Robust Control of Nonlinear Stochastic Systems by Modelling Conditional Distributions of Control Signals

Randa Herzallah David Lowe
NCRG, Aston University, UK
email: herzarom,d.lowe@aston.ac.uk

May 8, 2003

Abstract

We introduce a novel inversion-based neurocontroller for solving control problems involving uncertain nonlinear systems which could also compensate for multi-valued systems. The approach uses recent developments in neural networks, especially in the context of modelling statistical distributions, which are applied to forward and inverse plant models. Provided that certain conditions are met, an estimate of the intrinsic uncertainty for the outputs of neural networks can be obtained using the statistical properties of networks. More generally, multicomponent distributions can be modelled by the mixture density network. In this work a novel robust inverse control approach is obtained based on importance sampling from these distributions. This importance sampling provides a structured and principled approach to constrain the complexity of the search space for the ideal control law. The performance of the new algorithm is illustrated through simulations with example systems.

1 Introduction

In nonlinear stochastic control problems, once the objective functional is defined we would ideally seek a dynamic programming solution. This however, is practically unfeasible, not least because of the unbounded search space in which we need to maintain possible solution trajectories. The method of approximation we choose is to construct nonlinear neural network models for the forward and inverse plant dynamics. However we are interested in intrinsically stochastic systems. Since standard neural network approaches produce deterministic system approximations, we need

a way to allow for sampling from the (unknown) distribution of control signals which would be generated by the real stochastic system. We achieve this by employing the same neural networks to estimate error variances around the predicted mean values of the control values, thus characterising the distribution of the control signals as Gaussian. For inverse problems, the mapping can be often multi-valued, with values of the inputs (tracking signal) for which there are several valid values for the outputs (control signals). In this case, mixture density networks can be implemented to model the more general distribution of the control signal.

In recent years, neural network models have evolved into favourite candidates in the field of nonlinear system identification and control, due to their ability to approximate multi-variable nonlinear mappings. In addition to having nonlinear features, dynamic systems may have noise events affecting their inputs and outputs, and usually are time-variant. Because artificial neural networks can be adapted on line [15, 4, 13], usually they are capable of good approximation in such situations. However for most real control problems where disturbances play an important part and where a relatively big sampling interval is used, the predicted output of the neural network is inherently uncertain. Neural networks now have the ability to model general distributions rather than just producing point estimates, and in particular can produce an estimate of the uncertainty involved in the predictions [3, 7, 16]. Recent research interest has been to go beyond the classical methods for identification and control by accounting for model and system uncertainty explicitly in the modelling process. As examples, in [2] a systematic procedure that accounts for the structured uncertainty when a neural network model is integrated in an approximate feedback linearisation control scheme has been developed. The use of an adaptive critic controller when there is input uncertainty has been discussed in [5]. The application of recently developed minimal resource allocating network ($MRAN$) in a robust manner under faulty conditions has been demonstrated in [14]. A robust adaptive nonlinear control method for controlling a class of nonlinear systems in the presence of both unknown nonlinearities and unmodelled dynamics has been illustrated in [8]. In [3, 11] a new class of network models obtained by combining a conventional neural network with a mixture density model, has been used to model the conditional probability distribution for problems in which the mapping to be learned is multi-valued. Other computational approaches, namely forward and inverse modelling, and feedback error learning have been suggested in [15, 9] for acquiring the inverse dynamics model of the multi-valued functions. None of the recent works have considered the possibility of using the neural network's own estimate for error bars. In this paper we address for the first time the use of this extra knowledge to develop a robust control method for uncertain nonlinear systems. This paper aims to demonstrate that a promising approach to robust control can be provided by

this proposed framework.

The paper begins with a review of the principle of system model and error bar estimation. Next, we develop a nonlinear controller architecture based on approximate dynamic inversion and the use of error bar knowledge. This development is then employed to control a nonlinear stochastic simulated system.

2 Adaptive Inverse Control

The classical inverse adaptive control technique is shown in figure 1. The neural network is learning to recreate the input $u(t-d)$, that created the current output of the plant $y(t)$. The inverse controller contains adjustable parameters that control its impulse response. An adaptive algorithm is usually used to automatically adjust the controller parameters to minimise some function of the error (usually mean square error, though other error functions can also be used). The error is defined as the difference between the input of the plant $u(t-d)$, and the actual output of the controller $\hat{u}(t-d)$. Many such algorithms are described in the reports and textbooks by Narendra and Parthasarathy [13] and by White and Sofge [15].

When trained, the network should be able to take the desired response y_r and produce the appropriate control signal u, which is then supposed to make the plant output y approach the desired response y_r. This control architecture however, may not be efficient since the network may have to learn the response of the plant over a larger operational range than is actually necessary. This problem is related to the concept of persistent excitation, which acknowledges the importance of the inputs used to train learning systems. A preliminary discussion for this concept can be found in [12].

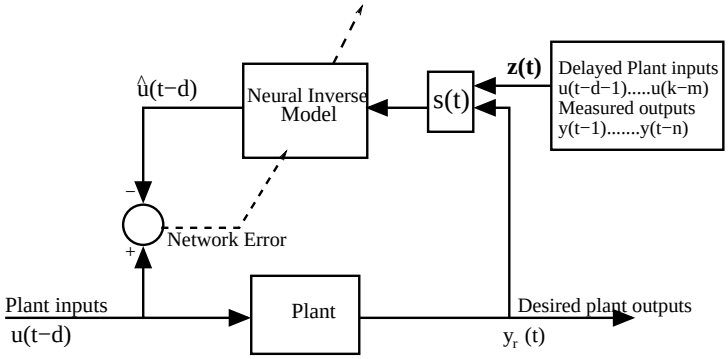

Figure 1: Training of an inverse controller.

3 Distribution Modelling

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 $\boldsymbol{z}(t) = [y(t-1), ..., y(t-n), u(t-d-1), ..., u(t-m)]$ and the desired output value $y_r(t)$ as an input, and outputs the control signals $u(t-d)$ (assuming a relative degree of d), which will move the plant output to the desired value. In this work the primary goal is to model the statistical properties of the control signals, $u(t-d)$, expressed in terms of the conditional distribution function $p(u(t-d)|\boldsymbol{s}(t))$. Here $\boldsymbol{s}(t) = [\boldsymbol{z}(t), y_r(t)]$ 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 a function f of its input $u(t-d)$ and the delayed vector $\boldsymbol{z}(t)$. 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 example 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 instance 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 cannot be defined uniquely, then the direct inverse mapping f^{-1}, found by using the sum of the square error function between the input of the system and the actual output of the controller, cannot 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, the assumption of a Gaussian distribution can lead to a very poor representation of the control signal. For this situation a more general framework for modelling conditional probability distributions is required. This general framework is based on the use of the mixture density network.

3.1 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. Different methods for estimating the uncertainty around the predicted output of a neural network have been presented in [3, 7, 16]. In this work the predictive error bar method will be used [7]. 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 $f_{opt}^{-1}(\boldsymbol{s}(t)) =< u(t-d)|\boldsymbol{s}(t) >$, and that the local variance of the target data can be estimated as $\|u(t-d) - f_{opt}^{-1}(\boldsymbol{s}(t))\|^2$. 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 [7], 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 2. 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 networks 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 $\sigma^2(\boldsymbol{s}(t))$.

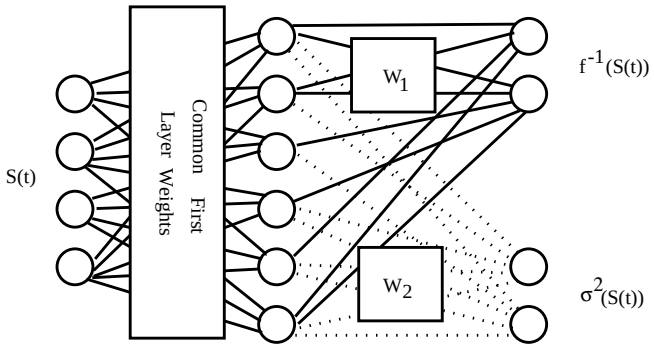

Figure 2: The architecture of the predictive error bar network.

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

3.2 Mixture Density Network

For multi-valued functions, Mixture Density Networks $MDNs$ [3] provide a general framework for modelling conditional probability density functions $p(u(t-d)|\boldsymbol{s}(t))$ 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 $\boldsymbol{s}(t)$ as inputs. The general conditional distribution function is given by

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

where $\alpha_j(\boldsymbol{s}(t))$ represents the mixing coefficients, and can be regarded as prior probabilities (which depend on $\boldsymbol{s}(t)$), $\phi_j(u(t-d)|\boldsymbol{s}(t))$ are the kernel distributions of the mixture model (whose parameters are also conditioned on $\boldsymbol{s}(t)$), 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

$$\phi_j(u(t-d)|\boldsymbol{s}(t)) = \frac{1}{(2\pi)^{c/2}\sigma_j^c(\boldsymbol{s}(t))} \exp\left(-\frac{\| u(t-d) - \mu_j(\boldsymbol{s}(t)) \|^2}{2\sigma_j^2(\boldsymbol{s}(t))}\right) \tag{2}$$

where c is the dimensionality of the target data $u(t-d)$, $\mu_j(\boldsymbol{s}(t))$ 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, using full covariance Gaussian 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 [3]. It follows that for any given value of $\boldsymbol{s}(t)$, the mixture model (1) provides a general formalism for modelling the conditional density function $p(u(t-d)|\boldsymbol{s}(t))$. To achieve this the parameters of the mixture model, namely the mixing coefficients $\alpha_j(\boldsymbol{s}(t))$, the means $\mu_j(\boldsymbol{s}(t))$ and the variance $\sigma_j^2(\boldsymbol{s}(t))$ are taken to be general continuous functions of $\boldsymbol{s}(t)$. These functions are modelled by the outputs of a feed-forward neural network that takes $\boldsymbol{s}(t)$ as input.

The neural network element of the MDN is implemented with a standard radial basis function network RBF of thin plate spline basis functions. The output vector from the RBF, Z, holds 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 determine 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 RBF 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. The constraints are such that

$$\sum_{j=1}^{M} \alpha_j(\boldsymbol{s}(t)) = 1 \tag{3}$$

$$0 \leq \alpha_j(\boldsymbol{s}(t)) \leq 1 \tag{4}$$

The first constraint ensures that the distribution is correctly normalised, so that $(\int p(u(t-d)|\boldsymbol{s}(t))\mathrm{d}u(t-d) = 1)$. These constraints can be satisfied by choosing $\alpha_j(\boldsymbol{s}(t))$ to be related to

the network's outputs by a 'softmax' function

$$\alpha_j(\boldsymbol{s}(t)) = \frac{\exp(z_j^\alpha)}{\sum_{l=1}^{M} \exp(z_l^\alpha)}. \tag{5}$$

The variances of the kernel represent scale parameters and always take positive values, so it is convenient to represent them in terms of exponentials of the corresponding outputs of the RBF network, z_j^σ

$$\sigma_j^2 = \exp(z_j^\sigma). \tag{6}$$

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 RBF network, z_{jk}^μ

$$\mu_{jk} = z_{jk}^\mu. \tag{7}$$

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 [3]. The likelihood of the training data set, $\{\boldsymbol{s}(t), u(t-d)\}$, can be written as

$$\begin{aligned} \mathcal{L} &= \prod_n p(\boldsymbol{s}_n(t), u_n(t-d)) \\ &= \prod_n p(u_n(t-d)|\boldsymbol{s}_n(t)) p(\boldsymbol{s}_n(t)) \end{aligned} \tag{8}$$

where 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

$$E = -\ln \mathcal{L} = -\sum_n \ln p(u_n(t-d)|\boldsymbol{s}_n(t)) - \sum_n p(\boldsymbol{s}_n(t)). \tag{9}$$

The second term in (9) is constant because it is independent of the network parameters, so it can be removed from the error function. The error function becomes

$$E = -\ln \mathcal{L} = -\sum_n \ln p(u_n(t-d)|\boldsymbol{s}_n(t)). \tag{10}$$

Next we substitute (1) into (10) and derive the negative log likelihood error function for the MDN

$$E = -\sum_n \ln\left\{ \sum_{j=1}^{M} \alpha_j(\boldsymbol{s}_n(t))\phi_j(u_n(t-d)|\boldsymbol{s}_n(t)) \right\}. \tag{11}$$

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 [3]. By first defining the posterior probability of the jth kernel, using Bayes theorem

$$\pi_j(\boldsymbol{s}(t), u(t-d)) = \frac{\alpha_j \phi_j}{\sum_{l=1}^{M} \alpha_l \phi_l} \tag{12}$$

the analysis of the error derivatives with respect to the network outputs is simplified. From (12) the posterior probabilities sum to unity

$$\sum_{j=1}^{M} \pi_j = 1. \tag{13}$$

Since the error function (11) is composed of a sum of terms $E = \sum_n E^n$, the computation of the error derivative can further be simplified by considering the error derivative with respect to each training pattern, n. The total error E is then defined as a sum of the error, E^n, for each training pattern

$$E = \sum_{n=1}^{N} E^n. \tag{14}$$

Each of the derivatives of E^n are considered with respect to the outputs of the network 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

$$\frac{\partial E^n}{\partial z_j^\alpha} = \alpha_j - \pi_j \tag{15}$$

$$\frac{\partial E^n}{\partial z_j^\sigma} = -\frac{\pi_j}{2}\left\{ \frac{\| u_n(t-d) - \mu_j \|^2}{\sigma_j^2} - c \right\} \tag{16}$$

$$\frac{\partial E^n}{\partial z_{jk}^\mu} = \pi_j \left\{ \frac{\mu_{jk} - u_k(t-d)}{\sigma_j^2} \right\} \tag{17}$$

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 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, $\int \phi_j(u(t-d)|\boldsymbol{s}(t))\mathrm{d}u(t-d) = 1$, the most probable branch is given by

$$\arg\max_j\{\alpha_j(\boldsymbol{s}(t))\}. \tag{18}$$

The required value of $u(t-d)$ is then given by the corresponding centre μ_j. In this work the MDN will be used to model the conditional density function in case of a multi-valued function.

4 Problem Formulation and Solution Development

Dynamic programming is a powerful tool in stochastic control problems [6, 10]. 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 stochastic control problems. We search for an algorithmic approach yielding numerical solutions to the minimisation problem. The proposed method is equivalent to sampling values from the distribution of u and using the function value alone to determine a reasonable minimisation of the objective, $J(t)$. Using the gradient information of $J(t)$, although it would be more efficient, is not exploitable here due to the random sampling nature of the algorithm. In the proposed method we assume that we know the set of decisions allowable at any stage which can be determined from the distribution of the control signals, the effect of these decisions or the model of the process, and the criterion by which we evaluate the control policy that is employed.

4.1 Neural Network Development for Incorporating Uncertainty

Once properly trained, the inverse model can be used to control the plant since it can generate 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 behaviour 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 needing to be considered. 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. Modelling the conditional distribution of the control signals permits the idea of implementing importance sampling of the control signal distribution, which define the set of allowable decisions at each stage producing 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 behaviour.

4.1.1 Incorporating Uncertainty For the Gaussian Distribution Function

Based on estimates of the distribution of control signal values, we can construct the following algorithm incorporating the uncertainty directly. The architecture of this algorithm is shown in figure 3.

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 the following neural network model

$$\hat{y}(t) \quad = \quad f(y(t-1), ..., y(t-n), u(t-d), ..., u(t-m)) \tag{19}$$

where $y(t)$ is the measured plant output, $u(t)$ is the measured plant input, n is the maximum delay in the output, m is the maximum delay in the input, and d is a known relative degree of the plant.

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. Assuming the following hidden variable of the neural network,

$$x(t) \quad = \quad f^{-1}(y(t), y(t-1), ..., y(t-n), u(t-d-1), ..., u(t-m)) \tag{20}$$

the conditional mean of the controller is $\hat{u}(t-d) = x(t)w_1$, and the conditional variance is $var_{u(t-d)} = x(t)w_2$. Here w_1 is the weight matrix of the linear layer estimated to predict the conditioned mean of the control signal, and w_2 is the weight matrix 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, d, 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

$$p(u(t-d) \mid \boldsymbol{s}(t)) = \frac{1}{(2\pi\sigma^2_{u(t-d)})^{\frac{1}{2}}} \exp(-\frac{(u(t-d)-\hat{u}(t-d))^2}{2\sigma^2_{u(t-d)}}) \tag{21}$$

where $\sigma^2_{u(t-d)}$ is the variance of the control signal $\boldsymbol{s}(t) = [y(t), y(t-1), ..., y(t-n), u(t-d-1), ..., u(t-m)]$.

5. Generate a vector of samples from the control signal distribution. Since Gaussian distribution, the Matlab random number generator can be used. That vector of samples is considered as the admissible control values at each instant of time. The number of samples is chosen based on the value of the predicted variance of the control signal as, $number\,of\,samples = K \times var_{u(t-d)}$. 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.

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.

$$J(t) = \underset{u \in U}{Min} \underset{v}{E}[(\hat{y}(t) - y_r(t))^2] \tag{22}$$

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 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.

7. Go to step 3.

4.1.2 Incorporating Uncertainty for the Mixture Density Network

Since we have discussed most of the proposed algorithm in our discussion for incorporating uncertainty in the Gaussian distribution case, we summarise here the main differences between the two algorithms:

1. The conditional distribution of the inverse model of the plant mentioned in step 2 for the Gaussian distribution function, is assumed to be described by the MDN given by equation (1).

2. For the non-sampling case, the value of the control signal in the MDN is assumed to be given by the centre μ_j of the most probable branch, where the most probable branch is given by

$$\arg\max_j \{\alpha_j(\boldsymbol{s}(t))\}. \tag{23}$$

   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 step 5. The admissible values of the control signal for the mixture density network, are assumed to be sampled from a MDN. 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.

Other steps are the same as in the Gaussian distribution case.

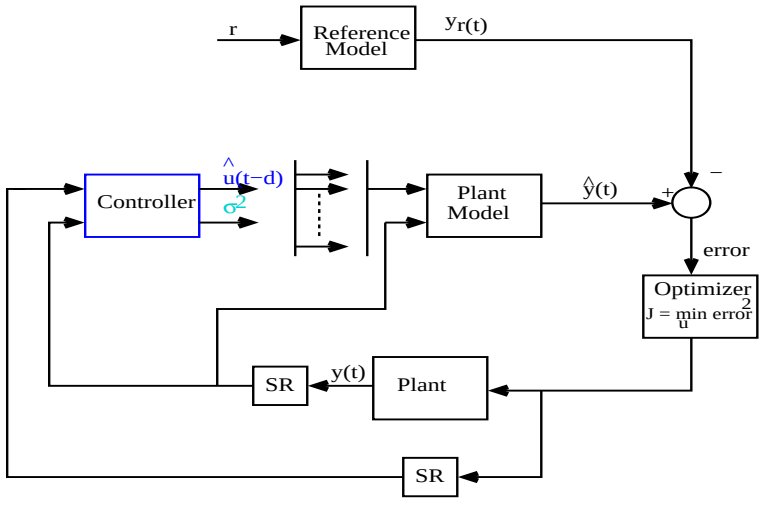

Figure 3: The architecture of the proposed optimisation method. The input and the output of the plant are passed through a shift register (SR) so as to generate the required past input and output values

5 Simulation 1, Gaussian Distribution

5.1 Introduction

In order to illustrate the validity of the theoretical developments, we consider the liquid-level system described by the following second order equation

$$\begin{aligned} y(t) &= 0.9722y(t-1) + 0.3578u(t-1) - 0.1295u(t-2) - 0.3103y(t-1)u(t-1) \\ &- 0.04228y^2(t-2) + 0.1663y(t-2)u(t-2) + 0.1087y(t-2)u(t-1)u(t-2) \\ &- 0.3513y^2(t-1)u(t-2) + 0.3084y(t-1)y(t-2)u(t-2) \\ &- \bar{v}y^2(t-1)y(t-2). \end{aligned} \tag{24}$$

This model has been used in [1] to illustrate theoretical developments for direct adaptive control. Because disturbances play an important part in real world processes, a stochastic component, $\bar{v}$, has been added to this model. This component is taken to be a Gaussian random variable $\mathcal{N}(0.03259, 0.2)$. The plant has been considered to be described by equation (24). Given the prior information concerning the order of the plant, a second order input-output model described by the following equation was chosen to identify the plant:

$$\hat{y}(t) = f(y(t-1), y(t-2), u(t-1), u(t-2))$$

where f is a Gaussian radial basis function network. This neural network model was trained using the scaled conjugate gradient optimisation algorithm, based on noisy input output data measurements taken from the plant with sampling time of $1s$. The input to the plant and the model was a sinc function followed by a sine wave in the interval $[-1, 1]$ with additive Gaussian noise $\mathcal{N}(0, \sigma^2)(\sigma = 0.3)$. Constructing an excitation signal capable of persistent excitation in nonlinear control systems is a known problem. In example (24) we found that the suggested plant input adequately explored the nonlinear control problem across the desired operating range. The single optimal structure for the neural network found by applying the cross validation method consisted of 6 Gaussian basis functions. If the order of the plant to be controlled is assumed to be unknown, cross validation method needs to be implemented to find the optimal order of the model.

Similarly an input-output model described by the following equation was chosen to find the inverse model of the plant:

$$\hat{u}(t-1) = f^{-1}(y(t-1), y(t-2), y(t), u(t-2))$$

where f^{-1} is a Gaussian radial basis function network. The training data was the same as in the forward model. By cross validation, a neural network again, but coincidentally, with 6 Gaussian basis functions was found to be the best model.

5.2 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 output value $y(t)$ at time t equal to the desired value $y_r(t)$

$$u(t-1) = f^{-1}(y(t-1), y(t-2), y_r(t), u(t-2))$$

where $y_r(t) = 0.2 * r(t-1) + 0.8 * y_r(t-1)$ and r is the set point. The predicted mean value from the neural network was forwarded to the plant. After running the process for about 600 time steps the output of this classical inverse control system was found to be unstable, and the classical inverse controller was unable to force the plant output to follow the reference output.

5.3 Proposed Control Approach

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. This control signal was obtained from a small number of importance samples from the Gaussian distribution, typically a maximum of 27 samples. The overall performance of the plant under the proposed method is shown in figure 4, where it is evident that the system outputs remain stable across the whole region, and that the proposed sampling approach managed to stabilise the plant. The control signal is shown in figure 5, and the variance of this control law is shown in figure 6. The error from the absolute difference between the plant output and the desired output of the classical inverse controller and the proposed sampling approach is shown in figure 7. More specifically, figure 7 is the plot of error $= |y - y_r|_{\text{sampling}} - |y - y_r|_{\text{classical inverse}}$ against time, y is the actual plant output. From this figure we can see that the sampling approach is no worse than taking the mean in the inverse control, and in addition, the sampling method remains stable in regions where the classical approach diverges.

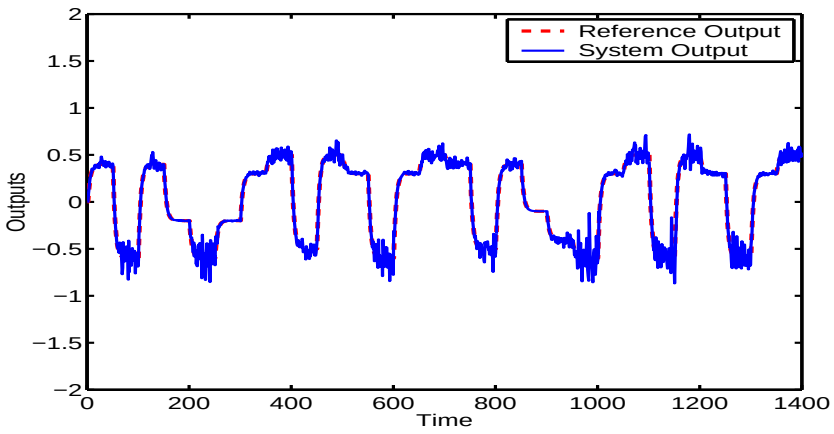

Figure 4: The desired and actual output values. The actual output of the plant (solid line) and the desired output (dotted line) are almost coincident, which indicates accurate control.

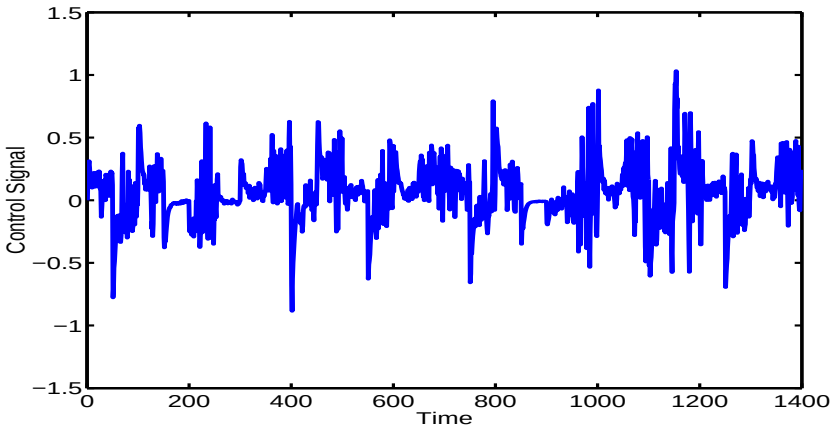

Figure 5: The Control Signal. The fluctuation in the control signal represents the stochastic nature of the control problem.

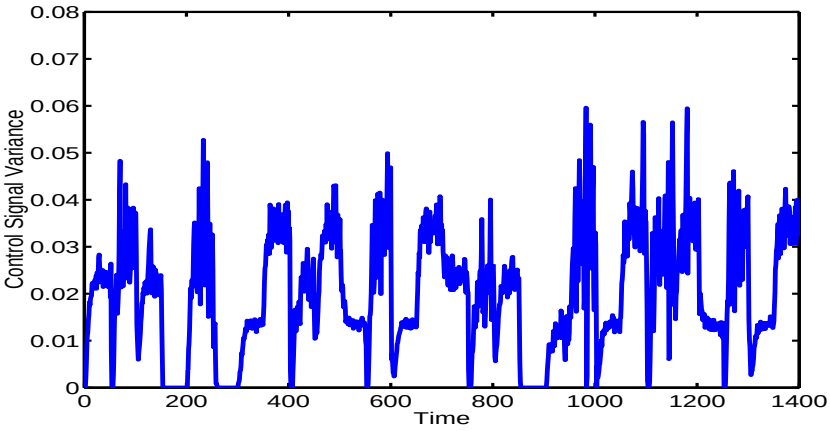

Figure 6: The Control Signal Variance. Compared to the variances added to the plant input ($\sigma^2 = 0.09$) and the plant output ($\sigma^2 = 0.2$), the predicted variance around the control signal is significantly smaller.

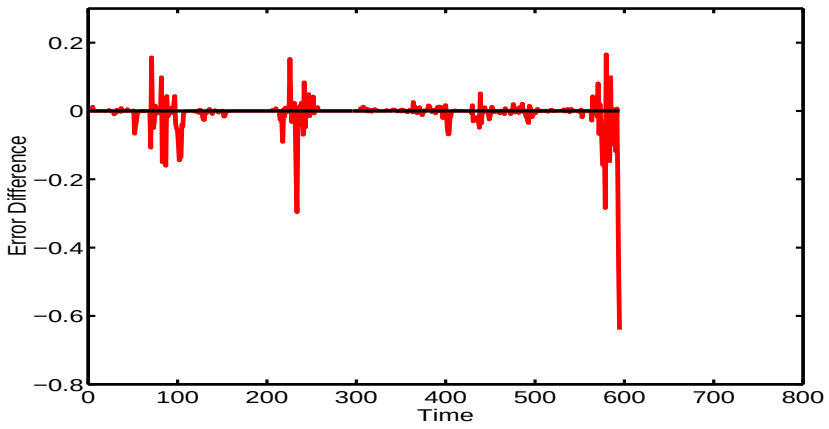

Figure 7: The Error Difference. The difference between the absolute tracking error of the proposed control method and the absolute tracking error of the classical control method. So the classical control method has more frequent and larger errors.

6 Simulation 2, Mixture Density Networks

6.1 Introduction

For inverse problems, the mapping can often be multi-valued and a unique solution cannot be found. If the Gaussian distribution approximates the inverse model, it will approximate the conditional average of the target data, and this will frequently lead to extremely poor performance. Here we will overcome this problem by appropriate use of a Mixture Density Network instead. In order to illustrate the application of the MDN with the proposed control approach we consider a simple example of single input single output given by the following equation

$$y(t) = u(t) + 0.3\sin(2\pi u(t)) + \epsilon \tag{25}$$

where ϵ is a random variable with uniform distribution in the interval $(-0.1, 0.1)$, $y(t)$ is the output variable, and $u(t)$ is the input variable. This example has been used in [3, 11] to demonstrate the use of the MDN. This equation represents a static system, since no delay exists between the input and the output variable. The plant has been considered to be given by equation (25). In order to identify the plant, an input-output model described by the following equation was chosen

$$\hat{y}(t) = f(u(t))$$

where f is a thin plate spline radial basis function network. Figure 8 shows a data set of 300 points generated by sampling equation (25). Also shown is the mapping represented by a thin plate spline radial basis function network after training using this data. The optimal structure for the neural network found by applying the cross validation method consisted of 5 thin plate spline basis functions. It was trained using the scaled conjugate gradient method. It can be seen that the network which is approximating the conditional average of the target data, gives an excellent representation of the underlying generator of the data.

6.2 Gaussian Distribution Model

We consider approximating the inverse mapping of the same problem and using the same training data as in the forward model by training a thin plate spline radial basis function network using least squares, which will lead to a Gaussian distribution assumption. Similarly an input-output model described by the following equation was chosen to find the inverse model of the plant,

$$\hat{u}(t) = f^{-1}(y(t)).$$

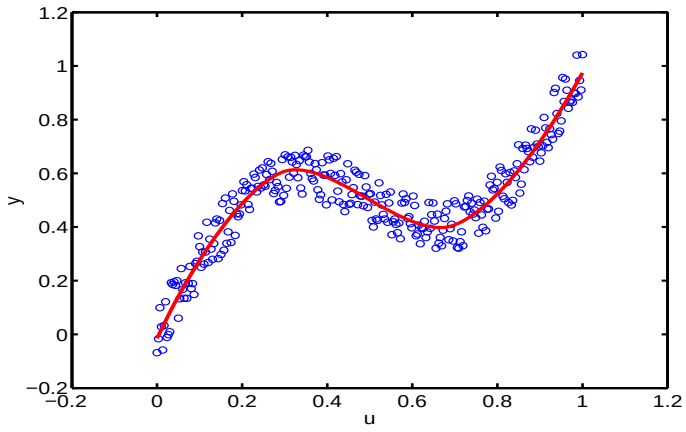

Figure 8: The forward model of the function $y(t) = u(t)+0.3\sin(2\pi u(t))+\epsilon$. The circles represent the samples generated from that function. The solid curve shows the result of training a thin plate spline radial basis function with 5 basis functions using a sum of square error function.

Again the network tries to approximate the conditional average of the target data, but now this corresponds to a very poor representation of the process as can be seen from figure 9. The network in this case had 15 thin plate spline basis functions and was trained using the scaled conjugate gradient optimisation method. This network was connected in series with the plant to generate the control signal required to cause the plant to follow the desired output. The desired output was given by

$$y_r(t) = r(t) + 0.3\sin(2\pi r(t))$$

where the input $r(t)$ has been chosen in such a way to generate data that have not been used in the training stage. The result is shown in figure 10, where it can be seen that there is a large error between the desired output and the plant output.

6.3 Mixture Density Network

In this section we apply an MDN to the same inverse problem, using the same data set as before. The appropriate number of kernel functions and the complexity of the neural network was decided by applying the cross validation method. It was found that the best structure for the MDN consisted of 7 thin plate spline basis functions with 9 outputs corresponding to 3 kernel functions . The MDN was trained using scaled conjugate gradient optimisation. Once trained the MDN predicts the conditional probability density of the target data (regarded as the input to the plant $u(t)$ in the inverse model) for each value of the input to the network (regarded as the output to the plant $y(t)$ in the inverse model). Having obtained a good representation for

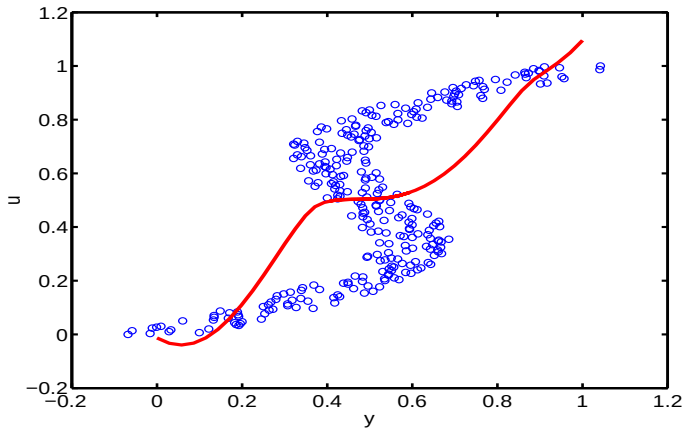

Figure 9: The inverse model of the function $y(t) = u(t) + 0.3\sin(2\pi u(t)) + \epsilon$. The circles represent the same data as in Figure 8. The solid curve shows the result of training a thin plate spline radial basis function with 15 basis functions using a sum of square error function.

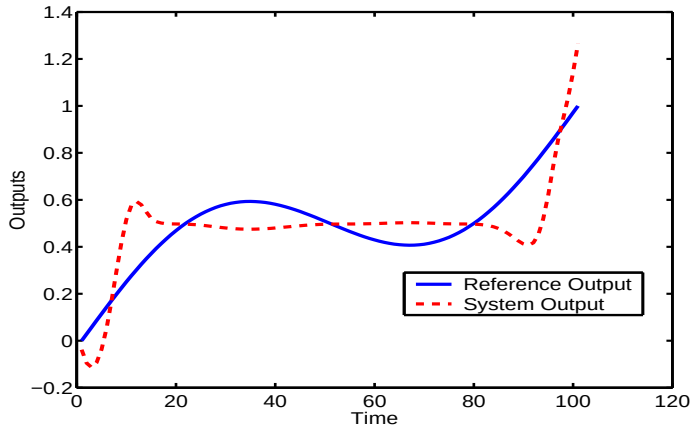

Figure 10: The control result extracted using the classical inverse controller.

the conditional density of the target data, we can in principle calculate any desired statistics from that distribution. In this control problem, since the conditional mean of the target data is a very poor approximation, we are interested in the evaluation of the centre of the most probable kernel according to equation (23), which gives the result shown in figure 11. Again this network has been connected in series with the plant to generate the control signal required to cause the plant to follow the same desired output as before. The result is shown in figure 12, where it can be seen that using the most probable value of the kernel functions has improved the performance of the controller significantly.

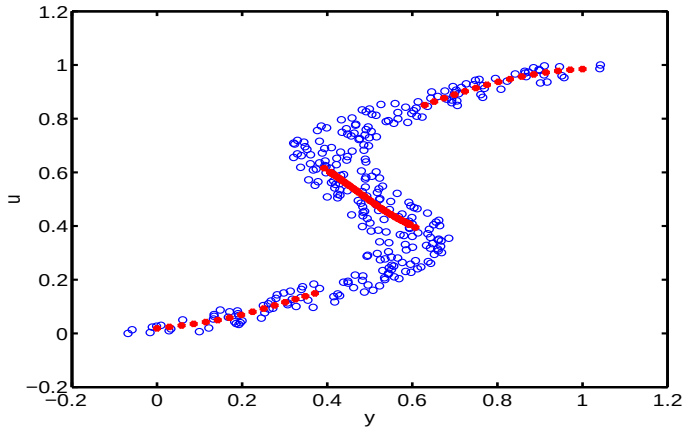

Figure 11: Plot of the central value of the most probable kernel as a function of $y(t)$ from the Mixture Density Network.

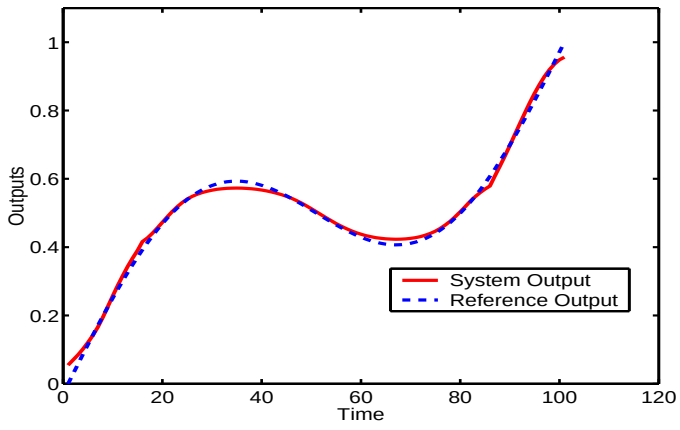

Figure 12: The control result from using most probable value of the Mixture Density Network as a control law.

6.4 Proposed Control Approach

The final demonstration of the utility of the approach, is to sample from the control signal distribution (from the mixture density distribution). In this proposed control approach the best control signal was found and forwarded to the plant, following the procedure presented earlier. The control signal was obtained from a small number of samples, typically 20 samples in this case. The overall performance of the plant under the proposed control approach is shown in figure 13. It can be seen from this figure that the proposed sampling approach is superior to finding the most probable centre value of the kernel function. The error from the absolute difference between the plant output and the desired output of the most probable value of the kernel function in the mixture density network, and the proposed sampling approach is shown in figure 14. From this figure we see that the sampling approach has reduced the error significantly.

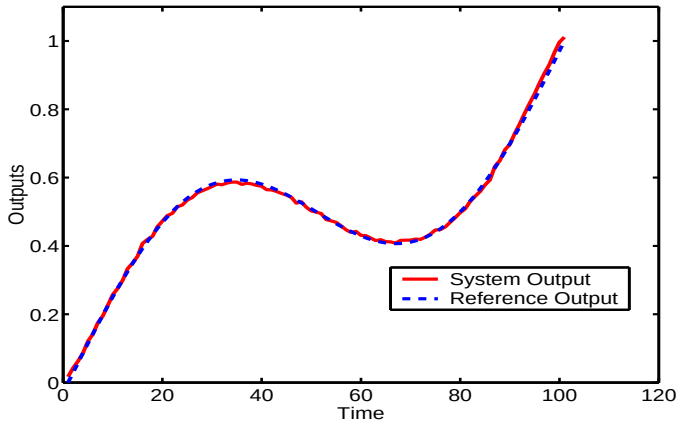

Figure 13: The control result from applying the proposed sampling approach from the mixture density network.

7 Conclusions

General inverse control can be considered to be a good control strategy if the model of the plant is invertible and accurate. We are assuming that the neural network approach allows us to construct accurate models such 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, the intrinsic uncertainty around the control signal can be estimated from 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 generalisation property of the controller. Simulation

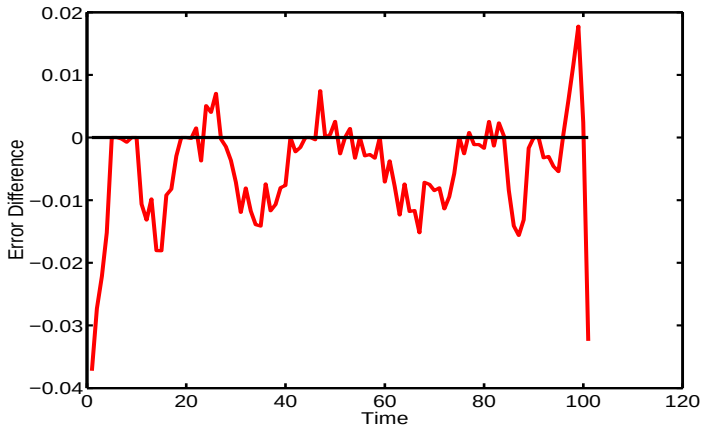

Figure 14: The Error Difference. The difference between the absolute tracking error of the proposed control method and the absolute tracking error of the classical control method.

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. Of course, in this paper we have considered a problem which is inherently stochastic. This leads to a control signal which is also stochastic. For dissipative or smoothly varying systems, the resulting control signals would also be smoothly varying.

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.

References

[1] M.S Ahmed. Neural-net-based direct adaptive control for a class of nonlinear plants. *IEEE Transactions on Automatic Control*, 45:119–124, 2000.

[2] 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.

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

[4] R. Herzallah. *Process Identification and Control Using Neural Networks*. MSc Thesis, University of Jordan, July 1996.

[5] Z. Huang and S.N. Balakrishnan. Robust adaptive critic based neurocontrollers for systems with input uncertainties. In *IEEE International Joint Conference on Neural Networks - IJCNN'00*, volume 3, pages 3067–3072, Como, Italy, 24-27 July 2000.

[6] 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.

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

[8] M.B. McFarland and A.J. Calise. Robust adaptive control of uncertain nonlinear systems using neural networks. *IEEE Transactions on Automatic Control*, pages 1–6, 1997.

[9] W.T. Miller, R.S. Sutton, and P.J. Werbos, editors. *Neural Networks for Control*. MIT Press, Cambridge, Massachusetts, London, England., 1990.

[10] 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.

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

[12] K.S. Narendra. Adaptive control using neural networks. In R.S. Sutton W.T. Miller and P.J. Werbos, editors, *Neural Networks for Control*, chapter 5, pages 115–142. MIT Press, Cambridge, Massachusetts, London, England., 1990.

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

[14] N. Sundararajan, Y. Li, and P. Saratchandran. Neuro-flight controllers for aircraft using minimal resource allocating networks (MRAN). *Neural Computation and Application*, 8:172–183, 2001.

[15] D.A. White and D. Sofge, editors. *Handbook of Intelligent Control*. Multiscience Press, Inc, New York, N.Y., 1992.

[16] 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.