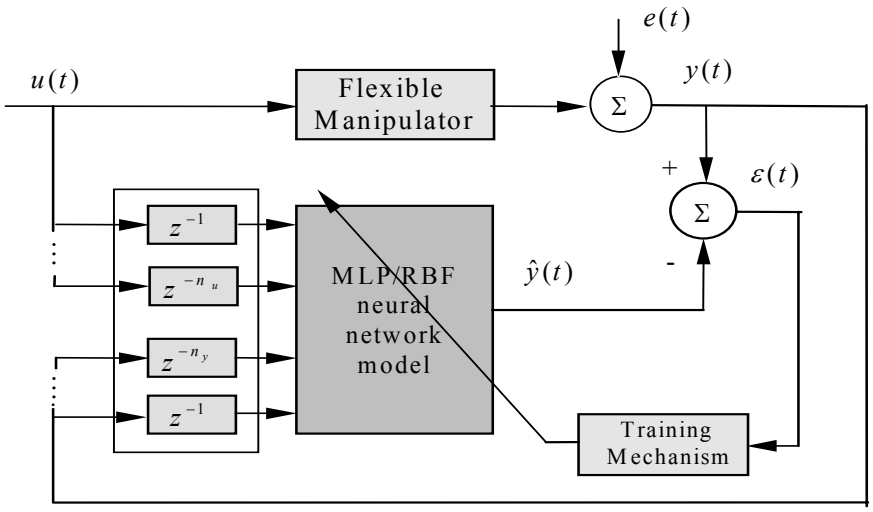
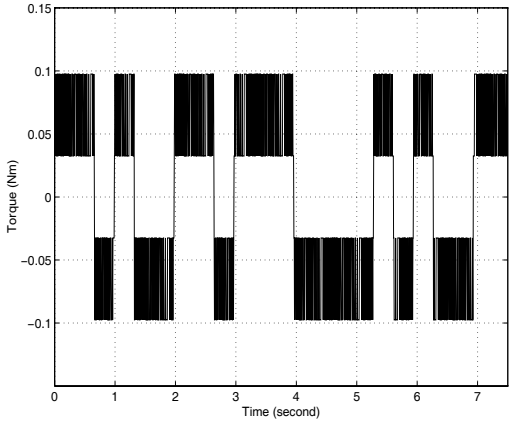
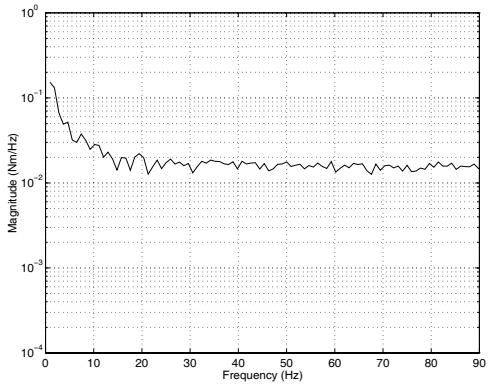


Figure 3: NARX model identification with MLP/RBF neural networks.

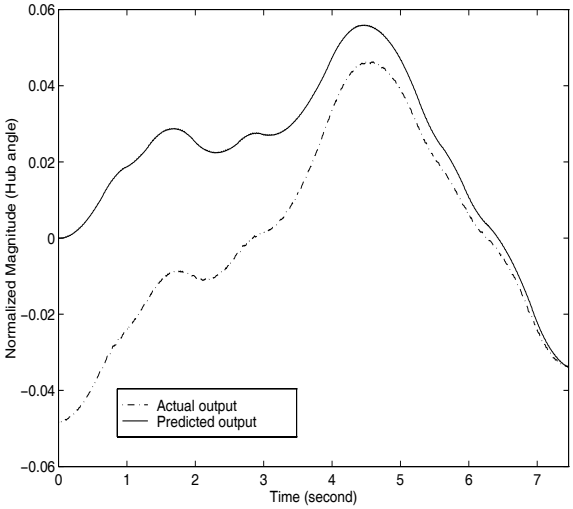


(a) Time domain.

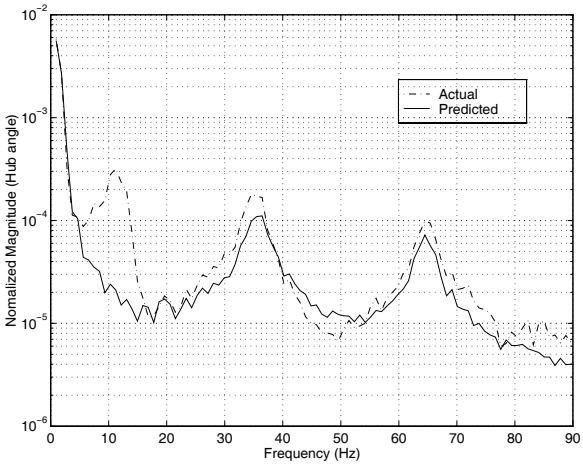


(b) Spectral density.

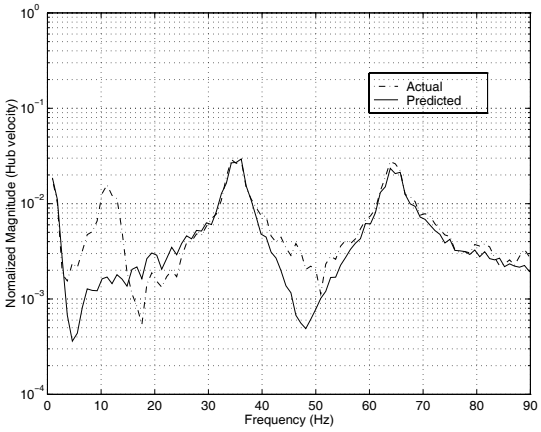
Figure 4: The composite PRBS torque input.



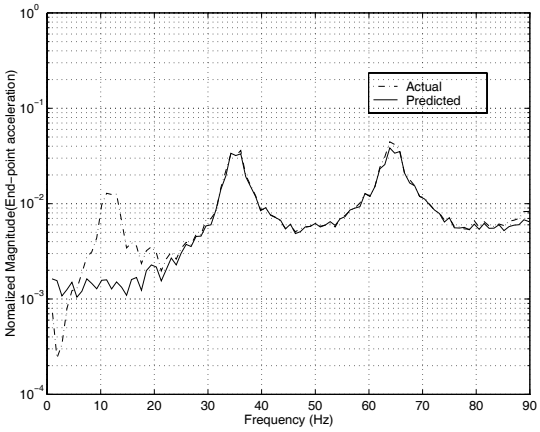
(a) Actual and predicted hub-angle output.



(b) Spectral density of the hub-angle output.

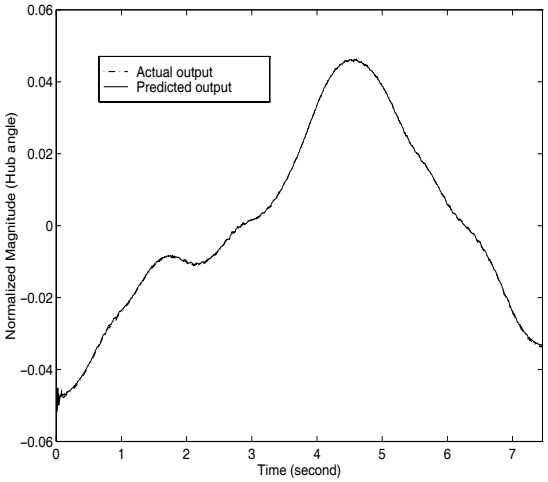


(c) Spectral density of the hub-velocity output.

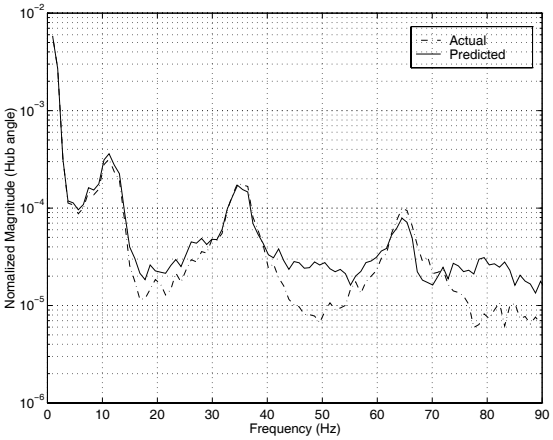


(d) Spectral density of the end-point acceleration output.

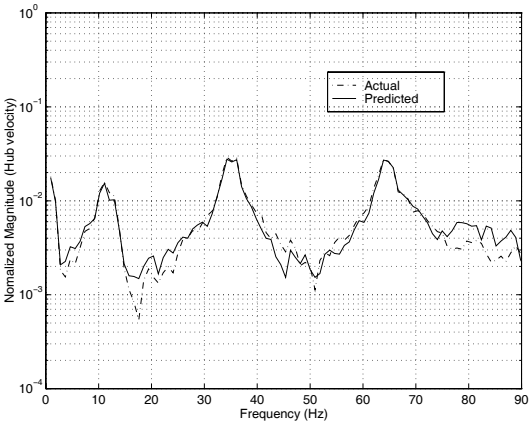
Figure 5: System model with the LMS algorithm.



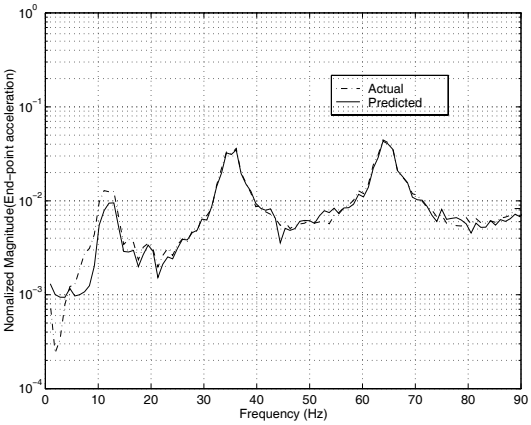
(a) Actual and predicted hub-angle output.



(b) Spectral density of the hub-angle output.

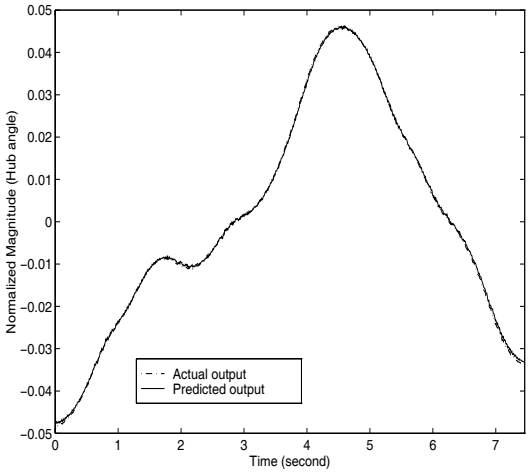


(a) Spectral density of the hub-velocity output.

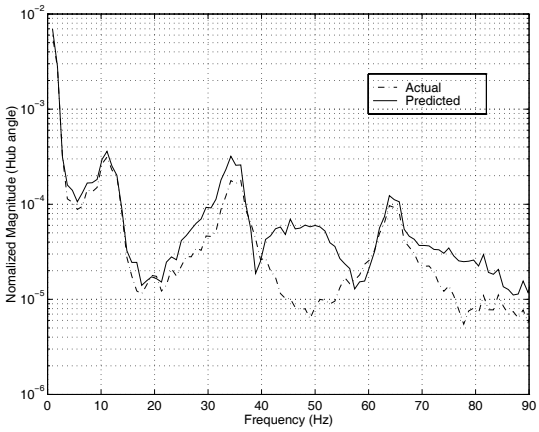


(d) Spectral density of the end-point acceleration output.

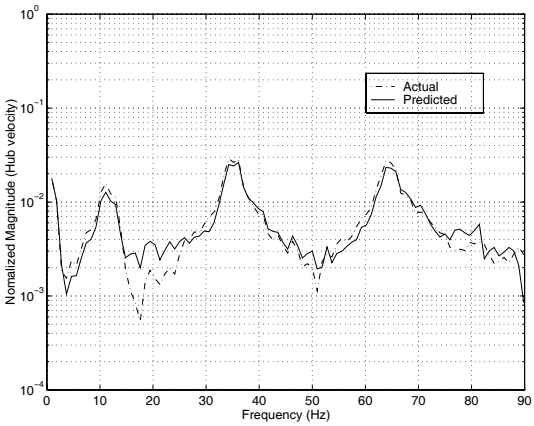
Figure 6: System model with the RLS algorithm.



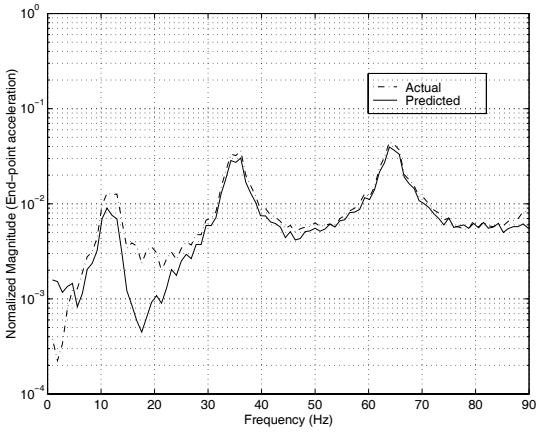
(a) Actual and predicted hub-angle output.



(b) Spectral density of the hub-angle output.

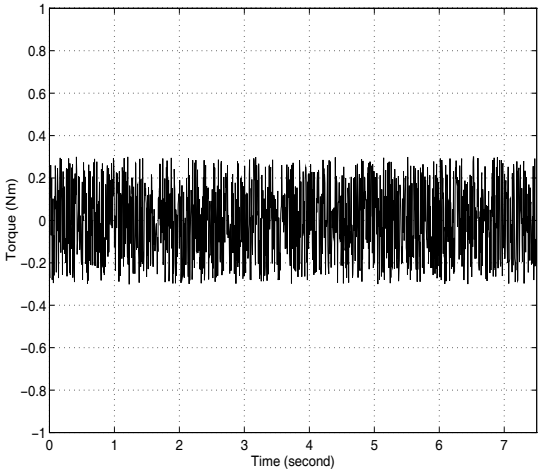


(c) Spectral density of the hub-velocity output.

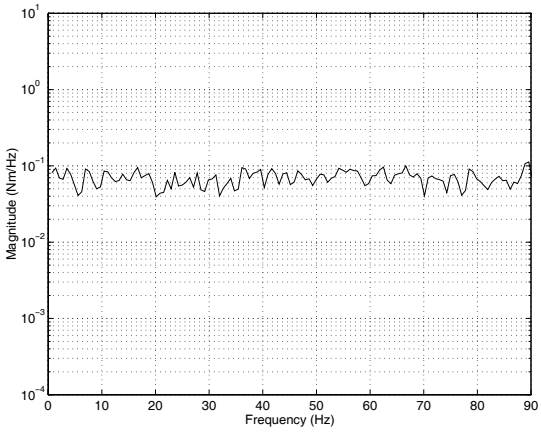


(d) Spectral density of the end-point acceleration output.

Figure 7: System model with the GA.

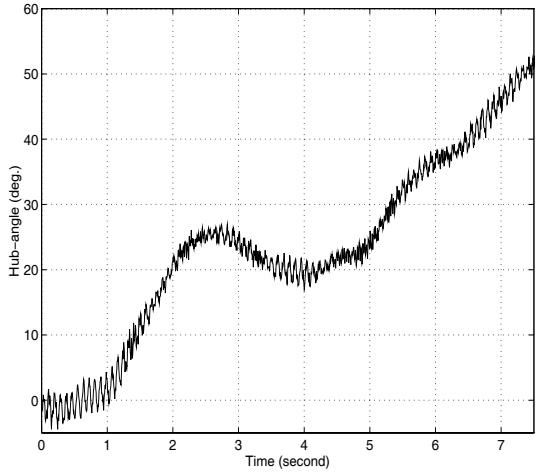


(a) Time domain.

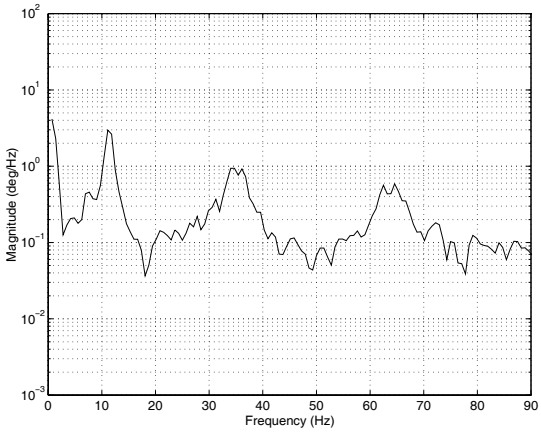


(b) Spectral density.

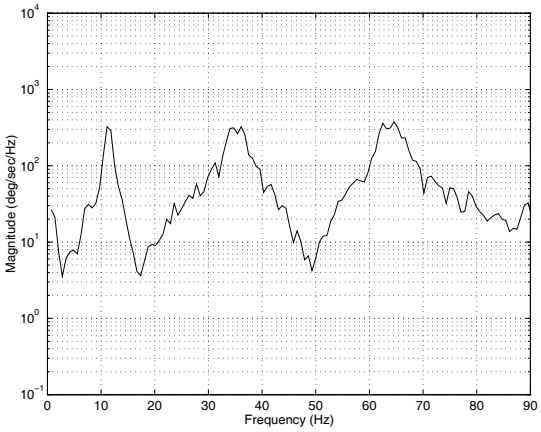
Figure 8: The white noise input to the system.



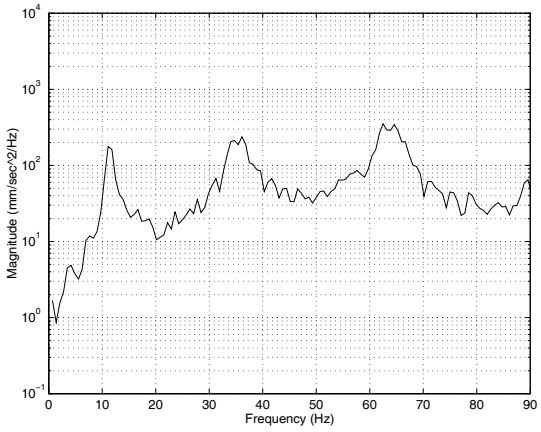
(a) Hub-angle.



(b) Spectral density of hub-angle.

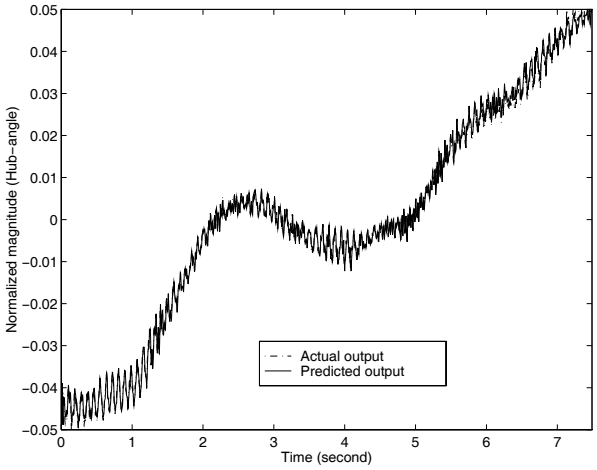


(c) Spectral density of hub-velocity.

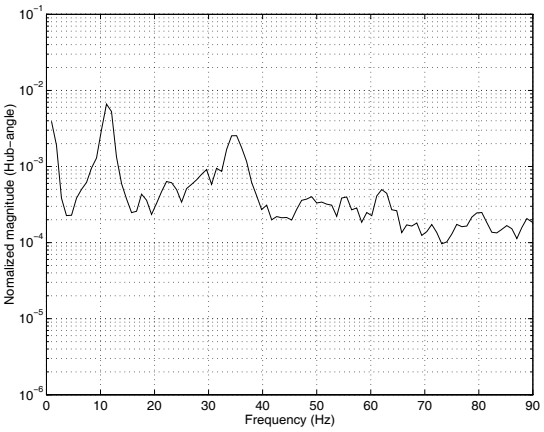


(b) Spectral density of end-point acceleration.

Figure 9: The system response to the white noise input.

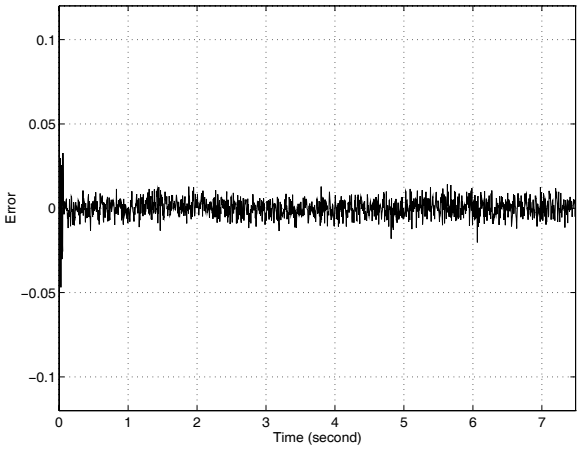


(a) Actual and predicted output.

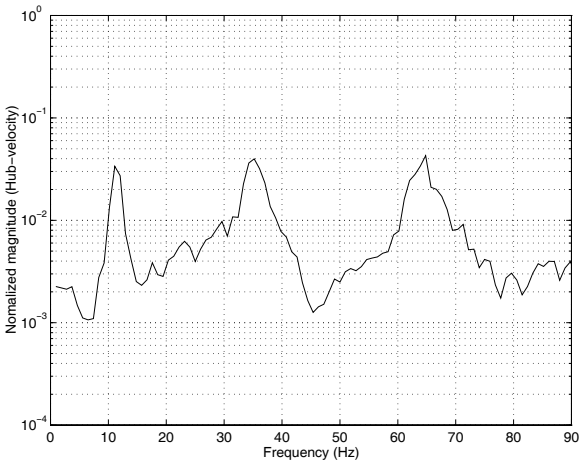


(b) Spectral density of the output.

Figure 10: One-step-ahead hub-angle prediction MLP-NN.

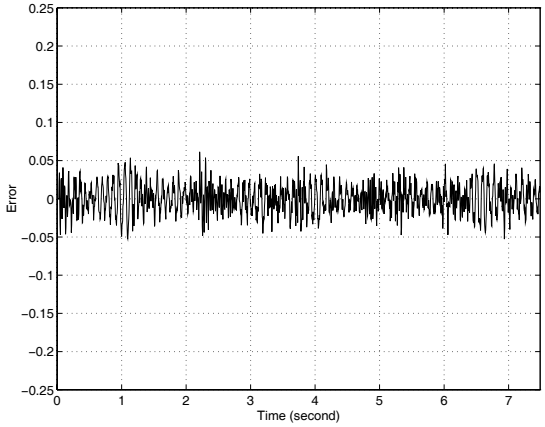


(a) Error between the actual and predicted output.

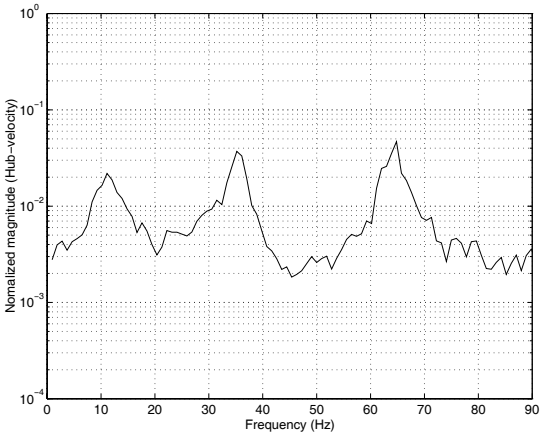


(b) Spectral density of the output.

Figure 11: One-step-ahead hub-velocity prediction with MLP-NN.

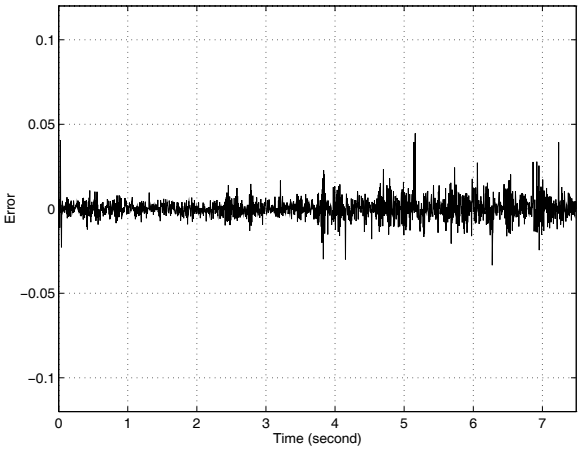


(a) Error between the actual and predicted output.

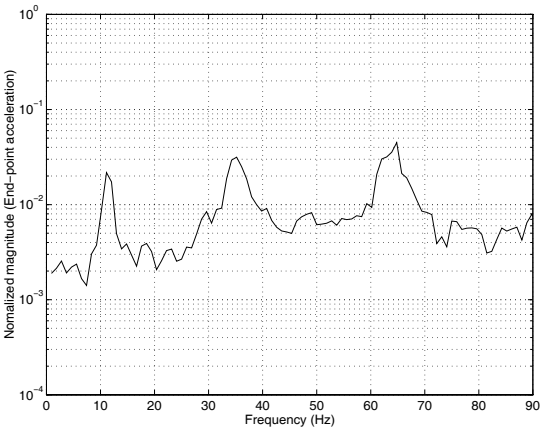


(b) Spectral density of the output.

Figure 12: Model predicted hub-velocity output with MLP-NN.

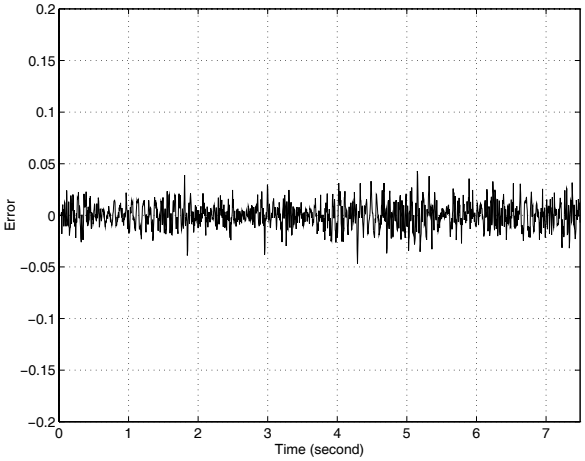


(a) Error between the actual and predicted output.

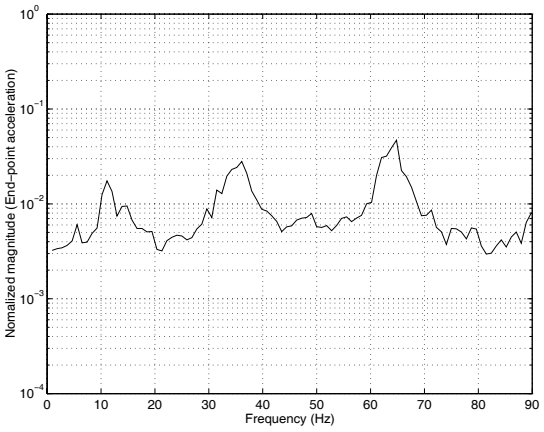


(b) Spectral density of the output.

Figure 13: One-step-ahead end-point acceleration prediction with MLP-NN.

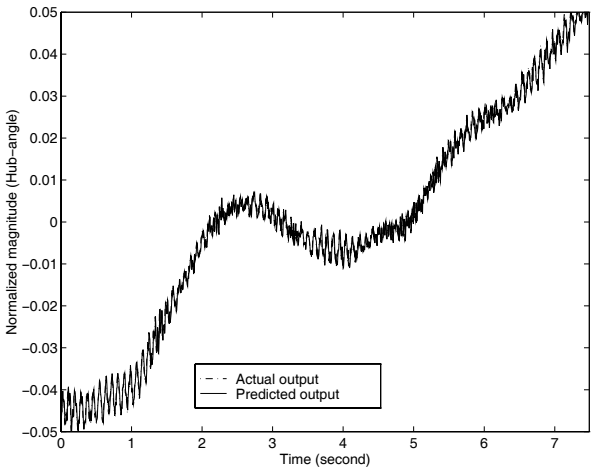


(a) Error between the actual and predicted output.

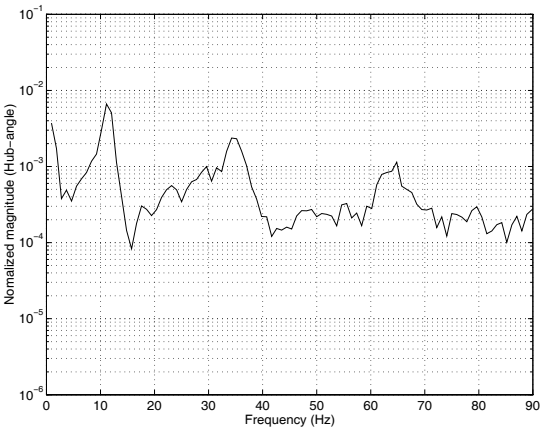


(b) Spectral density of the output.

Figure 14: Model predicted end-point acceleration output with MLP-NN.

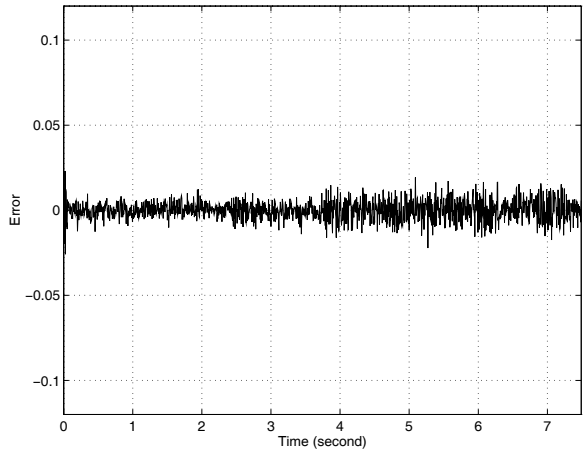


(a) Actual and predicted output.

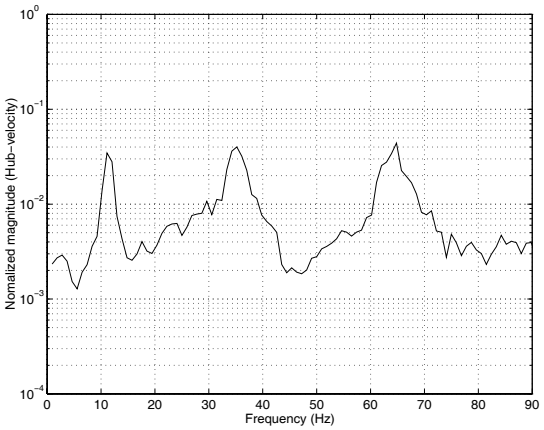


(b) Spectral density of the output.

Figure 15: One-step-ahead hub-angle prediction with RBF-NN.

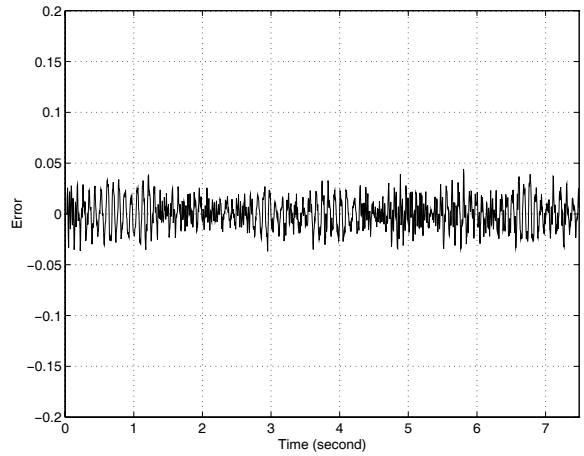


(a) Error between the actual and predicted output.

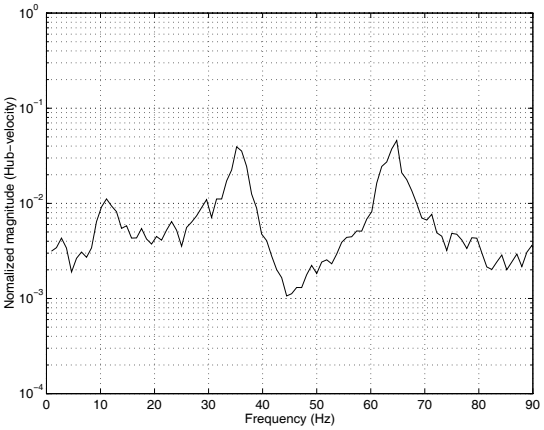


(b) Spectral density of the output.

Figure 16: One-step-ahead hub-velocity prediction with RBF-NN.

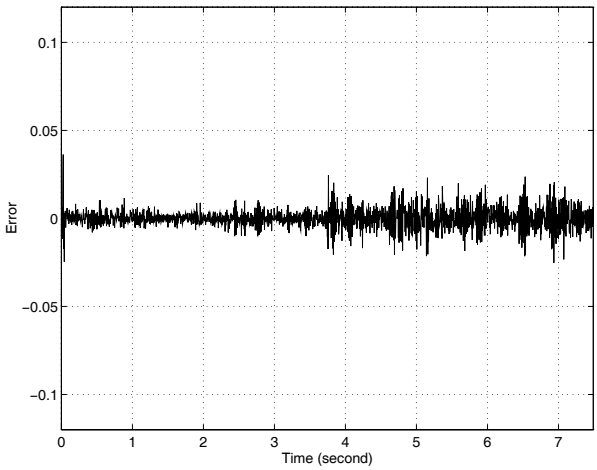


(a) Error between the actual and predicted output.

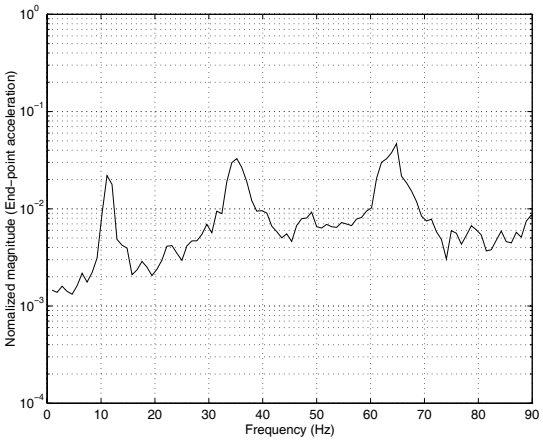


(b) Spectral density of the output.

Figure 17: Model predicted hub-velocity output with RBF-NN.

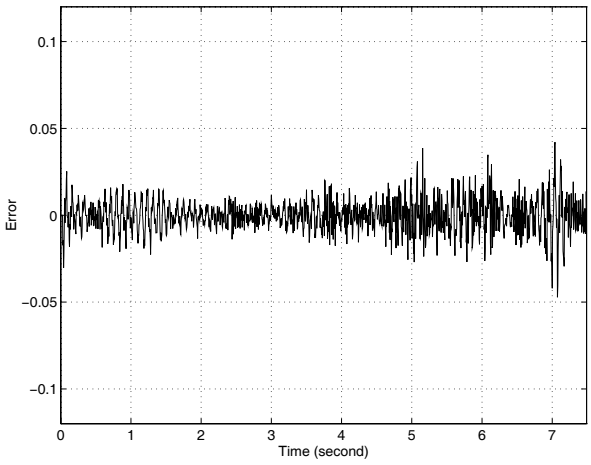


(a) Error between the actual and predicted output.

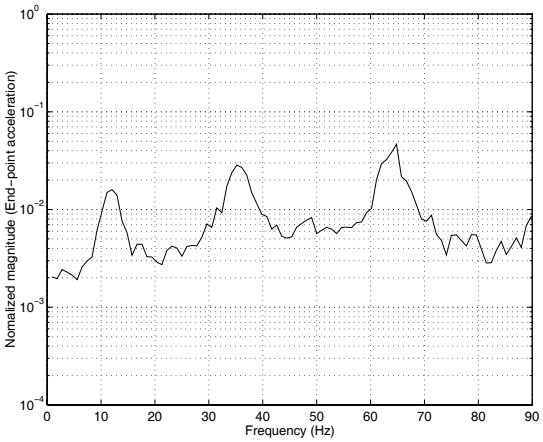


(b) Spectral density of the output.

Figure 18: One-step-ahead end-point acceleration prediction with RBF-NN.



(a) Error between the actual and predicted output.



(b) Spectral density of the output.

Figure 19: Model predicted end-point acceleration output with RBF-NN.