Conditional Distribution Modeling for Estimating and Exploiting Uncertainty in Control Systems

Randa Herzallah

NCRG, Aston University, UK

Fax:+44 (0) 121 333 4586

Email:herzarom@aston.ac.uk

David Lowe

NCRG, Aston University, UK

Fax:+44 (0) 121 333 4586

Email:d.lowe@aston.ac.uk

Abstract

This paper presents a general methodology for estimating and incorporating uncertainty in the controller and forward models for noisy nonlinear control problems. Conditional distribution modeling in a neural network context is used to estimate uncertainty around the prediction of neural network outputs. The developed methodology circumvents the dynamic programming problem by using the predicted neural network uncertainty to localize the possible control solutions to consider. A nonlinear multivariable system with different delays between the input-output pairs is used to demonstrate the successful application of the developed control algorithm. The proposed method is suitable for redundant control systems and allows us to model strongly non Gaussian distributions of control signal as well as processes with hysteresis.

I. Introduction

Uncertainty in process control can be related to different sources. Consider the following deterministic transformation problem

$$a(k+1) = g(a(k), b(k)) \tag{1}$$

This equation implies that the output of the model is known exactly before and after the occurrence of the transformation, which may result from applying a new input value. However, in real world problems the plants are usually subject to different sources of variation which can be due to:

- The incomplete knowledge we may have about the process itself.
- The transformation relationship between the input and the output variables may itself be a nondeterministic relationship. This can be related to several disturbances that can affect the input-output relationship.
- The lack of knowledge for determining the right cost function, which provides the basic ground to optimize the model which provides the prediction output.
- Since in most of the cases the description model is a parameterized function of the input, the parameters of the model itself can be uncertain.

In such situations, the choice of certain input values b may lead to non unique transformations. This means that the choice of one input value b may lead to a set of possible output values a.

To illustrate the variations in real world problems, consider the problem of predicting the value of the force in the pole balancing problem. The force is supposed to avoid failure, where failure in this problem is defined as the event of pole failing past a certain angle or the cart running into the bounds of its track.

To analyze this problem in some detail given the angle of pole, the angular velocity of pole, the horizontal position of the carts centre, the velocity of the cart, the velocity of the wind, the force of friction, and other information, we need to find the mathematical model to predict the amount of force needed to be applied to avoid possible failure by balancing the pole. Finding a suitable mathematical model will require combining all the experimental, physical, theoretical and computational programs to provide an estimation for the model parameters. Estimating the model parameters will require defining a cost function, which in turn should affect the accuracy of the estimation problem.

Even if all this is achieved and providing that a suitable mathematical model can be estimated, an exact prediction can still never be obtained. The first thing we will observe is that small changes in the initial conditions can result in significant changes in the predicted force value. This means that we are examining a highly unstable process.

Looking for an exact prediction can be considered to be very difficult and requires a very precise tool. Since exact solutions are then unattainable, approximate solutions are the only way.

In these situations if we wish to obtain a better prediction for the desired output, additional information in terms of the uncertainty knowledge should be included. There are many approaches and mathematical models for providing an estimate of the uncertainty. The best general way of estimating uncertainty can be described in terms of modeling the probability distribution for the desired prediction.

In this work the problem of decision making in control problems under uncertainty is introduced in the context of neural networks. In contrast to classical control approaches, suppose that the control vectors are generated from a probability distribution $p(u(k))$, and the output variables evolve with time according to

$$y(k+1) = g(y(k), u(k), \bar{v}(k)).$$

The objective in control problems is then to find the optimal control variables from the probability distribution $p(u(k))$ such that when applied to the system, the output of the system should be equal to a predetermined desired value $y_{ref}(k+1)$. This means that we are looking for an idealized optimal control vector $u(k)$, obtained from the distribution $p(u(k))$ such that

$$\mathrm{prob}[|y(k+1) - y_{ref}(k+1)| > 0] = 0 \tag{2}$$

However this cannot be applied directly to real world problems, because the effect of each control variable from the distribution $p(u(k))$ needs to be observed on the real world system. Since only one decision input to the real system can be applied, which is supposed to be optimal, the real output value $y(k+1)$ needs to be replaced by an estimate $\hat{y}(k+1)$. Consequently this implies that the solution provided in eq. (2) never occurs in practice because it requires that the estimator $\hat{y}(k+1)$ for $y(k+1)$ contains no error. Moreover to satisfy this condition the true probability distribution $p(u(k))$ needs to be known, where in practice only an estimate $\hat{p}(u(k))$ for the true distribution $p(u(k))$ can be obtained. In this current paper we will not consider this level of uncertainty, but assume that the estimated distributions are accurate.

To provide an estimate for the required distributions in this work a neural network is used. The principle feature of neural network estimation problems is that it assumes the availability of a set of m output variables $y = (y_1, y_2, ..., y_m)$, and a set of n control variables $u = (u_1, u_2, ..., u_n)$. The estimation problem is then to provide an estimate of the probability density function of the output variables $y(k+1)$ conditioned on the input variables $y(k), u(k)$,

$$\hat{q}[y(k+1)|y(k), u(k), W] \tag{3}$$

where W is the vector of model parameters. The control problem however is the inverse of this forward problem. The controller function needs to provide an estimation of the control variable $u(k)$ conditioned on the variables $y_{ref}(k+1), y(k)$,

$$\hat{p}[u(k)|y_{ref}(k+1), y(k), W] \tag{4}$$

In certain situations, particularly when dealing with inverse problems, mathematical constraints restrict estimating problems to well behaved functions g, and g^{-1}. Thus g, and g^{-1} are usually required to have continuous first derivatives and to have one-one mappings. These restrictions however have little effect on the statistical scope of the estimation problem.

Therefore any statistic $p[y(k+1)|y(k), u(k), W]$ or $p[u(k)|y_{ref}(k+1), y(k), W]$ should ideally provide an adequate description of the data.

Thus providing that a valid estimation for the true distribution $p(u(k))$ and the estimator of the forward model can be obtained, the statistical control optimization problem can be stated as follows. Given: a set U, consisting of all possible decisions $u \in U$ obtained from the probability density function $\hat{p}[u(k)|y_{ref}(k+1), y(k), W]$, and a performance criterion J which provides an evaluation of a given decision variables, two kinds of criteria can be taken:(1) a reward function, in which case it should be maximized, and (2) cost function in which case it should be minimized, and a set Y, the space of the output variables y, consisting of all possible outputs $y \in Y$ that may result from different decision variables, find the optimal control law that minimizes or maximizes the performance criterion J at each instant of time.

In this optimization method an estimation model of the real world system has been assumed, because the control decisions available from the estimated distribution of the controller need to be evaluated. However observing any other aspects of the system which provide information about different control decisions can be sufficient.

II. Forward and inverse models of the plant

In the neuro control field some of the control architectures are based on a forward model of the plant. On the other hand, inverse models are used as controllers. Forward models determine the forward relationship from the input to the plant output. For example the forward model in the pole balancing problem predicts the next state (angle of pole, angular velocity of pole, horizontal position of cart's centre and velocity of cart) given the current state and the force. Forward models have been used in indirect adaptive control, and in the adaptive critic methods. Inverse models however, invert the forward models by providing an estimate for the control, eg. the force applied to the cart in the pole balancing problem, given the desired pole angle and the current state. Since inverse models are responsible for providing control signals that are supposed to make the output of the system equal to the desired output value, they are suitable for use as controllers.

The forward models and the inverse models should then be adapted to provide a good representation for the output and the input of the system respectively. Adaptation procedures for the forward model can be seen to be straight forward. This is because forward models can be learned by supervised learning, comparing the predicted output of the forward model to the actual output. Learning inverse models on the other hand, can be seen to be more difficult. One of the simplest methods to learn inverse models is direct inverse control [1], [2], [9], [19]. In the direct inverse approach the correct control value responsible to make the output of the system equal to a desired output value is assumed to be known. The correct control value is then used as the training signal for the inverse controller. However the direct inverse approach has several drawbacks:

- The learning procedure is not goal directed, since in direct inverse control the error between the plant input $u(k)$ and the network output $\hat{u}(k)$, $e = |u(k) - \hat{u}(k)|$ is minimized, while the goal in control problems is usually to make the system output $y(k+d)$ follow a prespecified desired output $y_{ref}(k+d)$. Using the direct inverse control architecture to make the plant follow the desired response will work only if the desired response happens to be sufficiently close to the system outputs that were used during the training process. The successful application of this method depends largely on the ability of the neural network to generalize and interpolate in regions where the control signal is required for inputs that have not been in the training set. This in turn requires the training signal to be sampled over a wide range of system inputs to cover the possible operational range of the system.
- Obtaining the inverse of the system may not be possible in problems where the mapping is not one-one.

To overcome these problems, Psaltis [19] has suggested the use of a specialized learning architecture. In this approach both the forward and inverse model of the plant are used. The controller in this approach has the advantage that it is trained to span the desired operational output space. This means that the input to the inverse controller in this control architecture is the reference or command signal. The inverse model is then trained to minimize the error between the desired response and the system response, $e = |y(k+d) - y_{ref}(k+d)|$. The actual system output is replaced by the forward model output $\hat{y}(k+d)$ to allow calculating the required derivatives. The forward model is usually assumed to be a good representation of the plant and its parameters are not adapted with the inverse model parameters. Another method which has been suggested in [13] is called feedback error learning. Using these control architectures is supposed to overcome both of the above problems in direct inverse control. However, if the conventional neural networks are unable to cope with redundant systems, they need to be adapted each instant of time before they produce the appropriate control command. This in turn, typically produces large transient errors for training the feedforward controller.

In this work, modeling conditional distributions of control signals is suggested to overcome the uncertainty problems of the inverse controller.

III. Conditional Distributions of Forward and Inverse Models

As mentioned in the previous section, forward models are responsible for providing an estimation for the predicted output of the system given the previous and current outputs of the system and the previous inputs to the system

$$\hat{y}(k+d) = f(y(k), y(k-1), ..., y(k-q+1), u(k), u(k-1), ...u(k-p+1)) \tag{5}$$

where d is the relative degree of the plant, q is the maximum delay in the output and p is the maximum delay in the input.

The aim of the control is to provide an estimation for the control value which can achieve the desired output value

$$\hat{u}(k) = g(y_{ref}(k+d), y(k), ..., y(k-q+1), u(k-1), ..., u(k-p+1). \tag{6}$$

In this section, modeling conditional distributions for forward and inverse models is described. For this purpose the estimation problem for the conditional distribution of the following general neural network function will be described

$$\hat{t}(k) = N(\mathbf{s}(k), c(k), W) \tag{7}$$

where W is the model parameter vector, $\mathbf{s}(k) = [y(k), y(k-1), ..., y(k-q+1), u(k-1), ..., u(k-p+1)]$ is an input vector for both the forward and inverse models , $c(k) = u(k)$ for the forward model, $c(k) = y_{ref}(k+d)$ for the inverse model, and where $t(k) = y(k+d)$ for the forward model, and $t(k) = u(k)$ for the inverse model.

The basic goal is to model the statistical properties of $t(k)$, expressed in terms of the conditional distribution function $p(t(k)|\mathbf{s}(k), c(k), W)$.

In certain situations, particularly when dealing with inverse problems, the estimation problem is restricted to the case of one-to-one mapping functions. In this case and only in this case the inverse of the function denoted by g can be introduced. Therefore a feed-forward neural network trained using the sum of the square error function can be used to obtain the inverse and the forward models of the plant. For this case the distribution of the target data can be described by a Gaussian function with an input-dependent mean (given by the outputs of the trained network), and an input-dependent variance (given by the residual error value).

However, if the inverse of the function f can not be defined uniquely, then the direct inverse mapping found by minimizing the sum of the square error function, can not be used to obtain the inverse of the function. Therefore, assuming a Gaussian distribution can lead to a very poor representation of the control signal. In this case a more general framework for modeling conditional probability distributions is required. This general framework is based on the use of the mixture density network.

In the following sections the problem of Gaussian distribution modeling as well as the mixture density network is presented.

A. Gaussian Distribution Modeling.

If a neural network has been used to predict the target values given by eq (7)

$$\hat{t}(k) = N(\mathbf{s}(k), c(k), W) \tag{8}$$

then the conditional distribution of the target data can be estimated by modeling the conditional uncertainty involved in its own prediction. Estimating the uncertainty around the predicted output of the neural network can be obtained simply by measuring the errors between the predicted value from the neural network $\hat{t}(k)$ and the actual value $t(k)$, $e(k) = \| \hat{t}(k) - t(k) \|^2$. This approach is based on the important result that for a network trained on minimum square error the optimum network output approximates the conditional mean of the target data, or $N_{opt}(\mathbf{s}(k), c(k), W) =< t(k) \mid \mathbf{s}(k), c(k) >$, and that the local variance of the target data can be calculated as $\sigma^2(\mathbf{s}(k), c(k)) = \|t(k) - N_{opt}(\mathbf{s}(k), c(k), W)\|^2$. The assumption then, is that the distribution of the target data can be modeled by a Gaussian distribution with mean equal to the conditional average of the target data, and variance equal to the estimated residual errors

$$p(t(k) \mid \mathbf{s}(k), c(k), W) = \frac{1}{(2\pi\sigma^2(\mathbf{s}(k), c(k), \tilde{W}))^{1/2}} exp\left\{\frac{(t(k) - N_{opt}(\mathbf{s}(k), c(k), W))^2}{2\sigma^2(\mathbf{s}(k), c(k), \tilde{W})}\right\}. \tag{9}$$

To provide an estimation for the predicted variance two neural networks are suggested to be used [20]. The first network is trained so as to predict the conditional mean of the target data. After training the first network, the residual errors can be calculated and can then be used as the target values for the second network, with the inputs being the same as the inputs in the first network. This method is called the predictive error bar method [15], [20].

Other methods for estimating the uncertainty around the predicted output of the neural network can be found in [3], [22]. In this work the predictive error bar method will be used.

B. Mixture Density Network.

Unlike the direct inverse approach, specialized learning control architectures [9], [13], [19] as well as the feedback error learning [13] are able to acquire an accurate inverse model even for multivalued functions (redundant systems). Other approaches for solving the control problems for redundant systems are based on the use of multiple models [3], [5], [10]–[12], [21].

In this work the use of the mixture density network is proposed to acquire the inverse model for multivalued control problems. In its original formulation [4], [17], the mixture density network was specified in terms of a static system for solving regression problems. Recently, the formulation of the mixture density network has been extended to the dynamic case and used in the control context [8]. Furthermore, the multicomponent distribution has been used to search for the optimal control law locally, rather than taking a single estimated value corresponding to the most probable value as in the standard mixture density network [8].

Introducing the maximum likelihood and replacing the Gaussian distribution in (9) with a mixture model, which can model general distribution functions, the probability distribution of the target data can then be defined as

$$p(t(k) \mid \mathbf{s}(k), c(k)) = \sum_{j=1}^{M} \alpha_j(\mathbf{s}(k), c(k)) \phi_j(t(k) \mid \mathbf{s}(k), c(k)) \tag{10}$$

where $\alpha_j(\mathbf{s}(k), c(k))$ represents the mixing coefficients, and can be regarded as prior probabilities, $\phi_j(t(k) \mid \mathbf{s}(k), c(k))$ are the kernel distributions of the mixture model, and M is the number of kernels in the mixture model. Note that the mixing coefficients and the kernel functions are taken to be functions of the input vector $[\mathbf{s}(k), c(k)]$. Various choices are available for the kernel functions, but in this paper the choice will be restricted to spherical Gaussians of the form

$$\phi_j(t(k) \mid \mathbf{s}(k), c(k)) = \frac{1}{(2\pi)^{d/2}\sigma_j^d(\mathbf{s}(k), c(k))} \exp\left(-\frac{\| t(k) - \mu_j(\mathbf{s}(k), c(k)) \|^2}{2\sigma_j^2(\mathbf{s}(k), c(k))} \right) \tag{11}$$

where d is the dimensionality of the target data $t(k)$, $\mu_j(\mathbf{s}(k), c(k))$ represents the centre of the jth kernel, with components μ_{jk}. The spherical Gaussian assumption in (11) can be relaxed in a very straightforward way, by using a full covariance matrix for each Gaussian kernel. However using full covariance Gaussians is not necessary, because in principle a Gaussian mixture model with sufficiently many kernels of the type given by (11) can approximate any given density function arbitrarily accurately providing that the mixing coefficients and the Gaussian parameters are correctly chosen [3]. It follows then that for any given input vector $[\mathbf{s}(k), c(k)]$, the mixture model (10) provides a general formalism for modeling the conditional density function $p(t(k) \mid \mathbf{s}(k), c(k))$.

The parameters of the mixture model, namely the mixing coefficients $\alpha_j(\mathbf{s}(k), c(k))$, the means $\mu_j(\mathbf{s}(k), c(k))$ and the variance $\sigma_j^2(\mathbf{s}(k), c(k))$ are taken to be general continuous functions of $[\mathbf{s}(k), c(k)]$. Since they are continuous functions of the input vector, they can be modeled by a feed-forward neural network that takes $[\mathbf{s}(k), c(k)]$ as its input. In this work the neural network element of the MDN is implemented with a standard multi layer perceptron network. This combined structure of a feedforward network and a mixture model is shown in Figure 1.

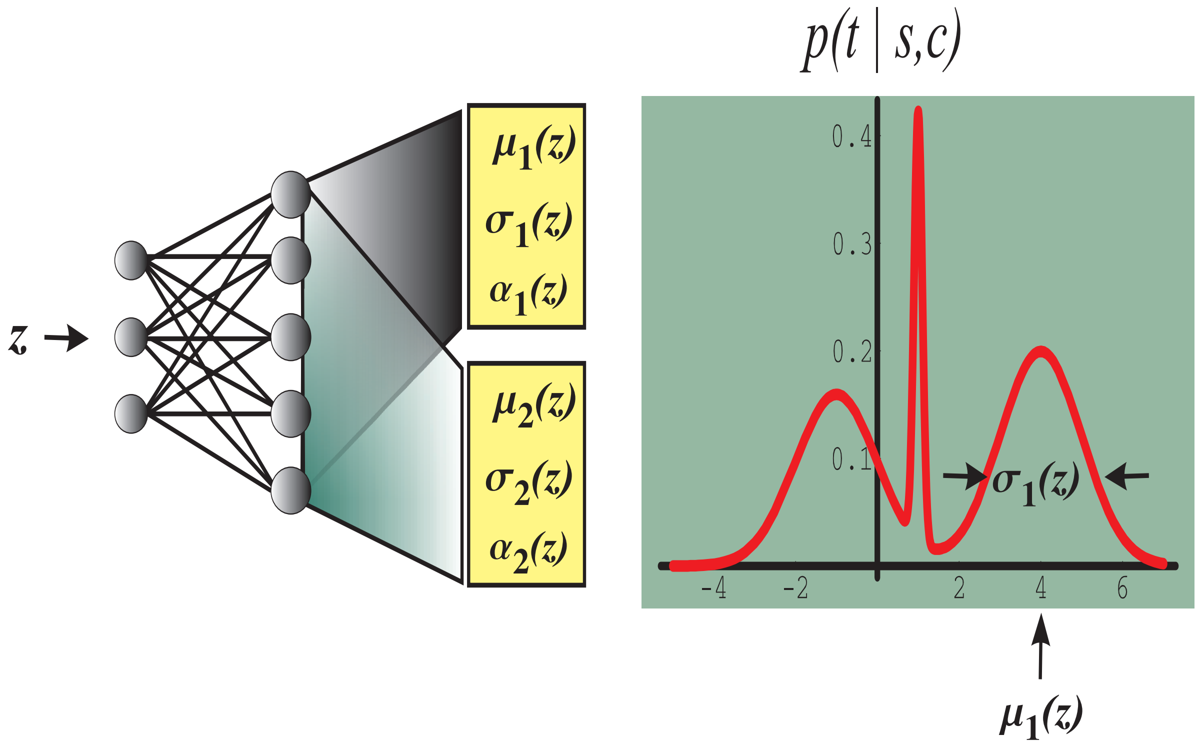


Fig. 1. The architecture of the mixture density network. Here $z = [\mathbf{s}(k), c(k)]$.

The outputs of the MLP approximate the parameters that define the Gaussian mixture model. The notation z_j will be used to denote the output variables. As compared with the usual d outputs for a MLP network used with a sum-of squares error function, the total number of network outputs in the mixture model (10) of M components sum to $(d+2)\times M$. To satisfy the constraints of the mixture model, the parameters of the MDN (the outputs of the MLP network) undergo some transformations . The mixing coefficients α_j must satisfy the following constraint

$$\sum_{j=1}^{M} \alpha_j(\mathbf{s}(k), c(k)) = 1 \tag{12}$$

$$0 \leq \alpha_j(\mathbf{s}(k), c(k)) \leq 1. \tag{13}$$

The first constraint ensures that the distribution is correctly normalized, so that $\int p(t(k) \mid \mathbf{s}(k), c(k)) dt(k) = 1$. These constraints can be satisfied by choosing $\alpha_j(\mathbf{s}(k), c(k))$ to be related to the network's outputs by a 'softmax' function

$$\alpha_j(\mathbf{s}(k), c(k)) = \frac{\exp(z_j^{\alpha})}{\sum_{l=1}^{M} \exp(z_l^{\alpha})}. \tag{14}$$

The variances of the kernel represent scale parameters and always take positive values. To achieve positive values for the variances of the kernel functions, the variances are taken to be exponentials of the corresponding outputs of the MLP network, z_j^{σ}

$$\sigma_j^2 = \exp(z_j^{\sigma}). \tag{15}$$

The centres μ_j of the Gaussians represent a location in the target space and can take any value within that space. Therefore they are taken directly from the corresponding outputs of the MLP network, z_{jk}^{μ}

$$\mu_{jk} = z_{jk}^{\mu}. \tag{16}$$

In order to optimize the parameters in a MDN a likelihood function needs to be constructed [3]. The negative logarithm of the likelihood function can then be used to define the error function. Training the mixture density network can then be proceeded by minimizing the error function (see Appendix for definition of the likelihood function and error function minimization procedure).

Once the network has been trained it can predict the conditional density function of the target data for any given value of the input vector. This conditional density represents a complete description of the generator of the data. More specific quantities can be calculated from this density function which may be of interest in different applications. One of the simplest statistics is the mean, corresponding to the conditional average of the target data. This is equivalent to the mean computed by a standard network trained by least squares. However, in control applications where unique solutions cannot be found, and where the distribution of the target data will consist of different numbers of distinct branches, one specific branch from the estimated conditional density of the MDN needs to be selected. Two examples of how to select a specific branch are the most likely, and the most probable output values. To a very good approximation the most likely output value in an MDN is given by the centre μ_j of the component with largest central value. The component of the largest central value is given by $\max_j\{\alpha_j(\mathbf{s}(k), c(k))/\sigma_j^d(\mathbf{s}(k), c(k))\}$. Alternatively, the most probable output value corresponding to the most probable branch can be calculated, since each component of the mixture model is normalized, $\int \phi_j(t(k) \mid \mathbf{s}(k), c(k))dt(k) = 1$. Therefore, the most probable branch is given by

$$\arg\max_j\{\alpha_j(\mathbf{s}(k), c(k))\}. \tag{17}$$

The required value of $t(k)$ is then given by the corresponding centre μ_j. In this work the MDN with the most probable output value will be used to model the conditional density function to allow for the possibility of a multi-valued function.

IV. Problem Formulation and Solution Development

Dynamic programming is a powerful tool in stochastic control problems [14], [16]. However, it performs poorly when the order of the system increases. The algorithm proposed here is based on incorporating the uncertainty knowledge from the neural network to avoid the computational requirements for the dynamic programming solution of stochastic control problems, see [7].

In direct inverse control the optimal control law is obtained by minimizing the following cost function

$$e = \| u(k) - \hat{u}(k) \|^2 \tag{18}$$

where $u(k)$ is the actual input to the system, and $\hat{u}(k)$ is the inverse model output. The probability distribution of the inverse model can then be estimated as described in Section III.

Modeling the probability distribution of the inverse model provides information about the uncertainty in the predicted output. The main objective in this work is to use this estimate for the uncertainty around predicted outputs of the inverse model so as to make the error between the actual output of the system and the desired output equal zero

$$(y(k+d) - y_{ref}(k+d))^2 = 0. \tag{19}$$

Equation (19) defines the objective functional. The output of the inverse model has already been trained to provide the control value which should bring the output of the process to follow the desired value. Accepting the fact that the predicted output can never be exact and using knowledge of uncertainty around its prediction, we can then try and maintain the condition given in eq. (19).

Since a probability distribution for the inverse model can be estimated, we search for an algorithmic approach yielding numerical solutions to the minimization problem. The proposed method is equivalent to sampling values from the distribution of the control signal $\hat{p}(u(k))$ and using the function value alone to determine a reasonable minimization of the objective functional. Using the gradient information of the objective functional, although it would be more efficient, is not exploitable here due to the random sampling nature of the algorithm and the potential stochastic nature of the plant. Because the inverse controller has been optimized over the entire range of possible output values the functional given in eq. (19) could be maintained locally, at each instant of time.

Maintaining the condition of eq. (19) will be subject to the possible control values resulting from the distribution of the inverse model for a specific input value

$$J(k) = \underset{u \in U}{Min}[(y(k+d) - y_{ref}(k+d))^2] \tag{20}$$

where $U = [u_1, u_2, ..., u_k]$ is the set of samples from the estimated distribution of the controller.

Equation (20) requires applying the sampled values from the control signal distribution to the actual process to see their effect. This however is not allowed in real world applications. Only one control value which is supposed to be optimal can be forwarded to the actual process. Nevertheless, it is common to build a stochastic model for the system output by simply writing

$$g(u(k), \mathbf{s}(k), W) = \hat{g}(u(k), \mathbf{s}(k), W) + \xi(u(k), \mathbf{s}(k), \tilde{W}) \tag{21}$$

where $\hat{g}(u(k), \mathbf{s}(k), W)$ is the forward model of the plant, $\xi(u(k), \mathbf{s}(k), \tilde{W})$ is the uncertainty around the predicted output of the forward model, and where $\mathbf{s}(k) = [y(k), y(k-1), ..., y(k-q+1), u(k-1), ..., u(k-p+1)]$.

Ignoring the uncertainty of the forward model and replacing the actual output of the system by the output of the forward model $\hat{g}(u(k), \mathbf{s}(k), W)$, yields

$$J(k) = \underset{u \in U}{\text{Min}} \; [(\hat{g}(u(k), \mathbf{s}(k), W) - y_{ref}(k+d))^2]. \tag{22}$$

In stochastic control problems the forward relationship from the input to the output can also be driven by a stochastic component

$$y(k+d) = g(\mathbf{s}(k), u(k), \overline{v}(k)). \tag{23}$$

If this is the case then the criterion function given by eq. (20) needs to be modified. The following new criterion function can now be introduced

$$J(k) = \underset{u \in U}{\text{Min}} \; \underset{\overline{v}}{\text{E}} \; [(\hat{g}(u(k), \mathbf{s}(k), \overline{v}(k), W) - y_{ref}(k+d))^2]. \tag{24}$$

The minimum is now taken over all admissible control values from the control signal distribution and the expected value over the stochastic component $\overline{v}$. Assuming that the probability distribution $p(\overline{v})$, of the stochastic component is known

$$\begin{aligned} J(k) &= \underset{u \in U}{\text{Min}} \; \underset{\overline{v}}{\text{E}} \; [(\hat{g}(u(k), \mathbf{s}(k), \overline{v}(k), W) - y_{ref}(k+d))^2] \\ &= \underset{u \in U}{\text{Min}} \left(\int [(\hat{g}(u(k), \mathbf{s}(k), \overline{v}(k), W) - y_{ref}(k+d))^2] p(\overline{v}) d\overline{v} \right) \end{aligned} \tag{25}$$

an approximation method can be used to evaluate the quantity inside the brackets in eq. (25). Since the true distribution of the stochastic component is assumed to be known, the integral can be approximated by the following finite sum

$$\int [(\hat{g}(u(k), \mathbf{s}(k), \overline{v}(k), W) - y_{ref}(k+d))^2] p(\overline{v}) d\overline{v} \approx \frac{1}{L} \sum_{i=1}^{L} [(\hat{g}(u(k), \mathbf{s}(k), \overline{v}_i(k), W) - y_{ref}(k+d))^2]. \tag{26}$$

Equation (26) implies that if the probability distribution for the output of the forward model is given, the expected value of the criterion function could be minimized.

A. Uncertainty in the forward model

In contrast to the uncertainty in the inverse model, uncertainty in the forward model can be considered by changing the performance index to be optimized. Defining the performance index as before

$$J(k) = \underset{u \in U}{\text{Min}} \; M^2(k) = \underset{u \in U}{\text{Min}} \; [(g(u(k), \mathbf{s}(k), W) - y_{ref}(k+d))^2] \tag{27}$$

where $M(k) = (g(u(k), \mathbf{s}(k), W) - y_{ref}(k+d))$ is the utility function, and where

$$g(u(k), \mathbf{s}(k), W) = \hat{g}(u(k), \mathbf{s}(k), W) + \xi(u(k), \mathbf{s}(k), \tilde{W}) \tag{28}$$

where $\xi(u(k), \mathbf{s}(k), \tilde{W})$ is assumed to be random noise with zero mean and variance σ_ξ^2. The uncertainty $\xi(u(k), \mathbf{s}(k), \tilde{W})$ in the forward model can be modeled as described in Section III assuming a Gaussian distribution in the error.

Substituting (28) into (27) yields

$$J(k) = \underset{u \in U}{Min} \; [(\hat{g}(u(k), \mathbf{s}(k), W) + \xi(u(k), \mathbf{s}(k), \tilde{W}) - y_{ref}(k+d))^2] \tag{29}$$

Since the system output is a random variable now, an optimal control law is a control law which minimizes the expected value of the performance index $J(k)$, $< J(k) >$. The expected value of the performance index is given by

$$\begin{aligned} < J(k) >_\xi &=< (\hat{g}(u(k), \mathbf{s}(k), W) + \xi(u(k), \mathbf{s}(k), \tilde{W}) - y_{ref}(k+d))^2 >_\xi \\ &= (\hat{g}(u(k), \mathbf{s}(k), W) - y_{ref}(k+d))^2 + \sigma_\xi^2 \end{aligned} \tag{30}$$

since

$$< M(k) \mid g(u(k), \mathbf{s}(k), W) >= \hat{g}(u(k), \mathbf{s}(k), W) - y_{ref}(k+d) \tag{31}$$

and

$$var(M(k) \mid g(u(k), \mathbf{s}(k), W)) = \sigma_\xi^2 \tag{32}$$

It is clear from (30) that the conditional probability density $p(M(k) \mid \hat{g}(u(k), \mathbf{s}(k), W))$ is given by that of $\xi(u(k), \mathbf{s}(k), \tilde{W})$ with $\xi(u(k), \mathbf{s}(k), \tilde{W}) = M(k) - \hat{g}(u(k), \mathbf{s}(k), W) + y_{ref}(k+d)$. From (30), the optimal control law is given by

$$u(k) = arg \; \underset{u \in U}{Min} \; [(\hat{g}(u(k), \mathbf{s}(k), W) - y_{ref}(k+d))^2 + \sigma_\xi^2] \tag{33}$$

Therefore, instead of minimizing the gap between the average of the forward model and the desired trajectory as in (24), the uncertainty in the forward model is included and the expected value of the performance measure evaluated over the uncertainty in the forward model is minimized.

B. *Optimal control law*

Once properly trained, the inverse model can be used to control the plant since it can create the necessary control signals to create the desired system output. Despite the fact that neural networks have been accepted as suitable models for capturing the behavior of nonlinear dynamical systems, it is also accepted that such models should not be considered exact. The algorithm proposed here circumvents the dynamic programming scaling problem whilst simultaneously allowing for the model uncertainty by using the predicted neural network error bars to limit the possible control solutions to consider. Accepting the inaccuracy of neural networks, the distribution of the output of the inverse control network can be approximated by a Gaussian distribution, or more generally by a multi-component distribution as discussed previously.

Using just the mean estimate of the control in the Gaussian case and the most probable value of the control in the multi-component distribution case is typically suboptimal in nonlinear systems. Modeling the conditional distribution of the control signals, permits the idea of implementing importance sampling of the control signal distribution, which defines the set of allowable decisions at each stage to obtain a better estimate of the control law than the mean or the most probable value. The calculated quantities from these distributions, namely the mean, the most probable value, and the variance are nonlinear functions of previous states, thus allowing for good models of forward and inverse plant behavior. Based on estimates of the distribution of control signal values, the following algorithm can be constructed for incorporating the uncertainty directly. The architecture of this algorithm is shown in figure 2.

1) *Estimate the conditional distribution of the forward model. The assumption here is that the distribution of the forward model is a Gaussian function, and is given by*

$$p(y(k+d) \mid \mathbf{s}(k), u(k)) = \frac{1}{(2\pi\sigma^2_{y(k+d)})^{\frac{1}{2}}} \exp\left(-\frac{(y(k+d)-\hat{y}(k+d))^2}{2\sigma^2_{y(k+d)}}\right) \tag{34}$$

2) *Estimate the conditional distribution of the inverse model, as a Gaussian function given by*

$$p(u(k) \mid \mathbf{s}(k), y(k+d)) = \frac{1}{(2\pi\sigma^2_{u(k)})^{\frac{1}{2}}} \exp\left(-\frac{(u(k)-\hat{u}(k))^2}{2\sigma^2_{u(k)}}\right) \tag{35}$$

or a mixture of Gaussians given by

$$p(u(k) \mid \mathbf{s}(k), y(k+d)) = \sum_{j=1}^{M} \alpha_j(\mathbf{s}(k), y(k+d))\phi_j(t(k) \mid \mathbf{s}(k), y(k+d)) \tag{36}$$

3) *At each instant of time k,*
   a) *Calculate the desired output from the reference model.*
   b) *Bring the control network online and calculate the control signal in addition to the variance of the control signal.*
   c) *Generate a vector of samples from the control signal distribution. This can be obtained as follows*
      i) *For the Gaussian function:*
         A random number generator need to be constructed and used directly to produce a sample from the Gaussian distribution.
      ii) *For the mixture density network:*
         Since Gaussian kernel functions are used, the samples can be generated from each kernel function randomly. This can be done by retrieving the components μ_{jk} *of the kernel centres* μ_j*, and the kernel widths* σ_j *of each kernel function. The number of samples from each component is determined randomly with more samples generated from the component with larger prior.*

      The vector of samples is considered as the set of admissible control values at each instant of time.
   d) *Based on the effect of each sample on the output of the model, the most likely control value is taken. The most likely value is assumed to be the value that minimizes the following cost function.*

$$J(k) = \underset{u \in U}{Min} \underset{\nu}{E}[(\hat{y}(k+d) - y_{ref}(k+d))^2 + \sigma_\xi^2] \tag{37}$$

      where U *is a vector containing the sampled values from the control signal distribution,* E *is the expected value of the cost function over the random noise variable* ν*, and* σ_ξ^2 *is the variance of the uncertainty in the forward model. Because we are using a neural network to model the system, and because the neural network predicts the mean value for the output of the model averaged over the noise on the data, the above function can be optimized directly.*

The stability analysis for the updating rule of the control law has been proved in [6].

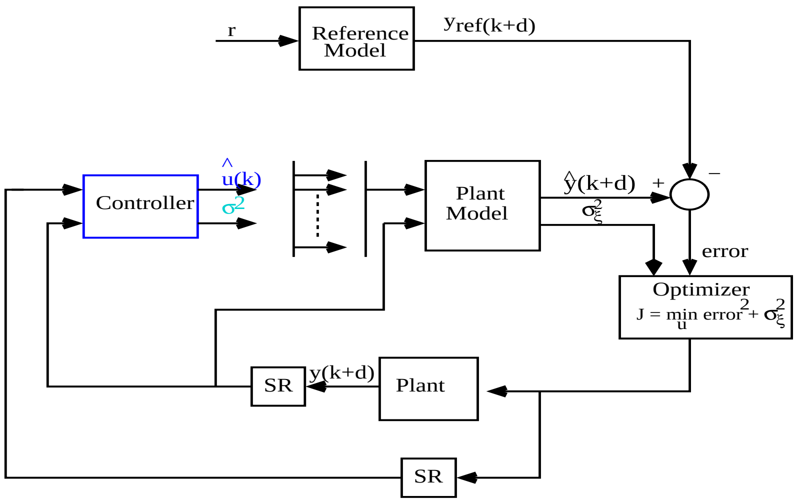


Fig. 2. The architecture of the proposed optimization method. The input and the output of the plant are passed through a shift register (SR) so as to generate the required past input and output values.

V. Simulation 1: Gaussian Distribution

A. Introduction

In order to illustrate the validity of the theoretical developments, we consider the third order system with two inputs and two outputs described by the following state equation :

$$\begin{aligned}
x_1(k+1) &= 0.9x_1(k)\sin[x_2(k)] + \left[2 + 1.5\frac{x_1(k)u_1(k)}{1 + x_1^2(k)u_1^2(k)}\right]u_1(k) + \left[x_1(k) + \frac{2x_1(k)}{1 + x_1^2(k)}\right]u_2(k) \\
x_2(k+1) &= x_3(k)\{1 + \sin[4x_3(k)]\} + \frac{x_3(k)}{1 + x_3^2(k)} \\
x_3(k+1) &= \{3 + \sin[2x_1(k)]\}u_2(k) \\
y_1(k) &= x_1(k) \\
y_2(k) &= x_2(k) \qquad (38)
\end{aligned}$$

where $x(k) = [x_1(k), x_2(k), x_3(k)]$ is the state, $\mathbf{u}(k) = [u_1(k), u_2(k)]$ is the control variable, and $\mathbf{y}(k) = [y_1(k), y_2(k)]$ is the output. This model has been used in [18] to illustrate theoretical developments for the indirect adaptive controller. In this system the delay from the inputs u_1, and u_2 to y_1 is unity, and the delay to y_2 is three from u_1, while it is two from u_2. The plant is considered to be described by equation (38), though assumed unknown to us. Although one neural network could be sufficient to identify the outputs of the plant, two neural networks have been used in this work following the procedure used in Narendra's one model for each output [18]. An input-output model described by the following two equations was chosen.

$$\hat{y}_1(k+1) = N_{f1}(\mathbf{y}(k), \mathbf{y}(k-1), \mathbf{y}(k-2), \mathbf{u}(k), \mathbf{u}(k-1), \mathbf{u}(k-2))$$
$$\hat{y}_2(k+2) = N_{f2}(\mathbf{y}(k), \mathbf{y}(k-1), \mathbf{y}(k-2), \mathbf{u}(k), \mathbf{u}(k-1), \mathbf{u}(k-2))$$

Where N_{f1}, and N_{f2} are multi-layer neural networks. This neural network model was trained using the scaled conjugate gradient optimization algorithm, based on input-output data measurements taken from the plant with sampling time of 1s. The inputs u_1 and u_2 to the plant and the model were generated uniformly over the intervals $[-1.5, 1.5]$ and $[-0.5, 0.5]$ respectively. The single optimal structure for the neural networks found by applying the cross validation method consisted of 21 hidden units for the first model and 17 hidden units for the second model. In the cross validation method both of the forward models were tested on new data that has not been seen in the training stage. The error function between the actual output and the model output, $e = \| y_i(k+d) - \hat{y}_i(k+d) \|^2$, was calculated for different model structures with different numbers of neurons in the hidden layer. The best optimal structure is then taken to be the model with the minimum error value in the validation stage. Similarly, an input-output model described by

$$\hat{\mathbf{u}}(k) = N_c(\mathbf{y}(k), \mathbf{y}(k-1), \mathbf{y}(k-2), \mathbf{u}(k-1), \mathbf{u}(k-2), [y_1(k+1), y_2(k+2)])$$

was chosen to find the inverse model of the plant, where N_c is a multilayer neural network. The training data was the same as in the forward model. A neural network with 7 hidden units was found to be the best model by cross validation. Here the same data used to validate the forward models is used to validate the inverse model. In the validation stage the model with the minimum error between the actual control value and the inverse model value, $e = \| \mathbf{u}(k) - \hat{\mathbf{u}}(k) \|^2$ is taken to be the best model.

B. Classical Inverse Control Approach

After training the inverse controller off line, the control network is brought on line and the control signal is calculated at each instant of time from the control neural network and by setting the two outputs $y_1(k+1)$, $y_2(k+2)$ equal to the desired values $y_{ref1}(k+1) = r_1(k)$, and $y_{ref2}(k+2) = r_2(k)$ respectively. where

$$r_1(k) = 0.65 \sin\left[\frac{2\pi k}{50}\right] + 0.65 \sin\left[\frac{2\pi k}{10}\right]$$
$$r_2(k) = 0.65 \sin\left[\frac{2\pi k}{30}\right] + 0.65 \sin\left[\frac{2\pi k}{20}\right].$$

The predicted mean value from the neural network was forwarded to the plant. The control result is shown in Fig. 3. The performance of the classic controller was seen to be poor with large overshoots around the desired response in the second output $y_2(k)$.

C. Proposed Control Approach: Ignoring uncertainty in the forward model

In our new approach, both the mean and the variance of the control signal were estimated. Following the procedure presented earlier, the best control signal was found and forwarded to the plant, ignoring the uncertainty in the forward model. This means that the performance index of (24) is minimized. Firstly, 30 samples were generated from the Gaussian distribution of each control signal. However the number of samples used to search for an optimal control law was 31^2, including the mean value from each distribution. The overall performance of the plant under the proposed method is shown in Fig. 4. The performance of the proposed controller is seen to be significantly better than the classic controller. However, because the model of the second output was found to be more inaccurate than for the first output, larger errors in the second output can be seen.

D. Proposed Control Approach: Including uncertainty in the forward model

Here, the optimal control value is taken to be the control value that minimizes (30). Similarly the number of samples from the control signal distribution is taken to be 31^2. The performance of the proposed sampling method including the uncertainty in the forward model is found to be slightly better than that where the uncertainty in the forward model is ignored. The average tracking error of the proposed sampling method without including the uncertainty in the forward model is found to be 0.0836. However, the tracking error of the sampling method by accounting for the uncertainty in the forward model is found to be 0.0738.

VI. Simulation 2: Mixture Density Networks

A. Introduction

In this section, the MIMO dynamical system of Section V described by equation (38) is reworked using the idea of mixture density networks to model the conditional distributions of control signals.

The conditional distribution of control signals for this example is calculated using input output data as follows

$$p(u(k) \mid \mathbf{s}(k), \mathbf{y}(k+d)) = \sum_{j=1}^{M} \alpha_j(\mathbf{s}(k), \mathbf{y}(k+d))\phi_j(u(k)|\mathbf{s}(k), \mathbf{y}(k+d))$$

where $\mathbf{s}(k) = [\mathbf{y}(k), \mathbf{y}(k-1), \mathbf{y}(k-2), \mathbf{u}(k-1), \mathbf{u}(k-2)]$ is the same input vector as in Section V. The same training data as for the Gaussian case of Section V is also used for training the mixture density network.

Similarly to test the validity of the mixture density network model, the different model structures have been tested in the validation stage, using the same validation data. The error between the actual control value and the mixture density network, $e = \| u(k) - \hat{u}(k) \|^2$, has been used in this example to find the best model. It was found that the best optimal structure for the inverse model is a mixture density network with 2 components and 7 hidden units in the the multilayer perceptron network.

However, the forward models for the first and the second outputs of the plant remain as before, modeled with a standard multilayer perceptron network.

B. Standard Mixture Density Network

Once the mixture density network is trained off line, its output can be used on line to calculate the control signals that are required to make the output of the system follow the desired output. The same reference signals, $r_1(k)$ and $r_2(k)$, that are used in Section V are used here.

The control signal, $u(k)$, from the MDN is taken to be the mean, $\mu_j[\mathbf{s}(k), \mathbf{y}_{ref}(k+d)]$, of the kernel function with the highest prior value, $\alpha_j(\mathbf{s}(k))$. The result of using the mixture density network as a controller in the MIMO system was found to be slightly better than using a standard network, this is shown in Fig 5. The average tracking error of the standard inverse control is found to be 0.5711, while that of the standard mixture density network is found to be 0.5353.

C. Proposed Control Approach of MDN: Ignoring uncertainty in the forward model

In this section the sampling method from a mixture density network for the MIMO control system is presented. Here the means of the kernel functions in addition to the variances are retrieved and used in the sampling method. Since each kernel is a Gaussian function, the random number generator in Matlab is used to sample each kernel with more samples taken from the kernel with the highest prior value.

The control result from sampling the mixture density network is shown in Fig 6. The number of samples used to search for the optimal control signal is taken to be 900. Again the performance of the controller by sampling the mixture density network is found to be slightly better than that of sampling the Gaussian function.

D. Proposed control method of MDN: Including uncertainty in the forward model

For the mixture density network, taking the optimal control value of the sampling method to be the one that minimizes (30) is again found to be slightly better than taking it to be the one that minimizes (24). The tracking error of the proposed sampling method without including the uncertainty in the forward model is found to be 0.0722. However, the tracking error of the sampling method by accounting for the uncertainty in the forward model is found to be 0.0693. The number of samples in both cases was 900.

VII. Conclusions

General inverse control can be considered to be a good control strategy if the model of the plant happens to be invertible and accurate. We are assuming that the neural network approach allows us to construct accurate models so that we can rely on their outputs as representing the correct conditional mean expectations. If this is not the case then the approach discussed in this paper can fail. Assuming accuracy of the model though, the intrinsic uncertainty around the control signal can be estimated by estimating the conditional distribution of the control signal.

The main contribution of this paper is that it provides a systematic procedure to use this uncertainty measure in order to improve the generalization and robustness property of the controller. Simulation experiments demonstrated the successful application of the proposed strategy to improve the controller performance for a class of nonlinear control dynamic and static systems. Since we are sampling our control signal from the estimated distribution and choosing one which better fits the model, the predicted value of the control signal in the next time step should be more accurate. By feeding back a better value of the control signal, another benefit is that there should be no need to change the controller parameters as long as we are dealing with stationary processes.

The examples given in this paper demonstrate the simplest representative of the conditional density distribution (Gaussian distribution function) in addition to a whole class of density-estimating neural networks (the mixture density network) and also points out a fruitful direction for control research: that of sampling control signals from estimated distribution functions which can incorporate even more information on the full distribution such as higher order moments beyond just the first two, representing the control law and the uncertainty around the control law. This more general approach is not constrained by assumptions of invertibility and it shows the ability to deal with multi-valued processes as well.

APPENDIX I
LIKELIHOOD FUNCTION OF MDN AND ERROR FUNCTION MINIMIZATION

For the training data set, $\{\mathbf{s}(k), c(k), t(k)\}$ the likelihood function can be written as

$$\begin{aligned} \mathcal{L} &= \prod_n p(\mathbf{s}_n(k), c_n(k), t_n(k)) \\ &= \prod_n p(t_n(k) \mid \mathbf{s}_n(k), c_n(k)) p(\mathbf{s}_n(k), c_n(k)) \end{aligned} \tag{39}$$

where in the above equation the likelihood is taken to be a product of probabilities, based on the assumption that each data point has been drawn independently from the same distribution. The negative log likelihood can then be used to define the error function, E

$$E = -\ln \mathcal{L} = -\sum_n \ln p(t_n(k)|\mathbf{s}_n(k), c_n(k)) - \sum_n p(\mathbf{s}_n(k), c_n(k)). \tag{40}$$

The second term in (40) is constant because it is independent of the network parameters, so it can be removed from the error function. The error function becomes

$$E = -\ln \mathcal{L} = -\sum_n \ln p(t_n(k)|\mathbf{s}_n(k), c(k)). \tag{41}$$

Next the error function (the negative log likelihood) for the MDN can be obtained by substituting (10) into (41)

$$E = -\sum_n \ln \left\{ \sum_{j=1}^{M} \alpha_j(\mathbf{s}_n(k), c(k)) \phi_j(t_n(k) \mid \mathbf{s}_n(k), c(k)) \right\}. \tag{42}$$

In order to minimize the error function, the derivatives of the error E with respect to the weights in the neural network must be calculated. Providing that the derivatives can be computed with respect to the outputs of the network, the errors at the network inputs may be calculated using the back-propagation procedure [3]. Since the error function (41) is composed of a sum of terms, one for each training pattern, the error derivative can be considered with respect to each training pattern, n. The total error E is then defined as a sum of the errors for each training pattern. A nonlinear optimization method can then be used to find the minimum of the error function E. In this work the scaled conjugate gradient method is used.

Since the error function of the mixture density network (42) is defined as the sum of the product of the conditional density functions ϕ_j and a prior probability α_j, the posterior probability of the jth kernel can be defined using Bayes' theorem as

$$\pi_j(\mathbf{s}(K), c(k), t(k)) = \frac{\alpha_j \phi_j}{\sum_{l=1}^{M} \alpha_l \phi_l}. \tag{43}$$

This simplifies the analysis of the error derivatives with respect to the network outputs. From (43) one can note that the posterior probabilities sum to unity

$$\sum_{j=1}^{M} \pi_j = 1. \tag{44}$$

Each of the derivatives of E^n are considered with respect to the outputs of the networks and their respective labels for the mixing coefficients, z_j^{α}, variance parameters, z_j^{σ} and centres or position parameters z_{jk}^{μ}. The derivatives are as follows

$$\frac{\partial E^n}{\partial z_j^{\alpha}} = \alpha_j - \pi_j \tag{45}$$

$$\frac{\partial E^n}{\partial z_j^{\sigma}} = -\frac{\pi_j}{2}\left\{\frac{\| t_n(k) - \mu_j \|^2}{\sigma_j^2} - d\right\} \tag{46}$$

$$\frac{\partial E^n}{\partial z_{jk}^{\mu}} = \pi_j\left\{\frac{\mu_{jk} - t_k(k)}{\sigma_j^2}\right\}. \tag{47}$$

References

[1] J.S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Transactions of the ASME Journal of Dynamic Systems, Measurements, and Control*, 97:220–227, 1975.

[2] D.R. Baughman and Y.A. Liu. *Neural Networks in Bioprocessing and Chemical Engineering*. Academic Press, Inc, 1995.

[3] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, N.Y., 1995.

[4] D.J. Evans, I.T. Nabney, and D. Cornford. Structured neural network modelling of multi-valued functions for wind vector retrieval from satellite scatterometer measurements. *Neurocomputing*, 30:23–30, 2000.

[5] S.G. Fabri and V.Kadirkamanathan. *Functional Adaptive Control: An Intelligent Systems Approach*. Springer-Verlag, February 2001.

[6] R. Herzallah and D. Lowe. Improved robust control of nonlinear stochastic systems using uncertain models. In *Controlo2002*, pages 507–512, Aveiro, Portugal, September 2002.

[7] R. Herzallah and D. Lowe. A novel approach to modelling and exploiting uncertainty in stochastic control systems. In *International Conference on Artificial Neural Networks, ICANN*, pages 801–806, Madrid, Spain, August 2002.

[8] R. Herzallah and D. Lowe. Multi-valued control problems and mixture density network. In *IFAC International Conference on Intelligent Control Systems and Signal Processing, ICONS*, pages 387–392, Faro, Portugal, April 2003.

[9] K.J. Hunt, D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop. Neural networks for control systems-a survey. *Automatica*, 28(6):1083–1112, 1992.

[10] R.A. Jacobs and S.J. Nowlan. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[11] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[12] M.I. Jordan and L. Xu. Convergence results for the EM approach to mixtures of experts architectures. Technical report, A.I. Memo no.1458, C.B.C.L. Memo no. 87, Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center of Biological and Computational Learning Department of Brain and Cognitive Sciences, 1993.

[13] M. Kawato. Computational schemes and neural network models for formation and control of multijoint arm trajectory. In R.S. Sutton W.T. Miller and P.J. Werbos, editors, *Neural Networks for Control*, chapter 9, pages 197–228. MIT Press, Cambridge, Massachusetts, London, England, 1990.

[14] R.E. Larson. *State Increment Dynamic Programming*. Number 12 in Modern Analytical and Computational Methods in Science and Mathematics. American Elsevier Publishing Company, New York, N.Y, 1968.

[15] D. Lowe and C. Zapart. Point-wise confidence interval estimation by neural networks: A comparative study based on automative engine calibration. *Neural Computing and Applications*, 8:77–85, 1999.

[16] L. Moreno, L. Acosta, J.A. Méndez, A. Hamilton, G.N. Marichal, J.D. Pñeiro, and J.L. Sánchez. Stochastic optimal controllers for a DC servo motor: Applicability and performance. *Control Engineering Practice*, 4(6):757–764, 1996.

[17] I.T. Nabney. *Netlab Algorithms for Pattern Recognition*. Springer, 2002.

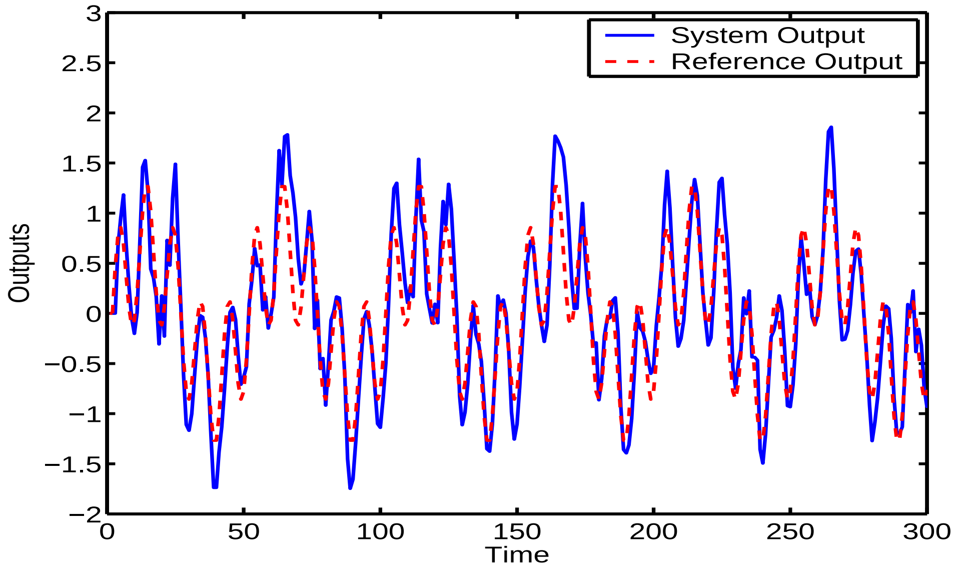
[18] K.S. Narendra and S. Mukhopadhyay. Adaptive control of nonlinear multivariable systems using neural networks. *Neural Networks*, 7(5):737–752, 1994.

[19] D. Psaltis, A. Sideris, and A.A Yamamura. A multilayered neural network controller. *IEEE Control Systems Magazine*, 88(2):17–21, 1988.

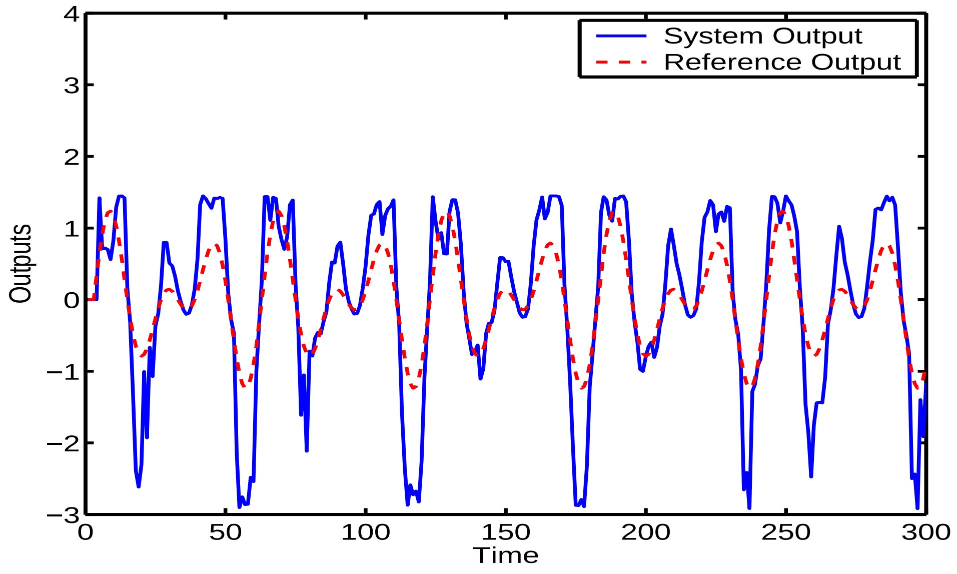
[20] C. Satchwell. Finding error bars (the easy way). *Neural Computation and Application*, 5, 1994.

[21] D.M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.

[22] W.A. Wright, G. Ramage, D. Cornford, and I.T. Nabney. Neural network modelling with input uncertainty: Theory and application. *Journal of VLSI Signal Processing*, 26:169–188, 1993.

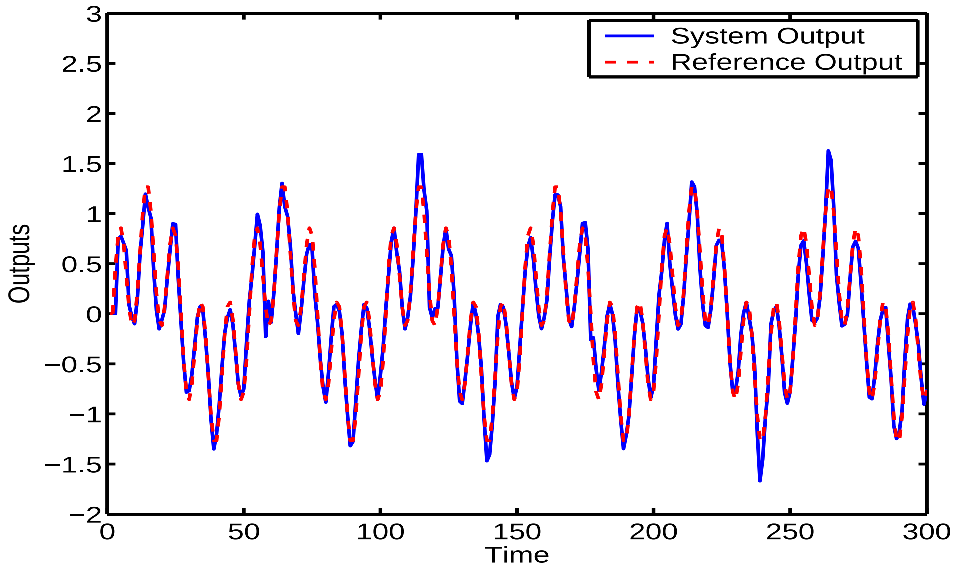


(a)

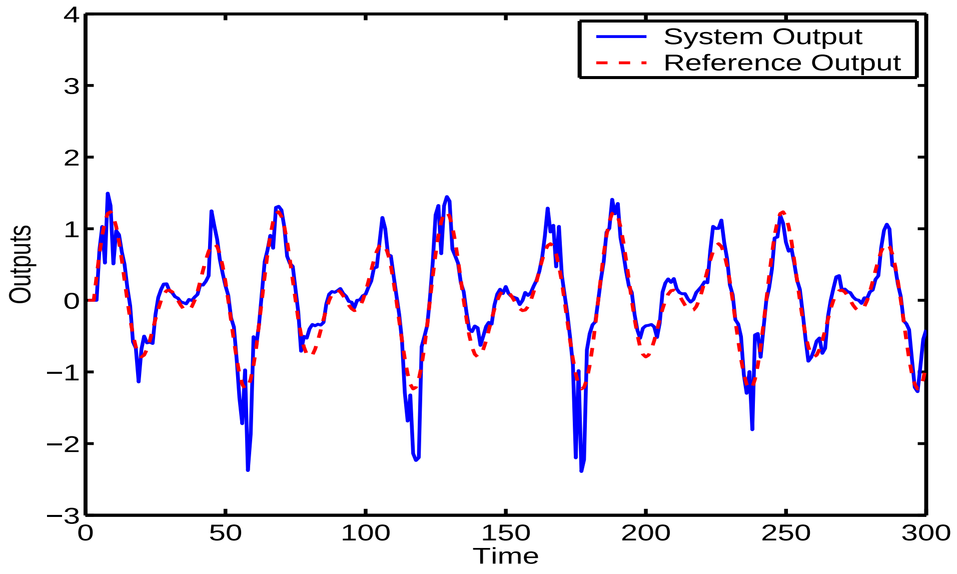


(b)

Fig. 3. Performance of the classical control approach: (a) the first output of the plant. (b) the second output of the plant.

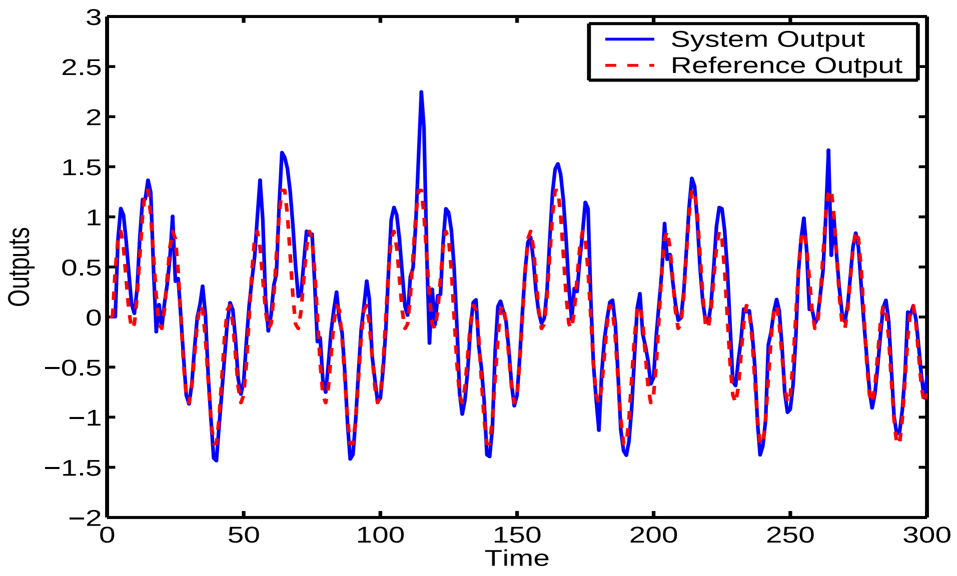


(a)

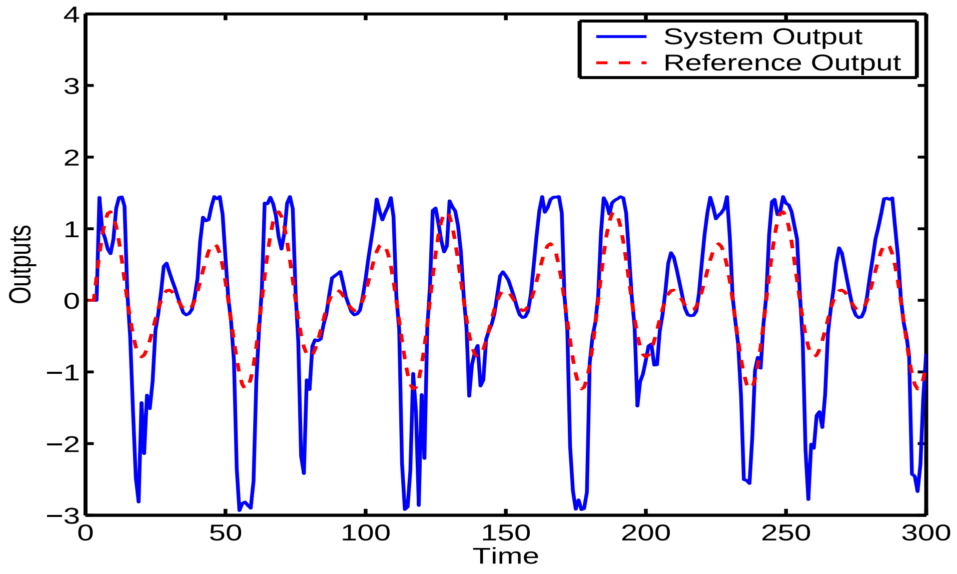


(b)

Fig. 4. Performance of the proposed control approach of a standard neural network for dynamical MIMO system: (a) the first output of the plant. (b) the second output of the plant.

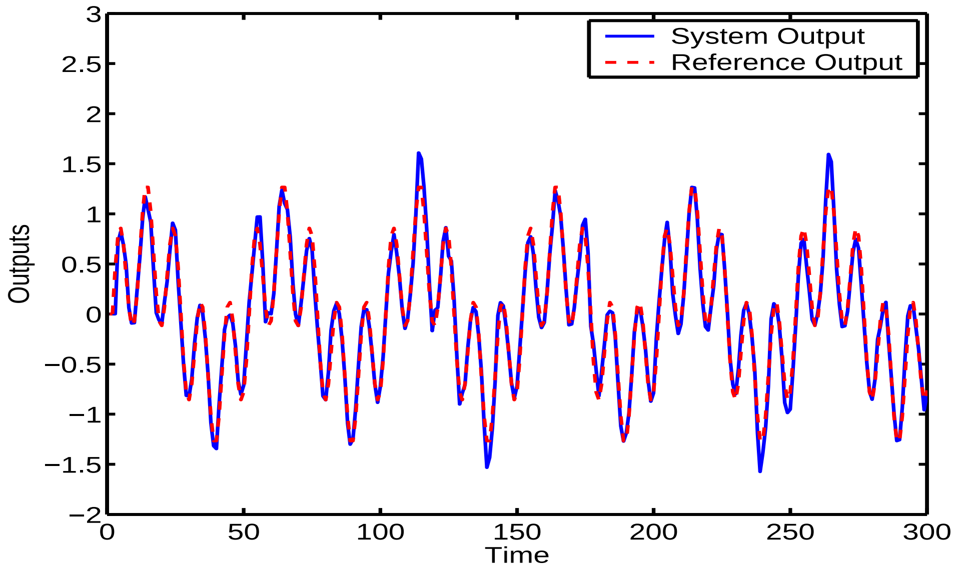


(a)

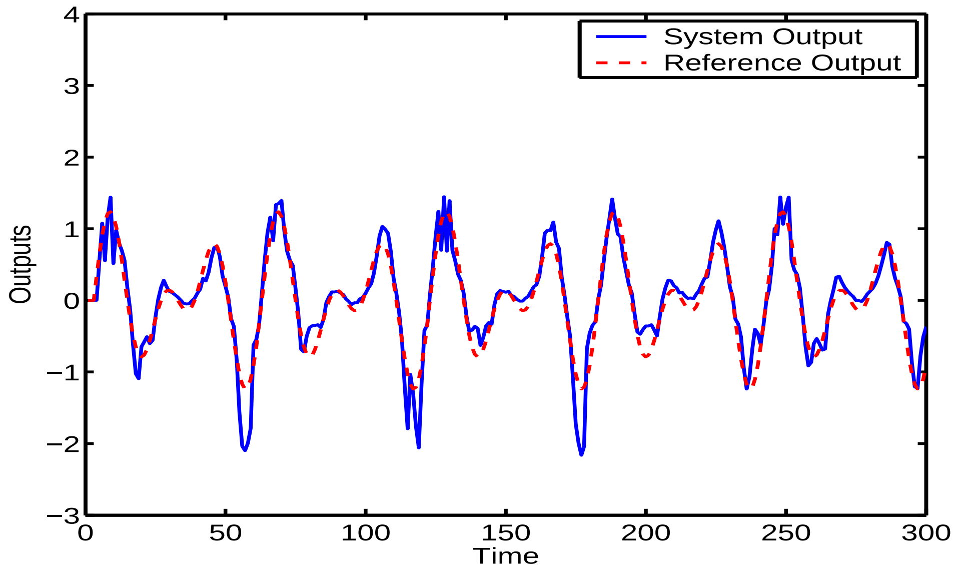


(b)

Fig. 5. Performance of the mixture density network as a controller: (a) the first output of the plant. (b) the second output of the plant.



(a)



(b)

Fig. 6. Performance of the proposed control approach of mixture density network for the dynamical MIMO system: (a) the first output of the plant. (b) the second output of the plant.