Probability Distribution Modelling to Improve Stability in Nonlinear MIMO Control

Randa Herzallah
NCRG, Aston University, UK
Email:herzarom@aston.ac.uk

David Lowe
NCRG, Aston University, UK
Email:d.lowe@aston.ac.uk

Abstract— **We consider the direct adaptive inverse control of nonlinear multivariable systems with different delays between every input-output pair. In direct adaptive inverse control, the inverse mapping is learned from examples of input-output pairs. This makes the obtained controller sub optimal, since the network may have to learn the response of the plant over a larger operational range than necessary. Moreover, in certain applications, the control problem can be redundant, implying that the inverse problem is ill posed. In this paper we propose a new algorithm which allows estimating and exploiting uncertainty in nonlinear multivariable control systems. This approach allows us to model strongly non-Gaussian distribution of control signals as well as processes with hysteresis. The proposed algorithm circumvents the dynamic programming problem by using the predicted neural network uncertainty to localise the possible control solutions to consider.**

I. Introduction

Recently, several authors have used neural networks for the identification and control of unknown nonlinear multivariable processes [7], [8]. Because of the nonlinearity of the plant, and the fact that the delays between input-output pairs can be different, the identification and control of multivariable systems is substantially more complex than that of (SISO) systems.

When the parameters of the plant are unknown, adaptive control is assumed. Indirect control architectures for multivariable systems, have been proposed in [7], [8], assuming exact models for identifier and controller. The most recent research interest is now to go beyond the classical methods for identification and control by accounting for model uncertainty. In [1] a systematic procedure that accounts for the structured uncertainty in the neural network model has been developed. It has been shown in this work that for the overall linear closed loop system, the propagation through the control loop of the structured uncertainty from the neural network parameters enables the construction of a polytopic uncertainty description. This in turn can provide a Lyapunov function for the uncertain system, and therefore proving robust stability of the overall control system. A new approach for adaptive output feedback control of uncertain nonlinear (MIMO) system has been introduced in [6]. In this approach an observer for the output tracking error rather than a state observer in the classical approaches has been proposed under the assumption that the system is feedback linearisable. A simple linear observer for the tracking error and a multi-layer perceptron neural network is used to cancel the modelling errors. Ultimate boundness of the error signals was shown through Lyapunov stability analysis.

An inversion based neurocontroller for solving control problems of uncertain nonlinear (SISO) systems has been presented in [3], [4]. The controller is designed to predict conditional distributions of control signals. A sampling approach is then used to search for a better value of control. The stability analysis for the updating rule of the control law was proven in [3]. The approach in [3], [4] is based on the assumption that the estimated distributions of control signals are Gaussian. A more general approach for sampling from non Gaussian distributions is discussed in [5].

In this paper we extend the approaches of [4], [5] to MIMO systems, which have different delays between every input-output pair. We show that the same computational procedure can be used except that the samples for searching for a better control law need to be generated from the estimated distribution of control signals in each dimension. This means that the number of samples grows exponentially with the dimension of control variables. A comparison between sampling from different distributions is also provided.

II. Distribution Modelling of Control Signals

In classical inverse control the challenge is to build a neural network that will take past values of the input and output of the plant $z(t) = [y(t-1), ..., y(t-n), u(t-2), ..., u(t-m)]$ and the desired output value $y_r(t)$ as an input, and outputs the control signals $u(t-d)$ (assuming d relative degree), which will move the plant output to the desired value. In this work the basic goal is to model the statistical properties of the control signals, $u(t-d)$, expressed in terms of the conditional distribution function $p(u(t-d)|s(t))$. Here $s(t) = [z(t), y_r(t)]$ is the input vector to the neural inverse model. For dynamical systems it is reasonable to assume that the output of the system $y(t)$ is function f of its input $u(t-d)$ and the delayed vector $z(t)$. Furthermore in the case of a one-to-one mapping, and only in this case, the inverse of the function denoted by f^{-1} can be introduced. In this case a feed-forward neural network trained using the sum of the square error function (between the input of the system and the actual output of the controller) can perform well. 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),

0-7803-7729-X/03/$17.00 ©2003 IEEE.

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 f^{-1} found by minimising the sum of the square error can not be used to tell us how to choose the control signal $u(t-d)$ so as to reach the desired response $y_r(t)$. Therefore, assuming a Gaussian distribution can lead to a very poor representation of the control signal. In this case a more general framework for modelling conditional probability distributions is required. This general framework is based on the use of the mixture density network.

A. Gaussian Distribution Modelling

If a neural network has been used to model the adaptive inverse controller, it can also model the conditional distribution of the target data (the control signal) by modelling the conditional uncertainty involved in its own predictions. In this work the predictive error bar method reported in [4] will be used. 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 $f_{opt}^{-1}(s(t)) =< u(t-d)|s(t) >$, and that the local variance of the target data can be estimated as $\|u(t-d) - f_{opt}^{-1}(s(t))\|^2$.

If this variance is used as a target value for another neural network, then the optimum output of this second network is again the conditional mean of that variance. As reported in [4], in the implementation of predictive error bars two correlated neural neural networks are used. Each network shares the same input and hidden nodes, but has different final layer links which are estimated to give the approximated conditional mean of the target data in the first network, and the approximated conditional mean of the variance in the second network. Thus the second network predicts the noise variance of the predicted mean by the first network. This architecture is shown in figure 1. Optimisation of the weights is a two stage process: The first stage determines the weights w_1 conditioning the regression on the mapping surface. Once these weights have been determined, the network approximations to the target values are known, and hence so are the conditional error values on the training examples. In the second stage the inputs to the network remain exactly as before, but now the target outputs of the network are the error values. This second pass determines the weights w_2 which condition the second set of output noise to the squared error values $\sigma^2(s(t))$.

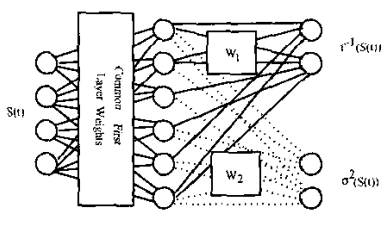


Fig. 1. The architecture of the predictive error bar network.

We will demonstrate the use of this noise (in a control architecture) soon, but first we discuss a more general method for distribution modelling which we need for multimodal and non-Gaussian control problems.

B. Mixture Density Network

For multi-valued functions, Mixture Density Networks (MDNs) [2] provide a general framework for modelling conditional probability density functions $p(u(t-d)|s(t))$ for the inverse mapping. The distribution of the outputs, $u(t-d)$, is described by a parametric model whose parameters are determined by the output of a neural network, which takes $s(t)$ as inputs. The general conditional distribution function is given by

$$p(u(t-d)|s(t)) = \sum_{j=1}^{M} \alpha_j(s(t))\phi_j(u(t-d)|s(t)) \quad (1)$$

where $\alpha_j(s(t))$ represents the mixing coefficients, and can be regarded as prior probabilities (which depend on $s(t)$), $\phi_j(u(t-d)|s(t))$ are the kernel distributions of the mixture model (whose parameters are also conditioned on $s(t)$), and M is the number of kernels in the mixture model. Various choices are available for the kernel functions, but in this paper the choice will be restricted to spherical Gaussians of the form

$$\phi_j(u(t-d)|s(t)) \alpha \exp\left(-\frac{\| u(t-d) - \mu_j(s(t)) \|^2}{2\sigma_j^2(s(t))}\right) \quad (2)$$

where $\mu_j(s(t))$ represents the centre of the jth kernel, with components μ_{jk}. A spherical Gaussian assumption can be relaxed in a very straightforward way, by using a full covariance matrix for each Gaussian kernel. However this complication is not necessary, because in principle a Gaussian Mixture Model (GMM) with sufficiently many kernels of the type given by (2) can approximate any given density function arbitrarily accurately providing that the mixing coefficients and the Gaussian parameters are correctly chosen [2]. It follows then that for any given value of $s(t)$, the mixture model (1) provides a general formalism for modelling the conditional density function $p(u(t-d)|s(t))$. To achieve this the parameters of the mixture model, namely the mixing coefficients $\alpha_j(s(t))$, the means $\mu_j(s(t))$ and the variance $\sigma_j^2(s(t))$ are taken to be general continuous functions of $s(t)$. These functions are modelled by the outputs of a feed-forward neural network that takes $s(t)$ as input.

The neural network element of the (MDN) is implemented with a standard multi-layer perceptron network (MLP) of tanh functions. The output vector from the MLP, Z, returns the parameters that define the Gaussian mixture model. For M components in the mixture model (1) the network will have $(c+2) \times M$ outputs, namely M outputs denoted by z_j^α which determines the mixing coefficients α_j, M outputs denoted by z_j^σ which determine the kernel width σ_j, and $M \times c$ outputs denoted by z_{jk}^μ which determine the components μ_{jk} of the kernel centres μ_j. This is compared with the usual c outputs for a MLP network used with a sum-of squares error function.

The outputs of the MDN undergo some transformations to satisfy the constraints of the mixture model [2], [5]. In order to optimise the parameters in a MDN, an error function is required that provides an indication of how well the model represents the underlying generating function of the training data. The error function of the mixture density network is motivated from the principle of maximum likelihood [2]. The likelihood of the training data set, $\{s(t), u(t-d)\}$, can be written as

$$\mathcal{L} = \prod_n p(u_n(t-d)|s_n(t))p(s_n(t)) \quad (3)$$

where here the assumption has been made that each data point has been drawn independently from the same distribution, and so the likelihood is a product of probabilities. Generally one wishes to maximise the likelihood function. However, in practice, it is often more convenient to consider the negative logarithm of the likelihood function. These are equivalent procedures, since the negative logarithm is a monotonically decreasing function. The negative log likelihood can be regarded as an error function, E

$$E = -\sum_n \ln p(u_n(t-d)|s_n(t)) - \sum_n p(s_n(t)) \quad (4)$$

In order to minimise the error function, the derivatives of the error E with respect to the weights in the neural networks 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 [2], [5]. By first defining the posterior probability of the jth kernel, using Bayes theorem

$$\pi_j(s(t), u(t-d)) = \frac{\alpha_j \phi_j}{\sum_{l=1}^{M} \alpha_l \phi_l} \quad (5)$$

the analysis of the error derivatives with respect to the network outputs is simplified. The computation of the error can further be simplified by considering the error derivative with respect to each training pattern, n. The total error, E, is defined as a summation of the error, E_n, for each training pattern. $E = \sum_{n=1}^{N} E^n$, where

$$E^n = -\ln\left\{\sum_{j=1}^{M} \alpha_j(s_n(t))\phi_j(u_n(t-d)|s_n(t))\right\} \quad (6)$$

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

$$\frac{\partial E^n}{\partial z_j^\alpha} = \alpha_j - \pi_j \quad (7)$$

$$\frac{\partial E^n}{\partial z_j^\sigma} = -\frac{\pi_j}{2}\left\{\frac{\| u_n(t-d) - \mu_j \|^2}{\sigma_j^2} - c\right\} \quad (8)$$

$$\frac{\partial E^n}{\partial z_{jk}^\mu} = \pi_j\left\{\frac{\mu_{jk} - u_k(t-d)}{\sigma_j^2}\right\} \quad (9)$$

for full derivation see [2]. 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. An example of these quantities is the mean, corresponding to the conditional average of the target data. This corresponds 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, this is a not valid solution. In such cases one may be interested in finding an output value corresponding to the most probable branch. Since each component of the mixture model is normalised, $\int \phi_j(u(t-d)|s(t))du(t-d) = 1$, the most probable branch is given by $\arg\max_j\{\alpha_j(s(t))\}$. The required value of $u(t-d)$ is then given by the corresponding centre μ_j.

III. Problem Formulation and Solution Development

Dynamic programming is a powerful tool in stochastic control problems. 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 for nonlinear control problems. In contrast to the classical control approaches, suppose that the control vectors are generated from some probability distribution $p(u(t-d))$, and the output vectors evolve with time according $y(t) = f(z(t), u(t-d))$. The objective in control problems is then to find the optimal control variables from the probability distribution $p(u(t-d))$ such that when applied to the system, the output of the system should be equal to a predetermined desired value $y_r(t)$. This means that we are looking for an optimal control vector $u(t-d)$, obtained from the distribution $p(u(t-d))$ such that

$$p[|y(t) - y_r(t)| > 0] = 0 \quad (10)$$

However this cannot be applied directly to real world problem, because we need to observe the effect of each control variable from the distribution $p(u(t-d))$ on the real world system. Since we can only apply one decision input to the real system which is supposed to be optimal, the real world system needs to be replaced by an estimation $\hat{y}(t)$. Consequently this implies that the solution provided in Eq 10 never occurs in practice because it requires that the estimator $\hat{y}(t)$ for $y(t)$ contains no error. Moreover to satisfy this condition we need to know the true probability distribution $p(u(t-d))$, where in practice we can only estimate $\hat{p}(u(t-d))$. In this work we do not consider this kind of uncertainty. Further development will be based on the assumption that the estimated distributions are accurate. Providing that we have a valid estimation for the true distribution $p(u(t-d))$ and the estimator of the forward

model, the statistical control optimisation problem proposed in this work can be stated as follows. Given:

- A set U, consisting of all possible decisions $u \in U$ obtained from the probability density function $\hat{p}(u(t-d))$
- A performance criterion J which provides an evaluation of a given decision variable
- 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 minimises or maximises the performance criterion J at each instant of time.

Based on the estimated distribution of control signals we can construct the following algorithm incorporating uncertainty directly.

1) Incorporating Uncertainty For the Gaussian Distribution Function:

1) Based on the pre-collected input-output data, an accurate model of the process is constructed and trained off line. It is assumed to be described by $\hat{y}(t) = f(z(t), u(t-d))$
2) An accurate inverse model of the plant should also be constructed, and trained off line to approximate the conditional mean of the control vector and the conditional variance. It is assumed to be described by the following neural network

$$h(t) = f^{-1}(y(t), z(t)) \tag{11}$$

where $\hat{u}(t-d) = h(t)w_1$, $var_{u(t-d)} = h(t)w_2$, and where $h(t)$ is the predicted hidden variable from the neural network at each instant of time t, w_1 is the weight of the linear layer estimated to predict the conditioned mean of the control signal, and w_2 is the weight of the linear layer estimated to predict the variance of the predicted control signal.
3) At each instant of time t the desired output is calculated from the reference model output, which should be chosen to have the same relative degree as that of the plant.
4) Bring the control network on line and at each time t estimate the appropriate control signal from the controller and the variance of that control signal. The control signal distribution is then assumed to be Gaussian and given by

$$p(u(t-d) \mid s(t)) \alpha \exp(-\frac{(u(t-d)-\hat{u}(t-d))^2}{2\sigma^2})$$

where σ^2 is the variance of the control signal $var_{u(t-d)}$.
5) Generate a vector of samples from the control signal distribution. For example for a two dimensional problem a vector of samples need to be generated from the distribution of each control signal. The admissible control values at each instant of time is then generated using a two dimensional grid to represent the error surface. The state however, does not need to be tracked since we consider that the neural network has been optimised off line to predict the control values and the uncertainty of the control values at each instant of time. This gives an advantage over the dynamic programming approach. The number of samples in each dimension can be chosen in two ways:(1)based on the value of the predicted variance of the control signal $number of samples = K \times var_{u(t-d)}$. This equation determines the number of samples based on the confidence of the controller about the predicted mean value of the control signal. So more samples are generated for larger variance. (2)a fixed number of samples can be used.
6) Based on the effect of each sample on the output of the model, the most likely control value is taken, which is assumed to be the value that minimises the following cost function.

$$J(t) = \min_{u \in U} \underset{\bar{v}}{E}[(\hat{y}(t) - y_r(t))^2] \tag{12}$$

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 $\bar{v}$. 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 optimised directly.

2) Incorporating Uncertainty For the Mixture Density Network: Since we have covered a lot of ground of the proposed algorithm in our discussion for incorporating uncertainty for the Gaussian function case, we summarise here the main differences between the two approaches:

1) The conditional distribution of the inverse model of the plant in step 2 for the Gaussian distribution function, is assumed to be described by a mixture density network given by equation (1).
2) For the non-sampling case, in the mixture density network the value of the control signal is assumed to be given by the centre μ_j of the most probable branch, where the most probable branch is given by

$$\arg\max_j \{\alpha_j(s(t))\} \tag{13}$$

While the predicted value of the control signal for the Gaussian distribution function is assumed to be equivalent to the mean of that distribution.
3) The admissible values of the control signal at each instant of time for the Gaussian distribution case are assumed to be sampled from that distribution, as in point 5. The admissible values of the control signal for the mixture density network, are assumed to be sampled from a mixture density network. Since we are using Gaussian kernel functions 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 width 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.

IV. Simulation Study

A. Introduction

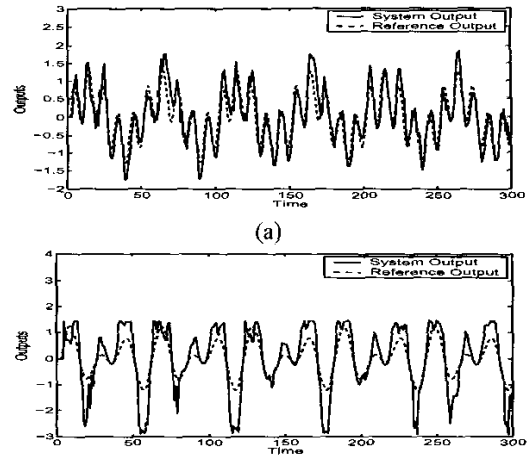
In order to illustrate the validity of the theoretical developments for MIMO systems, we consider a third order system with two inputs and two outputs described by the following state equation :

$$
\begin{aligned}
x_1(t+1) &= 0.9x_1(t)\sin[x_2(t)] \\
&\quad + \left[2 + 1.5\frac{x_1(t)u_1(t)}{1+x_1^2(t)u_1^2(t)}\right]u_1(t) \\
&\quad + \left[x_1(t) + \frac{2x_1(t)}{1+x_1^2(t)}\right]u_2(t) \\
x_2(t+1) &= x_3(t)\{1+\sin[4x_3(t)]\} + \frac{x_3(t)}{1+x_3^2(t)} \\
x_3(t+1) &= \{3 + sin[2x_1(t)]\}u_2(t) \\
y_1(t) &= x_1(t) \qquad y_2(t) = x_2(t)
\end{aligned} \tag{14}
$$

where $x(t) = [x_1(t), x_2(t), x_3(t)]$ is the state, $u(t) = [u_1(t), u_2(t)]$ is the control variable, and $y(t) = [y_1(t), y_2(t)]$ is the output. This model has been used in [8] 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 has been considered to be described by equation (14). 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. An input-output model described $\hat{y}_1(t+1) = N_{f1}(y(t), y(t-1), y(t-2), u(t), u(t-1), u(t-2)), \hat{y}_2(t+2) = N_{f2}(y(t), y(t-1), y(t-2), u(t), u(t-1), u(t-2))$ was chosen to find the forward model of the first and second outputs respectively, where N_{f1}, and N_{f2} are multi-layer neural networks. These neural network models were trained using the scaled conjugate gradient optimisation 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 network found by applying the cross validation method consisted of 21 hidden units for the first model and 17 hidden units for the second model. Similarly an input output model described by $\hat{u}(t) = N_c(y(t), y(t-1), y(t-2), u(t-1), u(t-2), [y_1(t+1), y_2(t+2)])$ 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.

B. Classical Inverse Control Approach

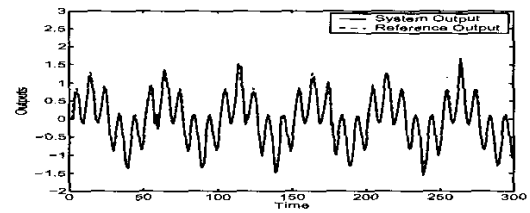
After training the inverse controller off line, the control network is brought on line and the control signal is calculated at each instant of time from the control neural network and by setting the two outputs $y_1(t+1)$, $y_2(t+2)$ equal to the desired values $y_{r1}(t+1) = r_1(t)$, and $y_{r2}(t+2) = r_2(t)$ respectively. Here $r_1(t) = 0.65\sin[\frac{2\pi t}{50}] + 0.65\sin[\frac{2\pi t}{10}]$, and



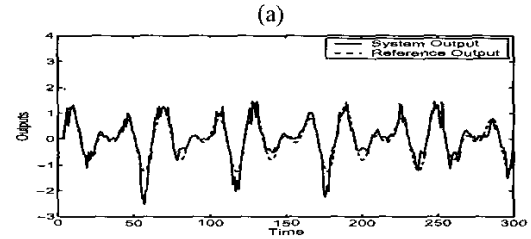
(a)

(b)

Fig. 2. Performance of the classical control approach: (a) the first output of the plant. (b) the second output of the plant.



(a)



(b)

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

$r_2(t) = 0.65\sin[\frac{2\pi t}{30}] + 0.65\sin[\frac{2\pi t}{20}]$. The predicted mean value from the neural network was forwarded to the plant. The control result is shown in Fig. 2. The performance of the classic controller was seen to be poor with large overshoots around the desired response in the second output $y_2(t)$.

C. Sampling from Gaussian distribution

In our new approach, the full distribution of the control signal was estimated. In the first experiment, the distribution of the control signal was estimated by Gaussian distribution. Following the procedure presented earlier, the best control signal was found and forwarded to the plant. Firstly 20 samples have been generated from the Gaussian distribution of each control signal. However the number of samples used to search for optimal control law was 21^2, including the mean value from

each distribution. The overall performance of the plant under the proposed method is shown in Fig. 3. 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. Although in this experiment we have used 21^2 samples, the determination of the number of samples still an open question. The effect of the number of samples on the accuracy for the optimal control strategy will be discussed in the next section.

D. Sampling from Other Distributions

In this section we are considering the problem of determining an appropriate number of samples. In addition, sampling from other distributions such as uniform and non-Gaussians will be considered. Firstly several experiments have been performed with different numbers of samples from the Gaussian distribution. The dashed curve in figure 4 shows the result. It is clear from this figure that the tracking error has dropped very quickly using only a small number of samples. Increasing the number of samples however, does not help in reducing the tracking error once the minimum has been reached.

Following the discussion in section II-B, the conditional distribution of control signals has been estimated by a mixture of Gaussians. A mixture density network with 2 components and 7 hidden units was found to be the best model using cross validation. The result of determining the optimal control law by sampling from non Gaussian function and for different number of samples is given by the solid curve in figure 4. A slights improvement compared to the Gaussian case has been achieved in this particular problem. The tracking error using the mixture density network is less than that obtained in the Gaussian case. In addition, sampling from a non-Gaussian function results in reducing the tracking error quicker than the Gaussian case.

Although we are suggesting estimating the distribution of the control signals and then sampling from that distribution to find the optimal control law, one may think of defining an arbitrary lower and upper bound around the control signals and searching uniformly for the optimal control values in that range. Indeed this can be done, but has several disadvantages. First, several experiments need to be done to find the right bounds for sampling. In addition, severe instability problems result if the control signal happens to be outside the operating range which may result because of sampling from the wrong distribution. Figure 4 shows several curves with different bounds for uniform sampling. It has been noticed during running these experiments that although the overall tracking error can be better than using the mean value from the estimator, the result of the controller can be very bad around certain operating points where the optimal control value cannot be found. This situation has not been noticed in the Gaussian and mixture density case.

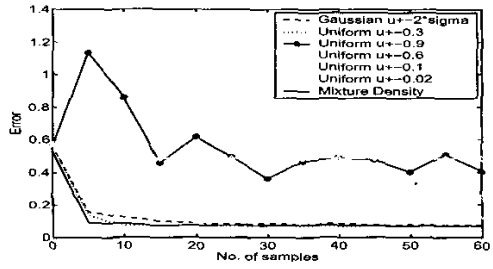


Fig. 4. The tracking error from sampling different distribution functions.

V. Conclusion

Incorporating uncertainty estimation to improve the performance of controllers for multivariable control problems is considered in this paper. A method that uses uncertainty around the predicted mean value of the control signal was proposed. The proposed method allows for the control signal to be adapted from its distribution, to obtain a better estimate of the control signal than the mean. The proposed control strategy in this work chooses the optimal control value almost in the same way as in dynamic programming. However, the proposed method is computationally more efficient and is not based on the use of the recurrence relation as in dynamic programming. This is because the estimated mean and variance are predicted from a neural network which is supposed to be optimised on the input output data and so constrains the sampling space just to the feasible region. By predicting the full distribution of the control signal from the neural network, searching for a better value of the control signal than the mean can be performed only in this region in which the optimal solution is expected to lie. Simulation experiments demonstrated the successful application of the proposed strategy to improve the controller performance for multivariable problems.

References

[1] Miguel Ayala Botto, Bart Wams, Ton van den Boom, and José Sá da Costa. Robust stability of feedback linearised systems modelled with neural networks: dealing with uncertainty. *Engineering Applications of Artificial Intelligence*, 13(6):659–670, 2000.

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

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

[4] 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.

[5] 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*, Faro, Portugal, April 2003.

[6] N. Hovakimyan and A.J. Calise. Adaptive output feedback control of uncertain multi-input multi-output systems using single hidden layer neural networks. In *Proceedings of the American Control Conference*, pages 1555–1560, Anchorage, Alaska, USA, May 2002.

[7] K.S. Narendra and K.Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1:4–26, 1990.

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