Exploiting Uncertainty in Nonlinear Stochastic Control Problems

Randa Herzallah

Doctor of Philosophy

ASTON UNIVERSITY

October 2003

ASTON UNIVERSITY

Exploiting Uncertainty in Nonlinear Stochastic Control Problems

RANDA HERZALLAH

Doctor of Philosophy, 2003

Thesis Summary

In nonlinear and stochastic control problems, learning an efficient feed-forward controller is not amenable to conventional neurocontrol methods. For these approaches, estimating and then incorporating uncertainty in the controller and feed-forward models can produce more robust control results. Here we introduce a general methodology for estimating and incorporating uncertainty in the controller and forward models. The new methodology is presented using the direct inverse control method.

In direct 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.

This work introduces 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. Based on importance sampling from these distributions a novel robust inverse control approach is obtained. This importance sampling provides a structured and principled approach to constrain the complexity of the search space for the ideal control law. The developed methodology circumvents the dynamic programming problem by using the predicted neural network uncertainty to localise the possible control solutions to consider. Convergence of the output error for the proposed control method is verified by using a Lyapunov function.

Several simulation examples are provided to demonstrate the efficiency of the developed control method. The manner in which such a method is extended to nonlinear multi-variable systems with different delays between the input-output pairs is considered and demonstrated through simulation examples.

Keywords: Uncertainty, conditional distribution modelling, stochastic control, direct inverse control

To my children Yazan, Zaid, and Anas for giving me the time to do this work,
my husband, Kefah, for his tolerance, and my mother.

Acknowledgements

I would like to express my gratitude to my supervisor David Lowe. I am deeply indebted to him for his encouragement, excellent guidance and support. His way to inspirit rather than to dictate knowledge is very much useful.

Special thanks to Dr. Ian Nabney for all the constructive criticism and stimulating discussions I received from him during my PhD.

I would like to especially acknowledge my husband, Kefah Qarout. Kefah has more love, respect, encouragement and acceptance than I have ever hoped for. A special thanks to my children Yazan, Zaid and Anas for patience and tolerance in the time I spent working on this thesis. Kefah, Yazan, Zaid and Anas I love you with all my heart.

Thanks to my mother for all her support during my study. I am indebted to her for many reasons, but most importantly for nursing my children in the first six months of my study. Without her and Kefah nothing could have been done.

A special thanks to the minister of education in Jordan, Prof. Khaled Toukan who was instrumental in my decision to pursue my PhD study.

Thanks also to all past and present members of the neural computing research group, Dr. David Evans, Dr. David Barber, Dr. Manfred Opper, Dr. David Saad, Dr. Laura Rebollo-Neira, Dr. Ian Nabney, Lehel Csato, Dr. Christopher James, and Dr. Dan Cornford. Lehel Csato deserves a special thanks for all his help with technical computer problems I experienced during my study. I am also grateful to Dr. David Evans for useful discussion on mixture density network. Also thanks to Vicky Bond who makes things run smoothly in the group and quietly deals with our demands.

Contents

List of Figures

List of Tables

List of Publications

Some aspects of this work have already appeared in the public domain, specifically:

[1] R. Herzallah and D. Lowe. Probability Distribution Modelling to Improve Stability in Nonlinear MIMO Control. In 2003 IEEE Conference on Control Applications, CCA2003, volume 2, pages 954-959, Istanbul, Turkey, June 2003.

[2] R. Herzallah and D. Lowe. Multi-valued Control Problems and Mixture Density Network. In IFAC International Conference on Intelligent Control Systems and Signal Processing, ICONS2003, volume 2, pages 387-392, Faro, Portugal, April 2003.

[3] R. Herzallah and D. Lowe. Robust Control of Nonlinear Stochastic Systems by Modelling Conditional Distributions of Control Signals. Neural Computing and Applications, 2003. (Accepted).

[4] R. Herzallah and D. Lowe. Improved Robust Control of Nonlinear Stochastic Systems Using Uncertain Models. In Controlo2002, pages 507-512, Aveiro, Portugal, September 2002.

[5] R. Herzallah and D. Lowe. A Novel Approach to Modelling and Exploiting Uncertainty in Stochastic Control Systems. In International Conference on Artificial Neural Networks, ICANN2002, pages 801–806, Madrid, Spain, August 2002.

Chapter 1

Introduction

This thesis focuses on finding an optimal control strategy for noisy nonlinear processes and for ill-posed problems. The approach is based on incorporating uncertainty explicitly by modelling statistical distributions, which are applied to the models of the dynamic system and the controller. Proposed control methods can be obtained based on importance sampling from these distributions. The importance sampling provides a structured and principled approach to constrain the complexity of the search space for the ideal control law.

The two considered types of problems, illustrated in Fig. 1.1, can be characterised as follows:

- Stochastic control problems: The general stochastic control problem is concerned with systems for which the present values of the state are known, but future values are affected by random forces. The system equations instead of depending only on state, and control, depend as well on a set of random variables, $\bar{v}_j, j = 1, 2, ..., b$. These variables are assumed to follow a known probability density function. In the discrete time case the system difference equation is:

$$x(k+1) = g[x(k), u(k), \bar{v}(k)]. \tag{1.1}$$

  In addition to the random forces affecting the dynamics of the system, the measurement system is affected by noise as well. Therefore the final measured value for the state of the system is described by:

$$z(K) = h[x(k), w(k)], \tag{1.2}$$

  with $w(k)$ being the measurement noise vector. The control problem is then to find the control sequence $u(k), k = 0, 1...., T$ to give the desired behaviour of the system. The general problem of finding optimal control for such systems has been solved by dynamic programming [69; 86; 109]. It is based on an indirect model-based methodology. The solution in [69] is first to find the best estimate of the state of the system from the measurements and then to apply the control that is optimal if the observed variable of the system is equal to this estimate. This however is practically infeasible, not least because of the unbounded search space which is needed to try and maintain all possible solution trajectories.

- multi-valued control problems: Multi-valued control problems in which there exists a well defined forward function, but undefined inverse function, and control problems with hysteretic transfer functions appear in a wide range of real world applications. Examples include, the control of industrial plants, robot kinematics, and analysis of spectral data.

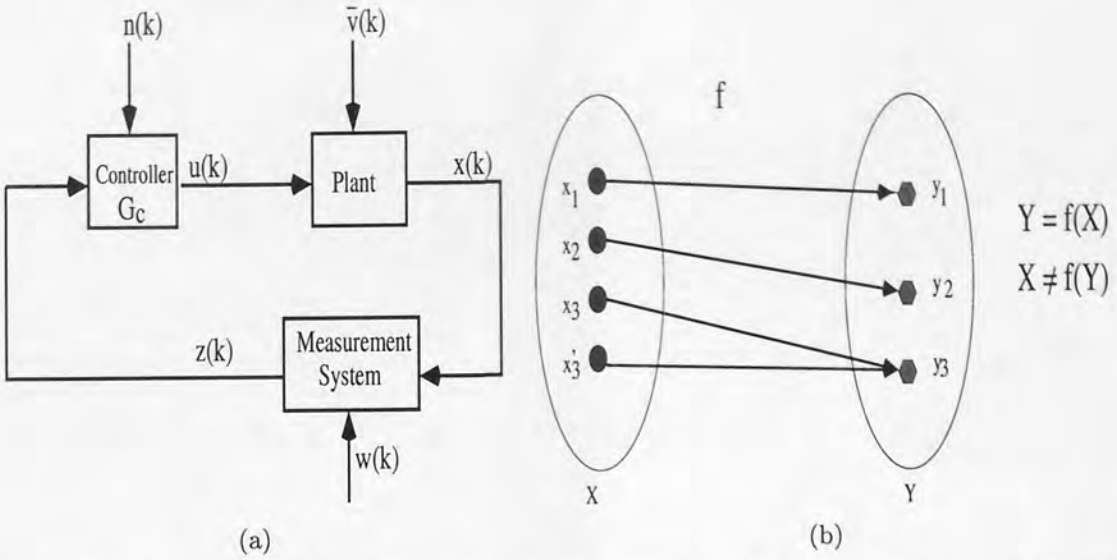


Figure 1.1: The two types of problems considered in this thesis: (a) the stochastic control problem. (b) Ill-Posed control problems.

Various systematic schemes exist for designing robust controllers for such systems. See, for example [45] for robotic applications, [72] for hybrid dynamical systems consisting of a family of continuous time subsystems, and [66] for processes with hysteretic transfer characteristics.

From what has come earlier, there are two general methods to design the controller. The first one is to design the controller directly from experimental data. Examples of such direct data-based methods are direct adaptive control algorithms, and inverse control algorithms. The second way for designing the controller is the indirect model-based methodology, where a dynamic model for the controlled system is firstly obtained through system identification. The controller is then computed based on this model. This thesis will focus on inverse control methods, for the design of the controller. In this thesis experimental data from stochastic processes are used to model the controller. In such models, illustrated in Fig. 1.1, the input signal to the controller is the noisy measured value of the state of the system, and G_c is an adaptive filter selected to give the output signal vector $\hat{u}(k)$ which is considered to be a function of the desired behaviour of the state of the system $x_r(k+1)$, the measured state of the system, and the criterion J_1 to be minimised.

$$\hat{u}_1(k) = G_{c1}[h\{x(k), w(k)\}, x_r(k+1), J_1]. \tag{1.3}$$

The problem with designing such filters is that the inevitable modelling errors constitute a potential source of performance degradation. If the models are not accurate, then the performance of the filter when applied to the actual system, will differ from what could be expected and could cause severe stability problems. The modelling error in the controller is represented by $n(k)$ in Fig. 1.1. To investigate the effect of model imperfections, and to reduce these effects, quantitative information about model uncertainty should be provided. Therefore instead of estimating a single output from the controller (which is the control signal) as in traditional control methods, another output representing the uncertainty around the control signal should be estimated as well. Graphically this can be illustrated in Fig. 1.2. Taking knowledge about the uncertainty in the predicted control signal to find the ideal control law, could improve the performance of the controller. Formally we may then

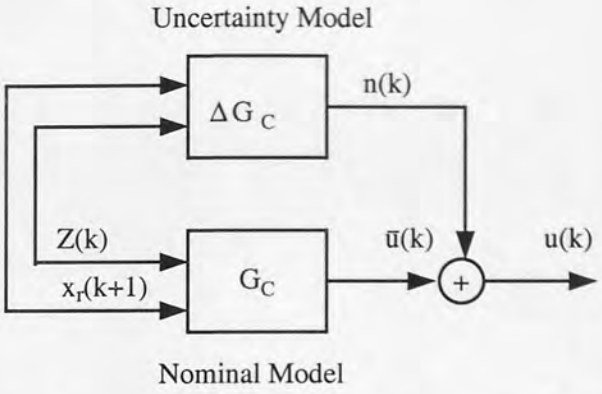


Figure 1.2: An example of a more complete control signal modelling. A nominal model G_C is here complemented by another uncertainty model, ΔG_C, which reflects the range of possible deviations from the nominal one.

replace Eq. (1.3) by the computation of a modified control law, $\hat{u}_2$

$$\hat{u}_2(k) = G_{c2}[h\{x(k), w(k)\}, x_r(k+1), J_2, n(k)], \tag{1.4}$$

where J_2 is now a performance criterion which is to be minimised with respect to the uncertainty in the controller $n(k)$. This is the aim of a method for robust design. To summarise:

The thesis is devoted to direct inverse control methods to design the controller. Uncertainty in the controller and forward model could then be explicitly estimated and taken into account to find a better estimate for the ideal control law.

The design methodology will be based on a probabilistic description for the output of the controller. The solution is simple and yet still powerful.

The primary goal of the present chapter is to give an introduction to the uncertainty problem. In the next section we will review various research in the literature concerned with uncertainty. We will see that different definitions for uncertainty and different ways for handling this problem have been proposed.

1.1 Literature survey

Dynamic programming [8; 9], currently is the only mathematical formalism under which a controller can be designed for nonlinear and stochastic systems. Techniques like feedback linearisation [120; 117] have been used for nonlinear control problems under limited conditions, such as equal numbers of inputs and outputs and accurate system modelling. Other available solutions for nonlinear controllers are based on linearised mathematical modelling to approximate the identification and control problems [122]. Examples of these methods are self tuning adaptive control, pole placement, and predictive control. However the above developed methods are found to be inadequate for most of the real world applications. Such systems are usually characterised by high nonlinearity with time delays, saturation limits, time varying parameters, high noise level and overall complexity. The difficulties that arise in these systems can be broadly classified into three categories: (1) computational complexity (2) presence of nonlinearities and (3) uncertainty.

In recent times, several authors have used neural networks to optimally solve nonlinear control problems based on exact system models [125; 84]. As a result, five basic design approaches have been developed in the literature. The simplest approach is supervised control, where neural nets are trained on a database that contains the correct control signals. Problems result from the use of the direct inverse control and solutions to these problems have been demonstrated in [84]. The use of neural nets instead of linear mappings in standard adaptive control have been discussed in detail in [94; 95; 59]. The fourth approach called back-propagation of utility, cannot efficiently account for noise and cannot provide real time learning [125]. More complex than the above four mentioned approaches, but the most efficient approach is the adaptive critic family. This approach approximates dynamic programming and has the most interesting structure. It is presented in [104; 114; 103]. The application of these methods to complex real world problems has demonstrated that artificial neural networks are suitable to cope with all three categories of difficulties mentioned earlier. The nonlinear components in neural networks make them suitable to approximate any nonlinear mapping to any desired degree of accuracy; their parallel nature permits the use of powerful and computationally efficient training methods. Because neural networks can adapt their parameters in real time, they can be used under different conditions of uncertainty. On the other hand, 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.

In addition to nonlinear control methods based on exact system models, significant progress has been made in solving problems for nonlinear systems with different types of uncertainties. In the following we will review a few examples from the literature. In [14] a systematic procedure that accounts for the structured uncertainty in the neural network model has been developed. In this work a multi-layer perceptron neural network is used to obtain the output of the process $y_{k+1} = f(x_k, u_k) = W\tanh(Vx_k + Gu_k + b) + d$, where x represents the state vector, and u is the process input. An approximate feedback linearisation is achieved by first performing the Taylor expansion of process model around the previous operating point (x_{k-1}, u_{k-1}) at each sampling instant, $y_{k+1} = f(x_{k-1}, u_{k-1}) + F(x_{k-1}, u_{k-1})\Delta x_k + E(x_{k-1}, u_{k-1})\Delta u_k$. Here F is the Jacobian of f with respect to x_{k-1}, and E is the Jacobian of f with respect to u_{k-1}. To allow for structured uncertainty, Taylor series expansions for the linearised model of the process is performed around the nominal weights of the neural network model. The propagation through the control loop of the structured uncertainty from the neural network parameters enables the following state space description for the uncertain linear closed loop system $\tilde{x}^l_{k+1} = (A + \delta A)\tilde{x}_k + (B + \delta B)\nu_k$, and $y_k = C\tilde{x}^l_k$. The terms δA and δB represent the overall linear closed loop uncertainty as a function of the weights of the neural network model. The closed loop stability for the uncertain system is then proved. An extension of this method to nonlinear and state dependent closed loop uncertainties can be found in [15].

A new approach for adaptive output feedback control of uncertain nonlinear single input single output (SISO) systems has been introduced in [48]. In this approach an observer for the output tracking error rather than a state observer in the classical approach has been proposed under the assumption that the system is feedback linearisable. A simple linear observer for the tracking error and a multi-layer neural network (MLNN) is used to cancel the modelling errors. Ultimate boundness of error signals is shown through Lyapunov stability analysis. The same approach subject to a set of mild restrictions has been implemented in [47] for an uncertain multi-input multi-output (MIMO) nonlinear system. Uncertainty according to unknown nonlinearities and unmodelled dynamics has been introduced in [81]. A neural network and direct adaptive control are used to compensate for

unknown nonlinear mappings. On the other hand, the dynamic nonlinear mapping is used to provide robustness to unmodelled dynamics. Different descriptions for the unmodelled dynamics have been suggested in [50]. In this work an extra control signal is added to stabilise the perturbed system. The application of the recently developed minimal resource allocating network (MRAN) in a robust manner under faulty conditions to an aircraft flight control was demonstrated in [115]. This work is based on the use of a feedback error learning mechanism in which the MRAN aids a conventional controller. Simulation results proved that satisfactory performance can be obtained even under faulty conditions. Another approach to compensate for faulty conditions in dynamical control systems has been proposed in [133]. This approach is based on using an intelligent on-line sliding mode control strategy to handle the control problem for systems suffering from incipient and abrupt failures. The proposed approach is to continuously monitor the system performance and identify the fault based on knowledge of the nominal system and nominal controller. Once a fault is detected the controller will adjust it's control signal by adding a sliding mode control signal to confine the system performance within a boundary layer. Simulation results show a significant improvement in trajectory following performance under the proposed method.

A nonlinear inverse controller together with an adaptive disturbance canceller have been suggested in [126] to reject internal plant disturbances. Disturbance cancelling for a nonlinear system is performed by calculating a digital copy $\hat{P}$ of the plant, from which the plant inverse Q of the best least squares is obtained. The input to Q is the output of the plant model which takes the plant noise as an input. The error between the output of Q and the noise is used to adapt the least squares plant inverse Q. The disturbance is filtered by a digital copy of Q and subtracted from the control signal. A more detailed study for this control scheme has appeared in [100]. Several applications for this scheme to linear and nonlinear systems have proved that excellent disturbance rejection can be achieved. In [119] radial basis function (RBF) network sequentially trained by an extended Kalman filter as a model of a nonlinear dynamic system has been suggested to deal with high observation noise. The state space representation of the RBF network parameters and output dynamics are assumed to be $x(k) = \phi[x(k-1), u(k-1)] + d_x(k-1), y(k) = Hx(k) + \nu(k)$. The state vector is the vector with the previous network outputs, $u(k-1)$ is the vector of previous inputs, $d(k)$ the process noise, and $v(k)$ the observation noise are assumed to be independent. $\phi(x(k), u(k)) = [f(s(k), u(k), w(k))\ s(k-1)\ ...\ s(k-\Delta_s+1)\ w(k)]$ with $s(k)$ being the output of the network. The on-line structure adaptation in this work is based on the growing and pruning criterion of the hidden units, which is derived using a Kalman filter's estimate of the state estimation error and the innovation statistics.

A methodology of signal sensitivity analysis to allow the selection of an ideal multi-layer perceptron (MLP) neural network model from a set of models has been demonstrated in [121]. The output of the model is assumed to be obtained from a multi-layer perceptron neural network $y = f(W, u_0)$. The input vector is perturbed from u_0 to $u_0 + \delta u_0$. The output error due to this perturbation is obtained as $\Delta y = f(W, u_0 + \delta u_0) - f(W, u_0)$. Based on the assumption that the input perturbations are small the output error is approximated by first order Taylor series expansion $\Delta y = \bar{C}(W, u).\Delta u_0$, with $\bar{C}$ is the gradient of the output with respect to the weight parameters. The output error variance for both additive and multiplicative input perturbations is then computed. This is used to choose MLPs with lower sensitivity by choosing activation function coefficients. This method requires a large number of experiments in order to select an accurate model.

Another kind of uncertainty which occurs only if the controller is trained by the direct inverse

approach has been addressed in [60]. This approach to acquire the inverse dynamic model is proposed and used in [55; 82; 67]. Two approaches in control theory have been proposed to solve this uncertainty problem. The first one is the forward and inverse modelling proposed in [88]. In this approach the forward model of the controlled system is firstly learned using input and output data. Next the desired performance is fed to the inverse model to calculate its output. The error between the desired performance and the output from the forward model is back-propagated through the forward model to calculate the error signal for training the inverse model. Several successful applications have been reported in [18; 89]. An alternative approach called feedback error learning is suggested in [61], where a summation of feedback and feed-forward control signals is used to control the system. Problems encountered with this method and solutions to overcome these problems have been presented in [23; 22]. The application of this method to identify the inverse dynamic of an arbitrary trans-femoral prosthesis exposed to perturbations and dynamic uncertainty has been presented in [58]. Simulation results showed the ability of the proposed control method to respond well to perturbations and environment changes.

Although these methods [14; 15; 48; 47] appear to be promising for enhancing the performance of control systems in the face of modelling uncertainties, they are based on linearised control methods. However since the real world is nonlinear, linearised control methods compared to nonlinear control methods are relatively ineffective and not robust when applied to the real world. The methods [126; 100; 119; 60; 55; 82; 67] on the other hand, handle a different level of uncertainty but they still assume exact system models. The assumption of exact system models can never be true in reality therefore, a control theory needs to be derived for systems with uncertainties.

This thesis is concerned with the question of how to use model uncertainties in developing a novel robust nonlinear control architectures for systems characterised by high noise levels, nonlinearities, hysteretic, and high input output dimensionality.

The main objective of the thesis, therefore, may be stated as the investigation of the possibility of estimating model uncertainties which then can be used to derive the optimal control law. The key to this problem is to bring together and apply certain ideas in various standards of control theory, neural network theory, dynamic programming and statistics.

One of the best general methods to estimate model uncertainties can be described in terms of modelling conditional distributions for the desired model output. Recent developments in neural networks allow us to model general distribution functions. This in turn can produce an estimate of the uncertainty involved in the modelled output. In this work the conditional distribution functions will be estimated using neural networks. In the next section a preliminary discussion for the probabilistic control framework proposed in this work is presented.

1.2 Preliminary discussion

In this section the problem of decision making in control problems under uncertainty in the context of neural networks is introduced. In contrast to the classical control approaches, suppose that the control vectors are generated from some probability distribution $p(u(k))$, and the state variables evolve with time according to eq. (1.1)

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

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 state of the system should be equal to a predetermined desired value $x_r(k+1)$. This means that we are looking for an optimal control vector $u(k)$, obtained from the distribution $p(u(k))$ such that

$$\text{prob}[|x(k+1) - x_r(k+1)| > 0] = 0. \tag{1.5}$$

However this cannot be applied directly to real world problems, because the effect of each control variable from the distribution $p(u(k))$ on the real world system needs to be observed. Since only one decision input can be applied to the real system, with the assumption that the applied input to the real system is the optimal decision input value, the real state value $x(k+1)$ needs to be replaced by an estimate $\hat{x}(k+1)$. Consequently this implies that the solution provided in Eq. (1.5) never occurs in practice because it requires that the estimator $\hat{x}(k+1)$ for $x(k+1)$ 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 work this higher level type of uncertainty is not considered. Further development will be based on the assumption that the estimated distributions are accurate.

To provide an estimate for the required distributions, a neural network is used in this work. The principal feature of a neural network estimation problem is that it assumes the availability of a set of m state variables $x = (x_1, x_2, ..., x_m)$, and a set of n control variables $u = (u_1, u_2, ..., u_n)$. The estimation problem in a neural network is then to provide an estimate of the probability density function of the state variables $x(k+1)$ conditioned on the the input variables $x(k), u(k)$.

$$\hat{p}[x(k+1)|x(k), u(k), W], \tag{1.6}$$

where W is the vector of model parameters. The control problem however is the inverse of this forward problem. The controller function is then to provide an estimation of the control variable $u(k)$ conditioned on the input variables $x(k+1), x(k)$.

$$\hat{p}[u(k)|x_r(k+1), x(k), W]. \tag{1.7}$$

In certain situations, particularly when dealing with inverse problems, mathematical constraints restrict estimation 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 a one-one mapping. These restrictions however have little effect on the statistical scope of the estimation problem.

Therefore estimating the conditional distributions of forward model $p[x(k+1)|x(k), u(k), W]$, or inverse model $p[u(k)|x_r(k+1), x(k), W]$, 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 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 $\hat{p}[u(k)|x_r(k+1), x(k), W]$,
- A performance criterion J which provides an evaluation of a given decision variables. Two kinds of criterion can be taken:(1) a reward function, in which case it should be maximised, and (2) a cost function in which case it should be minimised,

- A set X, the space of the state variables x, consisting of all possible outputs $x \in X$ 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.

In this optimisation method an estimation model for the real world system has been assumed. This is because the control decisions available from the estimated local conditional 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.

1.3 Thesis Contribution

As described before the main goal of this thesis is to explore for the first time the possibility of using a neural networks estimate for uncertainty information in a novel control architecture.

The key to this problem is to bring together and apply certain ideas that already exist in classical control theory, neural networks theory, dynamic programming and statistics.

In support of this aim the main contribution of this thesis to the neurocontrol field include:

1. **To develop a novel control architecture for finding an optimal control law for stochastic nonlinear control problems:** The developed control method 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 statistical properties of networks. In this thesis 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. This control method is applied on-line and shows that real time control can be achieved. It will be shown that the developed control algorithm can stabilise control systems when they become unstable. Simulation results will be presented to verify the proposed control algorithm, including examples of nonlinear stochastic control systems, deterministic control systems, SISO, MISO and MIMO plants, static and dynamical control problems.

2. **Stability analysis for the updating rule of the control law:** Convergence of the output error from updating the control signal will be verified by using a Lyapunov function approach.

3. **Extend the use of the mixture density network for modelling general conditional distributions to the dynamic system:** This contribution can be divided into two main areas.

   - In its original formulation, a mixture density network is used to model general conditional distributions to static regression problems. In this thesis the formulation of the mixture density network will be extended to the dynamic control case. Simulation experiments for using the mixture density network to model general conditional distributions of static and dynamic systems are provided.
   - The proposed sampling approach from the estimated distribution of the standard network will be extended to sampling from the estimated distribution of the mixture density network. The performance of the new algorithm will be illustrated through simulations with static and dynamic systems.

4. **As the proposed sampling method accounts for the uncertainty in the inverse model only, an improved sampling algorithm which includes the uncertainty of the forward model is also developed:** The proposed sampling method in its original formulation ignores the uncertainty of the forward model. The sampling method is enhanced by allowing estimating the stochastic model of the forward dynamics of the plant, and then included the stochastic model rather than the deterministic one in searching for the optimal control law.

1.4 Outline of the thesis

The thesis is divided into eight chapters and five appendices. Below, a summary of each chapter is presented.

Chapter 2: Classical Methods for Identification and Control. This chapter presents a review of classical adaptive theory for identification and control, which is required for the design methods proposed in this thesis. A survey of the current control architecture in the neurocontrol field is also provided.

Chapter 3: Uncertainty Estimation and Distribution Modelling. In this chapter we discuss several methods for estimating the prediction interval. The presented methods are based on modelling statistical distributions for the neural network prediction. This is required to achieve the aim of the thesis, which is to use uncertainty estimation from neural network to improve the controller performance under uncertain conditions.

Chapter 4: Incorporating Uncertainty Directly for Stochastic SISO Nonlinear Dynamical System. This chapter which constitutes the central part of the thesis, concerns the use of the estimated neural network uncertainty in control architectures in a profitable manner. The problem formulation is introduced, and the proposed control method is presented in detail. Based on the developed control method for incorporating uncertainty, a stability analysis for the updating rule of the control law is also provided. The method is verified by applying it to a simulation example which represent a stochastic dynamical control system.

Chapter 5: Multivariate Control Problems. This chapter deals with the design of multi-variable, feed forward controllers. Further simulation examples are provided in this chapter to verify the developed control method. Problems that may result from applying the developed methods to multi-variable control problems are also discussed in this chapter. The use of the developed method to improve the controller performance for MISO and MIMO highly nonlinear systems is demonstrated.

Chapter 6: Multi-valued Control Problems. Provides a general framework for modelling multicomponent distributions which is based on the use of the mixture density network. Standard neural networks perform poorly in situations where the mapping to be learned is multi-valued. In such situations the mixture density network is proposed [13] to model statistical distributions of the generator of the data. The use of the mixture density network to model multicomponent distributions for multi-valued control problems is demonstrated for the first time in this thesis. Moreover the proposed sampling method is extended to perform the sampling from the mixture density network. Simulation experiments are provided to demonstrate the successful application

of the proposed sampling method to the mixture density network. Both static and dynamical systems are used.

Chapter 7: Incorporating Uncertainty of the Forward Model. This chapter enhances the proposed sampling method presented in **Chapter 4** by allowing for the uncertainty in the forward model to be included in the search method. Several simple linear examples are presented to show the effect of ignoring the uncertainty in the forward model on the derived control law. It is demonstrated that a better control law from the sampling method can be obtained if the uncertainty in the forward model is considered when performing the search for the optimal control law.

Chapter 8: Conclusions. Here the thesis is concluded by presenting a summary of the results obtained. In addition, suggestions for future research are given and discussed.

Appendix A: Bayesian Error Bars. The derivation of the uncertainty in the neural network prediction by using the Bayesian method is given in this chapter. Both standard Bayesian error bar methods and Bayesian error bar methods with input noise are presented.

Appendix B: Weight Updates for the Radial Basis Function. The radial basis function network is used to demonstrate the Bayesian methods for error bar estimation in **Chapter 3**. Two cases are considered: updating the weights in the final layer of the RBF network and updating all the parameters of the RBF network. The Levenberg Marquardt algorithm is used for the first time to update the network parameters in a Bayesian framework. All the required derivations are given in this appendix.

Appendix C: Mean and Variance of Gaussian Processes. Gives the detailed derivation for the mean and the variance of the predictive distribution of Gaussian processes.

Appendix D: Recurrence Relation in Dynamic Programming. Provides the derivation of the recurrence relation in dynamic programming.

Appendix E: The Derivatives of the Error Function of an MDN. This appendix provides the derivation for the error gradients of the mixture density network with respect to the output of the feed-forward neural network.

Chapter 2

Classical Methods for Identification and Control

Most industrial processes are characterised by nonlinear behaviour, high dimensionality of the decision space, high noise level, and have time varying parameters. Motivated by the superior abilities of feed-forward artificial neural networks to cope with complex environments, they have been extensively used for modelling and control. Once the cost function of the control problem is determined, the parameters in the control network can be adapted in two ways: either determined via some prescribed off-line algorithm and remain fixed during operation, or adjusted on-line step by step.

The adaptive capability of neural networks has given rise to their use in adaptive control. The successful applications of neural networks to practical problems and the ability of these networks to cope with all difficulties mentioned earlier have been demonstrated in several publications [19; 102; 111; 39]. For control applications the most relevant neural network architectures have been feed-forward architectures with back-propagation as the standard training method. However such networks suffer from slow training and the attainment of global optimum solutions cannot be guaranteed due to the existence of local minima [28; 113]. Neural networks are characterised by their network topology, which means the number of parameters, the type and number of the nonlinear elements used, and the kind of training algorithms implemented for adapting the networks parameters.

The main objective of this chapter is to review the problem of identification and control in the neurocontrol field. Further, the different models and methods used to solve the adaptive control problem are presented.

This chapter starts by reviewing the general framework of adaptive control, in Section 2.1. In Section 2.2, the nonlinear approximation ability of the neural network together with the two different types of artificial neural networks that are used for identification and control, in addition to the principal features for these networks will be described. The various methods for parameter adjustment are discussed in Section 2.3. Section 2.4 discusses the optimal solution for neural networks. The identification problem of nonlinear systems is considered in Section 2.5. Section 2.6 discusses process modelling using neural network. Section 2.7 deals with the nonlinear adaptive control problem and discusses several methods that extends adaptive control principles to neural networks. The chapter ends with a short discussion.

2.1 Adaptive control

This section provides a survey for adaptive control, and an introduction to the adaptive control problem we are interested in, in this work.

Adaptive control of linear systems has been well established using developed techniques based on linear algebra, complex variable theory, and theory of ordinary linear differential equations. In adaptive control, the parameters of the controller are adjusted using an adaptive algorithm, so that the output of the plant follows prespecified trajectories, or more generally so as to optimise a specified performance criterion. For a long time two distinct approaches have been used to control a linear time invariant plant adaptively. In the first one, called direct control, the parameters of the controller are adjusted directly to minimise some error function. In the second approach, called indirect control, the controller parameters are estimated based on preestimated plant parameters. For more details about direct and indirect control the reader is referred to [89].

The robustness properties of adaptive control systems have been a research topic for over 20 years. The effect of external disturbances, unmodelled dynamics [65] and time variant parameters have been extensively investigated, and robust adaptive algorithms were developed. Under certain assumptions adaptive control methods have been generated such that the overall system is globally stable and the output error tends to zero asymptotically. In recent years adaptive control of nonlinear feedback linearisable system has been studied. The same approaches, which have proved successful for the adaptive control of linear plants, are used even when the control plants are nonlinear. However in place of the linear models in conventional adaptive control, nonlinear neural networks are used for both identification and control. Learning algorithms are used to adjust the weights so as to reduce the modelling error or the cost function of the controller. Moreover other approaches for handling the adaptive control problem have been implemented utilising neural networks properties, such as the direct inverse control and adaptive critic methods.

Similar to linear adaptive control, the stability analysis and convergence of the error have been proved for a class of nonlinear control systems [88]. Nonlinear adaptive control methods have been developed and improved to be able to deal with external disturbances, unmodelled dynamics and time variant parameters [50; 39].

2.2 Neural networks and nonlinear approximation

Networks with nonlinear components can be used in nonlinear functional approximation. The inherently parallel nature of the networks can make them suitable for solving problems at high rates. Since nonlinear relationships can be handled by neural nets, they are suited to be used in control problems. Nonlinear approximation methods have been discussed extensively in mathematics. An example of these methods is polynomial approximation which can approximate arbitrarily well a continuous function. Multi-layer neural networks and RBF networks are considered as additional approximation methods.

The important result that multi-layer feed-forward networks with a single hidden layer and sufficient numbers of hidden units, are capable of approximating any continuous function to any degree of accuracy has been proved in the literature. In [46] multi-layer feed-forward networks have been proven as a class of universal approximators. An approximation scheme is said to be a universal approximation if for a given function $f \in C(U)$, where U is a compact set in R^n, there exist an integer N_1 and sets

of real constants α_i, b_i and w_i where $i = 1, 2, ..., N_1$ and $p = 1, 2, ..., n$, such that $F(u_1, u_2, ..., u_n) = \sum_{i=1}^{N_1} \alpha_i \phi(\sum_{p=1}^{n} w_{ip} u_p + b_i)$ satisfies the condition $|F(u_1, u_2, ..., u_n) - f(u_1, u_2, ..., u_n)| < \epsilon$. This theorem is shown to be directly applicable to the multi-layer perceptron neural network. Failure of applications has been referred to the inadequate number of hidden units, inadequate learning, or the stochastic relation between inputs and targets. The two hidden layer networks was proved to have good approximation properties in [29]. The result for one hidden layer was for networks whose output functions for the hidden layer are sigmoidal, but the output functions for the input and the output layers are linear. The same result for a linear output layer and a single hidden layer of sigmoidal nodes is obtained in [21; 17].

However since polynomials can approximate nonlinear functions, the results provide no special motivation as to why neural networks should be preferred. The Stone-Weierstrass theorem provides the basis for these approximation methods [46; 21]. In addition to the case of polynomials, these theorems do not provide information about the number of terms needed to achieve the approximation, i.e the number of hidden units and how many hidden layers in neural network context.

The work in [20] gives the empirical result that networks with two hidden layers can provide more accurate approximation property than a single hidden layer. This is shown to be achieved using fewer processing units than in the case of a single hidden layer. A similar result has been provided in [73]. In this paper theoretical and heuristic considerations for selecting the number of nodes in neural networks for both of the single and two hidden layers are provided as well. In [116] a theory has been developed to allow one to specify a finite set of training data and the minimum number of parameters for a three layered networks, used for generalising and approximating a given continuous mapping.

The problem of exact approximation has been considered in [32], from the point of view of Kolmogorov's theorem. Kolmogorov's theorem states that any continuous multivariate function with N variables can be represented by a linear summations of fixed increasing continuous functions and a nonlinear continuous function of one variable, see [64] for details. For neural networks this means that every continuous function of N variables can be computed by a network having $N(2N+1)$ units in the first layer and $(2N+1)$ in the second layer. The result of this attempt has been that Kolmogorov's theorem can not be used to prove that a network with two hidden layers is a good representation. This is because of the following two reasons. The first reason is that in Kolmogorov's theorem the functions in the first hidden layer are required to be non-smooth, and if these functions are smooth then the theorem breaks down. This however leads to problems of extreme sensitivity to the input variables in the neural network field, and the smoothness property is one of the basic requirements for the generalisation performance of a network. The second reason is that in Kolmogorov's theorem the functions in the second hidden layer depend upon the particular function we need to approximate. However in a neural network, fixed activation functions are usually considered and a particular function is usually approximated by adjusting the number of hidden units and the values of the weights and biases.

The property of exact approximation has been considered further in [33; 101]. In these two works it has been shown that the approximation of a continuous function arbitrarily well, is not sufficient for characterising good approximation schemes. It is proposed that the key property is the property of best approximation. An approximation scheme has this property if for a given function f belonging to some prescribed set of functions ϕ, and given subset A of ϕ, there is one function a of A that has minimum approximating error for the function f to be approximated. a is then called the best approximation function to f from A. In [33] the main result is that the multi-layer neural networks of

the type used in back-propagation do not have the best approximation property, Secondly the RBF network is proved to have the best approximation property. The RBF network is a single hidden layer network and can be trained using linear optimisation techniques with guaranteed global solution once the basis functions have been determined [16].

The complexity of the RBF network (in terms of the number of basis functions to be used) has been discussed by several authors. In early stages of development, the number of basis functions was restricted to be equal to the number of the training patterns. However this strategy was considered to be poor. This is due to the fact that using as many basis functions as data points could result in misleading variations due to imprecise or noisy data. In addition, the number of basis functions required for approximation increases exponentially with the dimension of the input space. An alternative technique for using less basis functions than data points was proposed in [16]. It is based on linear optimisation methods and the use of pseudo inverse for interpolating in a high dimensional space. A method for characterising the complexity of the RBF network is proposed in [75]; it is based on estimating its effective degrees of freedom.

Another subject that has been discussed in the literature was regarding the choice of the neural network to be used, i.e. multi-layer neural network or RBF network. In the multi-layer neural network the parameters appear in a nonlinear fashion. This consequently implies that nonlinear optimisation methods become mandatory for the adjustment of the parameters. Using gradient methods for optimising the parameters allows convergence to a local minima, and can be very slow. The slow rate of convergence is due to the contribution of many hidden units to the determination of the output value for a given input. During training the outputs from the hidden units when linearly combined by the final layer of weights should generate the correct outputs for a range of possible input values. Interference and cross coupling between the hidden units result in the network training process being highly nonlinear with problems of flat regions in the error function arising from near cancellations in the effects of different weights. This can result in a very slow convergence even with advanced optimisation strategies. On the other hand the RBF network is characterised by local tunability. So for example, the output from the RBF is zero when the input is far away from μ_j, and is very large when the input is close to a centre μ_j. This local tunability of RBF network is suited to fast learning. In addition, since the parameters in the output layer enter linearly, least squares techniques can be used to compute their optimal values. This however requires large numbers of basis functions to obtain the best approximation. In addition, prior information is needed about the magnitudes of the input signals to locate the centres of the RBF network. Multi-Layer neural networks do not require such information, and are therefore preferred when the magnitudes of the inputs are not known a priori.

The fact that RBF networks require large numbers of basis function to give the best approximation property, implies that the RBF network is impractical for high dimensional input space. This is because, for high dimensional input space the number of nodes needed grows with the dimension of the input space. To overcome this problem, the basis functions of the RBF network have been considered to be chosen based on the complexity of the output data. This in turn requires the adaptation of the centres μ_j of the network in addition to the output layer weight parameter W. The widths of the network can also be adapted for better approximation [38]. Since the output of the RBF network depends nonlinearly on the centres of the basis function μ_j and the width σ_j, nonlinear optimisation methods are used to search for the optimal parameters.

From the above discussion, the question as to whether multi-layer neural network or RBF network is to be used for approximating certain problem depends on the prior information that is available

about the problem in hand.

2.3 Methods for parameter adjustment

In the following chapters the multi-layer neural network as well as the RBF network will be used for identification and control of dynamical systems. In both cases the parameters of the neural networks need to be adjusted using information available in the input/output data. The adjustment of the parameters is required to approximate the output of the dynamical system and the control law. The nonlinear dependence of the outputs of RBF network and multi-layer neural network on their parameters, implies that the parameters need to be adjusted using a nonlinear optimisation method. One of the basic requirements of these nonlinear optimisation techniques is the estimation of the gradient of the error function with respect to the parameters. The most popular method to estimate the gradient of the error function with respect to the parameters is the back-propagation algorithm. This method is presented in [96] and it has been shown that it can also be suitable with some modification for the adaptation of recurrent networks (with feedback of the network output). This has been called dynamic back-propagation in contrast to the static back-propagation used for adapting static neural networks (without self feedback).

In this work only static networks are used. Therefore, only static back-propagation is required, see [13] for details. For dynamic back-propagation the reader is refered to [96].

2.4 Optimal solution for neural network

Neural networks provide a practical framework for approximating arbitrary nonlinear multivariate mappings.

In practice the optimisation of the network parameters is performed so as to find the minimum of the cost function. Nonlinear optimisation methods are not exact and do not converge to the global minimum of the cost function. This however does not mean a bad approximation to the function. Neural networks normally approximate the optimal solution.

In the following the optimal performance that could be expected from the neural network will be determined. The derivation of the result will not be dependent on the choice of the network architecture, or even whether we are using a neural network at all. It can be for any adaptive nonlinear filter given that the representation for the nonlinear mapping is sufficiently general.

This theorem has been discussed in [31]. The theorem states that if the outputs of the network are trained by minimising a sum of square error function of the form

$$\mathrm{Err} = \sum_{k} \{y_k - \hat{y}_k(u, W)\}^2, \tag{2.1}$$

then the optimal predicted output of the network will be given by:

$$\hat{y}_k = \mathrm{E}[y_k|u], \tag{2.2}$$

where $\hat{y}_k$ is the optimal output vector of the network, y_k is the vector of the target data, and u is the input vector to the network.

To prove this theorem, suppose that $\hat{y}_k$ is the optimal estimate, and $\tilde{y}_k$ is some other estimate. In the following it will be shown that the error from $\tilde{y}_k$ is larger than that obtained from $\hat{y}_k$. Now

consider

$$\begin{aligned} Err(\tilde{y}_k) &= E[\| y_k - \tilde{y}_k \|^2], \\ &= E[\| y_k - < y_k \mid u > + < y_k \mid u > - \tilde{y}_k \|^2], \\ &= E[\| y_k - < y_k \mid u > \|^2] + E[\| < y_k \mid u > - \tilde{y}_k \|^2], \\ &+ 2E[(y_k - < y_k \mid u >)^T (< y_k \mid u > - \tilde{y}_k)], \\ &\geq Err(\hat{y}_k) + 2E[(y_k - < y_k \mid u >)^T (< y_k \mid u > - \tilde{y}_k)]. \end{aligned} \tag{2.3}$$

The second term on the right hand side is zero. The following is an illustration for this result

$$E[(y_k - < y_k \mid u >) \mid u] = 0. \tag{2.4}$$

Since $< y_k \mid u > - \tilde{y}_k$ is a deterministic function of u

$$E[(y_k - < y_k \mid u >)^T (< y_k \mid u > - \tilde{y}_k) \mid u] = 0. \tag{2.5}$$

Then by iterated expectation

$$\begin{aligned} E\{E[(y_k - < y_k \mid u >)^T (< y_k \mid u > - \tilde{y}_k) \mid u]\} &= E[(y_k - < y_k \mid u >)^T (< y_k \mid u > - \tilde{y}_k)], \\ &= 0, \end{aligned} \tag{2.6}$$

and therefore

$$Err(\tilde{y}_k) \geq Err(\hat{y}_k). \tag{2.7}$$

The derived result for the optimal predicted output of the neural network (2.2) is based on three assumptions. Firstly, the training data set must be sufficiently large. Secondly, the approximation function $\hat{y}_k$ of the network must be general, such that there are sufficiently many parameters which can be adapted to make the error between the predicted neural network output and the desired output small. Thirdly, the parameter optimisation method should be performed in such a way so as to find the minimum of the cost function.

Given the above three requirements neural networks can in principle approximate the conditional average of the target data to arbitrary accuracy.

Although this result is important and desirable in the control field, model uncertainty can also be derived based on this result. Uncertainty estimation can be performed by subtracting this average from the target values and the result is then squared, $[y_k - < y_k | u >]^2$. This in turn approximate the variance $\sigma^2(u)$ of the target values. The assumption then is that the distribution of the target data could be approximated by a Gaussian function with mean equal to the conditional average of the target data and variance equal to the residual error.

In this way a stochastic model rather than a deterministic one could be estimated to describe the target data. This development represents the best general method for estimating model uncertainties, and can be utilised in a useful way so as to obtain a better control result as will be demonstrated later.

The development in this section for estimating conditional distributions of the target data is the simplest method, however, it is not suitable for some control problems as we will see later in Chapter 6.

2.5 Identification models for nonlinear dynamical systems

As mentioned in Section 2.2, neural networks have the ability to approximate large classes of nonlinear functions sufficiently accurately. The use of static and dynamic back-propagation for adjusting their parameters makes them attractive to be used as an identifiers and controllers. For neural networks to be used as an identifier or controller, an identification model which describes the input-output model should be chosen. In this section four models which were introduced in [94] for the representation of discrete time single input single output (SISO) nonlinear plants are presented. These models are motivated by the models which have been used in the adaptive literature for the identification and control of linear dynamical systems. Since all models are nonlinear, linear models can be considered as a special cases. The output of these models is a function of the past inputs and outputs of the dynamical systems, as in the case of the autoregressive moving average (ARMA) model for linear systems.

2.5.1 Plant model characterisation

The four classes of the plant models are

- Model I: In this model class the output y of the unknown nonlinear plant is assumed to depend linearly on its past values and nonlinearly on the past values of the input u.

$$y(k+d) = \sum_{i=0}^{q-1} \alpha_i y(k-i) + g[u(k), u(k-1), ..., u(k-p+1)], \tag{2.8}$$

  where d is the relative degree of the plant, which corresponds to the minimum delay from the input to the output, and p is the maximum delay in the input. This means that the plant has a time delay equal to d, therefore the control applied at time k affects the output of the plant only at time instant greater than or equal to $k+d$.

- Model II: Here the output depends linearly on the input u and nonlinearly on its own past values

$$y(k+d) = \sum_{i=0}^{p-1} \beta_i u(k-i) + f[y(k), y(k-1), ..., y(k-q+1)], \tag{2.9}$$

  where q is the maximum delay in the output. This model is particularly suited for control problems due to the linear dependency of the output on the input. This makes it directly applicable to control.

- Model III: In this model the unknown plant output is considered as the sum of two nonlinear functions. The first nonlinear function has the past values of the model as an input, while the second one inputs the past values of the dynamical system input.

$$y(k+d) = f[y(k), y(k-1), ..., y(k-q+1)] + g[u(k), u(k-1), ..., u(k-p+1)]. \tag{2.10}$$

- Model IV: This is the most general of the earlier models and subsumes all of them. In this case the output of the unknown plant is a nonlinear function of its past values and the past values of the input.

$$y(k+d) = f[y(k), y(k-1), ..., y(k-q+1), u(k), u(k-1), ..., u(k-p+1)]. \tag{2.11}$$

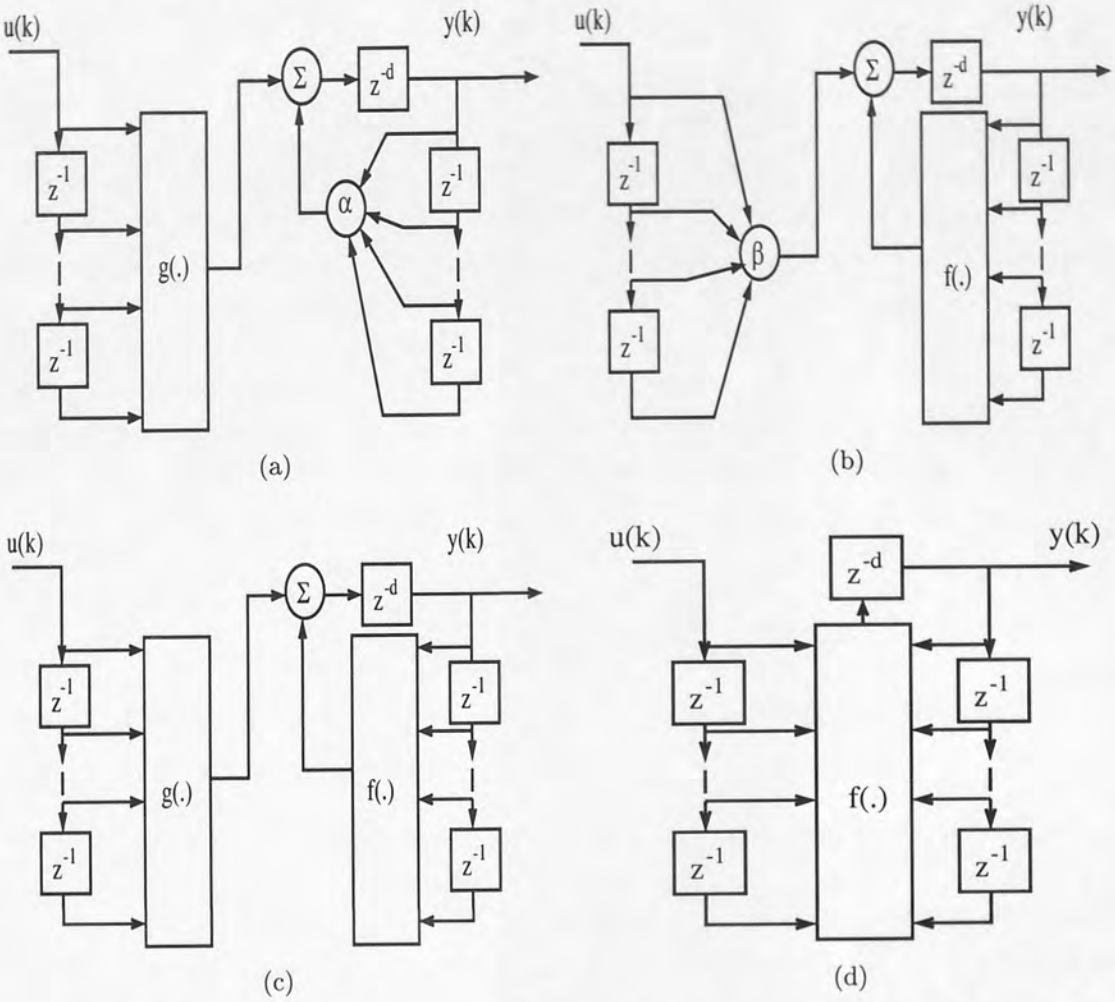


Figure 2.1: The four input-output models for nonlinear systems: (a) Model I. (b) Model II (c) Model III (d) Model IV.

The block digram of the various models are shown in Fig. 2.1, the elements labeled z^{-1} at the input and output ends represent unit delay elements.

For multi-input multi-output (MIMO) systems $[\mathbf{u}(k), \mathbf{y}(k)]$ represent input-output vector pair of dimension n and m respectively, $\mathbf{u}(k) = [u_1(k), u_2(k), ..., u_n(k)]$ and $\mathbf{y}(k) = [y_1(k), y_2(k), ..., y_m(k)]$. Taking Model IV , the representation of the multi-variable system with specified relative degree is

$$
\begin{aligned}
y_1(k+d_1) &= f[\mathbf{y}(k), \mathbf{y}(k-1), ..., \mathbf{y}(k-q+1), \mathbf{u}(k), \mathbf{u}(k-1), ..., \mathbf{u}(k-p+1)], \\
y_2(k+d_2) &= f[(\mathbf{y}k), \mathbf{y}(k-1), ..., \mathbf{y}(k-q+1), \mathbf{u}(k), \mathbf{u}(k-1), ..., \mathbf{u}(k-p+1)], \\
&\quad ..... \qquad\qquad ..... \qquad\qquad ..... \\
y_m(k+d_m) &= f[\mathbf{y}(k), \mathbf{y}(k-1), ..., \mathbf{y}(k-q+1), \mathbf{u}(k), \mathbf{u}(k-1), ..., \mathbf{u}(k-p+1)],
\end{aligned} \tag{2.12}
$$

where the relative degree d_i of the i^{th} output can be defined from the relative degree d_{ij} for each input output pair (u_j, y_i) in such a manner $d_i = \min_j d_{ij}$. This corresponds to the minimum time delay between any one of the inputs and the i^{th} output.

In this work an input-output model represented by model IV is used for both identification and control.

2.5.2 Identification procedure

As mentioned in Section 2.3 the adaptive identification problem is based on setting up a suitably parameterised identification model. In the neurocontrol field the identification model of the nonlinear plant is composed of neural networks and tapped delay lines.

To identify the plant a suitable identification model is firstly chosen based on prior information concerning the class to which it belongs. The model parameters are then adjusted so as to optimise a performance function which in most cases tries to minimise the difference between the output of the plant denoted by $y(k+d)$ and the identification model output $\hat{y}(k+d)$. The error $e(k+d) \triangleq y(k+d) - \hat{y}(k+d)$ is used to update the parameters W of the neural network. This means that the identification process is based on adjusting the neural network parameters, so that the plant and the model outputs are described by identical input-output equations.

Two identification procedures have been reported in the literature. The two procedures are presented below for a nonlinear plant which assumes model class I mentioned in Section 2.5.1. The delays in the input and the output are set to $p = 2$ and $q = 2$ respectively and the nonlinear function g is replaced by a neural network.

$$y(k+d) = \alpha_0 y(k) + \alpha_1 y(k-1) + g[u(k), u(k-1)]. \tag{2.13}$$

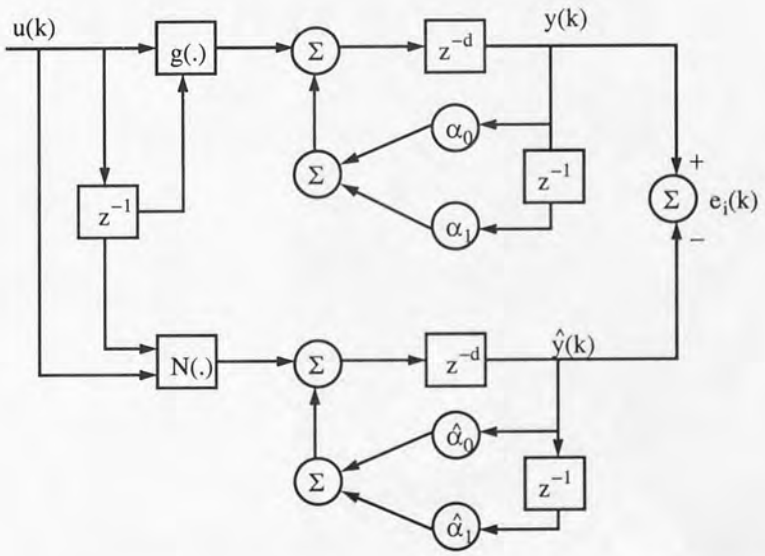


Figure 2.2: Parallel identification model. In this model the output of the neural network is fed back to its input to represent the delayed vector of the plant output.

- parallel model

  In this case the output of the identification model $\hat{y}(k+d)$ at time $k+d$ is assumed to be a linear combination of its past values, and depends nonlinearly on the past values of the input.

  $$\hat{y}(k+d) = \hat{\alpha}_0 \hat{y}(k) + \hat{\alpha}_1 \hat{y}(k-1) + N[u(k), u(k-1)]. \tag{2.14}$$

  The block diagram for this model is shown in Fig. 2.2. Due to the self feedback from the output of the neural network to its input dynamic back-propagation based on the error $e(k+d)$ between the model output $\hat{y}(k+d)$ and the actual output $y(k+d)$, $e(k+d) \triangleq y(k+d) - \hat{y}(k+d)$, is used for estimating $\hat{\alpha}_i$ as well as the parameters of the neural network.

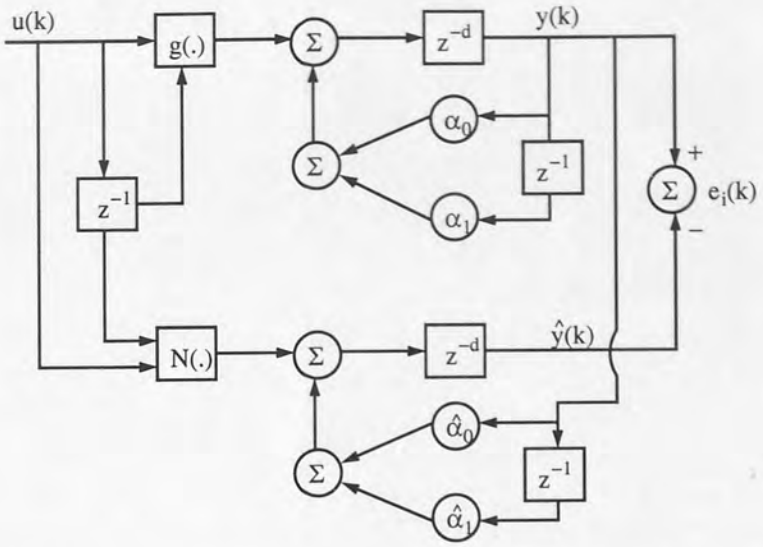


Figure 2.3: A series parallel identification model. In this model the actual output of the plant is fed back to the neural network input to represent the delayed vector of the plant output.

- series parallel model
  In contrast to the parallel model the output of the identification model $\hat{y}(k+d)$ is assumed to be a linear combination of the past values of the plant output (rather than the identification model).

$$\hat{y}(k+d) = \hat{\alpha}_0 y(k) + \hat{\alpha}_1 y(k-1) + N[u(k), u(k-1)]. \tag{2.15}$$

  The series parallel identification model has the form shown in Fig. 2.3. The parameters in this model $\hat{\alpha}_i$ as well as the neural network parameters can be adapted using static back-propagation, since no feedback loop exists in the model. Although the neural network is a static network in the series parallel model, it assumes a dynamic behaviour by embedding it in models I through IV described in Section 2.5.1.

Since the plant is bounded-input bounded-output (BIBO) stable, the series parallel method can be assured to be globally stable. In contrast to this, the boundness of the signal of the parallel models can not be guaranteed.

To assure stability for identification and control, the series parallel model is used in this work.

2.6 Process modelling

In the neurocontrol field, some of the control architectures, as will be seen in Section 2.7, are based on a model of the plant. Thus to design a good controller, an accurate model of the plant is required. Generally, real world systems are characterised by:

1. The system can be too complex to understand or to represent in a simple way, which may be due to high nonlinearity and random forces affecting its dynamics.

2. Time varying parameters.

3. Large environmental disturbances, which are difficult to predict.

4. The model is difficult or expensive to evaluate.

Motivated by superior abilities of neural networks to cope with complex problems of the type described above, significant results exist on modelling of nonlinear dynamical systems [95; 94; 11].

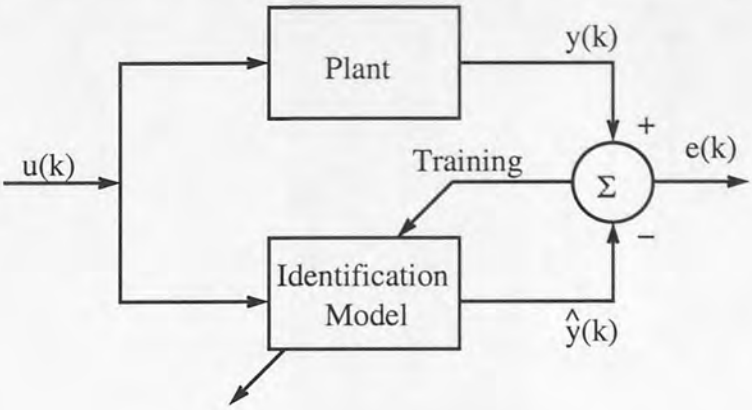


Figure 2.4: Forward identification model.

The problem of identification is shown in Fig. 2.4. It consists of three main elements: (1) the input of the plant $u(k)$ (sometimes called manipulated variables) (2) measured variable of the actual response of the plant $y(k)$ (3) and predicted response of the process $\hat{y}(k)$. The input $u(.)$ is assumed to be a uniformly bounded function of time. The plant is assumed to be stable with unknown values of the parameters. The error signal $e(k) = (y(k) - \hat{y}(k))$ is used for training the network and approximating the plant parameters. This is represented by the diagonal arrow passing through the model.

After training when the model and the actual plant are subjected to the same input $u(k)$, the model output $\hat{y}(k)$ is supposed to approximate the actual plant output $y(k)$.

Several methods have been developed to test the validity of the identification model. In [11] a model validity test is provided for the nonlinear autoregressive moving average model with exogeneous input, (NARMAX). In his test, Billings has shown that several conditions should hold if the fitted nonlinear model is adequate. The model validity test proved to be efficient in detecting the inadequacy of the fitted model whether it is due incorrect input node assignment, insufficient hidden nodes, noisy data, or a network that has not converged. Another method known as the cross-validation method is also presented in [13]. This method is based on generating three sets of data, the training data, the validation data and the testing data. This method is shown to be efficient to test the validity of the model and will be used in this work.

2.7 Adaptive control using neural networks

Adaptive control problems are concerned with controlling the output of the plant that has a known structure, but unknown parameters. More specifically, for known plant with input-output pair $\{u(k), y(k)\}$, and a reference model with input-output pair $\{r(k), y_{ref}(k)\}$, an optimal control is attempted so as to make the plant output $y(k)$ match the reference model output $y_{ref}(k)$ asymptotically

$$\lim_{k \to \infty} \| y(k) - y_{ref}(k) \| \leq \epsilon, \tag{2.16}$$

for some specified constant $\epsilon \geq 0$.

Different control architectures have been proposed in the neurocontrol field to solve this problem. Basically there are three different design methods for using neural networks as an adaptive controller[125]. The simplest approach is direct inverse modelling. Here the plant input u is selected and applied to the plant to obtain the corresponding output y. The network is then trained to reproduce the control input u as a result of the output y. When trained, the network should be able to take the desired response y_{ref} and produce the appropriate control signal u, which is then supposed to make the plant output y approach the desired response y_{ref}. 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 [88]. Here is just a brief discussion.

Assuming that θ is the adjustable parameter vector in a system and $\ddot{\theta}$ is the parameter vector of the actual system, which can be unique or belong to a set S. If $\theta \equiv \ddot{\theta}$, the system has the same transfer function as the model. When a specific input is applied to the system, the error between the desired response y_{ref} and the system response y may tend to zero even if $\theta \in S$.

However, the parameters of the unknown system can be defined exactly if the input to the system is persistently exciting, $\theta \rightarrow \ddot{\theta} \in S$. In the neural network context, if the input to the plant is sufficiently general and the weights of the neural network are adjusted using a stable method and for a long time, the error $\| \mathcal{L}_u - \hat{\mathcal{L}}_u \|$ can be made small for any input $u \in U$. Here $\mathcal{L}$ represents the operator of the plant and $\hat{\mathcal{L}}$ is the operator of the identification model. This general input can then be considered to be persistently exciting.

A second approach which has been used by Narendra [95] is called indirect adaptive control (Kawato in [60] refers to this structure as feed-forward and inverse modelling). The indirect adaptive control aims to overcome problems in the direct inverse control. In the indirect adaptive approach a trained forward model of the system is placed in parallel with the plant. As in direct inverse control, the controller precedes the system and receives as input a training signal representing the desired operational output space. The controller in this case is designed to solve an optimal control problem by minimising the gap between the desired output $y_{ref}(k)$ and the actual plant response $y(k)$

$$U(k) \triangleq \frac{1}{2} \sum_{k} (y(k) - y_{ref}(k))^2. \tag{2.17}$$

Back-propagation of utility is then used to solve this optimisation problem.

The third approach is the adaptive critic approach. This approach can be defined as a set of methods that approximate dynamic programming. It is based on the basic concept common to all forms of dynamic programming [49]. The user needs to supply a utility function U and a stochastic model of the plant to be controlled. Dynamic programming is used to solve for another function called the cost function J, which is assumed to be a function of the state variable at time k of the plant to be controlled, $x(k)$. Adaptive critic designs are defined more precisely as designs that include two neural networks: the critic network which tries to approximate the cost function J or its derivatives, and the action network which should be adapted so as to maximise J in near term future. The input to both the action and the critic networks is the state vector $x(k)$.

Based on the output supposed to be approximated by the critic network and the method for adapting the action network, three different critic designs have been proposed in the literature: (1) Heuristic dynamic programming (HDP), which adapts a critic network whose output is an approximation of

J(x(k)), (2) Dual heuristic programming (DHP), which adapts a critic network whose outputs represent the derivative of J(x(k)) [5], and (3) Globalized DHP (GDHP), which adapts a critic network whose output is an approximation of J(x(k)), but adapts it so as to minimise errors in the implied derivatives of J, as well as J itself. The reader is refered to [123; 105] for full discussion about critic designs. In the following, DHP will be assumed to illustrate the adaptive critic method. Assuming the following type of cost function:

$$J[x(k)] = U(x(k), u[x(k)]) + < J[x(k+1)] >, \tag{2.18}$$

where J[x(k)] is the cost to go from time k to the final time. $U(x(k), u[x(k)])$ is the utility, which is the cost from going from time k to time k+1. And $< J[x(k+1)] >$ is assumed to be the minimum cost from going from time $k+1$ to the final time. By defining a new variable called $\lambda(k)$ as the derivative of the cost function with respect to the state x(k) at time k

$$\lambda[x(k)] \equiv \frac{\delta J[x(k)]}{\delta x(k)}, \tag{2.19}$$

then

$$\begin{aligned} \lambda[x(k)] &= \frac{\delta U[x(k), u(k)]}{\delta x(k)} + \frac{\delta U[x(k), u(k)]}{\delta u(k)} \frac{\delta u[x(k)]}{\delta x(k)} \\ &+ \langle \lambda[x(k+1)] \frac{\delta x(k+1)}{\delta x(k)} \rangle + \langle \lambda[x(k+1)] \frac{\delta x(k+1)}{\delta u(k)} \frac{\delta u[x(k)]}{\delta x(k)} \rangle. \end{aligned} \tag{2.20}$$

Since $\langle \lambda[x(k+1)] \rangle$, $U[x(k), u(k)]$ and the system model derivatives are known, then $\lambda[x(k)]$ can be calculated. Finally the optimality equation is defined as

$$\frac{\delta J[x(k)]}{\delta u(k)} = 0. \tag{2.21}$$

The above two equations are usually used in dynamic programming to solve an infinite or finite horizon control policy.

The training process for the adaptive critic network comes in two stages. The training of the action network which outputs the optimal control policy u[x(k)] and the training of the critic network which approximates the derivative of the cost function $\lambda[x(k)]$. As a first step in the training process, the critic and the action networks need to be designed and the initial weights of these networks should be randomised. Since the derivative of the utility function can be calculated, this in combination with the critic outputs and the system model derivatives, allow the use of Eq. (2.20) to calculate the target value of the critic $\lambda^*[x(k)]$. The difference between $\lambda^*[x(k)]$ and $\lambda[x(k)]$ is used to correct the critic network, until it converges. The output from the converged critic is used in Eq. (2.21) solving for the target $u^*(k)$ which is then used to correct the action network. Since the model of the system can be linear or nonlinear, this latter equation can be solved by using one of the numerical analysis methods, for example Newton's method [63]. These two steps continue until a predetermined level of convergence is reached.

Other approaches to solve the adaptive control problem have been discussed in [125]. Among these are the back-propagation of utility using back-propagation through time, and the feed back error learning.

2.8 Summary and discussion

In this chapter the use of neural networks for identification and control was introduced. Models of the nonlinear plant as well as the identification procedure have been discussed. Three different control architectures for the use of neural networks in adaptive control problems were presented. To achieve the goal of this work which is estimating the uncertainty around the predicted output of the controller and using this estimate in a profitable manner, the inverse control architecture is chosen to be used in this thesis as an adaptive controller. The advantage of using the inverse control architecture in this work is that it provides an explicit way for estimating the uncertainty of the controller, as will be shown in chapter 4. In the inverse control architecture, the controller can be trained using supervised learning, in which the desired response (target values) in the training data plays a role of a teacher.

Since the series parallel identification model discussed in Section 2.5.2 can be assured to be globally stable, it will be used in this work. Providing that the series parallel model is chosen, only static neural networks will be required.

The decision of whether to use a RBF or MLP neural network is problem dependent. Therefore, both the RBF and MLP networks are used in this thesis.

Chapter 3

Uncertainty Estimation and Distribution Modelling

As demonstrated in Chapter 2, several control architectures have been developed in the literature to optimally solve nonlinear control problems. These architectures are developed to handle three main difficult problems in process control: complexity, nonlinearity and uncertainty. Because neural networks consist of nonlinear processing elements, they can handle nonlinearities in control processes easily. Compared with other nonlinear approximation methods, neural network are relatively less sensitive to noise and incomplete information, thus they can deal with higher levels of uncertainty in control problems. Resolving the uncertainty of neural network outputs however, has not been addressed in all of the developed control architectures. All the developed architectures in the neurocontrol field are based on the assumption that the output of a neural network is reliable. This assumption however may be inappropriate for the output obtained from a neural network presented with novel or unrepresentative input data. Moreover, in real world control problems it is important for the user to know how reliable is the predicted output from the controller to avoid instability problems if possible. In these situations qualitative measures for the uncertainty of the neural network output become necessary to validate the prediction from the neural network.

Different methods have been suggested in the literature for estimating neural network confidence intervals. Such methods allow us to approximate the distribution of the output given the model parameters and data, rather than predicting a single value (the mean) as in classical control approaches. Different assumptions of model specifications and data noise distributions have been used for different approaches with trade-offs in terms of complexity and efficiency among each approach. In real world control problems where real time control is necessary, knowledge about uncertainty should be provided in a timely manner, even if this information is suboptimal. Hence knowledge of the best error bar becomes useless if it takes too long to compute, or even if it requires expensive computations.

In this chapter we will make a comparative study of error bars obtained by three different neural network approaches. We will investigate the use of these three methods in providing an estimate for the error bar in control problems.

The definition of the confidence interval is firstly introduced in section 3.1. In section 3.2 the standard Bayesian method for estimating error bars is described. Different cases for modelling the confidence interval are also discussed. Section 3.3 discusses the Bayesian technique with input noise. Section 3.4 presents the Gaussian processes. The predictive error bar method is presented in sec-

tion 3.5. Other methods including maximum likelihood approaches and the bootstrapping method for error bar estimation are also discussed in section 3.6. A comparative study applied to a synthetic sine function between the Bayesian method, Gaussian processes and predictive error bar methods is given in section 3.7. In section 3.8, the three methods for estimating error bars are applied to a dynamical control example. Finally the discussion is given in section 3.9.

3.1 Confidence intervals

In this work we are concerned with control problems, so we focus on regression problems for error bar estimation. The development in this chapter will be based on using the RBF network with Gaussian basis functions, though other basis functions can be assumed under the same development.

In this work, model IV will be assumed as a representation model for the plant and the controller. For example in SISO dynamical systems the data in the approximation problem come in the form of a set of input output pairs $[u(k), y(k+d)]$ [1]. Therefore for the input vector $u(k) = u^1(k), ..., u^N(k)$ in the set $u(k)$, there is a target vector $y(k+d) = y^1(k+d), ..., y^N(k+d)$ belonging to the set $y(k+d)$.

The inference is made from a set of data $D = \{u(k), y(k+d)\}$, the set of inputs and targets. As introduced in chapter 1, in real world problems noise usually affects the relation between the inputs and outputs. The targets are usually related to the inputs through a nonlinear function $y(k+d) = g[u(k), \mathbf{x}(k), \bar{\nu}(k)]$, with the noise being assumed to have a nonlinear effect on the relationship between the inputs and outputs. Here $\mathbf{x}(k)$ is a vector of the previous inputs and outputs of the system, $\mathbf{x}(k) = [y(k), ..., y(k-q+1), u(k-1), ..., u(k-p+1)]$ [2]. q, and p are the maximum delays of the output and input respectively. The regression problem is then to try and find a regression function $\hat{g}[u(k), \mathbf{x}(k), \boldsymbol{W}]$ conditioned on the input values $[u(k), \mathbf{x}(k)]$ and the weights of the model $\boldsymbol{W}$.

In the limit of infinite amount of data the regression function $\hat{g}[u(k), \mathbf{x}(k), \boldsymbol{W}]$ approaches the conditional mean of the target data, by averaging over noise values. However, due to the noise on the target data, we cannot predict the output with certainty. The uncertainty in the predicted output is known as the predicted variance $\sigma^2_{y(k+d)}$, and the confidence interval is specified by $\pm\sigma_{y(k+d)}$. Several methods have been suggested in the literature to estimate the confidence limits for this prediction problem. Some of these methods calculate the residual error result from the regression problem $|g[u(k), \mathbf{x}(k), \bar{\nu}(k)] - \hat{g}[u(k), \mathbf{x}(k), \boldsymbol{W}]|$. An estimation for the variability caused by the error term allow calculating the confidence interval, or more generally the prediction variance for the model. Other methods decompose the confidence interval of the model into several terms and provide an estimation for each term. They assume that the total variance around the predicted output of the nonlinear approximator can be expressed as the sum of estimated noise of the target values $\sigma^2_{\bar{\nu}}$ and the estimated variance around the parameters, $\sigma^2_{y(k+d)} = \sigma^2_{\bar{\nu}} + \sigma^2_{\boldsymbol{W}}(u(k), \mathbf{x}(k))$. In the following sections we will review three pragmatic methods for providing confidence interval estimation.

3.2 Bayesian techniques

In the Bayesian inference method to error bar modelling discussed in [77; 78; 13], there are basically two sources of error: the first is the deviation of the estimated neural network parameters from the

[1] since we are using model IV as a representation model for the forward and inverse problems, the development for error bar estimation will be constructed assuming the dynamical model IV.

[2] Because the input to the neural network is a vector of the previous input output, the problem can be considered as a MISO

real ones, the second is the intrinsic noise on the target data which is usually taken to be constant. These two terms are treated as independent and hence the total output error can be written as

$$\sigma^2_{y(k+d)}(\mathbf{s}(k)) = \sigma^2_{\hat{v}}(\mathbf{s}(k)) + \sigma^2_{\boldsymbol{W}}(\mathbf{s}(k)), \tag{3.1}$$

where $\mathbf{s}(k) = [u(k), \mathbf{x}(k)]$, $\sigma^2_{\boldsymbol{W}}(\mathbf{s}(k))$ is the variance of the output due to the neural network parameter uncertainty, $\sigma^2_{\hat{v}}(\mathbf{s}(k))$ is the variance of the target data, and $\sigma^2_{y(k+d)}(\mathbf{s}(k))$ is the overall output variance. The errors due to model complexity are not considered in the above equation. Given the Gaussian approximation to the posterior weight distribution, and the assumption that the output depends linearly on the weight $\boldsymbol{W}_{MP}$ by its linear expansion around $\boldsymbol{W}_{MP}$

$$\hat{g}(\mathbf{s}(k), \boldsymbol{W}) = \hat{g}(\mathbf{s}(k), \boldsymbol{W}_{MP}) + G^T(\boldsymbol{W} - \boldsymbol{W}_{MP}), \tag{3.2}$$

where $G \equiv \partial\hat{g}(\mathbf{s}(k), \boldsymbol{W})/\partial\boldsymbol{W}$ is the vector of derivatives of the outputs with respect to the weights measured at $\boldsymbol{W}_{MP}$, the distribution $p(y(k+d)|\mathbf{s}(k), D)$ is Gaussian with mean $\hat{g}(\mathbf{s}(k), \boldsymbol{W}_{MP})$ and variance $\sigma^2_{y(k+d)}(\mathbf{s}(k))$ given by Eq. (3.1), Appendix A.1.

The first term of $\sigma^2_{\hat{v}}(\mathbf{s}(k))$ is usually assumed to be constant and can be approximated by [13]

$$\sigma^2_{\hat{v}}(\mathbf{s}(k)) = \beta^{-1} = \frac{2E_D}{N - \gamma}. \tag{3.3}$$

The second term in Eq. (3.1) is approximated by (Appendix A.1)

$$\sigma^2_{\boldsymbol{W}}(\mathbf{s}(k)) = G^T(\mathbf{s}(k))\boldsymbol{A}^{-1}G(\mathbf{s}(k)), \tag{3.4}$$

where $\boldsymbol{A}$ is the Hessian matrix of the error function measured at the most probable value of the weight $\boldsymbol{W}_{MP}$, N is the number of training examples, γ is the number of well determined parameters in the model and E_D is the error measured on the training set. The parameter γ according to the full Bayesian treatment of the error bars can be estimated by [13]

$$\gamma = K - \alpha \ \mathrm{Trace}(\boldsymbol{A}^{-1}), \tag{3.5}$$

where K is the number of parameters in the model and α is a regularisation parameter estimated from

$$\alpha = \frac{\gamma}{2E_{\boldsymbol{W}}(\boldsymbol{W}_{MP})}, \tag{3.6}$$

where $E_{\boldsymbol{W}}(\boldsymbol{W}_{MP}) = (1/2)\boldsymbol{W}^T_{MP}\boldsymbol{W}_{MP}$. In implementing the full Bayesian approach both α and β are reestimated. In this chapter the development for error bars assuming an RBF network was implemented using two different approaches. In the first approach, Bayesian error bars were estimated for a linear output layer only. In the second approach, the error bars were estimated by taking the derivatives with respect to the weights in the output layer in addition to the means and the variances of the radial basis functions.

3.2.1 Bayesian error bars for linear output layer only

For this case, the calculation of the approximate Bayesian error bars is simplified since the Hessian matrix is given by the following exact formula [13]

$$\boldsymbol{A} = \beta\phi^T\phi + \alpha\mathbf{I}, \tag{3.7}$$

where ϕ is chosen to be an unnormalised Gaussian basis function. Because the output of the network depends linearly on the weights, direct matrix inversion was used to estimate the weight values using the following formula.

$$W = (\beta\phi\phi^T + \alpha I)^{-1}(\beta\phi^T)y(k+d), \tag{3.8}$$

where W is the weight vector in the output layer only, β^{-1} is the added noise to the output values, α is the prior of the weights, and $y(k+d)$ are the target values. The derivation of this formula is discussed in more detail in (Appendix B.1).

3.2.2 Bayesian error bars by differentiating with respect to all parameters of the RBF network

In this case the Hessian matrix was calculated using the outer product approximation and is found to be

$$A = \beta G^T G + \alpha I, \tag{3.9}$$

where G is the weight gradient of the neural network output. But because in this case the error function is a nonlinear function of the network parameters, Levenberg-Marquardt training or other gradient methods become mandatory for the adjustment of the parameters. This, in turn, implies that the parameters can converge to local minima. Further, the adjustment of a single parameter of the network affects the output globally [129; 37]. Hence, in general, all the weights have to be adjusted simultaneously for each training data set. The updated formula for the weights was found to be, (since this is a new result, Appendix B.2 gives a more detailed derivation of this formula).

$$W_{new} = W_{old} - (\beta G^T G + (\alpha+\lambda)I)^{-1}[\beta G^T \underline{f}(W_{old}) + \alpha W_{old}], \tag{3.10}$$

where $\underline{f}$ is the difference between the output of the network and the target values, G is the weight gradient of the network output, W_{new} is the new weight vector which includes the weights in the output layer and the means and variances of the RBF, W_{old} is the old weight vector.

3.3 Bayesian regression with input noise

Wright [132] considered the possibility of accounting for uncertainty in the input in the Bayesian framework. His argument was that an estimate of uncertainty in the standard Bayesian method is incomplete, since in real world problems the input can also be subjected to noise as is the target. In his work, three sources of error are considered to contribute to the predictive error bar of the model. The first two are the deviation of the estimated neural network parameters from the real ones, and the intrinsic noise on the target data which are the same as in the standard Bayesian method. The third is the intrinsic noise on the input data provided that a model of the noise process exist. Again these three terms are assumed to be independent and hence the total output error can be written as

$$\sigma^2_{y(k+d)}(s(k)) = \sigma^2_W(s(k)) + \sigma^2_v(s(k)) + \sigma^2_I(s(k)), \tag{3.11}$$

where $\sigma^2_W(s(k))$ is the variance of the output due to neural network parameter uncertainty, $\sigma^2_v(s(k))$ is the variance of the target data, $\sigma^2_I(s(k))$ is the variance of the input data, and $\sigma^2_{y(k+d)}(s(k))$ is the overall output variance. Similarly the error due to the model complexity is not considered in the above equation.

Again using the Laplace approximation the predicted distribution $p[y(k+d)|(\mathbf{s}(k), D)]$ is Gaussian with mean $\hat{g}(\mathbf{s}(k), \boldsymbol{W}_{MP})$ and variance $\sigma^2_{y(k+d)}(\mathbf{s}(k))$ given by Eq. (3.11), (Appendix A.2).

The updating equations for the first two terms in Eq. (3.11), ($\sigma^2_{\mathbf{W}}(\mathbf{s}(k))$, and $\sigma^2_{\hat{v}}(\mathbf{s}(k))$), are given by Eq. (3.4), and Eq. (3.3) respectively. The last term in the predicted variance $\sigma^2_{y(k+d)}(\mathbf{s}(k))$ is approximated by

$$\sigma^2_I(\mathbf{s}(k)) = \mathbf{h}^T \Sigma_{\mathbf{s}(k)} \mathbf{h}, \tag{3.12}$$

where $\mathbf{h} = \partial \hat{g}(\mathbf{s}(k), \boldsymbol{W})/\partial \mathbf{s}(k)$ is the input gradient of the neural network output measured at the noisy input data $\mathbf{z}(k)$, and $\Sigma_{\mathbf{s}(k)}$ is the full covariance matrix of the input noise which is usually assumed to be a known constant small additive Gaussian noise.

The same updating formulas in Section 3.2.1, Eq. (3.8) for the linear output layer, and in Section 3.2.2, Eq. (3.10) for all the parameters of the RBF network can be used for the Bayesian method with input noise.

3.4 Gaussian processes

Another approach to error bar estimation is the Gaussian process model [79; 97; 127; 128]. This approach can be considered as the generalisation of a Gaussian distribution over a finite vector space to a function space of infinite dimensions. Just as a Gaussian distribution can be specified by its mean and covariance matrix, a Gaussian process is also specified by a mean and a covariance function. In general the mean and the variance of the predictive distribution are given by [85], Appendix C

$$E[y^{N+1}(k+d)|y^1(k+d), .., y^N(k+d)] = y^{N+1}(k+d) = \mathbf{K}^T \mathbf{C}_N^{-1} \mathbf{y}(k+d), \tag{3.13}$$

$$\sigma^2[y^{N+1}(k+d)|y^1(k+d), .., y^N(k+d)] = \sigma^2_{y^{N+1}(k+d)}(\mathbf{s}(k)) = \kappa - \mathbf{K}^T \mathbf{C}_N^{-1} \mathbf{K}, \tag{3.14}$$

where $\mathbf{s}^N(k) = [u^1(k), .., u^N(k), \mathbf{x}^1(k), .., \mathbf{x}^N(k)]$, is the set of the training examples, $\mathbf{y}(k+d) = y^1(k+d), ..., y^N(k+d)$ is the set of training targets, $\mathbf{K} = [C(\mathbf{s}^{N+1}(k), \mathbf{s}^1(k)), ..., C(\mathbf{s}^{N+1}(k), \mathbf{s}^N(k))]^T$, $\mathbf{C}_N$ is the $N*N$ covariance matrix for the training data from Eq. 3.15, and κ is the covariance matrix of the predicted data $C(\mathbf{s}_i^{N+1}(k), \mathbf{s}_j^{N+1}(k))$. $\mathbf{C}_N(\mathbf{s}_i(k), \mathbf{s}_j(k))$ is an a-priori specified covariance function that determines the shape of the prior distribution taken over function space. In this work it is chosen to be

$$\mathbf{C}_N(\mathbf{s}_i(k), \mathbf{s}_j(k)) = \theta_1 \exp\left(-\frac{1}{2} \sum_{i=1}^{I} \frac{(\mathbf{s}_i(k) - \mathbf{s}_j(k))^2}{r_i} \right) + \theta_2, \tag{3.15}$$

where $\mathbf{s}_i(k)$ is the i-th component of $\mathbf{s}(k)$, I is the dimensionality of the vector $\mathbf{s}(k)$, and θ_1, θ_2, r_i $\in \Theta$. In practice we will not have sufficient prior knowledge about the appropriate values for the hyper-parameters that define the covariance (θ_1, θ_2, and r_i). Therefore a prior distribution to the hyper-parameters should be given. Prediction is then made by estimating the posterior distribution of the hyper-parameters $p(\theta|D) \alpha p(\mathbf{y}(k+d)|\mathbf{s}^N(k), \theta) p(\theta)$. This in turn requires computation of the log likelihood (the log of $p(\mathbf{y}(k+d)|\mathbf{s}^N(k), \theta)$) based on the N observed data

$$\mathcal{L} = -\frac{N}{2} \log(2\pi) - \frac{1}{2} \log \det \mathbf{C}_N - \frac{1}{2} \mathbf{y}^T(k+d) \mathbf{C}_N^{-1} \mathbf{y}(k+d). \tag{3.16}$$

The derivative of $\mathcal{L}$ with respect to the hyper-parameter θ have also to be computed to enable estimation of θ

$$\frac{\partial \mathcal{L}}{\partial \theta} = -\frac{1}{2} \mathrm{Trace}(\mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta}) + \frac{1}{2} \mathbf{y}^T(k+d) \mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta} \mathbf{C}_N^{-1} \mathbf{y}(k+d). \tag{3.17}$$

Hyper-parameters of the Gaussian process can then be estimated using one of the nonlinear optimisation methods. In this thesis the scaled conjugate gradient training method is used. This approach has two main disadvantages. Firstly, the requirement of inverting an $N \times N$ matrix which can be time consuming for large data sets. Secondly, numerical inaccuracies might result from that inversion.

3.5 Predictive error bars

The third approach to error bar estimation by neural networks is called predictive error bar estimation. It was first suggested by Satchwell in [112]. This approach is based on the fact that for a network trained on minimum square error the optimum network output approximates the conditional mean of the target data, or $y_{opt}(k+d) =< y(k+d)|\mathbf{s}(k) >$, see Section 2.4. Hence for each input pattern $\mathbf{s}(k)$ the local variance can be estimated as $\|y(k+d) - y_{opt}(k+d)\|^2$. 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 $\sigma^2(\mathbf{s}(k)) =<\| y(k+d) - < y(k+d)|\mathbf{s}(k) >\|^2 |\mathbf{s}(k) >$. In [112], two neural networks are used to model the mean $y_{opt}(k+d)$ and the variance $\sigma^2(\mathbf{s}(k))$ of the distribution model.

Lowe [76] performed a comparative study for three different methods of error bars estimation: The Bayesian technique, Gaussian processes, and the predictive error bar. As reported in [76], in the implementation of predictive error bars two 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 Fig. 3.1.

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 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 weights to the squared error values $\sigma^2(\mathbf{s}(k))$.

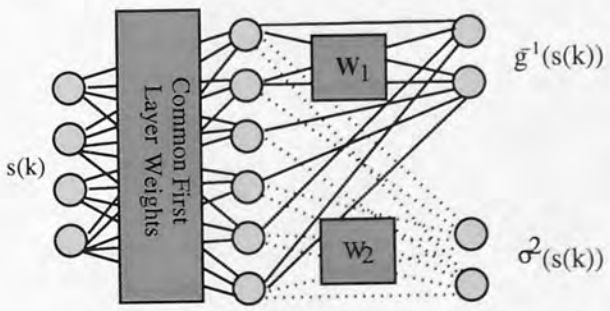


Figure 3.1: The architecture of the predictive error bar.

3.6 Other methods of error bar estimation

So far we have discussed the most suitable methods for providing confidence interval estimation. However, there are other approaches to error bar estimation discussed in the literature. They can be divided into two groups: bootstrapping techniques and maximum likelihood approaches. Here we review these methods for completeness.

3.6.1 Maximum likelihood

An approach called the delta method [118; 107] can provide similar results to the Bayesian method but using a maximum likelihood approach. It is based on the assumption that the target is Gaussian distributed with $\sigma^2_{y(k+d)}$ being the estimated variance by maximum likelihood

$$\sigma^2_{y(k+d)}(\mathbf{s}(k))\alpha \sum_{l=1}^{N} [y_l(k+d) - \hat{g}(\mathbf{s}_l(k), \boldsymbol{W})]^2, \tag{3.18}$$

where N is the number of training examples and $y(k+d)$ is the desired target. Given the assumption that the outputs $\hat{g}(\mathbf{s}(k), \boldsymbol{W})$ depend linearly on the weights around the optimal value

$$\hat{g}(\mathbf{s}(k), \boldsymbol{W}) \approx \hat{g}(\mathbf{s}(k), \boldsymbol{W}_{MP}) + G^T \Delta \boldsymbol{W}, \tag{3.19}$$

the confidence variance is given by

$$\sigma^2_{\boldsymbol{W}}(\mathbf{s}(k)) = \sigma^2_{\hat{v}} \mathbf{G}^T(\mathbf{s}(k)) \boldsymbol{A}^{-1} \mathbf{G}(\mathbf{s}(k)), \tag{3.20}$$

here $\boldsymbol{A} = \partial^2 E_D / \partial \boldsymbol{W}^2$ is the Hessian $G(\mathbf{s}(k)) = \partial \hat{g}(\mathbf{s}(k), \boldsymbol{W}) / \partial \boldsymbol{W}$ measured at $\boldsymbol{W}_{MP}$ is the gradient, and $E_D(\boldsymbol{W}) = 1/2 \sum_{l=1}^{N} (y_l(k+d) - \hat{g}(\mathbf{s}_l(k), \boldsymbol{W}))^2$.

An estimate for the intrinsic noise in the target data is obtained by minimising the negative log likelihood $p(D|\boldsymbol{W}, \beta) = (\beta/2\pi)^{(N/2)} \exp(-\beta E_D(\boldsymbol{W}))$ with respect to $\sigma^2_{\hat{v}} = \beta^{-1}$

$$\sigma^2_{\hat{v}} = \frac{2E_D}{N}. \tag{3.21}$$

The total predictive variance is then

$$\sigma^2_{y(k+d)}(\mathbf{s}(k)) = \frac{2E_D}{N} + \sigma^2_{\hat{v}} G^T(\mathbf{s}(k)) \boldsymbol{A}^{-1} G(\mathbf{s}(k)). \tag{3.22}$$

A similar approach called the sandwich estimator is also considered in [118]. This approach can perform well under model misspecification.

An alternative approach which is more efficient for estimating the confidence interval called error estimation by series association is developed in [62]. It can be understood as an improved approach to the stacked generalisation proposed by Wolpert [130].

In error estimation by series association two neural networks are used. The first network is trained to approximate the output function $y(k+d)$ by a nonlinear approximator $\hat{g}(\mathbf{s}(k), \boldsymbol{W})$. The corresponding error signal from this approximation is then calculated $\epsilon(\mathbf{s}(k), \boldsymbol{W}) = |y_l(k+d) - \hat{g}(\mathbf{s}_l(k), \boldsymbol{W})|$. The second network takes as input the input vector $\mathbf{s}(k)$, the output vector of the first network, and provides an approximation to the error term ϵ. Because the output from the first network is directly connected to the inputs of the error estimation network, this method is called error estimation by series association.

Another approach for estimating the noise variance on the predicted output of the neural network is called the mixture density network [13]. Here the distribution of the targets is modelled by a Gaussian mixture with means variances and mixing coefficients as input dependent variables which are modelled using the outputs of the neural network. This approach can approximate any distribution to arbitrary accuracy. Full discussion of this approach is deferred to Chapter 6.

3.6.2 Bootstrapping

A straightforward method is based on bootstrapping [24; 107]. Here B independent samples are obtained, each sample is generated with n data points from the original data set. The n data points are drawn randomly with replacement from the available data set. Weight parameters of the regression model are determined for each bootstrap sample. The outputs of the different obtained regression models are averaged to provide an estimate of the mean value for the output

$$\hat{g}(\mathbf{s}(k)) = \frac{1}{B}\sum_{l=1}^{B}\hat{g}(\mathbf{s}(k), \boldsymbol{W}_l). \tag{3.23}$$

Since the bootstrap outputs $\hat{g}(\mathbf{s}(k), \boldsymbol{W}_l)$ are noisy measurement of the true function $g(\mathbf{s}(k))$, then estimate of the confidence interval can be given by

$$\sigma^2_{\boldsymbol{W}}(\mathbf{s}(k)) = \frac{1}{B-1}\sum_{l=1}^{B}[\hat{g}(\mathbf{s}(k), \boldsymbol{W}_l) - \hat{g}(\mathbf{s}(k))]^2. \tag{3.24}$$

The predictive variance of the bootstrap samples is then given by

$$\sigma^2_{y(k+d)}(\mathbf{s}(k)) = \sigma^2_{\nu} + \sigma^2_{\boldsymbol{W}}(\mathbf{s}(k)). \tag{3.25}$$

Although bootstrap methods provide the most accurate estimate for errors of predicted values as concluded in [118], this approach is impractical in applications of neural networks. This is because an optimisation of the model complexity and parameters need to be carried for each bootstrap sample. Moreover this approach is inappropriate for providing on-line estimation for the confidence interval as required in control problems.

3.7 Synthetic example

As a simple example for comparing the behaviour of the three different approaches to estimating error bars, consider the noisy sine wave regression problem. For comparison the same training data was used for each. There were 40 data points evenly sampled from a sine wave $y = \sin(2\pi x)$, $x = [0, 1]$. Gaussian noise $\mathcal{N}(0, \sigma_t^2)(\sigma_t = 0.1)$ was added to y to generate the noisy target t^N, and $\mathcal{N}(0, \sigma_x^2)(\sigma_x = 0.1)$ was added to x. Fig. 3.2 shows the result of the regression curves, the data used in the training stage, the actual function, and the calculated error bar displayed as 2σ curves around the local predicted means, by using an unnormalised Gaussian function as a basis function for the RBF network. Note that all models produce good regression curves, but the error bars are different. The Gaussian process error bars are small in the region of high noise, which should not be the case, in addition to that it requires significant computation time. The Bayesian error bars seem to be unrealistic by updating the weights in the output layer only. More realistic error bars are obtained by updating all the parameters of the network, since in this method we can see that the error bars are broad in the region of low

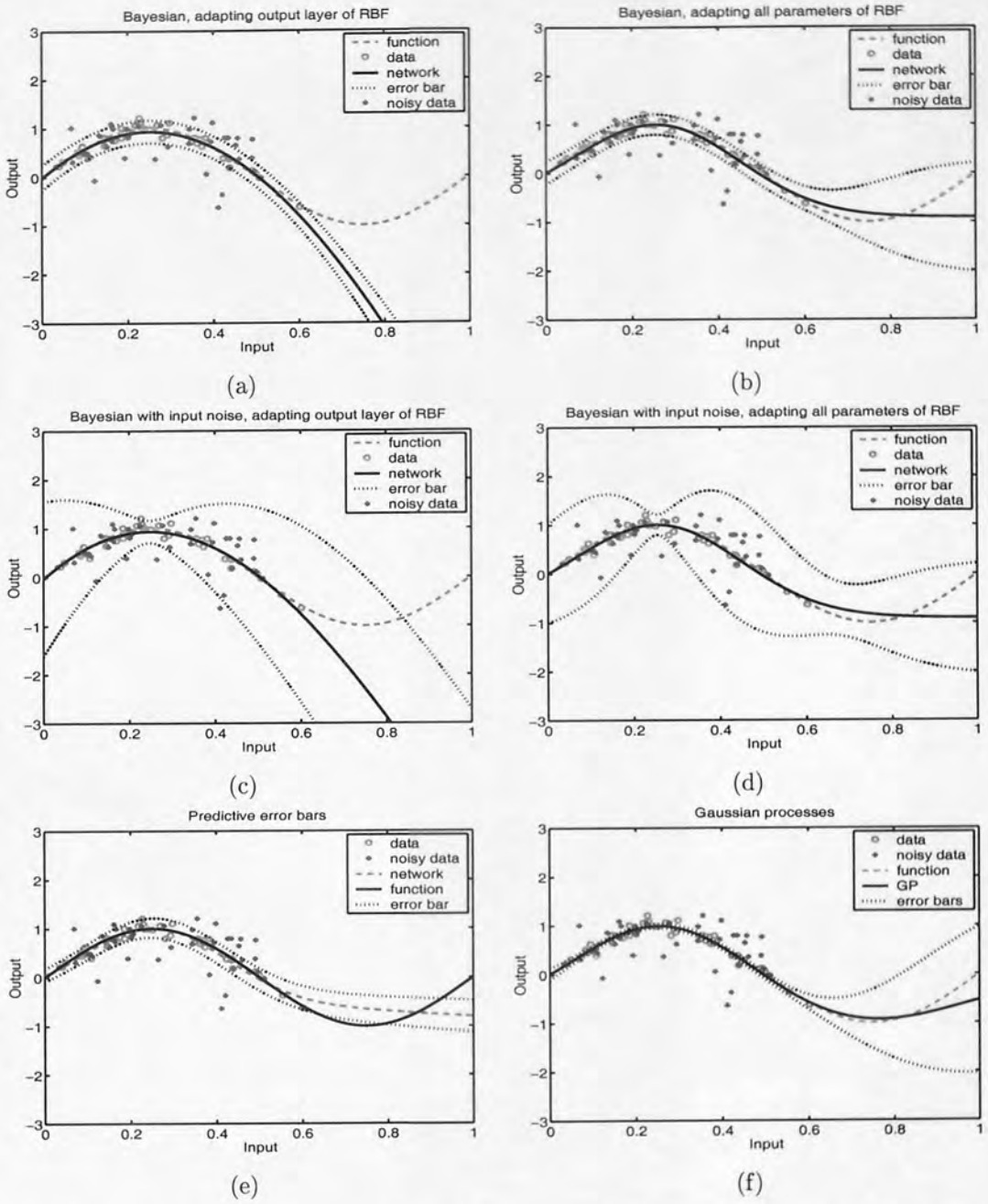


Figure 3.2: Error bar estimation from three different methods: (a) RBF network with standard Bayesian error bars (input noise term is not included), adapting the linear output layer only with 5 outer loops, in this case the final values for the hyper-parameters were found to be, $\alpha = 0.0006, \beta = 79.337$, and the number of hidden units was 2. (b) RBF network with standard Bayesian error bars (input noise term is not included), adapting all the parameters with 5 outer loops, in this case the final values for the hyper-parameters were found to be, $\alpha = 1.8, \beta = 101.8$, and the number of hidden units was 2. (c) RBF network with Bayesian error bars accounting for the input noise, adapting the linear output layer only with 5 outer loops, in this case the final values for the hyper-parameters were found to be, $\alpha = 0.0006, \beta = 79.337$, and the number of hidden units was 2. (d) RBF network with Bayesian error bars accounting for the input noise, adapting all the parameters with 5 outer loops, in this case the final values for the hyper-parameters were found to be, $\alpha = 1.8, \beta = 101.8$, and the number of hidden units was 2. (e) RBF network with predictive error bars, adapting all the parameters, the number of hidden units was 2. (f) Gaussian process model predictions with error bars, here the final values of the hyper-parameters were found to be, bias = 0.39, noise = 0.0095, inverse length scale = 15.27, vertical scale = 0.716

data density, it also displays large errors in the regions of noisy data. However, the predictive error bars require least processing time. In addition, since we use a neural network to model the residual errors on the target data this method can be considered to provide an estimation for any variability in the output, which can be due to the noise on the target data or due to the uncertainty in model parameters.

3.8 Control example

To demonstrate the problem of error bar estimation of dynamical systems as in control applications, the plant described by the following equation is taken as an example

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k). \tag{3.26}$$

A series parallel identification model described by the following equation was chosen to identify the plant

$$\hat{y}(k+1) = N_f[y(k), u(k)], \tag{3.27}$$

where N_f is an RBF network with 2 inputs, a single output and 6 Gaussian basis functions.

In the training stage, the input u to the plant were generated uniformly over the interval $[-2, 2]$. Only 100 training patterns were used. For testing the resulting models, the input to the plant was generated from the following sine wave function

$$u(k) = \sin\left[\frac{2\pi k}{25}\right] + \sin\left[\frac{2\pi k}{10}\right]. \tag{3.28}$$

The result of three different approaches for estimating error bars is shown in Fig. 3.3. All models produce good approximations for the target data. The figures on the right hand side show the value of the estimated standard deviation against time from the three methods. From these figures we see that the Gaussian process error bars are very small compared to the Bayesian and predictive error bars. The result from the Gaussian process reflects the fact that there has been no additive or multiplicative noise added to the target values. Although the Bayesian method provides higher values for the standard deviation at the peaks of the target data, it is clear that the variability in the standard deviation of predictive error bars is higher.

3.9 Discussion

So far several methods for estimating error bars on neural network output predictions have been studied: the Bayesian error bar, Gaussian processes, and the predictive error bar. Bayesian methods for estimating the confidence interval of regression problems based on estimating a fixed, but unknown quantity. Three source of noise in the improved Bayesian method could be estimated. On the other hand the predictive error bar provides an estimation for a random variable, the error in the forecast. It is the most used method in control applications [125; 84], although ignored in all the current control architectures.

The predictive error bar method can provide a general stochastic model for the required prediction. However in this work it is assumed that the covariance matrix of the error, $< ee^T >$ is diagonal, and that the noise in the forecast follows a normal distribution.

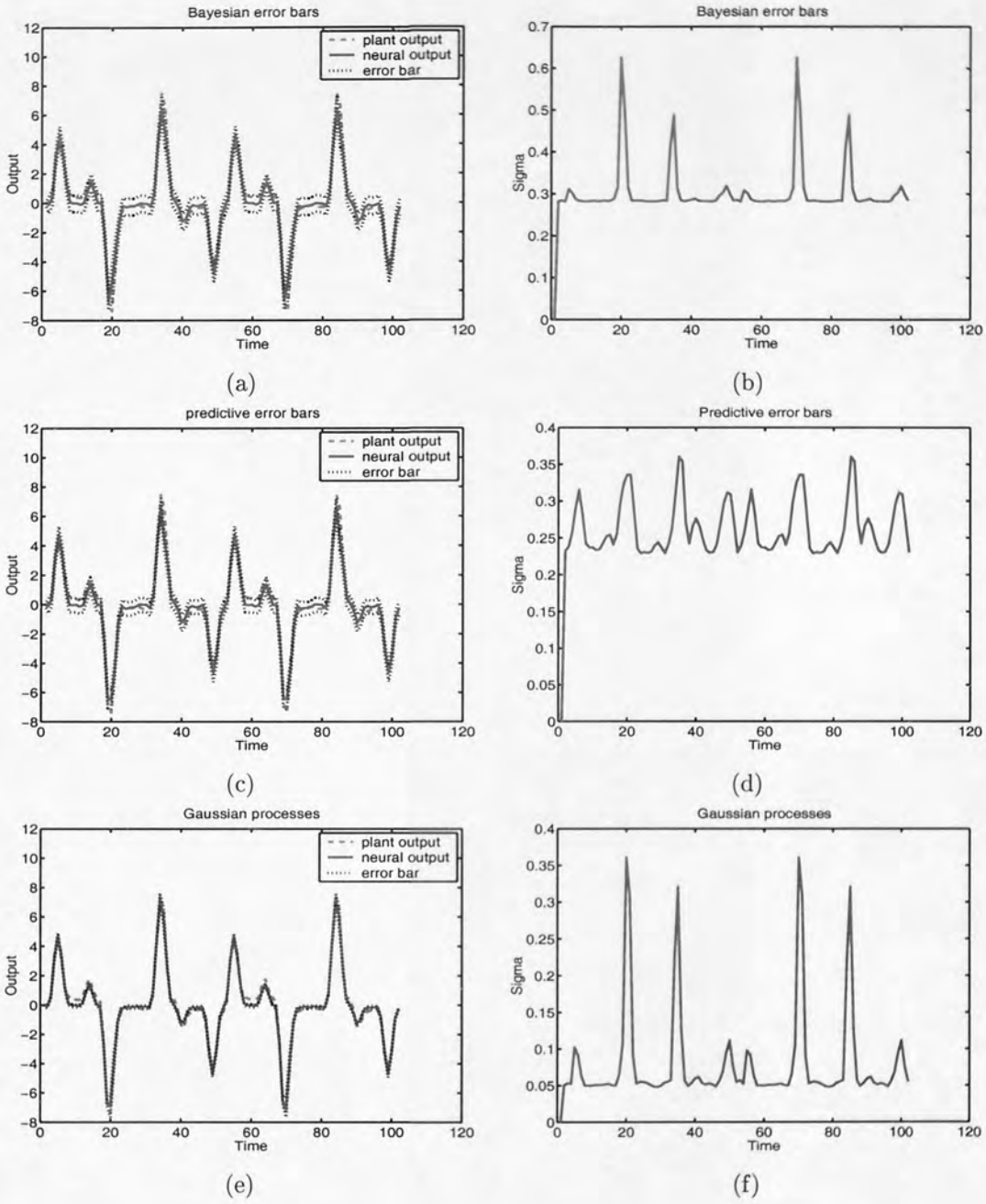


Figure 3.3: Error bar estimation for a dynamical control problem from three different methods: (a) RBF network with standard Bayesian error bars (input noise term is not included), adapting all the parameters with 3 outer loops, in this case the final values for the hyper-parameters were found to be, $\alpha = 0.4225, \beta = 860.74$, and the number of hidden units was 6. (b) The value of the Bayesian estimation of standard deviation σ against time. (c) RBF network with predictive error bars, adapting all the parameters, the number of hidden units was 6. (d) The value of the predictive error bar estimation for the standard deviation σ against time. (e) Gaussian process model predictions with error bars, here the final values of the hyper-parameters were found to be, bias = 1.265, noise = 0.0008, inverse length scale = 0.422, vertical scale = 15.913. (f) The value of the Gaussian processes estimation for the standard deviation σ against time

Although it has not been considered in the literature, Bayesian methods can also provide an estimate for the full covariance matrix of the neural network model uncertainty. Since the estimate of the confidence interval is accomplished by assuming a prior distribution functions for the weight parameters and the target values of the neural network, an estimate for the covariance matrix of the noise terms in the Bayesian method can be provided as long as the posteriors of the distributions can be calculated. More specifically, the likelihood function in Bayes theorem has the following form, see Appendix A

$$\begin{aligned} p(D|\boldsymbol{W}) &= \prod_{i=1}^{N} p(\mathbf{y}_i(k+d)|\mathbf{s}_i(k), D), \\ &= \frac{1}{Z_D(\Sigma)} \exp\left(-\frac{1}{2}\sum_{i=1}^{N}\{\mathbf{y}_i(k+d) - \hat{\mathbf{g}}[\mathbf{s}_i(k), \boldsymbol{W}]\}\, \Sigma\, \{\mathbf{y}_i(k+d) - \hat{\mathbf{g}}[\mathbf{s}_i(k), \boldsymbol{W}]\}^T\right), \end{aligned} \tag{3.29}$$

given that the output is a vector of dimension m now. As long as the normalisation factor $Z_D(\beta)$, which is the product of N independent m dimensional Gaussians can be calculated, the rest of the derivation for estimating the confidence interval remains the same as for the multiple input single output case given in Appendix A.

$$Z_D(\Sigma) = \int \exp\left(-\frac{1}{2}\sum_{i=1}^{N}\{\mathbf{y}_i(k+d) - \hat{\mathbf{g}}[\mathbf{s}_i(k), \boldsymbol{W}]\}\, \Sigma\, \{\mathbf{y}_i(k+d) - \hat{\mathbf{g}}[\mathbf{s}_i(k), \boldsymbol{W}]\}^T\right) d\mathbf{y}_1 ... d\mathbf{y}_N. \tag{3.30}$$

This integral can be easily evaluated to give,

$$Z_D(\Sigma) = (2\pi)^{\frac{N \times m}{2}} \mid \Sigma \mid^{-\frac{N}{2}}. \tag{3.31}$$

In real world modelling problems and control applications, adaptation to unexpected changes is essential, so it is important to obtain a fast estimate of the neural network output and error bars. Bayesian and Gaussian processes cannot be used in on-line learning methods. Also in its conventional form Bayesian methods do not include all possible sources of uncertainty. On the other hand, although not explicitly, predictive error bars include all possible sources of variation in the predicted output, whether it is due to noise added to the output, noise added to the input, or even due to weight uncertainty. Moreover, predictive error bars have been found to be more efficient in terms of computational power, and they can be used in case of on-line learning. Based on this, predictive error bar estimation will be used in this work.

Chapter 4

Incorporating Uncertainty for Stochastic SISO Nonlinear Dynamical System

In modern control theory nonlinear stochastic optimal control is of great importance, due to its immediate application to real world problems, where disturbances play an important part in the performance of control processes. In such situations, 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 which we need to try and maintain possible solution trajectories. The method of approximation we choose is to try and model the conditional uncertainty of the control signal by modelling its statistical properties, expressed in terms of a conditional distribution function, which would be generated by the real stochastic system under ideal circumstances. Since we are interested in nonlinear stochastic systems, a nonlinear controller is required to satisfactorily control such plants. The nonlinear controller can be viewed as a nonlinear approximation problem.

Recently, 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. Besides having nonlinear features, dynamic systems may have noise events affecting their inputs and outputs, and usually they are time-variant. Because artificial neural networks can be adapted on line [125; 95], usually they are capable of good performance 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.

Many researchers have considered the use of the uncertainty measure to build a more robust controller. Some of the recent works on the use of uncertainty have been discussed in Section 1.1.

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 its own predictions as discussed in Chapter 3. None of the recent works have considered the possibility of using the neural network's own estimate for error bars. In this chapter we address the use of this extra knowledge to approximate the conditional distribution of the control signal for stochastic systems of the form

$$y(k+d) = f(y(k), y(k-1), ..., y(k-q+1), u(k), ..., u(k-p+1), \bar{v}(k)), \tag{4.1}$$

where $y(k)$ is the measured plant output, $u(k)$ is the measured plant input, $\bar{v}(k)$ is the noise affecting the plant output, q is the maximum delay of the output, p is the maximum delay of the input, d is

the relative degree of the plant.

After modelling the conditional distribution of the control signal we search for the optimal control law from the estimated control signal distribution, almost in the same way as dynamic programming, where the optimal control law is chosen to minimise a utility function

$$u_{opt} = \arg \min_{u \in U} \underset{\tilde{v}}{E} J(M), \tag{4.2}$$

but with the advantage of avoiding the computation requirements for the dynamic programming.

The chapter starts by discussing dynamic programming in section 4.1. The adaptive inverse control problem together with a literature survey on the use of this method and its disadvantages is presented in section 4.2. The various sources of uncertainty in the control field together with a general discussion about the best way of estimating uncertainty are provided in section 4.3. Section 4.4 deals with distribution modelling of the inverse model which in turns provides an estimate about the uncertainty in the controller. The problem formulation for incorporating uncertainty of the inverse model to improve the controller performance in addition to the developed algorithm for incorporating uncertainty and the advantages of this algorithm is presented in Section 4.5. Section 4.6 discusses the stability analysis for the updating rule in the proposed control algorithm. A SISO stochastic dynamical system is used in section 4.7 to verify the theoretical development for the proposed control algorithm. Section 4.8 provides some analysis on the expected distribution in the output space results from applying different control values to the forward model of the system. Section 4.9 discusses alternative decision criteria. The chapter discussion is give in section 4.10.

4.1 Dynamic programming

In control theory, dynamic programming provides the best approach for solving highly nonlinear systems, stochastic control problems and constrained nonlinear systems. The computational procedure of dynamic programming, which is based on the direct search method, guarantees the achievement of the absolute minimum cost, avoids the troublesome questions of existence and uniqueness, and can handle constraints.

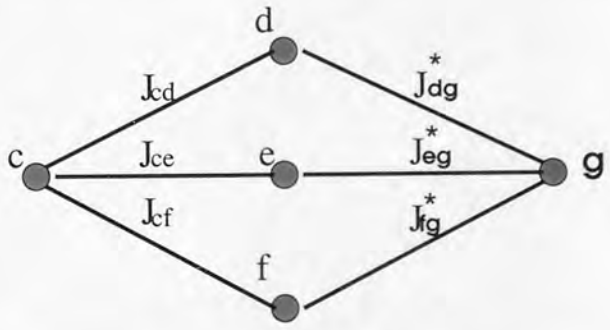


Figure 4.1: Optimal paths resulting from all admissible decisions at state c.

The computational procedure of dynamic programming employs Bellman's principle of optimality to search for a sequences of decisions which define an optimal control strategy. For the process shown in Figure 4.1, if the current state is given by c and the goal state is given by g, then there are several optimal paths to the goal state g, which may result from all admissible decisions at c [124]. According to the principle of optimality if the segment $c-d$ is the initial segment of the optimal path from $c-g$,

then $d-g$ is the terminal segment of the optimal path. Applying the same principle to paths $c-e$ and $c-f$, yields that the decisions of paths $c-d$, $c-e$, and $c-f$ are the only admissible decisions for achieving the optimal trajectory from c to g. The optimal trajectory from state c to g can then be obtained by comparing the minimum cost of all the admissible decisions:

$$\begin{aligned} J^*_{c,d,g} &= J_{c,d} + J^*_{d,g}, \\ J^*_{c,e,g} &= J_{c,e} + J^*_{e,g}, \\ J^*_{c,f,g} &= J_{c,f} + J^*_{f,g}, \end{aligned} \tag{4.3}$$

where $J^*_{c,d,g}$ is the minimum cost to go from the current state c to the goal state g via intermediate state d.

For multistage control processes this can be written in the following general form

$$J^*_{k,k+1,N} = J_{k,k+1} + J^*_{k+1,N}. \tag{4.4}$$

Consider the optimal control problem where finding the optimal control strategy that minimises a prespecified performance measure is the main objective. Minimising a performance measure requires finding a minimum value of a functional, which is a problem in the calculus of variations.

The optimal control problem then reduces to that of choosing $u(k)$ so as to minimise the time integral of a measure of instantaneous cost

$$\begin{aligned} J &= \int_0^T l(\mathbf{x}(t), \mathbf{u}(t), t)dt, \\ &= \sum_{k=0}^{K} l(\mathbf{x}(k), \mathbf{u}(k), k), \end{aligned} \tag{4.5}$$

where $\mathbf{x}(k)$ is the state at time k, $\mathbf{u}(k)$ is the control at time k, and l is the instantaneous cost which is a function of the state and control variables, and where $\mathbf{u}(k)$ and $\mathbf{x}(k)$ are connected by the following difference equation

$$\mathbf{x}(k+1) = g(x(k), u(k), k). \tag{4.6}$$

Applying the principle of optimality to this problem provides an iterative procedure for determining the optimal control [70; 8]. This iterative procedure is given by the following recurrence relation

$$J(k) = \underset{\mathbf{u}\in U}{\text{Min}}\Big(l[\mathbf{x}, \mathbf{u}, k] + J[g(\mathbf{x}, \mathbf{u}, k), k+1] \Big). \tag{4.7}$$

Full derivation of Eq. (4.7) is given in Appendix D. Equation (4.7) describes an iterative relation for determining $J(k)$ from knowledge of $J(k+1)$. The optimal control is then defined as

$$\mathbf{u}^*(k) = \arg \underset{\mathbf{u}}{\text{Min}}\Big(l[\mathbf{x}, \mathbf{u}, k] + J[g(\mathbf{x}, \mathbf{u}, k), k+1] \Big). \tag{4.8}$$

The range of the state variables and the admissible controls can vary with time. For stochastic control problems where the state is driven by a random forcing vector $\bar{\nu}$, $\mathbf{x}(k+1) = g(\mathbf{x}(k), \mathbf{u}(k), \bar{\nu})$, the functional equation can be determined by minimising the expected value over the random forcing vector $\bar{\nu}$

$$J(k) = \underset{\mathbf{u}\in U}{\text{Min}}\Big(\underset{\bar{\nu}}{E}\Big\{ l[\mathbf{x}, \mathbf{u}, k] + J[g(\mathbf{x}, \mathbf{u}, k), k+1] \Big\} \Big). \tag{4.9}$$

Dynamic programming is the only exact and efficient method for calculating the optimal control law in noisy and nonlinear environments. However, the cost of finding an optimal control law using

dynamic programming is proportional to the number of possible state in the plant, which in turn grows exponentially with the number of variables in the plant. Moreover, dynamic programming performs poorly when the order of the system increases. In addition to the computational complexities involved in the dynamic programming methods, dynamic programming assumes the presence of an accurate forward model for the plant to search ideally for the optimal control law.

The purpose of the techniques described in the next sections is to find an approximate solution for the dynamic programming problem. Basically, the optimal control law is derived by modelling the conditional distribution of the controller (the inverse model of the plant) which provides information about the inherent uncertainty of the controller. The conditional distribution of the controller is then used to search for a better control law provided that a representative forward model of the plant can be obtained.

4.2 Adaptive inverse control

The classical inverse adaptive control technique is shown in Fig. 4.2. The neural network is learning to recreate the input that created the desired output of the plant. So in this case the desired response for the adaptive controller is the plant input $u(k)$. The inputs to the inverse model can be put into three different groups: (1) past process inputs $[u(k-1), ..., u(k-p+1)]$ (2) past and present process outputs $[y(k), ..., y(k-q+1)]$ (3) desired process output $y(k+d)$. More specifically the inverse model is given by the following functional relationship [51; 106; 2; 7]

$$\hat{u}(k) = N_f^{-1}(y(k+d), y(k), ..., y(k-q+1), u(k-1), ..., u(k-p+1)). \tag{4.10}$$

The implementation of the neural inverse model in process control has been considered by other authors. In [10] the use of the inverse model in process control as part of the internal model control, IMC is proposed. In this control scheme both the neural forward model and the inverse models are employed. The use of the content addressable memories, CAM, as an inverse dynamic model is proposed in [4]. In this case the learning can be done in one trial and the error signal need not be calculated. Albus [2] has proposed a unique control scheme called the cerebellar model articulation controller, (CMAC). This control scheme is basically a table look up technique for representing complex, nonlinear functions $f(x)$, where x is the discrete input state vector of given dimension. Each point in the input space X maps to C locations in a memory having the same dimension as the state vector. The value of the function $f(x)$ is then determined by summing the values at each of the corresponding locations in the memory. This scheme has been considered further in [83].

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, although other error functions can also be used). The error is defined as the difference between the input of the plant $u(k)$, and the actual output of the controller $\hat{u}(k)$. Many such algorithms are described in the reports and textbooks by Narendra and Parthasarathy [95] and by White and Sofge [125]. In this work the scaled conjugate gradient method is used. For details of this method see [13].

Using the input of the system $u(k)$ as a training signal in the inverse controller will force the network of the inverse model to represent the inverse of the plant. However this approach has several drawbacks:

- The learning procedure is not goal directed, since in direct inverse control we are trying to minimise the error between the plant input $u(k)$ and the network output $\hat{u}(k)$, $e = |u(k) - \hat{u}(k)|$, 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 generalise and interpolate in regions where it requires to find the control signal 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 actual operational range of the system.

- Obtaining the inverse of the system may not be possible in problems where the mapping is not one-one. This problem will be considered further in Chapter 6.

To overcome these problems, Psaltis [106] has suggested the use of a specialised learning architecture. In this approach both forward and inverse models 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 minimise the error between the system response and the desired 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. Using this control architecture is supposed to overcome both of the above problems in the direct inverse control. Other control architectures have also been proposed to overcome these problems, see Chapter 1.

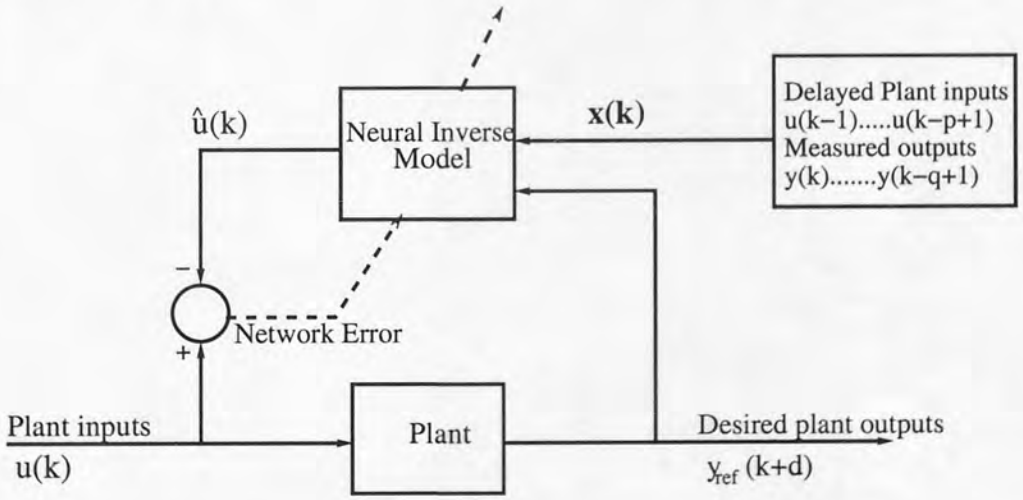


Figure 4.2: Training of an inverse controller.

4.3 Uncertainty

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

$$a(k+1) = g(a(k), b(k)). \tag{4.11}$$

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. Opposite to this is the situation where the choice of certain input values b may lead to a non-unique transformation. This means that the choice of one input value b may lead to a set of possible output values a. Taking the value $b(3)$ as an input to the model at time instant 3, the transformation will yield the output $a(4_1)$, sometimes $a(4_2)$ and so on. Both of the above situations are shown in Figure 4.3.

This can be due to different sources of variations.

- It 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 optimise the model which provides the prediction output.
- Since in most of the cases the description model is a parameterised function of the input, the parameters of the model itself can be uncertain.

To illustrate these points, 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 analyse this problem in some detail given the angle of pole, the angular velocity of pole, the horizontal position of the cart's 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 estimate.

Even if all this is achieved and providing that a suitable mathematical model could be estimated, an exact prediction can never be obtained. The first thing that will be observed is that small changes in the initial conditions, will result in a large change in the predicted force value. This means that an unstable process is under consideration.

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 a better prediction for the desired output is required, 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 modelling the probability distribution for the desired prediction.

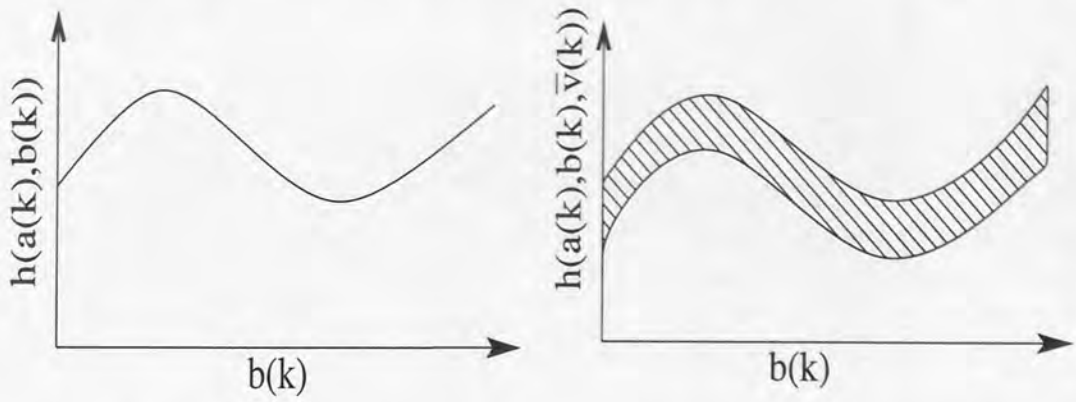


Figure 4.3: Stochastic versus deterministic transformation: (a) The transformation relationship between the input and the output can be represented by a deterministic function h. (b) The transformation relationship between the input and the output is affected by a random forcing vector $\bar{v}(k)$

4.4 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 $\mathbf{x}(k) = [y(k), ..., y(k-q+1), u(k-1), ..., u(k-p+1)]$ and the desired output value $y_{ref}(k+d)$ as an input, and outputs the control signals $u(k)$ (assuming d relative degree), which will move the plant output to the desired value. In this work the aim is to model the statistical properties of the control signals, $u(k)$, expressed in terms of the conditional distribution function $p(u(k)|\mathbf{s}(k))$. Here $\mathbf{s}(k) = [y_{ref}(k+d), \mathbf{x}(k)]$ is the input vector to the neural inverse model.

For dynamical systems it is reasonable to assume that the output of the system $y(k+d)$ is a function f of its input $u(k)$ and the delayed vector $\mathbf{x}(k)$. 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 defined. In this case a feed-forward neural network trained using the sum of the square error function (between the input of the system $u(k)$ and the actual output of the controller $\hat{u}(k)$) 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), 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 using the sum of the square error function between the input of the system and the actual output of the controller can not be used to tell us how to choose the control signal $u(k)$ so as to reach the desired response $y_{ref}(k+d)$. 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, and will be discussed in Chapter 6. In the following, only Gaussian functions for control signals will be discussed.

4.4.1 Gaussian distribution of control signals

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 Chapter 3. In this work the predictive error bar

method will be used.

In Section 2.4 it has been shown that the optimum network output approximates the conditional mean of the target data. In the inverse model this means $f_{opt}^{-1}(\mathbf{s}(k)) =< u(k)|\mathbf{s}(k) >$. Therefore the local variance of the control signal can be estimated as $\|u(k) - f_{opt}^{-1}(\mathbf{s}(k))\|^2$.

For implementation of predictive error bars, we follow the method presented in [76], where two neural networks are required. See Section 3.5 for details, here is just a brief discussion. For the RBF networks, both networks share the same inputs and hidden layer parameters. Only the final layer parameters need to be optimised given different target values in both of the networks. The first network is firstly optimised to produce the control signals. Once the parameters of the first network are optimised, the residual error can be calculated. The parameters in the second network are then optimised to predict the residual errors $\|u(k) - f_{opt}^{-1}(\mathbf{s}(k))\|^2$.

The same method can be used for the multi-layer perceptron network. Here two different networks with different parameter values but with the same input are used. The parameters of the first network are optimised such as to produce the control signal. The parameters of the second network however, is optimised such as to produce the calculated variance which can be obtained after optimising the first network.

4.5 Problem formulation and solution development

Dynamic programming is a powerful tool in stochastic control problems [86; 69]. 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, see [41].

Before the proposed method is discussed, the objective of the control problems we are interested in is firstly formulated. In this work the optimal control law is obtained by minimising the following cost function

$$e = \| u(k) - \hat{u}(k) \|^2, \tag{4.12}$$

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 utilising properties of the least square error function. Using the predictive error bar method, the probability distribution of the control signal is estimated. It is assumed to be Gaussian given by

$$p(u(k) \mid y(k+d), \mathbf{x}(k)) = \frac{1}{(2\pi\sigma^2(k))^{\frac{1}{2}}} \exp\left(- \frac{(u(k) - \hat{u}(k))^2}{2\sigma^2(k)} \right), \tag{4.13}$$

where $\hat{u}(k)$ is the mean value conditioned on the input, and $\sigma^2(k)$ is the variance conditioned on the input as well.

Modelling the probability distribution of the inverse model provides information about the uncertainty in the predicted output of the inverse model. 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 probability of the deviation between the process output $y(k+d)$ and the desired output $y_{ref}(k+d)$ greater than zero equal to zero

$$\text{prob}[|y(k+d) - y_{ref}(k+d)| > 0] = 0. \tag{4.14}$$

Equation. (4.14) defines the objective functional. However, 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 predicted output of the inverse model can never be exact and taking knowledge of uncertainty around its prediction, we can then try and maintain the condition given in Eq. (4.14).

Since uncertainty of the predicted output of the inverse model is modelled by estimating the probability distribution of the inverse model, 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 the control signal u and using the function value alone to determine a reasonable minimisation 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. To suit the development of the minimisation problem, the objective functional of Eq. (4.14) is rewritten in the following form

$$\| y(k+d) - y_{ref}(k+d) \|^2 = 0. \tag{4.15}$$

Since the inverse controller has been optimised over the entire range of possible output values the functional given in Eq. (4.15) could be maintained at each instant of time without carrying the integral over time as in Eq. (4.5) for the dynamic programming case.

Therefore an optimal control law so as to achieve the condition of Eq. (4.15) is required. Maintaining this condition will be subjected to the possible control values resulting from the distribution of the inverse model for a specific input value

$$J(k) = \underset{u \in U}{Min}[(y(k+d) - y_{ref}(k+d))^2], \tag{4.16}$$

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

Equation (4.16) 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

$$y(k+d) = g(u(k), \mathbf{x}(k)) = \hat{g}(u(k), \mathbf{x}(k), W) + \xi(u(k), \mathbf{x}(k), \tilde{W}), \tag{4.17}$$

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

Ignoring the uncertainty, $\xi(u(k), \mathbf{x}(k), \tilde{W})$, in the forward model and replacing the actual output of the system by the predicted mean value of the forward model yields,

$$J(k) = \underset{u \in U}{Min}[(\hat{g}(u(k), \mathbf{x}(k), W) - y_{ref}(k+d))^2]. \tag{4.18}$$

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

$$y(k+d) = g(\mathbf{x}(k), u(k), \bar{v}(k)). \tag{4.19}$$

If this is the case then the criterion function given by Eq. (4.18) needs to be modified. The following new criterion function can now be introduced

$$J(k) = \underset{u \in U}{Min}\left(\underset{\bar{v}}{E}[(\hat{g}(u(k), \mathbf{x}(k), \bar{v}(k)) - y_{ref}(k+d))^2] \right). \tag{4.20}$$

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

$$\begin{aligned} J(k) &= \underset{u\in U}{\text{Min}}\left(\underset{\bar{v}}{E}[(\hat{g}(u(k),\mathbf{x}(k),\bar{v}(k)) - y_{ref}(k+d))^2]\right), \\ &= \underset{u\in U}{\text{Min}}\left(\int [(\hat{g}(u(k),\mathbf{x}(k),\bar{v}(k)) - y_{ref}(k+d))^2]p(\bar{v})d\bar{v}\right). \end{aligned} \tag{4.21}$$

An approximation method can be used to evaluate the quantity inside the brackets in Eq. (4.21). Since the true distribution of the stochastic component is assumed to be known, the integral can be approximated by the following finite sum

$$\int [(\hat{g}(u(k),\mathbf{x}(k),\bar{v}(k)) - y_{ref}(k+d))^2]p(\bar{v})d\bar{v} = \frac{1}{L}\sum_{i=1}^{L}[(\hat{g}(u(k),\mathbf{x}(k),\bar{v}_i(k)) - y_{ref}(k+d))^2], \tag{4.22}$$

where $\bar{v}_i(k)$ represents a sample from $p(\bar{v})$, and hence Montecarlo sampling method. Equation. (4.22) implies that if the probability distribution for the output of the forward model is given, the expected value of the criterion function could be minimised.

4.5.1 Neural network development for incorporating uncertainty

Once properly trained, the inverse model can be used to control the plant since it can create the necessary control signals to cause the desired system output. Despite the fact that neural networks have been accepted as a suitable model 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 by using the predicted neural network error bars to limit the possible control solutions to consider. Accepting the inaccuracy of neural networks and assuming the output of the inverse control network can be approximated by a Gaussian distribution of control signals, the mean and variances can be obtained as discussed previously. Using just the mean estimate of the control is typically suboptimal in nonlinear systems. Even though the Gaussian assumption used here is an approximation, using the on-line variance estimate of the neural network determines a region around the predicted mean value where sampling can be used to obtain a better estimate of the control signal than the mean. The distribution assumption is Gaussian but the predicted mean and variance are nonlinear functions of previous states, thus allowing for good models of forward and inverse plant behaviour provided the inverse plant is a function. If this is not the case a similar approach, but using Gaussian mixtures (a mixture density network) could be employed. The use of the mixture density network to model general conditional distributions will be discussed in Chapter 6.

Based on estimates of the mean and variance of the distribution of control signal values, we can construct the following algorithm incorporating the uncertainty directly. The block diagram of this algorithm is shown in figure 4.4.

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:

$$\hat{y}(k+d) = N_f(y(k), y(k-1), ..., y(k-q+1), u(k), ..., u(k-p+1)). \tag{4.23}$$

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

$$\mathbf{h}(k) = N_f^{-1}(y(k+d), y(k), y(k-1), ..., y(k-q+1), u(k-1), ..., u(k-p+1)), \tag{4.24}$$

$$\hat{u}(k) = \mathbf{h}(k)\boldsymbol{W}_1, \tag{4.25}$$

$$var_{u(k)} = \mathbf{h}(k)\boldsymbol{W}_2, \tag{4.26}$$

where $\mathbf{h}(k)$ is the predicted hidden variable from the neural network at each instant of time k. $\boldsymbol{W}_1$ is the weight of the linear layer estimated to predict the conditioned mean of the control signal. $\boldsymbol{W}_2$ is the weight of the linear layer estimated to predict the variance of the predicted control signal.

3. Define the desired response of the plant by defining a suitable reference model, which should be chosen to have the same relative degree as that of the plant.

4. At each instant of time k the desired output is calculated from the reference model output.

5. Bring the control network on-line and at each time k 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

$$p(u(k) \mid y_{ref}(k+d), \mathbf{x}(k)) = \frac{1}{(2\pi\sigma^2(k))^{\frac{1}{2}}} \exp\left(-\frac{(u(k)-\hat{u}(k))^2}{2\sigma^2(k)}\right), \tag{4.27}$$

where σ^2 is the variance of the control signal $var_{u(k)}$, and $\mathbf{x}(k) = [y(k), y(k-1), ..., y(k-q+1), u(k-1), ..., u(k-p+1)]$.

6. Generate a vector of samples from the control signal distribution. 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, number of samples = $K \times var_{u(k)}$. 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.

7. Feed these samples to the system model and calculate the output from each sample.

8. 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.

$$J(k) = \underset{u \in U}{\text{Min}} E_{\bar{v}}[(\hat{y}(k+d) - y_{ref}(k+d))^2], \tag{4.28}$$

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 $\bar{v}$. Since a neural network is used in this work 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.

9. Go to step 4.

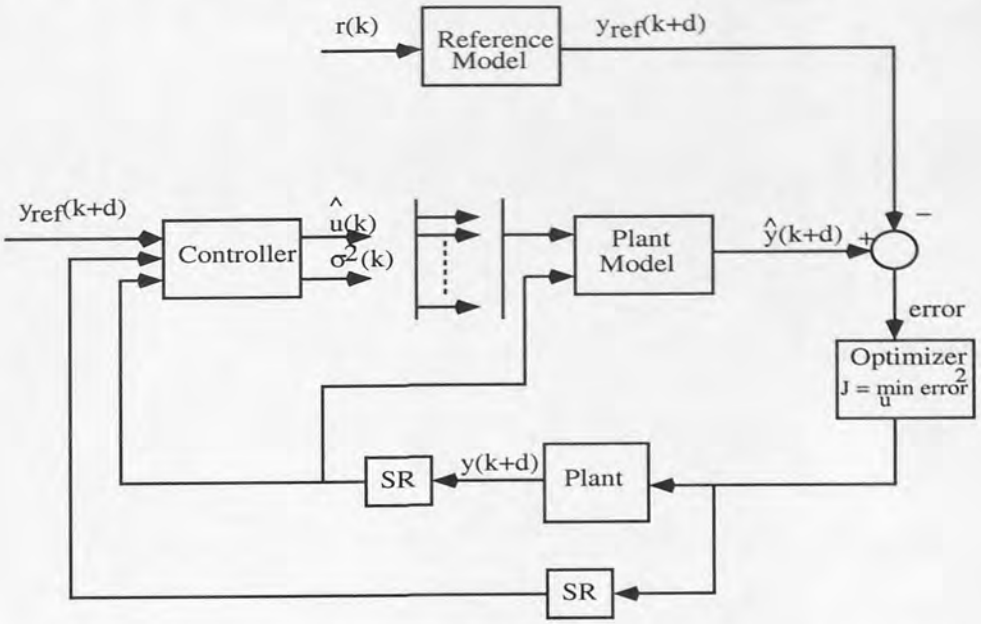


Figure 4.4: The block diagram of the proposed optimisation method. The input and the output of the plant are passed through a shift register (SR) so as to generate the required past input and output values.

4.5.2 Advantages of proposed method

The classical inverse controller is trained so as to minimise the error between the actual input of the system and the inverse model output $e = \| u(k) - \hat{u}(k) \|^2$. In this architecture the inverse model is trained on the data that is coming from the plant. After training the inverse model it should be able to take the desired response as an input and produce the appropriate control value so as to make the output of the plant $y(k+d)$ approach the desired output $y_{ref}(k+d)$. However this will work only if the desired output happens to be very close to one of the system outputs that have been used in the training stage.

Therefore, although the training of the inverse model in the direct inverse approach is not goal directed, an estimate for the probability distribution of the inverse of the plant could be made. This means that the distribution of the inverse model $p(u(k))$ tells us about a generic unconditional range of possible control values, irrespective of what we want to use it for. This can be compared to the hidden layer of the RBF network. It is only when the final layer is added that the RBF can be specialised to classification, regression or other objectives.

The probability distribution of the control signal which characterises the inverse model of the plant can then be optimised using a utility function which suites the objective of the problem. As an example consider the problem of tracking a given desired trajectory. In this case and according to the proposed method, sampling from the distribution of the control signal can be assumed so as to minimise the error between the system output and the desired trajectory $J(k) = \underset{u \in U}{Min}[(y(k+d) - y_{ref}(k+d))^2]$, with U being the set of samples from the distribution of the inverse model. The same distribution can also be used to find the shortest distance path through a maze or finding the most likely path to arrive, but with different utilities.

This is one of the main advantages of the proposed approach where the distribution of control signal $p(u(k))$ which characterises the inverse model of the plant is approximated using a neural

network, but this is not optimised using knowledge of how the plant output should be used or what is the operational range of the plant output (since this information is not known a priori).

For situations where the inverse model could be accurate enough to predict output values over the entire operational range, irrespective of what is the objective of the control problem, the proposed sampling approach would have no advantage over the direct inverse approach. This could be easily verified by looking at the values of the predicted uncertainty of the inverse model. If the uncertainty around the predicted value of the inverse model is small, and if the forward model is insensitive to small variations in the input signal then this means that the optimal control value is close to the mean value and little benefit would be obtained from sampling. On the other hand, if the predicted value of the uncertainty around the predicted mean value of the inverse model is large, then a better control value than the mean could be obtained. More details are given in Table 4.1.

Taking samples from the distribution of the inverse model and searching for the value that minimises a utility function rather than using gradient information of a utility function guarantees obtaining the absolute minimum of utility rather than a relative minimum. This has been addressed in Bellman's book [9]. Restricting the allowable control values at each stage to come from the distribution of the inverse model simplifies the search process and reduces the computing effort.

Finally we come to the point where the forward model is considered to be driven by a random forcing component. In these situations the integral of the utility function over the random variable needs to be carried out. This is not easy to be done analytically. Searching for the optimal control law allows us to approximate this integral by the finite sum as given in Eq. (4.22).

Sensitivity of forward model	Variance of inverse model	Sampling benefit
sensitive	large	√
sensitive	small	√
insensitive	large	×
insensitive	small	×

Table 4.1: Sampling benefit compared to the variance of the inverse model and the sensitivity of the forward model.

4.6 Stability analysis

The problem of dynamical control systems stability is one of the important subjects in real world systems. As an example, when a controller is designed for chemical control processes it should be shown first that this controller cannot destroy the plant and the controller should provide a proof of stability. Lyapunov methods played an important part in proving the stability of dynamical control systems; however in control theory the problem of stability is the problem of whole system stability made up of the unknown system, the controller and the updating rule for both of the controller and the system in certain situations.

According to the Lyapunov method a dynamical control system with the state vector x around a stationary point is said to be stable if one could find a positive definite continuous function of x, $V(x)$ called a Lyapunov function such that its time derivative $\partial V(x)/\partial t$ along a system trajectory is non positive definite [3].

Although the Lyapunov method is shown to be applicable to nonlinear dynamical systems, obtaining successful results, however, may not be an easy task [99].

In control theory partial or complete linearisation for the dynamical systems has been applied to prove stability of nonlinear systems [71]. This means that one can proceed easily by applying the theory of stability analysis of linear systems. However, it has been shown that although linear approximations could be used to prove stability around stationary operating points, nonlinearities should be examined to provide proof for asymptotic stability [110].

One of the successful fields for the theory of linear approximation is the feedback linearisation control methods. In this control area proof of stability has been provided for both the classical feedback linearisation control theory and the neural network. Here, a linear approximation of the system is firstly obtained. The problem of adaptive control is then to find a smooth state feedback control law defined locally around a stationary point, with the property that the corresponding closed loop system is asymptotically stable at that point [52].

Another important subject for the stability of nonlinear control systems is the adaptation procedure. This can be proved by showing the convergence of either:(1) the parameters of the nonlinear adaptive filter (2) the output error. For the objective of this work, the convergence of the output error is more important, which will be shown in the next section.

4.6.1 Error convergence

The development of the previous section was based on updating the control signal from its estimated distribution. Here the Lyapunov method is used to determine stability and convergence of the output error for the proposed updating rule of the control signal [40]. The Lyapunov function V_j is defined as

$$V_j = e_j^2. \tag{4.29}$$

If V_j can be shown to be positive and decreasing when updating the control signal from its distribution, then it can be concluded that the output error converges and the updating process is stable. By definition, V_j is positive.

Therefore, to prove error convergence for the proposed sampling method it is required to find whether V_j is decreasing or not. The variation of the Lyapunov function from iteration j to iteration $j+1$ is:

$$\Delta V_j = V_{j+1} - V_j = [e_{j+1}^2 - e_j^2]. \tag{4.30}$$

Assuming that the input, $\mathbf{x}$ and the desired output, y_{ref} remain the same from iteration j to iteration $j+1$, then

$$e_{j+1} = f(u_{new}, \mathbf{x}_j) - y_{ref_j}, \tag{4.31}$$

$$e_j = f(u_{old}, \mathbf{x}_j) - y_{ref_j}, \tag{4.32}$$

where u_{old} is initially set to be equal to the predicted mean value of the control signal $\hat{u}$, $u_{new} = U(j)$, and where $U = [u_1, u_2, ..., u_N]$ are the admissible control signal values sampled from the control signal distribution where N equals the number of the generated samples.

In order to find the minimum of the performance measure criterion using the value of the performance criterion function only, the updating rule is:

$$\begin{aligned} &\text{If} && e_{j+1}^2 \leq e_j^2 && u_{old} = u_{new}, \\ &\text{Otherwise} && && u_{old} = u_{old}. \end{aligned} \tag{4.33}$$

Since $e_{j+1} \leq e_j$, ΔV_j is always negative and consequently the error is decreased by updating the control signal in this way.

4.7 Single input single output stochastic process

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

$$\begin{aligned} y(k) &= 0.9722y(k-1) + 0.3578u(k-1) - 0.1295u(k-2) - 0.3103y(k-1)u(k-1) \\ &- 0.04228y^2(k-2) + 0.1663y(k-2)u(k-2) - 0.3513y^2(k-1)u(k-2) \\ &+ 0.3084y(k-1)y(k-2)u(k-2) + 0.1087y(k-2)u(k-1)u(k-2) \\ &- \bar{\nu}y^2(k-1)y(k-2). \end{aligned} \tag{4.34}$$

This model has been used in [1] to illustrate theoretical developments for the direct adaptive controller. Because disturbances play an important part in real world processes, a stochastic component, $\bar{\nu}$, is added to this model. This component is considered to be a Gaussian random variable with a mean of 0.03259 and a 0.2 variance. The plant has been considered to be described by Eq. (4.34). 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:

$$\hat{y}(k) = N_f(y(k-1), y(k-2), u(k-1), u(k-2)),$$

where N_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 (4.34) 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, a cross validation method needs to be implemented to find the optimal order of the model.

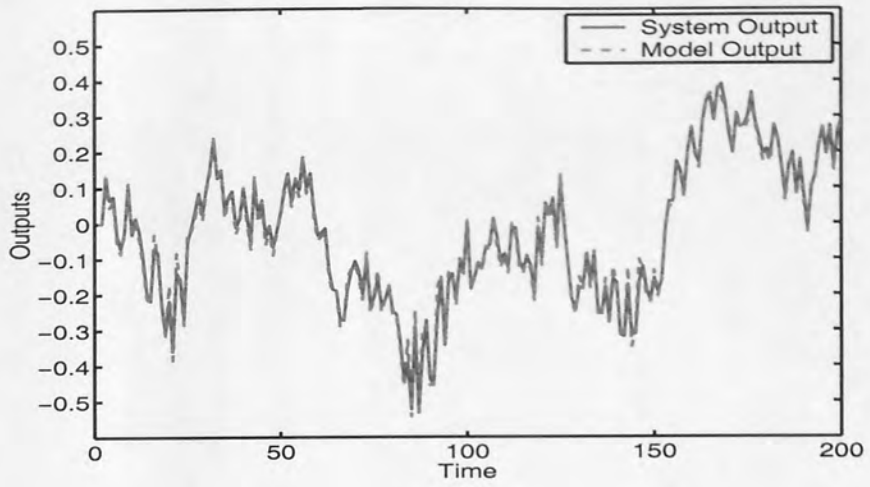
Similarly an input-output model described by the following second order equation was chosen to find the inverse model of the plant

$$\hat{u}(k-1) = N_f^{-1}(y(k-1), y(k-2), y(k), u(k-2)),$$

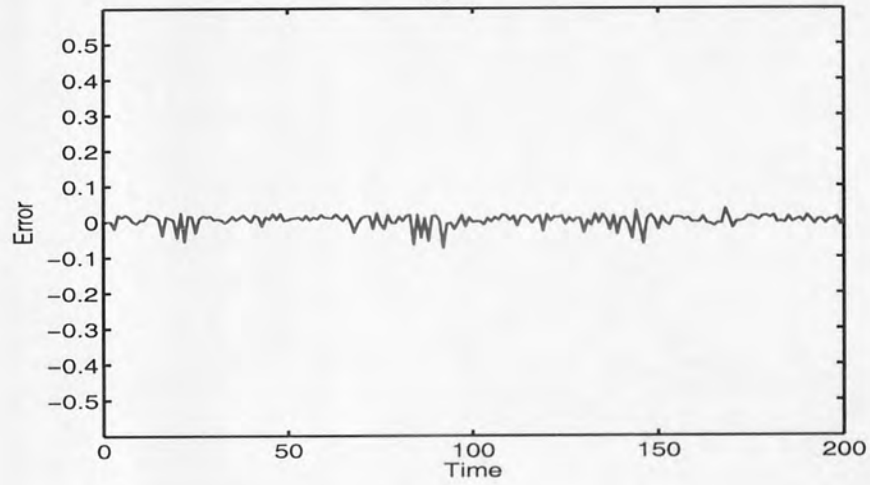
where N_f^{-1} is a Gaussian radial basis function network. The training data has been the same as in the forward model. A neural network with 6 Gaussian basis functions was found to be the best model.

4.7.1 Identification

After training the forward model off-line, a random input signal generated uniformly in the interval $[-0.4, 0.4]$ is used to test the model. The result is shown in Fig. 4.5. Very good approximation is achieved. The model could capture all of the high frequency peaks of the plant.



(a)



(b)

Figure 4.5: Actual and model outputs for the SISO stochastic process: (a) The actual and model output values. (b) The identification error for the SISO stochastic process.

4.7.2 Direct adaptive inverse control

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 $y(k)$ at time k equal to the desired value $y_{ref}(k)$

$$u(k-1) = f^{-1}(y(k-1), y(k-2), y_{ref}(k), u(k-2)),$$

where $y_{ref}(k) = 0.2 * r(k-1) + 0.8 * y_{ref}(k-1)$ and r is the set point. The predicted mean value from the neural network was forwarded to the plant. After running the process for 600 time steps the output of the system becomes unstable, and the classical inverse controller was unable to force the plant output to follow the reference output, as shown in Fig. 4.6.

4.7.3 Proposed sampling control method

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

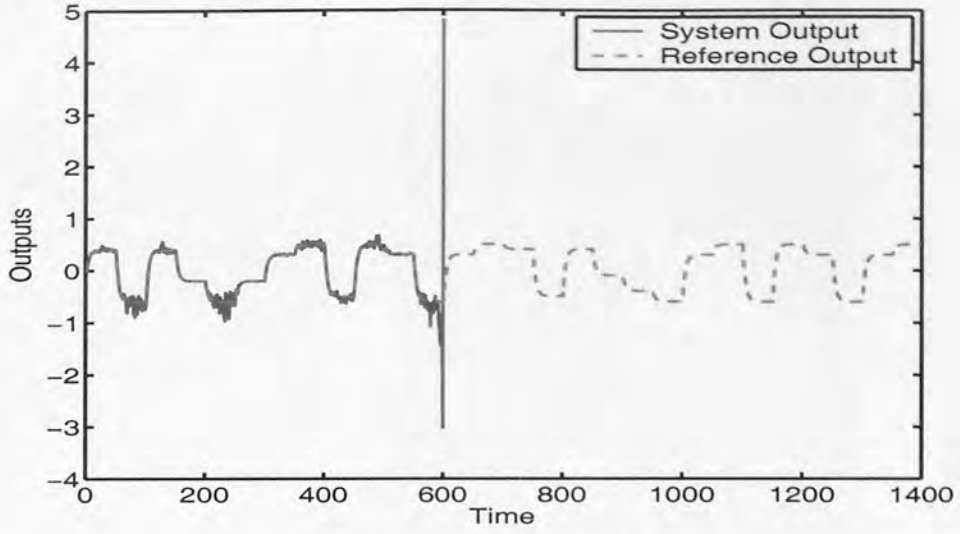


Figure 4.6: The desired and actual output values of direct inverse control.

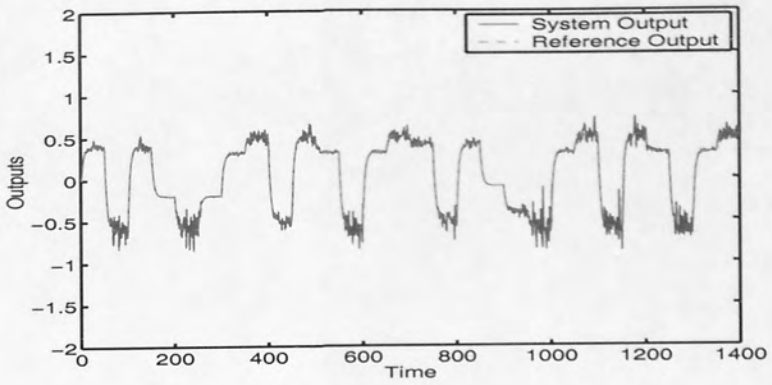
control signal was obtained from a small number of samples from the Gaussian distribution, typically a maximum of 27 samples. The overall performance of the plant under the proposed method is shown in Fig. 4.7, where it is evident that the system outputs remain stable across the whole region, and that the proposed sampling approach managed to stabilise the plant. The control signal is shown in Fig. 4.7.b, and the variance of this control law is shown in Fig. 4.7.c. 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 Fig. 4.7.d. More specifically Fig. 4.7.d is the plot of error $= |y - y_{ref}|_{\text{sampling}} - |y - y_{ref}|_{\text{classical inverse}}$ against the 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.

4.8 Output space

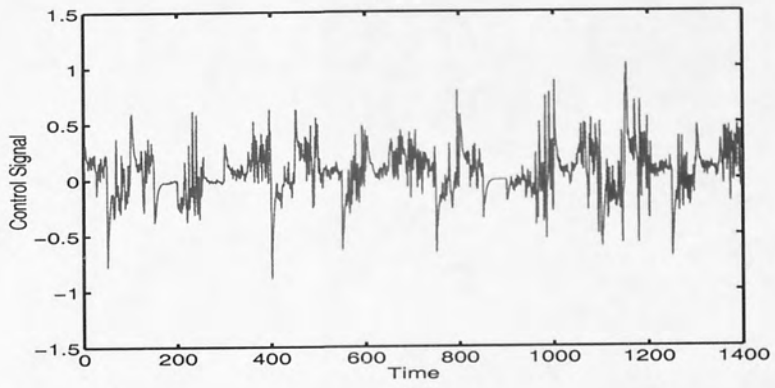
In this section some analysis regarding the distribution of the output of the model which could be evaluated at different samples of the control signals are provided. Since a variable number of samples from the control signal distribution may not provide enough samples to plot the histogram in the output space, a fixed number of samples is used. In this experiment 20000 samples are generated from the control signal distribution. Fig 4.8.a shows the histogram of the model output at time instant 7. At this instant of time the desired value of the output determined from the reference model is 0.2689. In addition to the histogram this figure shows the value of the desired output plotted as a dashed line, and the value of the model output obtained from the mean value of control signal plotted as a solid line. It is clear from this figure that the desired output value is very near to the output value obtained from the mean of control signal. In addition, a better control value could be obtained to make the probability of the deviation of model output and desired output greater than zero, equal to zero

$$\text{prob}[|\hat{y}(k) - y_{ref}(k)| > 0] = 0.$$

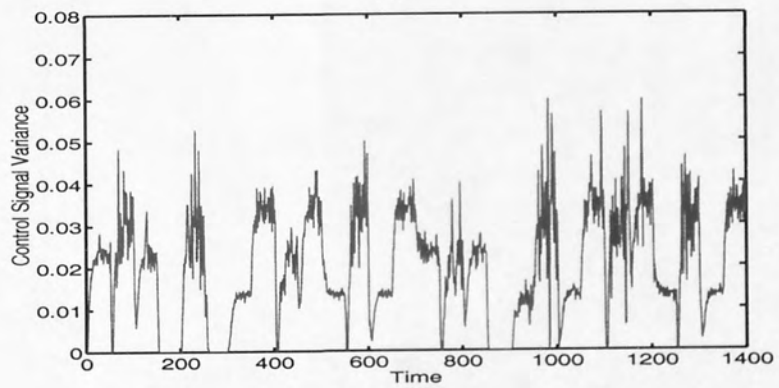
The estimated probability density function of the control signal depends on the previous input



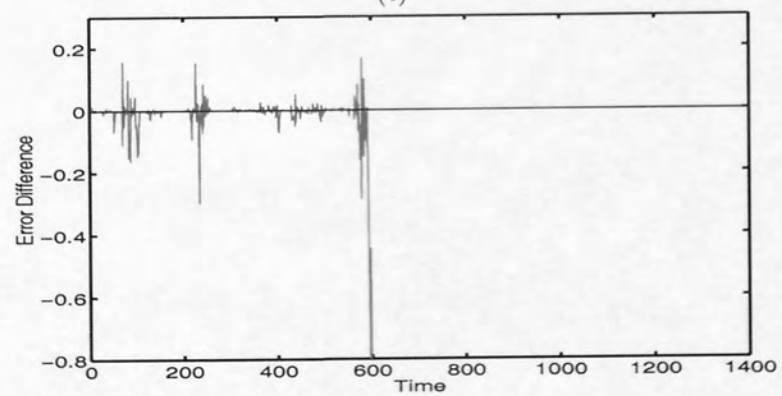
(a)



(b)



(c)



(d)

Figure 4.7: Stochastic SISO Control result: (a) The desired and actual output values. (b) The Control Signal. (c) The Control Signal Variance. (d) The Error Difference

and output values in addition to the desired output value, $p(u(k-1) \mid y_{ref}(k), \mathbf{x}(k))$. Searching for the optimal control value which minimises an expected value of the cost function is confined for that estimated density function.

Since the model has been shown to be a good approximator for the actual plant, better control has been obtained. Fig 4.8.b shows the value of the error function $|\hat{y}(k) - y_{ref}(k)|^2$, where $\hat{y}(k)$ evaluated at different samples of control signal. This figure shows that the minimum value of the cost function could be obtained from the input space U specified by the distribution of control signal. A similar result has been obtained at different instants of time. This is shown in Fig 4.8.c,d,e, and f.

Although the value of the model output has been very near to the desired output value at most of the operating points, it has been noticed that there is big uncertainty at other operating points. This is demonstrated in Fig 4.8.g. It is clear from this figure that the variability is very large in the output space. In addition, the value of the model output corresponding to the mean value of the control signal was far from the desired value.

The histogram for the output of the forward model shown in Figure 4.8 is constructed by applying different samples from the control signal distribution to the forward model. Taking the general case where the distribution of the system output is assumed to be approximated by a Gaussian distribution function, $p(y(k) \mid u(k-1), \mathbf{x}(k))$, the mean and the variance of this distribution can be calculated. However, the assumption of a Gaussian function is conditioned on the input value. Given that different input values are applied to the estimated distribution of the system output, different mean values and different variances are obtained. This is shown in Figure 4.10. The histogram of the system output Figure 4.8 is then constructed by calculating the frequency for the mean values of the forward model, which is a deterministic function of the control values. Given that different mean values are obtained from applying different control signal values to the forward model, the histogram of the forward model is not going to be Gaussian unless the applied control values happened to fall within a linear range of the output model. Since the variance of the predicted mean value from the inverse model is usually small, a linear approximation is expected and the histogram of the forward model is expected to look like a Gaussian function. Moreover, since the control signal distribution is sampled using importance sampling, most of the applied control values to the forward model are expected to be centred around the predicted mean value. This means that the histogram of the forward model is expected to have mean value equal to the output value of the forward model corresponding to the mean value of the control signal.

On the other hand the histogram of the utility function, $M(k) = \hat{y}(k) - y_{ref}(k)$ is generated from the output values of the forward model which result from different control values. It is clear that the utility function is a linear function of the forward model. Since the utility function is a linear function of the forward model, the histogram of the utility function will follow the histogram of the forward model. This means that if the histogram of the forward model is found to be Gaussian, the histogram of the utility function will be Gaussian as well. The mean values of the histogram of the utility function are shifted by the value of the reference model, y_{ref}. The mean value of the histogram of the utility function is expected to be equal to the mean value of the histogram of the forward model minus the reference model value, $\bar{M}(k) = \bar{y}(k) - y_{ref}(k)$. This result can be derived as follows: Assuming that the resulting distribution of the system output from applying different control values from the control signal distribution is Gaussian

$$p(y(k) \mid u(k-1), \mathbf{x}(k)) = \frac{1}{(2\pi\sigma^2(k))^{1/2}} \exp\left\{ -\frac{(y(k) - \bar{y}(k))^2}{2\sigma^2(k)} \right\}, \tag{4.35}$$

where $\bar{y}(k)$ and $\sigma^2(k)$ are the mean and the variance of the system output calculated from applying different control values from the control signal distribution to the forward model. Since the utility function is a linear function of the system output, defined as

$$M(k) = y(k) - y_{ref}(k), \tag{4.36}$$

the distribution function of the utility can be obtained by substituting $M(k) + y_{ref}(k) = y(k)$ into Eq. (4.35). This yields

$$p(M(k) \mid y(k)) = \frac{1}{(2\pi\sigma^2(k))^{1/2}} \exp\left\{ - \frac{(M(k) + y_{ref}(k) - \bar{y}(k))^2}{2\sigma^2(k)} \right\}. \tag{4.37}$$

From Eq. (4.37) it is clear that the distribution of the utility function has a mean value equal to $\bar{y}(k) - y_{ref}(k)$ and a variance equal to the calculated variance of the system output.

The results obtained support these equations. Take as an example Figure 4.9.d, the mean on this histogram is shown to be equal to -0.05. According to Eq. (4.37), this can be obtained by subtracting the desired value of the system output at this instant of time -0.6 which can be taken from Table 4.2 from the mean value of the system output at this instant of time (the mean value shown on the histogram of Figure 4.8.g) which is seen to be -0.65. The variance in Figure 4.9.d is seen to be equal to the variance in Figure 4.8.g. This again agrees with the derived equation from the utility distribution function.

Time, k	$y_{ref}(k)$	Variance(k)
7	0.2689	0.0108
750	0.4	0.0252
1250	0.3	0.013
1300	-0.6	0.0373

Table 4.2: The values of desired output and variance at different instants of time.

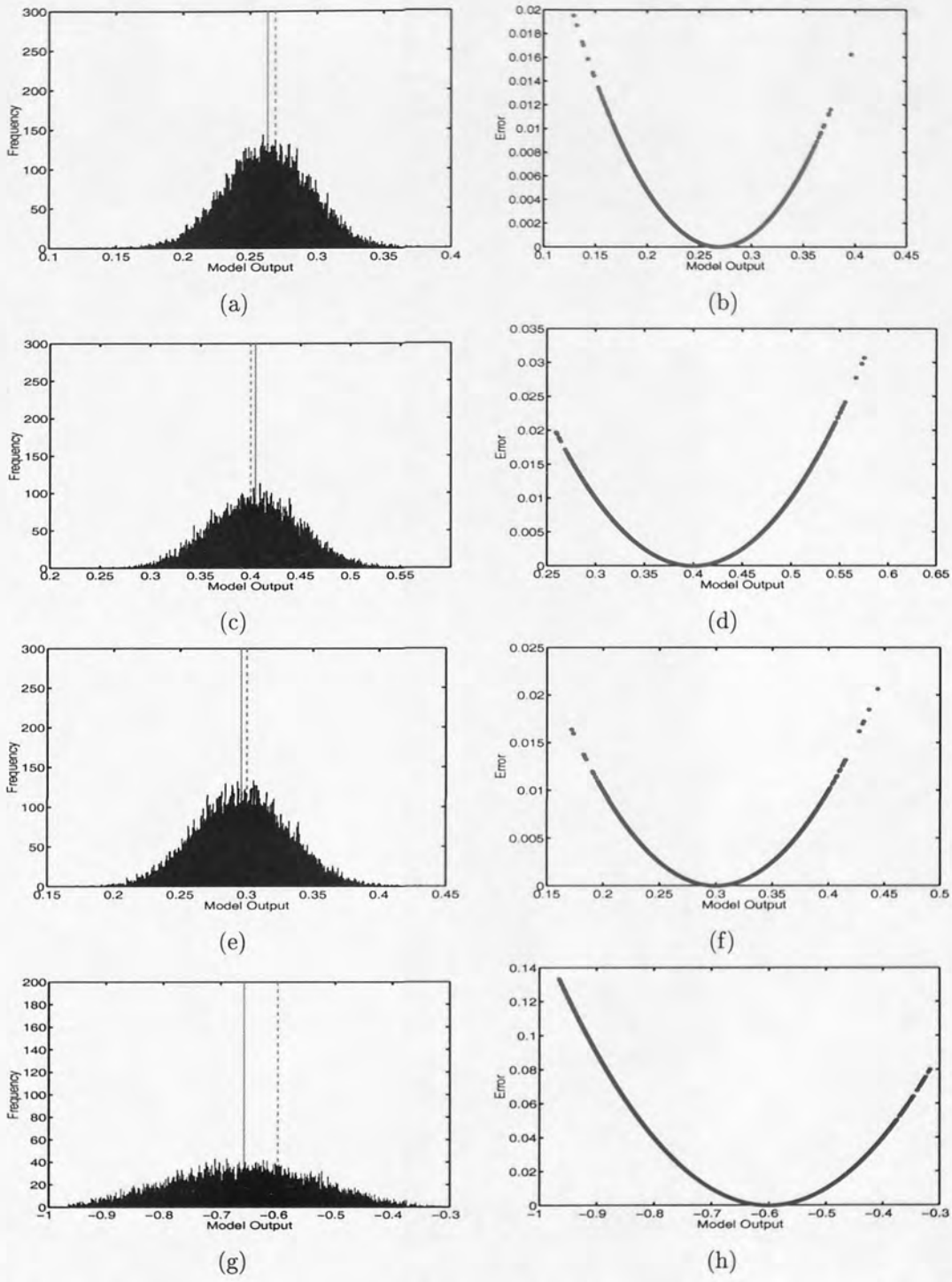


Figure 4.8: The output model histogram of the Stochastic SISO example: (a) The histogram of the model output resulting from the different samples of control signal at time 7, the desired output value at this time is 0.2689. (b) The value of the error function resulting from different samples of control signal at time 7. (c) The histogram of the model output resulting from the different samples of control signal at time 750, the desired output value at this time is 0.4 . (d) The value of the error function resulting from different samples of control signal at time 750. (e) The histogram of the model output resulting from the different samples of control signal at time 1250, the desired output value at this time is 0.3. (f) The value of the error function resulting from different samples of control signal at time 1250. (g) The histogram of the model output resulting from the different samples of control signal at time 1300, the desired output value at this time is -0.6. (h) The value of the error function resulting from different samples of control signal at time 1300.

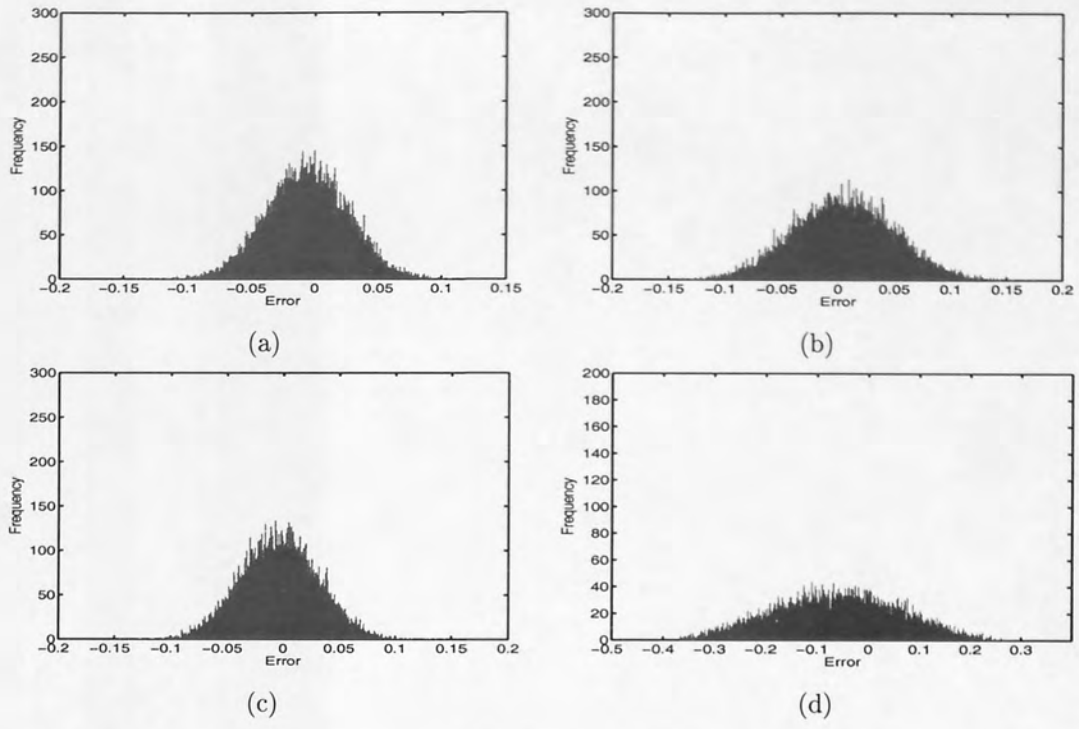


Figure 4.9: The error function histogram of the Stochastic SISO example: (a) The histogram of the error values resulting from the different output values at time 7. (b) The histogram of the error values resulting from the different output values at time 750. (c) The histogram of the error values resulting from the different output values at time 1250. (d) The histogram of the error values resulting from the different output values at time 1300.

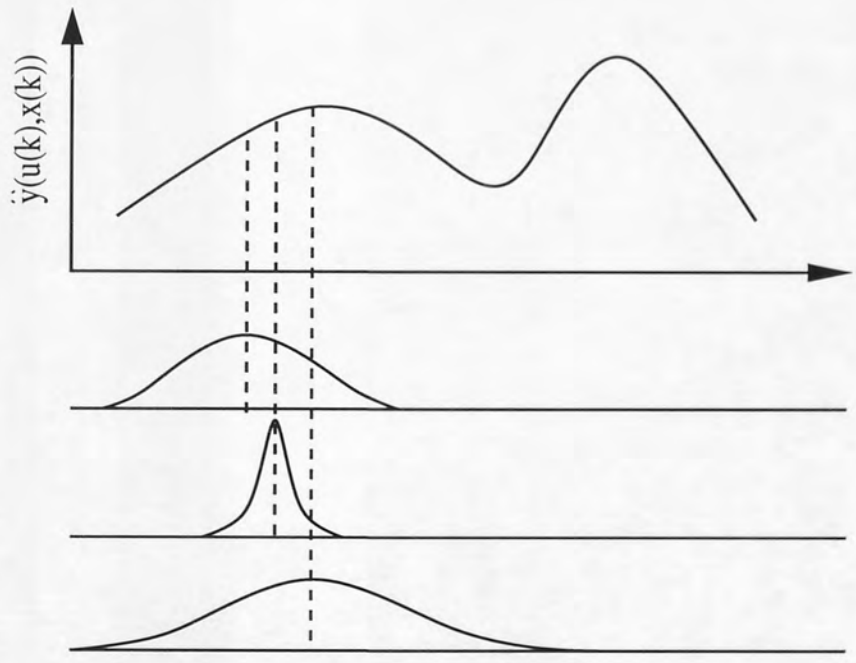


Figure 4.10: The result of applying different control values from the control signal distribution to the distribution of the system output. For each applied control signal, a different mean value and a different variance are obtained.

4.9 Alternative decision-making criteria

The proposed sampling approach in this work is concerned with finding the minimum of an error function which is defined as the square difference between the model output and the desired output

$$J(k) = \underset{u \in U}{\text{Min}} E_{\hat{y}}[(\hat{y}(k+d) - y_{ref}(k+d))^2]. \tag{4.38}$$

By contrast, the classical Bayesian approach for this problem requires integration over all possible values of the control signal. The integration over all possible values of the control signal can be evaluated by defining different integrand functions. For the purpose of this work, two integrand functions were chosen:

- The first integrand function is defined to be the forward model function. Here the expected value of the forward model resulting from applying a number of control signals from the control signal distribution is evaluated. In other words the problem is to evaluate the following integral

$$< \hat{y}(k+d) >= \int p(u(k) \mid \mathbf{s}(k)) N_f(u(k), \mathbf{x}(k)) du(k), \tag{4.39}$$

  where $p(u(k) \mid \mathbf{s}(k))$ represents the estimated distribution function of the control signal, and $N_f(u(k), \mathbf{x}(k))$ is the neural forward model.

  Since the conditional distribution function of the control signal can be estimated and is generally directly available, the integral (4.39) can be approximated with the finite sum

$$< \hat{y}(k+d) >= \frac{1}{L} \sum_{i=1}^{L} N_f(u_i(k), \mathbf{x}(k)), \tag{4.40}$$

  where $u_i(k)$ represents a sample of control signal from the estimated conditional distribution of the control signal $p(u(k) \mid \mathbf{s}(k))$.

  Given that the integral (4.40) has been evaluated, the optimal control value can then be taken as the value of control signal from the set of generated samples which minimises a prespecified distortion function

$$\underset{u \in U}{\text{Min}}[D(< \hat{y}(k+d) >, N_f(u(k), \mathbf{x}(k)))]. \tag{4.41}$$

  One of the possible definitions for the distortion function D, is the square difference between the evaluated expected value of the forward model and the value of the forward model resulting from each sample of control signals

$$D = [\{< \hat{y}(k+d) > - N_f(u(k), \mathbf{x}(k))\}^2]. \tag{4.42}$$

  This however, does not contain information about the desired trajectory to be followed by the real world process. Therefore, defining the distortion function as in (4.42) or more precisely evaluating the expected value of the forward model can be seen to be not goal directed and may lead to a bad tracking performance.

  To test the validity of this control method, the SISO stochastic control problem described in Equation (4.34) has been used. The forward model is firstly estimated in addition to the conditional distribution of the control signal. The expected value of the forward model is then calculated as defined in (4.40). The optimal control value is taken to be the control value which

minimises the square difference between the expected value of the forward model and the forward model value resulting from 1000 samples from the control signal distribution (4.42). The performance of this control method is shown in Fig 4.11. As expected the performance of the controller derived from this method was found to be unacceptable. Indeed, the performance of this controller can be seen to be very similar to the performance of the controller derived by applying the mean value of the control signal to the real plant.

- To avoid the lack of information about the desired trajectory found in the first method, the integrand in the Bayesian approach is taken to be the error function rather than the forward model function. Therefore, instead of evaluating the expected value of the forward model, the expected value of the cost function is evaluated in this method

$$< E\{\hat{y}(k+d), y_{ref}(k+d)\} >= \int p(u(k) \mid \mathbf{s}(k)) E\{\hat{y}(k+d), y_{ref}(k+d)\} du(k), \tag{4.43}$$

where $E\{\hat{y}(k+d), y_{ref}(k+d)\} = [\hat{y}(k+d) - y_{ref}(k+d)]^2$ and where $p(u(k) \mid \mathbf{s}(k))$ represents the estimated distribution of the control signal.

In the same way the integral (4.43) can be approximated with the finite sum

$$< E\{\hat{y}(k+d), y_{ref}(k+d)\} >= \frac{1}{L} \sum_{i=1}^{L} E\{N_f(u_i(k), \mathbf{x}(k)), y_{ref}(k+d)\}, \tag{4.44}$$

Once the expected value of the cost function has been evaluated, the optimal control value can be taken as the control signal from the set of generated samples from the control signal distribution which minimises a prespecified distortion function

$$\underset{u \in U}{Min}[D(< E(\hat{y}(k+d), y_{ref}(k+d)) >, E(\hat{y}(k+d), y_{ref}(k+d)))]. \tag{4.45}$$

Equation (4.45) states that the distortion function is a general function of the expected value of the error function and the error function itself. Therefore, it can be defined as the square difference between the expected value of the error function and the error function

$$D = [\{< E(\hat{y}(k+d), y_{ref}(k+d)) > - E(\hat{y}(k+d), y_{ref}(k+d))\}^2]. \tag{4.46}$$

Comparing the distortion function defined in (4.46) and (4.42), it can be seen that more information is included in the former function.

The same SISO stochastic control problem is used here to demonstrate the performance of this Bayesian method. The forward model and the conditional distribution of the inverse model are firstly estimated, and indeed they are taken to be the same structure defined in Section 4.7. The number of samples to calculate the expected value of the cost function and to find the minimum of the distortion function is taken to be 5000 samples. The performance of this controller is shown in Fig 4.12. Significant overshoots in the process output can be seen, which indicates bad controller performance.

The experiments of the two Bayesian methods, by minimising the square difference between the average model output and the model output and by minimising the square difference between the average cost function and the cost function, have been repeated with different sample sizes. It is found that no matter what is the number of generated samples from the control signal

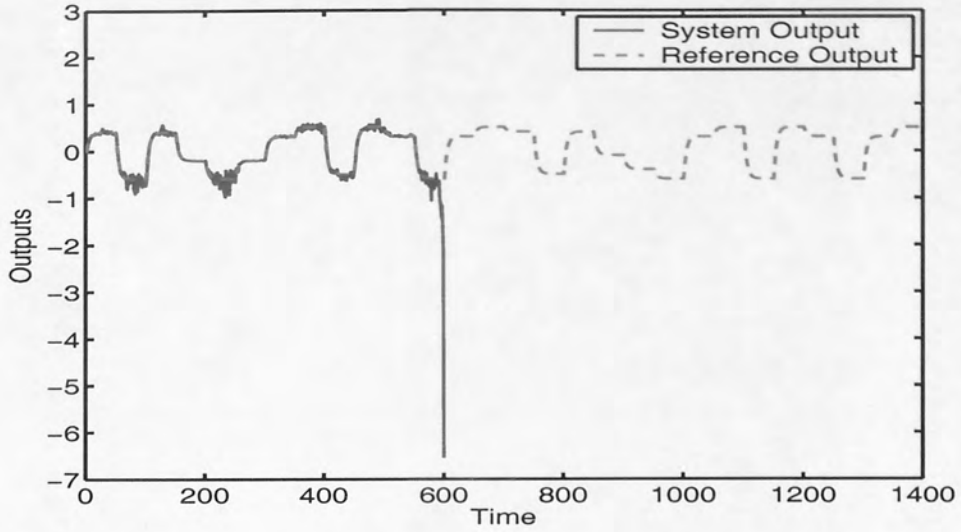


Figure 4.11: The desired and actual output values of the controller derived by calculating the average of the forward model output and minimising the distortion function, $D = [\{< \hat{y}(k+d) > -N_f(u(k), \mathbf{x}(k))\}^2]$.

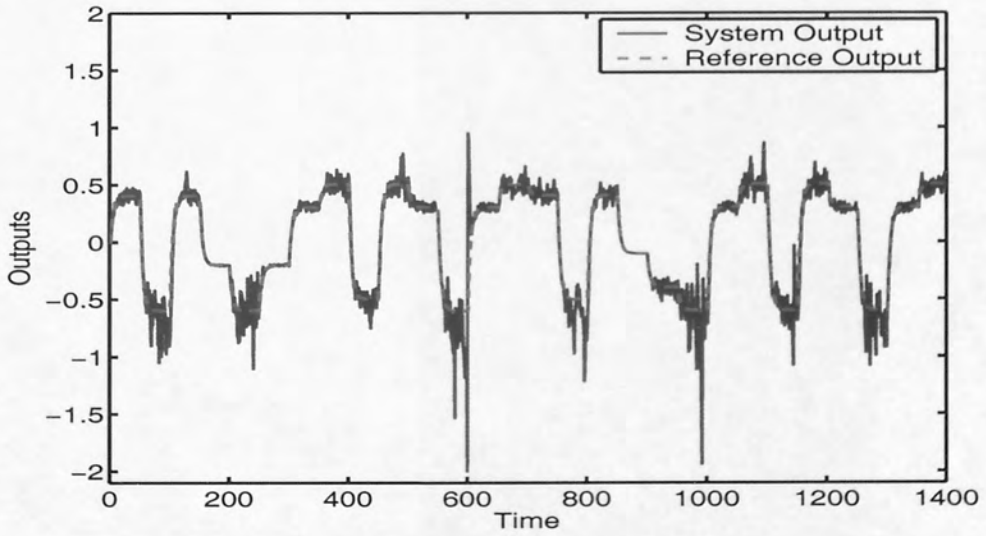


Figure 4.12: The desired and actual output values of the controller derived by calculating the average of the forward model output and minimising the distortion function, $D = [\{< E(\hat{y}(k+d), y_{ref}(k+d)) > -E(\hat{y}(k+d), y_{ref}(k+d))\}^2]$.

distribution the first Bayesian method, by evaluating the average of the forward model output, can not stabilise the process output. On the other hand, the second Bayesian method where the average of the error function is evaluated was found to be sensitive to the number of samples. For example, when the number of samples from the control signal distribution is taken to be less than 5000 samples this method could not stabilise the process output. In contrast, for larger number of samples the method is found to be able to stabilise the process output, but still with significant overshoots.

In the Bayesian method the expected value of either the forward model or the cost function is calculated in terms of the significance of each control signal to this function. The estimated distribution of the control signal can be seen as a weighting factor for the control signal. After all, only one control value from the conditional distribution of the control signal, the value which minimises the distortion function in either method, is chosen. This is different from the proposed method in this work in that the average value of either the forward model or the cost function is firstly calculated. The optimal control value is then taken to be the control value which makes the output of the forward model or the error function follow the evaluated average forward output value or the average cost function value respectively.

The sampling approach on the other hand, takes the control value which makes the output of the forward model, assumed to be close to the actual output value, follow a prespecified desired output value. The proposed sampling method logically derives the optimal control law for adaptive control problems, since the objective in adaptive control problems is to make the system output follow the desired output rather than to make the cost function follow the average cost function or to make the forward model output follow the model average output.

In conclusion the proposed sampling method is found to be more appropriate for the purpose of this work, and therefore the two Bayesian methods will not be considered further in the rest of this work.

4.10 Discussion

This chapter focused on modelling the conditional distribution of the control signal. The estimation of the distribution function is based on the use of neural networks as an approximation method to approximate the inverse model of the plant.

Since the use of the predicted mean value of the control signal may not be optimal for any given performance function, a new method that uses the uncertainty measure around the predicted mean value of the control signal was proposed. The new 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. Convergence of the output error from updating the control signal was verified by using a proper Lyapunov function. The proposed control strategy in this work chooses the optimal control value almost in the same way as in the dynamic programming as mentioned in the introduction. 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 optimised on the entire range of the input output data. By predicting the mean and the variance of the control signal from the neural network, searching for a better value of the control signal than the mean can be restricted to this region in which the optimal trajectory is

expected to lie.

Simulation experiments demonstrated the successful application of the proposed strategy to improve the controller performance and to stabilise a class of nonlinear control systems with large uncertainties. Since we are sampling our control signal from the estimated distribution and choosing which one better fits the model, the predicted mean 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 example given in this chapter is perhaps the simplest representative of a whole class of density-estimating neural networks (such as 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 mean and the uncertainty around the mean. This more general approach is not constrained by assumptions of invertibility and can deal with hysteretic processes as well.

Chapter 5

Multi-variable Control Problems

Recently, several authors have used neural networks for the identification and control of unknown nonlinear multi-variable processes [94; 93]. 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 multi-variable 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 multi-variable systems, have been proposed in [94; 93], 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 [14] 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 [47]. 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 is used for the tracking error and a multi-layer neural network is used to cancel the modelling errors. Ultimate boundness of error signals is shown through Lyapunov stability analysis.

An inversion based neurocontroller for solving control problems of uncertain nonlinear SISO systems has been presented in Chapter 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 has been given in Section 4.6. The approach in Chapter 4 is based on the assumption that the estimated distributions of control signals are Gaussian.

In this chapter we extend the approach of Chapter 4 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 better control laws 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.

The chapter starts in section 5.1 by discussing the application of the sampling approach to the MIMO control systems. The curse of dimensionality problem which occurs when the dimension of the input space increases is discussed in section 5.2. Section 5.3 discusses the preprocessing for the

training data. A MISO stochastic control problem is given in section 5.4 which verifies the successful application of the sampling method. In section 5.5 a highly nonlinear MIMO control problem is used to demonstrate the successful application of the sampling method to high dimensional problems. The discussion is presented in section 5.6.

5.1 Proposed control method

The same sampling procedure used for the SISO problems can be used to obtain an optimal control law for MISO and MIMO problems. The only difference here is that the sampled values need to be generated from the control signal distributions. Although the predictive error bar method for estimating predicted output variances can provide a general stochastic model for the output variances, it is assumed in this work that the error matrix $< ee^T >$ of the predicted outputs is diagonal, and consequently a Gaussian distribution for the random variables. Therefore, the samples have to be generated independently from the estimated Gaussian distribution of each control signal.

For example, for a two-dimensional problem a vector of samples needs to be generated from the distribution of the first control signal and of the second control signal independently. The admissible control values at each instant of time are then generated using a two dimensional grid so as to represent the surface of the model output. The state or the output however, does not need to be tracked since 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 sampling procedure can then be summarised as follows: since local conditional distributions for the inverse model and the forward model are estimated off-line, and according to the proposed sampling method, a vector of samples has to be generated from the inverse model distribution. These samples can then be propagated forward through the forward model of the plant. The resulting vector of outputs is then compared with the desired trajectory specified for the plant to follow. The control signal that is responsible for the output value which is the nearest to the desired value is then taken as the optimal control value. The optimal control values can then be forwarded to the real plant. These steps should be repeated at each instant of time. The full architecture for sampling from the Gaussian distribution is shown in Figure 5.1.

5.2 Problems with increasing the dimensionality of the input

In the proposed sampling method, samples need to be generated from the control signal distribution. Those sampled values are then used to calculate the state or the output values of the system model. Consider for example the problem of minimising the following general vector performance measure by taking samples from the distribution of control signals

$$J(k) = \underset{u \in U}{\text{Min}} \underset{\bar{v}}{E}[(\hat{g}(\mathbf{u}(k), \mathbf{x}(k), W) - \mathbf{y}_{ref}(k+d))^2], \tag{5.1}$$

where

$$\hat{\mathbf{y}}(k+d) = \hat{g}(\mathbf{u}(k), \mathbf{x}(k), W), \tag{5.2}$$

and where $\mathbf{x}(k) = [\mathbf{y}(k), ..., \mathbf{y}(k-q+1), \mathbf{u}(k-1), ..., \mathbf{u}(k-p+1)]$, $\mathbf{x}(k)$ is an N-dimensional vector, $\mathbf{y}(k)$ is an m-dimensional vector, and $\mathbf{u}(k)$ is an n-dimensional vector.

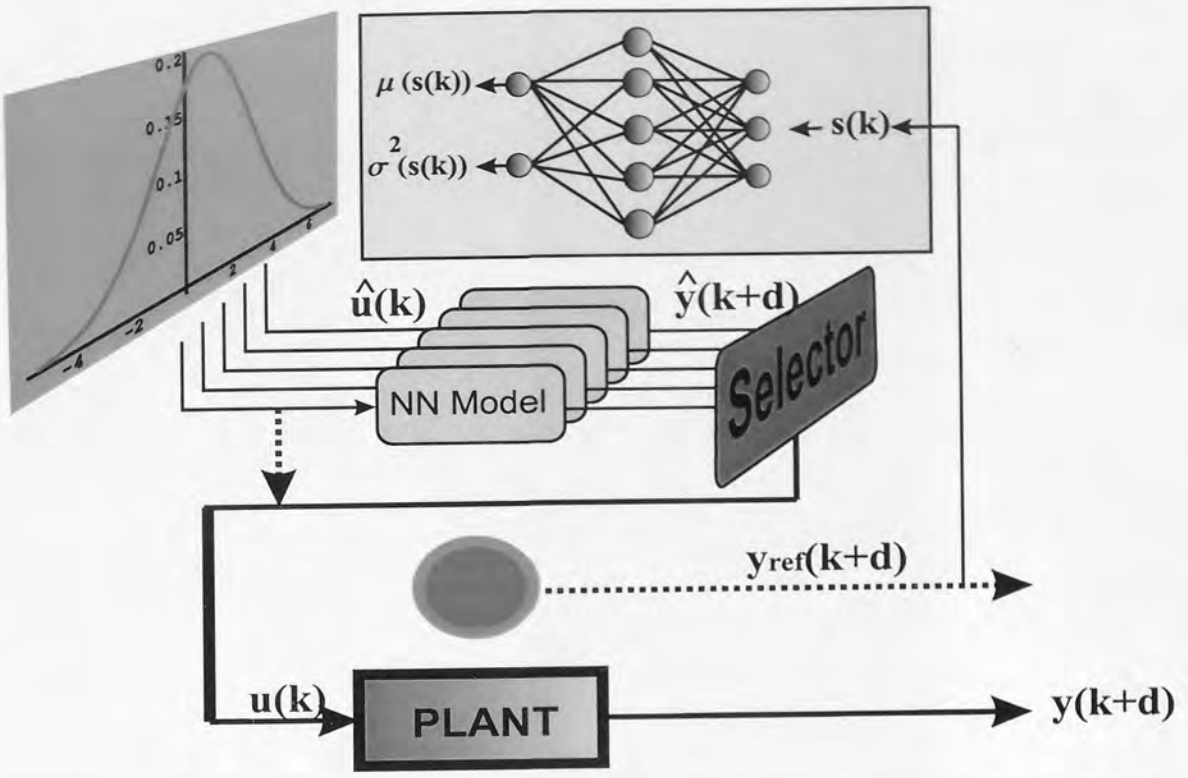


Figure 5.1: The architecture of the proposed sampling method from a Gaussian density function. The inputs to the neural network of the inverse model are $\mathbf{s}(k)$, the outputs are the expected value and the variance of the control signal.

As discussed in the previous section the same algorithm can be used regardless of the dimension of $\mathbf{u}(k)$ or the dimension of $\mathbf{y}(k)$. But the question to be asked is what may happen to the computational feasibility, and consequently to the computational time?

Since the number of samples from the distribution of control signals determines the size of the output space vector, for a single input function the number of output values to be considered at each stage can be seen to be equal to the number of samples from the input distribution. This is because a function of one variable $R = R(v)$, can be visualised as a curve in the plane.

However a function of two variables $R = R(v, w)$ is a surface in three dimensions. This means that, 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 are then generated using a two input dimensional grid as shown in Fig. 5.2 to represent the output surface, and consequently the error function. The size of the output space vector to be monitored is then determined from the number of samples generated from the two dimensional grid.

If $R = R(v)$ is specified over 100 samples of v, then 100 values of R need to be monitored at each stage. If $R = R(v, w)$ is specified over a two dimensional grid of 100 samples of v, and 100 samples of w, then $100 \times 100 = 10^4$ values in the R space are required now to be monitored. The problem becomes more troublesome in the computational time for higher input dimensionality.

The time required to search for better control values from the distribution of control signals increases with the dimensionality of the control signals. Despite this, it has been noticed that a small number of samples from the control signal distribution is enough to find a better control value than the mean value. Therefore, only a small number of samples need to be generated from the control

signal distribution. Increasing the number of samples does not improve the accuracy of the proposed sampling method significantly. This means that the proposed sampling method can be implemented in real time.

The state of the output however, does not need to be tracked since 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 (local conditional distributions were estimated off-line). This gives an advantage over the dynamic programming approach, where the state of the system needs to be tracked and where the optimal control sequence can be found by working backward in time using the recurrence relation.

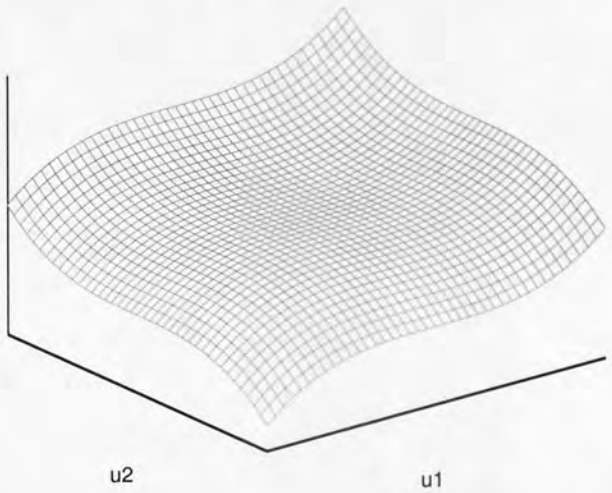


Figure 5.2: The generated samples from a two-dimensional control signal.

5.3 Preprocessing the training data

For training data, where the input and output ranges have significantly different magnitudes, some kind of normalisation for the input and output data sets is required. This normalisation is very important, since some variables may be given more significance because of the differences in the magnitude. Take as an example an input variable 1 with a magnitude of 20000, and an input variable 2 with a magnitude of 20. In order for input 2 to have any significance the assigned weight from the second input to any node in the hidden layer must be much greater than that of the first input weight. Moreover, the nonlinear functions of the hidden layers are usually of exponential type. Since there are limitations on the computational processing of computers, these functions cannot distinguish between two different input values when both are very large, because both will yield identical threshold values.

The same thing can be said about the RBF network, except that the problem becomes worse since the values of the variances need to be chosen carefully if they are chosen beforehand, and the convergence will almost be impossible if the variances need to be updated using the gradient information of the error function.

To avoid such problems all the input and output variables have been normalised for training purposes. The method taken in this work is to divide each variable in the training set u_i by its maximum value

$$u_{i,norm} = \frac{u_i}{u_{i,max}}, \tag{5.3}$$

where $u_{i,norm}$ is the normalised variable and $u_{i,max}$ is a normalisation factor.

This method is simple, but is an adequate method for the simulations considered in this work. The transformation of the input and output variables given by Eq. (5.3) can be considered as a reduction in the magnitude of these variables. In addition all information in the input output sets will be reserved. For other normalisation methods the reader is refered to [7].

5.4 Multiple input single output stochastic process

In this simulation a MISO experiment is conducted to illustrate the applicability of the proposed sampling method to multivariate control problems. A two input one output nonlinear process described by the following equation is taken

$$y(k+1) = y(k)\sin(u_1(k)) + u_2(k) - 0.1u_1(k-1)y(k-1) + 0.5u_2^2(k-1) + \bar{v}(k). \quad (5.4)$$

The output tracking ability and robustness of the proposed method is tested in this simulation. This simulation experiment is used in [30] to illustrate the theoretical development of the diagonal recurrent neural network (DRNN). A noise term has been added to the equation used in [30]. This noise is considered to be a Gaussian random variable $\mathcal{N}(0, 0.1)$, with zero mean and 0.1 variance. The noise component presented to the system equation is an additive noise component. The system given by Eq. (5.4) is a strongly nonlinear system since the dynamic gain between the output $y(k+1)$ and the first input $u_1(k)$ changes in both the amplitude and sign. The manipulated variables in this experiment are $u_1(k)$ and $u_2(k)$.

In the training stage the first input to the plant and the model u_1 is taken to be a sinc function with an additive Gaussian noise $\mathcal{N}(0, \sigma^2)(\sigma = 0.3)$. The range of this input is [−1.0 to 2.5]. The second input to the plant and the model however, is taken to be a sine function with an additive Gaussian noise $\mathcal{N}(0, \sigma^2)(\sigma = 0.3)$. The range of this input is [−1.8 to 1.8]. Using these input values the output values are ranged between [−3 to 9].

In this system the delay from the first and the second inputs is 1. Therefore, the relative degree of this system is 1. The plant is considered to be described by Eq. (5.4). For the proposed sampling method an accurate model for the plant is required. The inverse model of the plant is also required to calculate the control signals. Nonlinear models described by neural network models are taken to find the inverse and the forward models of the plant as described in the next sections.

5.4.1 Identification

Since an accurate forward model is required for the proposed sampling approach, a nonlinear neural network model described by the following input-output model

$$\hat{y}(k+1) = N_f(y(k), y(k-1), \mathbf{u}(k), \mathbf{u}(k-1)),$$

is taken to represent the actual system given by Eq. (5.4). Here $\mathbf{u}(k)$ is the vector of both the inputs, $\mathbf{u}(k) = [u_1(k), u_2(k)]$, and N_f is an RBF neural network. This neural network model was trained using the scaled conjugate gradient optimisation algorithm based on input output data taken from the plant and sampled every 1s. Using the cross validation method (based on the error between the model predicted output and the actual output, $\| y(k+1) - \hat{y}(k+1) \|^2$) it is found that the best optimal structure for the forward model is an RBF network with 18 Gaussian basis functions. In the validation

stage the data used to validate the model has been taken to be not seen in the training stage. For that purpose the first input and the second input are generated from a uniform distribution.

After training the forward model and choosing the best model structure, the model is tested on new data. The first and the second inputs were generated uniformly with additive Gaussian noise $\mathcal{N}(0, \sigma^2)(\sigma = 0.3)$. The first input has been generated from a uniform distribution defined on the interval [−0.4 to 0.4], while the second input has been generated from a uniform distribution defined on the interval [−0.8 to 0.8]. The result is shown in Fig 5.3. Very good approximation is achieved. From the figure one can see that the predicted output of the model is the conditional average of the actual output, averaged over the noise.

5.4.2 Direct adaptive inverse control

An input output model described by

$$\hat{\mathbf{u}}(k) = N_f(y(k), y(k-1), y(k+1), \mathbf{u}(k-1)),$$

is taken to find the inverse model of the plant. The same input output data used for training the forward mode is used to train the inverse model. The optimal structure for the inverse model is found using the cross validation method (based on the error between the model predicted output $\hat{\mathbf{u}}(k)$ and the actual output $\mathbf{u}(k)$, $\| \mathbf{u}(k) - \hat{\mathbf{u}}(k) \|^2$) to be a RBF network with 7 Gaussian basis functions. Here the same validation data used in the validation stage for the forward model is used to validate the inverse model.

After training the inverse model off-line and choosing the best structure, the inverse model is brought on-line and the control signal is obtained at each instant of time from the predicted output of the inverse model. In this stage the system output $y(k+1)$ is replaced by the desired output value $y_{ref}(k+1) = r(k)$, where the reference signal $r(k)$ is taken to be a sinusoidal signal given by

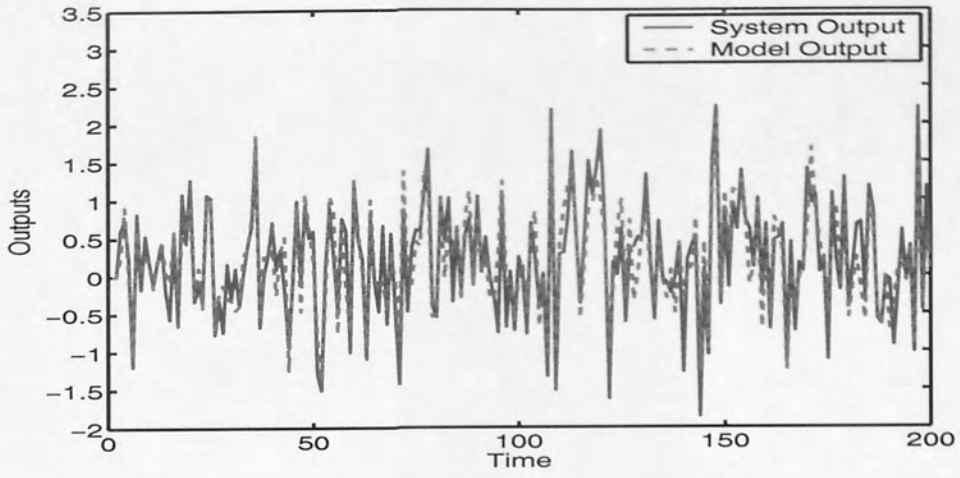
$$r(k) = 1.5 \sin\left[\left(\frac{\pi}{40}\right)k\right].$$

In the direct inverse control the predicted mean value from the inverse model is forwarded to the plant. The control result is shown in Fig 5.4. From this figure one can see that a bad tracking performance was obtained from the inverse controller. This bad tracking ability is clear especially at the negative part of the desired trajectory.

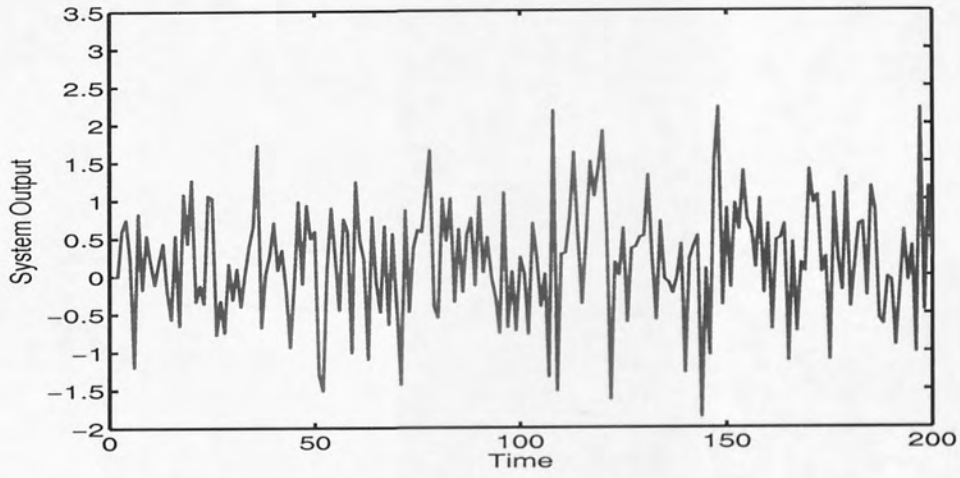
5.4.3 Proposed sampling control method

In the proposed sampling approach, both the predicted mean and the variance from the inverse model have been used to search for the optimal control values. Since the considered system in this simulation is two inputs single output, the variance of each control signal is estimated independently as presented earlier in Section 3.5. In the control stage 21 samples are taken from the distribution of each control signal. This makes the total number of samples sum to 21^2.

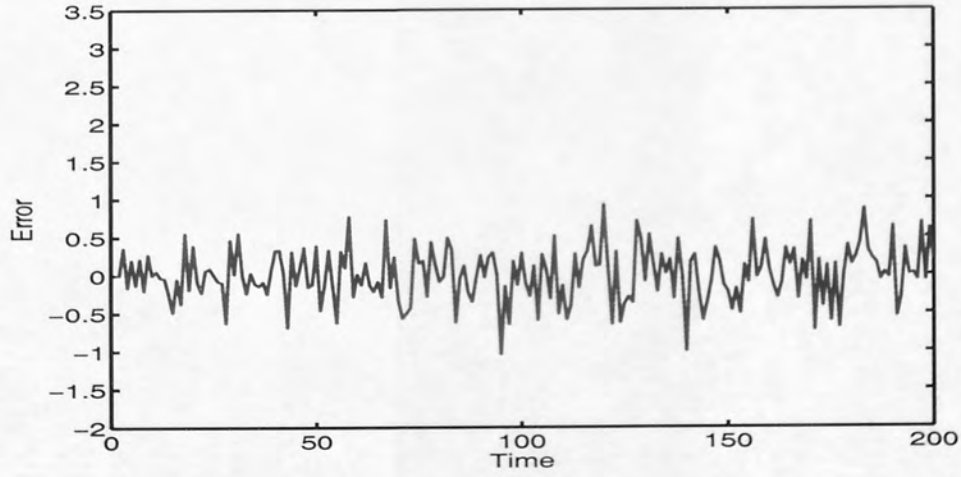
The performance of the sampling method is given in Fig 5.5. It is clear from Fig 5.5 that the controlled output of the proposed sampling method tracks the reference signal in a better way. This indicates that a better tracking ability could be obtained using the proposed sampling method. The overall average error from the sampling method and the direct inverse method is given in Table 5.1. Table 5.1 shows that the overall average error for the proposed sampling method is less than that of



(a)



(b)



(c)

Figure 5.3: The actual and the model outputs of the MISO system: (a) the actual and model outputs for the MISO stochastic process. (b) the system output for the MISO stochastic process. (c) the identification error for the MISO stochastic process

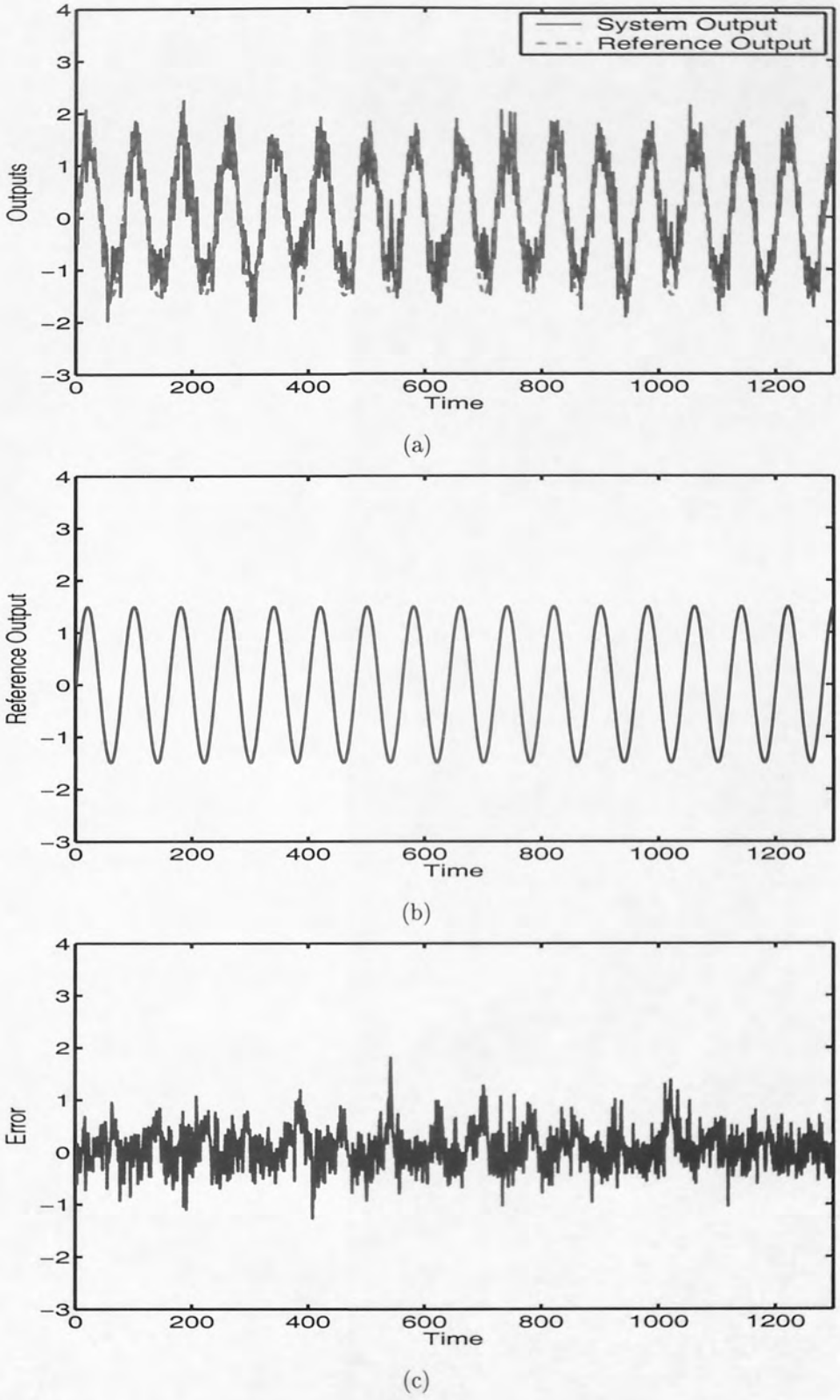


Figure 5.4: The performance of the classical control approach of the MISO system: (a) the actual and reference model outputs for the MISO stochastic process. (b) the reference model output for the MISO stochastic process. (c) the tracking error of the classical control approach for the MISO stochastic process

the direct inverse approach. The error difference between the absolute tracking error of the proposed sampling method and the absolute tracking error of the direct inverse control is shown in Fig 5.6. It is clear from Fig 5.6 that the error difference between the absolute tracking errors of the proposed sampling method and the direct inverse approach is like a Gaussian noise. Although the error difference between the tracking ability of the sampling and direct inverse approaches looked like Gaussian noise, a better tracking performance was obtained from the sampling approach by comparing between Fig 5.5 and Fig 5.4.

	Direct inverse	Sampling method
Tracking error	0.1559	0.1298

Table 5.1: The average tracking error of both the direct inverse control and the proposed sampling method.

5.5 Multiple-input multiple-output process

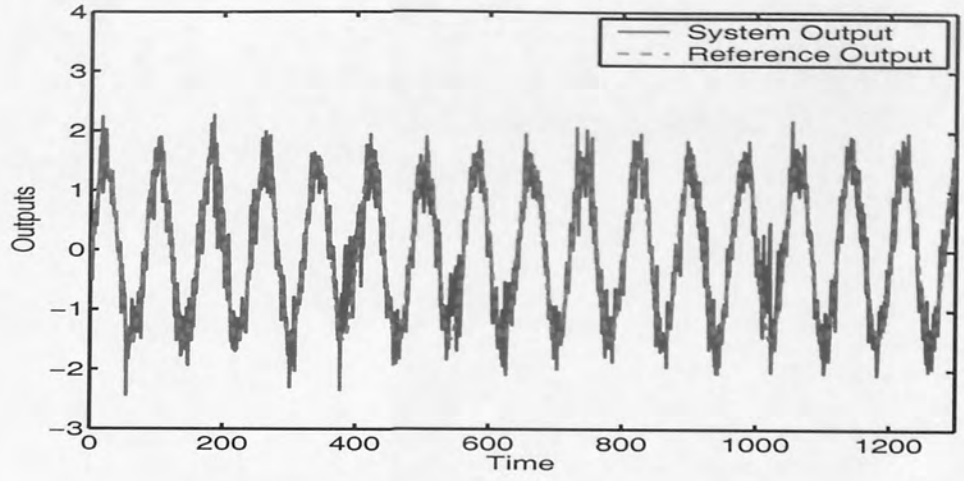
In order to illustrate the validity of the theoretical developments for MIMO systems, we consider a third order system with two inputs and two outputs described by the following state equation :

$$\begin{aligned}
x_1(k+1) &= 0.9x_1(k)\sin[x_2(k)] + \left[2 + 1.5\frac{x_1(k)u_1(k)}{1+x_1^2(k)u_1^2(k)}\right]u_1(k) + \left[x_1(k) + \frac{2x_1(k)}{1+x_1^2(k)}\right]u_2(k),\\
x_2(k+1) &= x_3(k)\{1+\sin[4x_3(k)]\} + \frac{x_3(k)}{1+x_3^2(k)},\\
x_3(k+1) &= \{3+\sin[2x_1(k)]\}u_2(k),\\
y_1(k) &= x_1(k),\\
y_2(k) &= x_2(k),
\end{aligned} \tag{5.5}$$

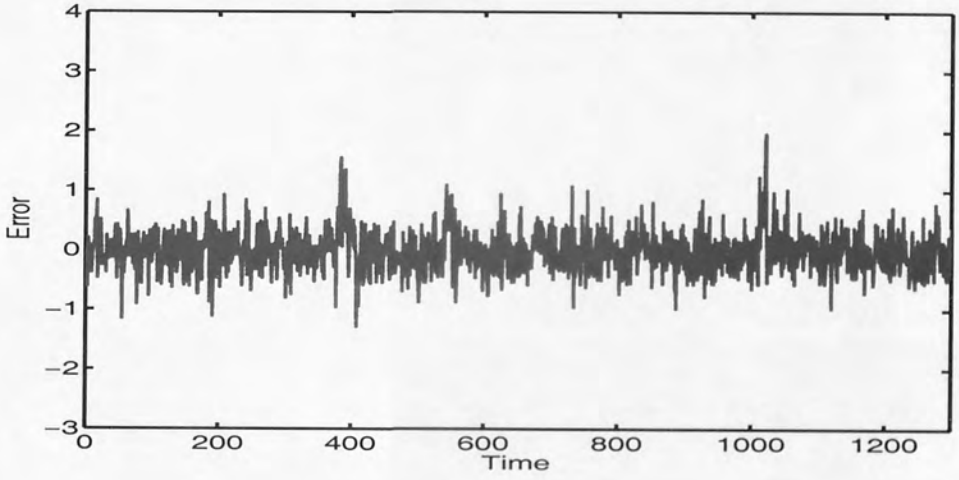
where $\mathbf{x}(k) = [x_1(k), x_2(k), x_3(k)]$ is the state, $\mathbf{u}(k) = [u_1(k), u_2(k)]$ is the control variable, and $\mathbf{y}(k) = [y_1(k), y_2(k)]$ is the output. This model has been used in [93] to illustrate theoretical developments for the indirect adaptive controller. In this system 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 Eq.(5.5). 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 [93]. An input-output model described by the following two equations was chosen.

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

where N_{f1}, and N_{f2} are multi-layer neural networks. This neural network model was 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 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 have been tested on new data that has not been seen in the training stage.



(a)



(b)

Figure 5.5: The performance of the proposed control approach of the MISO system: (a) the actual and reference model outputs for the MISO stochastic process. (b) the tracking error of the proposed control approach for the MISO stochastic process

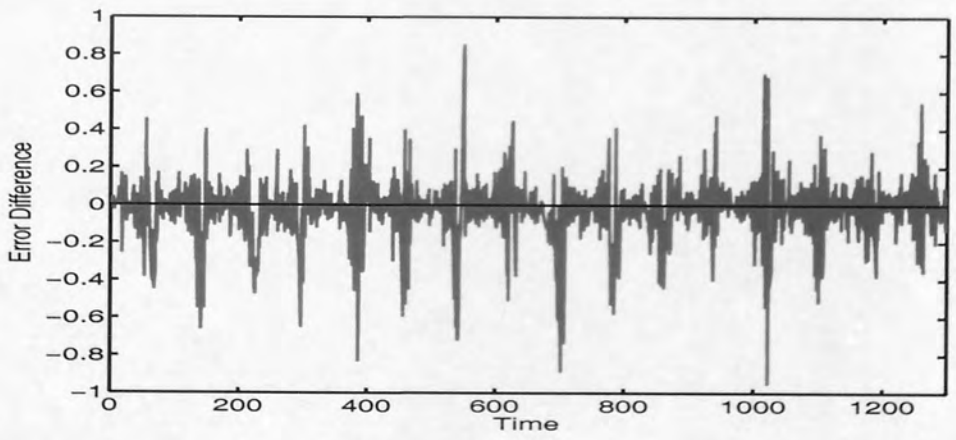


Figure 5.6: The error difference between the absolute tracking error of the proposed sampling method and the absolute tracking error of the direct inverse control for the MISO stochastic control problem.

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 number 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 multi-layer 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, $e = \| u(k) - \hat{u}(k) \|^2$ is taken to be the best model.

5.5.1 Identification

After training the forward models of the system described by Eq. (5.5) and choosing the structure of the neural network for both of the models, the models are tested on new input values. Here the first and the second inputs $u_1(k), u_2(k)$, are taken to be sine and cosine functions respectively

$$u_1(k) = 0.5 \sin\left(\frac{2\pi k}{25}\right), \tag{5.6}$$

$$u_2(k) = 0.3 \cos\left(\frac{2\pi k}{25}\right). \tag{5.7}$$

The actual output and the model output of the first and second models are shown in Fig 5.7. From this figure it can be seen that a better model has been obtained for the first output. The model of the second output could not capture the high frequency component. Although the second model could not capture the high frequency component, it can be seen from Fig 5.7.b that the second model could model the average output values for the second output.

5.5.2 Direct adaptive inverse control

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. 5.8. The performance of the classic controller was seen to be poor with large over shoots around the desired response in the second output $y_2(k)$.

5.5.3 Proposed sampling control method

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. Firstly

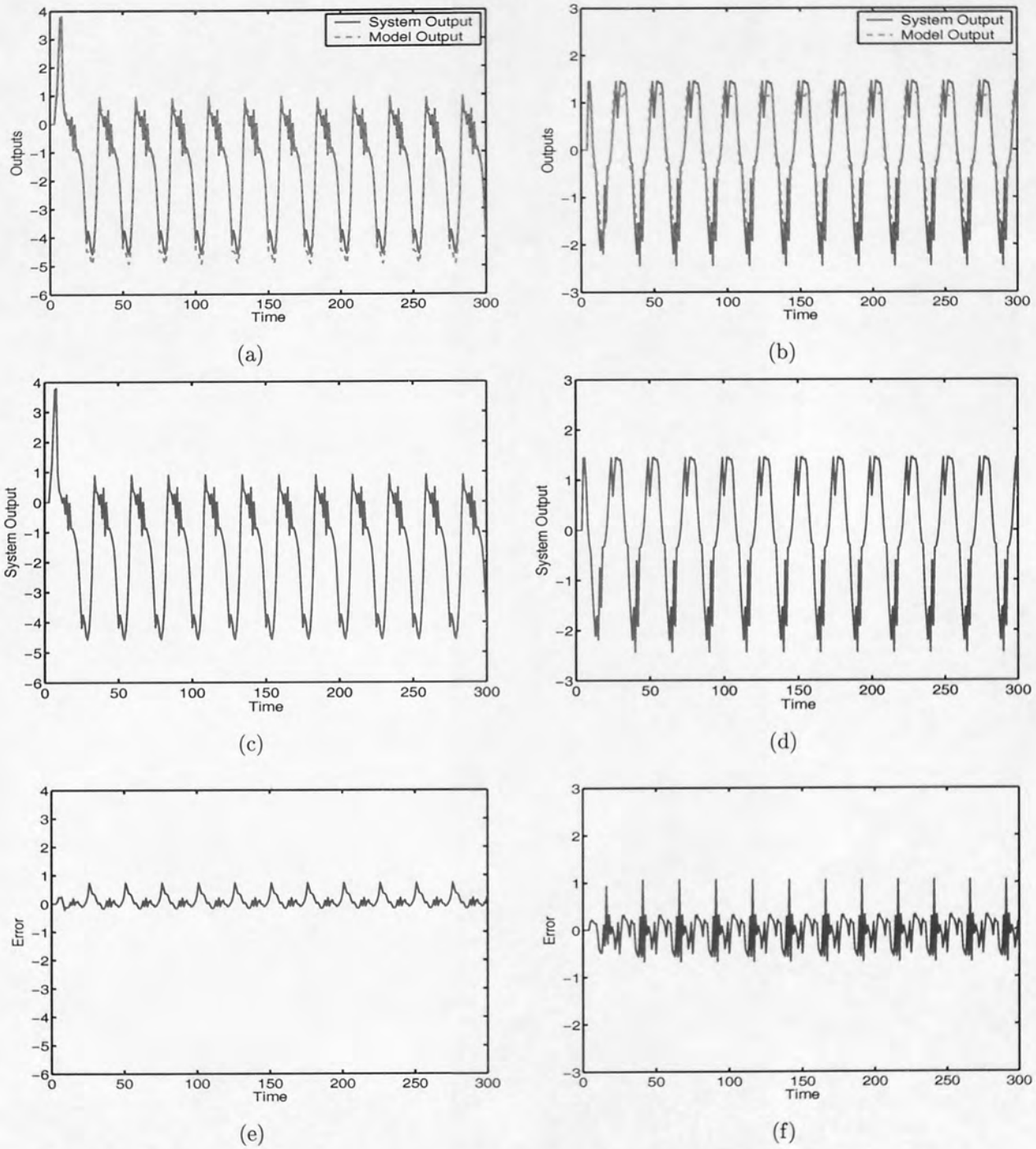


Figure 5.7: The actual and the model outputs for the first and second output of the MIMO system: (a) the actual and model outputs for the first output of the plant. (b) the actual and model outputs for the second output of the plant. (c) the first output of the plant. (d) the second output of the plant. (e) the identification error of the first output of the plant. (f) the identification error of the second output of the plant.

20 samples have been generated from the Gaussian distribution of each control signal. However the number of samples used to search for the 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. 5.9. 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.

5.6 Discussion

Incorporating uncertainty estimation to improve the performance of controllers for multi-variable control problems was considered in this chapter. A method that uses uncertainty around the predicted mean value of the control signal was proposed by modelling the distributions of control signals, assumed to be Gaussian. 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 mean and the variance 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 multi-variable problems.

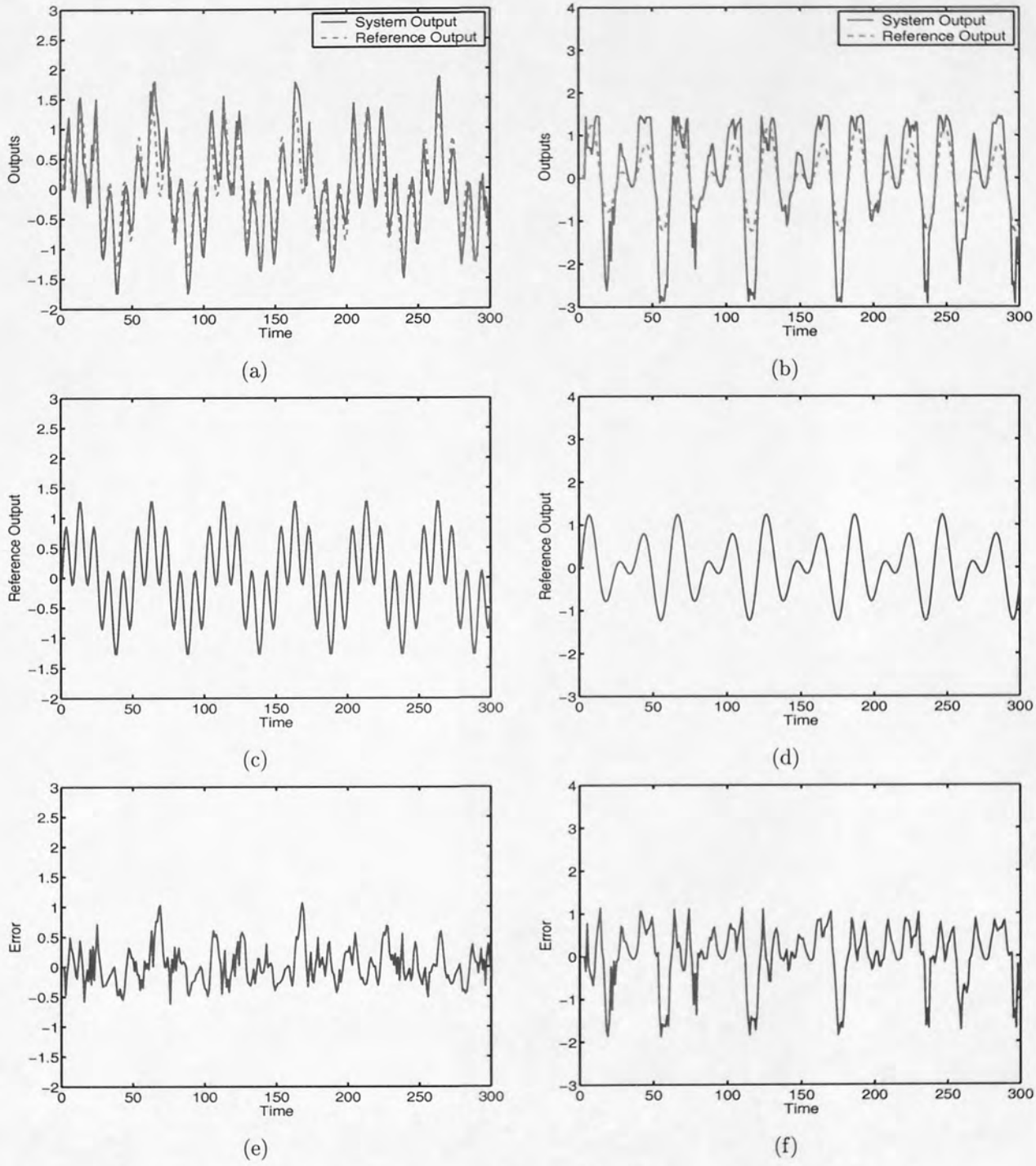


Figure 5.8: Performance of the classical control approach: (a) the actual and reference model outputs for the first output of the plant. (b) the actual and reference model output for the second output of the plant. (c) the reference model output for the first output of the plant. (d) the reference model output for the second output of the plant. (e) the tracking error of the first output of the plant from the direct inverse control. (f) the tracking error of the second output of the plant from the direct inverse control.

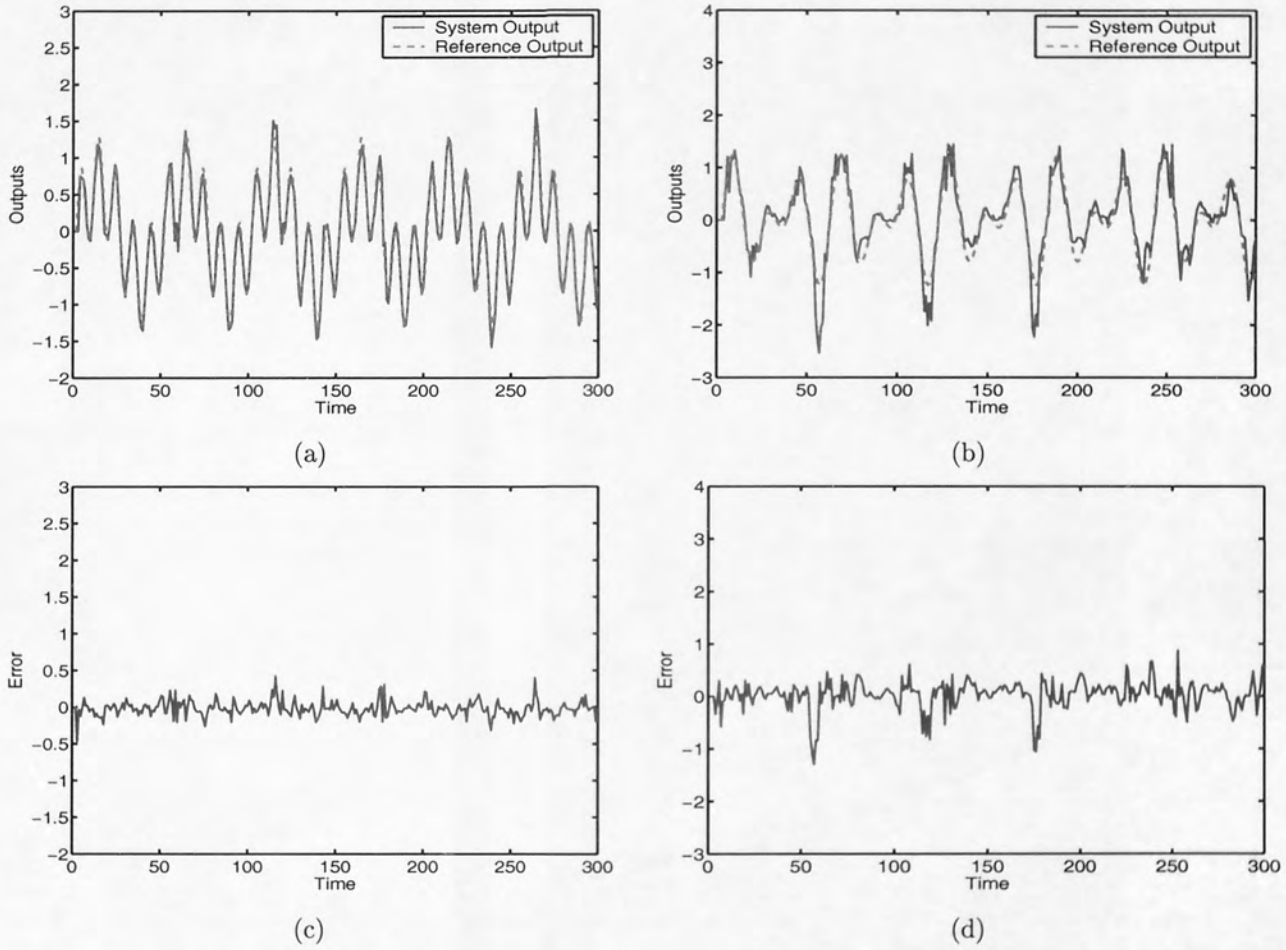


Figure 5.9: Performance of the proposed control approach: (a) the actual and reference model outputs for the first output of the plant. (b) the actual and reference model output for the second output of the plant. (c) the tracking error of the first output of the plant from the proposed control method. (d) the tracking error of the second output of the plant from the proposed control method.

Chapter 6

Multi-valued Control Problems

Although neural networks are accepted to be capable of representing complex nonlinear control systems, complex networks are also characterised by slow learning and poor generalisation. Nevertheless, in order to be able to represent more complex systems and to treat higher uncertainty levels, neural networks should be accompanied by a number of complementary techniques.

In practice there exist a number of processes that can not be represented by a single model. In fact these processes are characterised by a number of distinct modes of behaviour during their operation. This type of plant complexity is called multi-modality [27].

In the control literature, three types of multi-modality are considered. The first one is temporal multi-modality: this situation occurs when the plant operates in suddenly changing environment or when a fault condition occurs. The second type of multi-modality is called spatial multi-modality: it occurs when the plant is characterised by a highly nonlinear complex function, which exhibits different characteristics over different operating zones or operating spaces. The third type of multi-modality which will be considered in this work occurs when one tries to acquire the inverse dynamics of the plant using supervised learning. This is an ill posed problem, where there is a well defined forward solution, but the solutions to the inverse problem are not unique. Most motor control problems are ill-posed in the sense that there is a well defined forward solution, but the inverse solution is not unique.

Take as an example, the two link robot arm shown in Fig 6.1.a. The forward problem in this case is to map the joint angles (θ_1, θ_2) to the Cartesian coordinates (x_1, x_2). This mapping is a single valued mapping. However in the inverse problem, one is interested in calculating the joint angles (θ_1, θ_2) such that end effector of the arm can move to a desired location in the Cartesian coordinate. In general, this inverse problem is not a single valued problem, this is shown in Fig 6.1.b. Figure 6.1.b shows that the same end effector position can be reached from two configurations of the joint angles, elbow up and elbow down.

As mentioned in the previous chapters, training the inverse controller based on the assumption that the actual control commands are known is called the direct inverse control. This approach leads to extremely poor performance when applied to inverse problems in which the mapping to be learned is multi-valued or involves hysteretic transfer characteristics [87] as in the case of the two link robot arm. This is due to the fact that when a least square approach is applied to an inverse problem, it will then approximate the conditional average of the target data. However, the average of several solutions is not necessarily a correct solution.

In [87; 26] 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 [84; 125] for acquiring the inverse dynamics model of multi-valued functions. However the output from the above mentioned approaches has been an estimation for the control value only.

Although a mixture density network models the conditional probability distribution, it uses only a single control value when used as a controller in the control loop. This value is the mean value of one of the kernel functions corresponding to the most probable branch. Recently growing interest in robust control by accounting for model and system uncertainty has produced new results. For example, in [14] 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.

In this work a different way for accounting for the uncertainty around the predicted output of the inverse controller has been proposed and presented in Chapters 4, and 5 and published in [41; 40]. The controller is designed to predict both the control law and the uncertainty around that control law, which leads to the assumption that the inverse controller can be approximated by a Gaussian function. A sampling approach is used to search for a better value of the control signal than the mean in this region where the optimal solution is expected to lie. The stability for the updating rule of the control law has been proved in Section 4.6.1, see [40] as well.

However the Gaussian assumption is not always possible, as for example problems where the inverse mapping can be multi-valued. This chapter will go beyond the Gaussian description of the distribution of the inverse controller. The main idea is to use the mixture density network to model the multicomponent distributions of the inverse model of the plant. The idea of the mixture density network is not new [13; 87; 26; 25], but it has not been exploited in a control context before. The work presented here differs from [87] in that we consider the multicomponent distribution to search for the optimal control law, rather than taking a single estimate value corresponding to the most probable value. In Chapters 4, and 5 only the Gaussian distribution is considered. We extend this work by considering more general distributions, which create a general framework for searching for the optimal control law from an arbitrary probability distribution [42; 44].

The use of the mixture density network to obtain the inverse mapping will be compared with that of the direct inverse control. Furthermore, a comparison study for sampling from different conditional distributions [43] will be provided. It will be shown that sampling from the right distribution of the generator of the data, provides the best and the fastest result. This implies that sampling can be used in real time control to obtain a better control law, in situations where the predicted output of the neural network is uncertain.

Section 6.1 provides a general overview of the current methods in the control field which deal with the multi-modality problem. The theory of the mixture density network as an alternative method to find the control law for ill-posed problems is discussed in section 6.2. The extension of the proposed sampling method to the mixture density network is discussed in section 6.3. A synthetic example to demonstrate the successful application of sampling from an arbitrary distribution function, by sampling from the distribution function modelled using a mixture density network is given in section 6.4. Section 6.5 provides a general discussion about the probability density function modelled for the synthetic example in addition to a general discussion about the prior values from the mixture density network. In section 6.6 the application of the mixture density network to a highly nonlinear MIMO

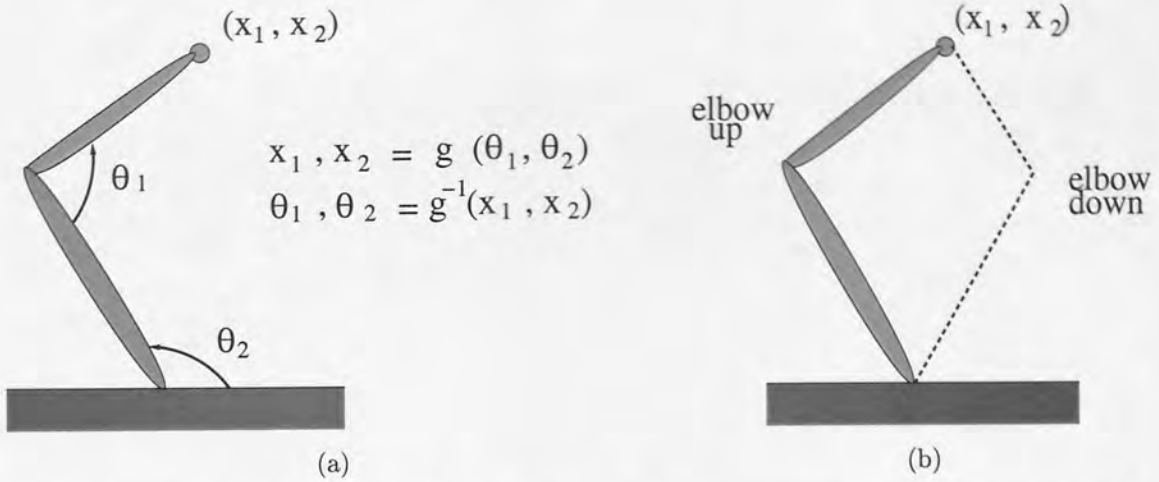


Figure 6.1: The ill-posed problem of motor control: (a) the schematic diagram of a two link robot arm, the forward kinematic problem is the mapping from the joint angle (θ_1, θ_2) to the Cartesian coordinates (x_1, x_2), which is a single valued mapping. The mapping from the Cartesian coordinates (x_1, x_2) to the joint angles is the inverse kinematic problem, and is usually a multi-valued problem. (b) This figure shows the two possible configurations for the joint angles (elbow up and elbow down), which can move the end effector of the arm to the position (x_1, x_2), .

dynamical system together with the application of the proposed sampling method from the modelled distribution of the mixture density network is demonstrated. Section 6.7 provides a general discussion about the probability density function of the mixture density network for the MIMO dynamical system. Sampling from other conditional distribution such as the uniform distribution is presented in section 6.8. The chapter discussion is given in section 6.9.

6.1 Current methods for multi-modal systems

One way of treating the systems that are subject to multi-modal complexity is to use several models plus a gating network that is responsible for taking the decision of which model should be used for the different modes of operation. This leads to the use of multiple model techniques. Multiple model techniques have been used in the control field to handle spatial multi-modality, temporal multi-modality and ill-posed problems [27].

6.1.1 Temporal multi-modality

Adaptive control is one of the most efficient methods which can be used to design satisfactory controllers for dynamical systems possessing a high degree of uncertainty. On the other hand standard adaptive control methods are not suitable to handle the problem of temporal multi-modality even if on line adaptation is conducted. This is because of the abrupt nature of temporal multi-modality which slows the adaptation procedure and makes it difficult to reach the required state value in short time.

One of the possible solutions to handle temporal multi-modality is through the use of multiple models. The multiple model control approach has been developed from the partitioning theory of adaptive control [68]. The successful use of multiple models in real time applications for control reconfiguration has been widely reported in the literature [108; 36; 80]. Control reconfiguration is

the control technique in which the component of failure is first detected and then switching to the controller which has been designed for that particular failure.

The multiple model for control reconfiguration consists of one model representing the no fault condition in addition to several models, one for each particular fault condition. For such uncertain systems it is assumed that the actual system under consideration can adequately be represented by a stochastic model, with an uncertain (either due to failure or due to noise component affecting the dynamic of the system) parameter vector affecting the parameters defining the structure of the system model. A finite number of parameters (Ω_i for $i = 1, \ldots, N$) are then defined by discretising the parameters of the system model Ω. Then an estimation procedure (Kalman filter has been used in all the reference papers used above in this section) is designed for each choice of parameter value, which provides an estimate for the state under the assumption that the parameter has value Ω_i. In addition to providing an estimate for the state value, each Kalman filter provides an estimate for the residual. The residual value of each filter is then used to calculate the conditional probability that the i^{th} model is correct. Based on the parameter value Ω_i of each filter, a feedback controller is designed and cascaded with the corresponding Kalman filter.

Using multiple models for control reconfiguration allows the determination of a new plant model from a set of candidate models very quickly if the plant parameters have suddenly changed. This assumes that at least one of the set of estimated models is close to the new plant mode. For the new plant mode, two methods have been suggested for calculating the control signal. In the first method the new control signal is taken to be the output of the controller with the highest conditional probability. In the second method, the new control signal is taken to be the probability weighted average of the outputs of all controllers.

The advantage of this method over the standard adaptive control method is that one model from a set of candidate models can be determined to represent the new mode of the system once change in plant parameters occurs. This ensures better transient response than the standard adaptive control method. The multiple model control approach as discussed previously can be seen as a model selection process rather than model estimation process. The disadvantage of the multiple model approach is that a large number of models are required especially if the discretisation process is required to be done at a high resolution.

To make use of the advantages of both the standard adaptive control method and the multiple control method, Narendra [92] suggested combining the two approaches. In this case, once the switching to a particular model which best represents the current plant mode occurs, adaptation for the plant and the corresponding controller can be initiated if the system mode is not accurately represented by the chosen model. In this way, a good performance is achieved even if the parameter discretisation has been done at a high granularity. Moreover, a more accurate control result has been achieved by allowing the candidate models to grow in time. In this case, in addition to the set of prelearned candidate models with parameter vectors (Ω_i for $i = 1, \ldots, N$), corresponding to N different environments, a new identification model I_{N+1} described by the parameter Ω_{N+1} and corresponding to the new environment can be learned on-line. Once a certain steady state error value has been reached and Ω_{N+1} is determined, the new model is added to the set of candidate models and is used for all the future performance of the system. Similar approach has been used in the context of model reference adaptive control to improve the transient response of linear time invariant systems for which a stability proof is also provided [91; 90].

6.1.2 Spatial multi-modality

Some functions of real world problems exhibit spatial multi-modality. For these situations, it is usually very difficult to use a single neural network to identify the nonlinear dynamics of the system over the full range of operating space. For functions with spatial multi-modality, the principle of divide and conquer has been used [27]. The main idea is to divide the operating space into a number of regions and different local models can then be used to identify each region over which the dynamics exhibit similar features. In addition to the local models a validity function (one for each local model) which is responsible for dividing experience between the different models is also required. A global model is then taken to be the weighted average of the outputs of all the local models. This method has been given different names; in [53] it is called a mixture of experts, and in [54] it is called regime decomposition.

6.1.3 ill-posed problems

Previous studies of adaptive control using neural networks [106; 2; 51], addressed the problem of how to calculate the error signal for a neural network to be used as a feed-forward controller. In supervised learning Barto [6] has shown that a neural network feed-forward controller can be trained using actual outputs and inputs data from the plant. This implies that the error for adapting the feed-forward controller should be command error rather than trajectory error (plant performance error).

Problems with inverse identification have been discussed in [6; 51; 106]. It has been shown that the major problem with the inverse identification is that they can be ill-posed in the sense that the solutions for these problems are not unique solutions. In this case, the output from the network feed-forward controller will be the conditional average of the target data, which is not necessarily an inverse.

Different neural network models have been proposed for solving ill-posed problems. In [106; 51; 60] forward and inverse modelling is suggested. In this method the forward model of the plant is firstly acquired. The feed-forward controller is then trained so as to minimise a utility function, which is taken to be the difference between the model output and the desired output of the system, $\| y - y_{ref} \|^2$. Another method which has been suggested in [60] is called feedback error learning. In feedback error learning the output of a feedback controller is used as the error for training the feed-forward controller.

However, if the conventional neural networks were unable to cope with redundant systems, they will need to be adapted every instant of time before they could produce the appropriate control command. This in turn, would produce large transient and tracking errors [34; 131].

Consequently, the use of the multiple neural network models have been suggested in the literature. Based on the principle of divide and conquer a general methodology for learning multiple models to handle complex problems has been developed. This methodology has been introduced in the architecture for supervised learning, and is called mixture of experts [53; 57; 56; 13]. The mixture of expert architecture consists of a set of function approximators called expert networks, and a gating network to specify which expert network should be activated at certain times. Each expert network is assumed to be responsible for a Gaussian region of the input space. There is one output from the gating network corresponding to each expert network. The gating network uses soft clustering of the input data and therefore, allows data to be processed by multiple experts. Training the mixture of experts is achieved by minimising the negative logarithm of the likelihood function,

$E = -\sum_n \ln \left\{ \sum_{i=1}^{M} \alpha_i(x^n)\phi_i(t^n \mid x^n) \right\}$, simultaneously with respect to the parameters of the expert networks and the gating network. Here $\phi_i(t \mid x)$ are assumed to be Gaussian functions with mean $\mu_i(x)$ and unit covariance matrices. The output of the expert network is a vector representing the mean of the Gaussian function $\mu_i(x)$, where x is the input vector. The mixing coefficients $\alpha_i(x)$ are taken to be the softmax transforms of the output of the gating network. During the training procedure the gating network learns how to split the input space into regions, where particular expert networks are specialised such that each expert network generates the outputs for input vectors falling within each region.

The mixture of experts approach has been applied to a vowel discrimination task [53] and proposed for the basal ganglia and adaptive motor control [35], where the basal ganglia are neural structures within the motor and cognitive control circuits in the mammalian forebrain and are interconnected with the neocortex by multiple loops. The mixture of experts approach has been extended to a recursively defined hierarchical mixture of experts (HME) and presented in a robotics application [56]. A set of expert networks defined over a relatively small region of input space is the result of the splitting procedure. This allows fitting a simple linear function to the data. Based on the expectation maximisation (EM) principle, a maximum likelihood learning algorithm for the HME architecture has been derived [56]. A theoretical analysis for the EM algorithm in addition to the convergence proof of the algorithm is provided in [57].

In [131], a theoretical development has been established for the use of multiple paired forward and inverse models for motor control. The forward model is assumed to make a prediction for the state based on the current state and the control command. The controller is assumed to predict the control command based on the desired state value. Nonlinear function approximators have been used to estimate the forward and inverse models. A set of forward models have been assumed and trained to learn how to partition experience by supervised learning. This means that the predictions from the n forward models are compared to the actual state value. The training of each forward model is then gated by a responsibility signal λ_t^i. The responsibility signal has been defined as the softmax transform of the errors. The same responsibility signal λ_t^i has been used to weight the learning of each controller. Two control architectures have been used to train the controller, direct inverse control in which supervised learning has been assumed and feedback error learning. A modification for the switching procedure between controllers has been proposed. This is done by allowing the controllers to switch based on purely contextual information. To achieve this purpose a third model which predicts the responsibility signal is proposed. The input to this model is taken to be the contextual and sensory feedback signals.

6.2 Conditional distribution modelling

Training a feed-forward neural network using the sum-of-squares error function leads to modelling the conditional distribution of the target data, $p(t \mid x)$, in terms of a Gaussian distribution with an input-dependent mean and an input-dependent variance as discussed in Chapter 3. However for processes with multi-modal complexity, Gaussian distributions can lead to a very poor representation of the data.

A more general network called the mixture density network has been used for representing general probability density functions $p(t \mid x)$ in parameterised form. This is achieved through a weighted sum

of density kernels $\beta_j(t \mid x)$ in the following form

$$p(t \mid x) = \sum_{j=1}^{M} \alpha_j(x)\beta_j(t \mid x), \tag{6.1}$$

where the parameters α_j are called mixing coefficients and the density kernels β_j are usually taken to be Gaussian. The mixture density network architecture is the combined structure of density model and neural network.

In its original formulation, the mixture density network was specified in terms of static systems for solving regression problems. However the formulation of the density network could easily be extended to the dynamic case. This will be presented in the next section for the purpose of this work.

6.2.1 Theory of mixture density network

For multi-valued functions, Mixture Density Networks (MDNs) [13; 74; 12; 98] provide a general framework for modelling conditional probability density functions $p(u(k)|\mathbf{s}(k))$ for the inverse mapping. Here $\mathbf{s}(k) = [y(k+d), \mathbf{x}(k)]$, where $\mathbf{x}(k) = [y(k), ..., y(k-q+1), u(k-1), ..., u(k-p+1)]$. The distribution of the outputs, $u(k)$, is described by a parametric model whose parameters are determined by the output of a neural network, which takes $\mathbf{s}(k)$ as inputs. The general conditional distribution function is given by

$$p(u(k)|\mathbf{s}(k)) = \sum_{j=1}^{M} \alpha_j(\mathbf{s}(k))\phi_j(u(k)|\mathbf{s}(k)), \tag{6.2}$$

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

$$\phi_j(u(k)|\mathbf{s}(k)) = \frac{1}{(2\pi)^{c/2}\sigma_j^c(\mathbf{s}(k))} \exp\left(-\frac{\| u(k) - \mu_j(\mathbf{s}(k)) \|^2}{2\sigma_j^2(\mathbf{s}(k))}\right), \tag{6.3}$$

where c is the dimensionality of the target data $u(k)$, $\mu_j(\mathbf{s}(k))$ 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 6.3 can approximate any given density function arbitrarily accurately providing that the mixing coefficients and the Gaussian parameters are correctly chosen [13]. It follows then that for any given value of $\mathbf{s}(k)$, the mixture model (6.2) provides a general formalism for modelling the conditional density function $p(u(k)|\mathbf{s}(k))$. To achieve this the parameters of the mixture model, namely the mixing coefficients $\alpha_j(\mathbf{s}(k))$, the means $\mu_j(\mathbf{s}(k))$ and the variances $\sigma_j^2(\mathbf{s}(k))$ are taken to be general continuous functions of $\mathbf{s}(k)$. These functions are modelled by the outputs of a feed-forward neural network that takes $\mathbf{s}(k)$ as input. This combination of a density model and a feed-forward neural network is represented schematically in Fig 6.2.

The neural network element of the MDN is implemented with a standard radial basis function network RBF of thin plate spline basis functions, or a multi-layer perceptron network for high dimensional problems. The output vector from the RBF, Z, holds the parameters that define the Gaussian

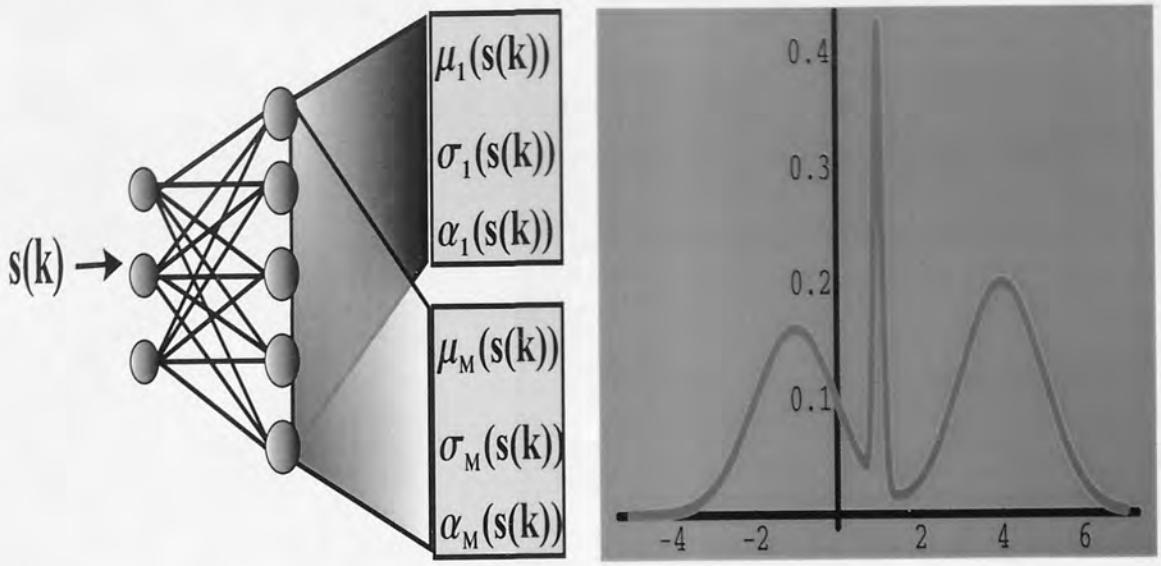


Figure 6.2: The architecture of the mixture density network.

mixture model. For M components in the mixture model 6.2 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 variance σ_j^2, 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 or 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. The constraints are such that

$$\sum_{j=1}^{M} \alpha_j(\mathbf{s}(k)) = 1, \tag{6.4}$$

$$0 \leq \alpha_j(\mathbf{s}(k)) \leq 1. \tag{6.5}$$

The first constraint ensures that the distribution is correctly normalised, so that $\int p(u(k)|\mathbf{s}(k))du(k) = 1$. These constraints can be satisfied by choosing $\alpha_j(\mathbf{s}(k))$ to be related to the network's outputs by a 'softmax' function.

$$\alpha_j(\mathbf{s}(k)) = \frac{\exp(z_j^\alpha)}{\sum_{l=1}^{M} \exp(z_l^\alpha)}. \tag{6.6}$$

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

$$\sigma_j^2 = \exp(z_j^\sigma). \tag{6.7}$$

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 network, z_{jk}^μ

$$\mu_{jk} = z_{jk}^\mu. \tag{6.8}$$

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 [13].

The likelihood of the training data set, $\{\mathbf{s}(k), u(k)\}$, can be written as

$$\begin{aligned} \mathcal{L} &= \prod_n p(\mathbf{s}_n(k), u_n(k)), \\ &= \prod_n p(u_n(k)|\mathbf{s}_n(k))p(\mathbf{s}_n(k)), \end{aligned} \tag{6.9}$$

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

$$E = -\ln \mathcal{L} = -\sum_n \ln p(u_n(k)|\mathbf{s}_n(k)) - \sum_n \ln p(\mathbf{s}_n(k)). \tag{6.10}$$

The second term in (6.10) is constant because it is independent of the network parameters, so it can be removed from the error function. The error function becomes

$$E = -\ln \mathcal{L} = -\sum_n \ln p(u_n(k)|\mathbf{s}_n(k)). \tag{6.11}$$

Next we substitute (6.2) into (6.11) and derive the negative log likelihood error function for the mixture density network

$$E = -\sum_n \ln\left\{ \sum_{j=1}^{M} \alpha_j(\mathbf{s}_n(k))\phi_j(u_n(k)|\mathbf{s}_n(k)) \right\}. \tag{6.12}$$

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

$$\pi_j(\mathbf{s}(k), u(k)) = \frac{\alpha_j \phi_j}{\sum_{l=1}^{M} \alpha_l \phi_l}, \tag{6.13}$$

the analysis of the error derivatives with respect to the network outputs is simplified. From (6.13) one can note that the posterior probabilities sum to unity

$$\sum_{j=1}^{M} \pi_j = 1. \tag{6.14}$$

Since the error function (6.12) 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

$$E = \sum_{n=1}^{N} E^n. \tag{6.15}$$

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

$$\frac{\partial E^n}{\partial z_j^{\alpha}} = \alpha_j - \pi_j, \tag{6.16}$$

$$\frac{\partial E^n}{\partial z_j^{\sigma}} = -\frac{\pi_j}{2}\left\{\frac{\| u_n(k) - \mu_j \|^2}{\sigma_j^2} - c\right\}, \tag{6.17}$$

$$\frac{\partial E^n}{\partial z_{jk}^{\mu}} = \pi_j\left\{\frac{\mu_{jk} - u_k(k)}{\sigma_j^2}\right\}. \tag{6.18}$$

The full derivation is presented in Appendix E. 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. Assuming that the component kernels of the density function are not too strongly overlapping, then 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 $\max_j\{\alpha_j(\mathbf{s}(k))/\sigma_j^c(\mathbf{s}(k))\}$. Otherwise, the maximum of the conditional density $p(u(k) \mid \mathbf{s}(k))$ needs to be computed by a nonlinear optimisation method. Alternatively, the most probable output value corresponding to the most probable branch can be calculated, since each component of the mixture model is normalised, $\int \phi_j(u(k) \mid \mathbf{s}(k))du(k) = 1$. Therefore, the most probable branch is given by

$$\arg\max_j\{\alpha_j(\mathbf{s}(k))\}. \tag{6.19}$$

The required value of $u(k)$ 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.

6.3 Neural network developments for incorporating uncertainty for non-Gaussian distributions

Since the proposed sampling algorithm for the Gaussian function has been covered in Chapters 4, and 5, we summarise here the main steps to apply the same algorithm considering sampling from a more general distribution. The full architecture of the proposed sampling method is shown in Figure 6.3

1. An accurate model of the process needs to be constructed based on the pre-collected input-output data, and to be trained off-line. In the general case, it is assumed to be described by the following neural network model:

$$\hat{y}(k+d) = N_f(u(k), \mathbf{x}(k)). \tag{6.20}$$

2. The conditional distribution of the inverse model of the plant should also be constructed. It is assumed to be described by a mixture density network given by Eq. (6.2).

3. 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

$$\arg\max_j\{\alpha_j(\mathbf{s}(k))\}. \tag{6.21}$$

4. For the sampling approach the following steps need to be carried out at each instant of time, k:

   (a) 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.

   (b) Calculate the components μ_{jk} of the kernel centres μ_j, and the kernel width σ_j of each kernel function, based on the desired output value.

   (c) The admissible values of the control signal for the mixture density network, are then assumed to be sampled from a mixture density network. Since Gaussian kernel functions are used, the samples can be generated from each kernel function randomly using the retrieved components μ_{jk} of the kernel centres μ_j, and the kernel width σ_j of each kernel function. A component to generate samples from is firstly chosen. The covariance matrix for that component is then constructed using the retrieved component parameters. Finally the samples are generated using a random number generator. The number of samples from each component is determined randomly with more samples generated from the component with larger prior. For more details see [87].

   (d) 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.

$$J(k) = \min_{u\in U} E_{\bar{v}}[(\hat{y}(k+d) - y_{ref}(k+d))^2], \tag{6.22}$$

   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 $\bar{v}$. 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.

6.4 Synthetic example

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

$$y(k) = u(k) + 0.3\sin(2\pi u(k)) + \epsilon, \tag{6.23}$$

where ϵ is a random variable with uniform distribution in the interval $(-0.1, 0.1)$, $y(k)$ is the output variable, and $u(k)$ is the input variable. This example has been used in [87; 13] to demonstrate the use of the Mixture Density Network. Equation (6.23) represents a static system, since no delay exists

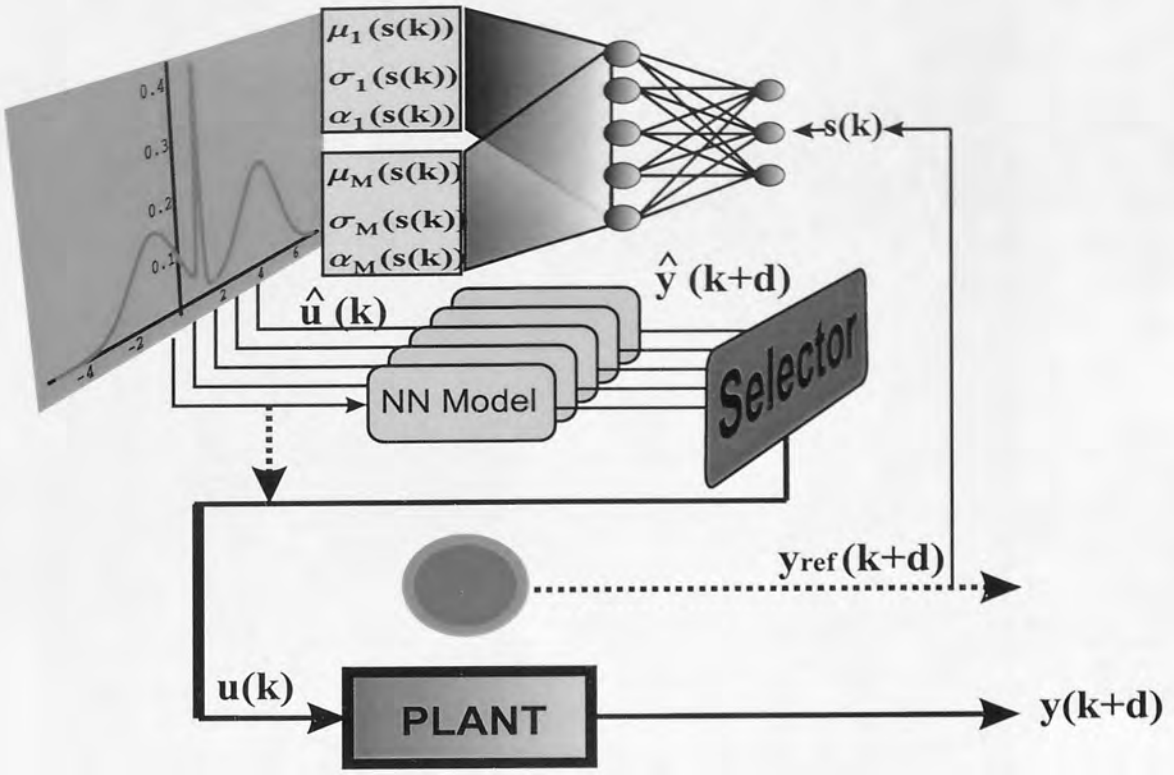


Figure 6.3: The architecture of the proposed sampling method from Mixture Density Network. The inputs to the neural network of the inverse model are $\mathbf{s}(k)$, the outputs are the parameters of the mixture density model.

between the input and the output variables. The plant has been considered to be given by Eq. (6.23). In order to identify the plant, an input-output model described by $\hat{y}(k) = f(u(k))$ was chosen, where f is a thin plate spline radial basis function network. Figure 6.4 shows a data set of 300 points generated by sampling Eq. (6.23). 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. In the cross validation the model with the minimum difference between the actual output of the system and the neural network output, $e = \| y(k) - \hat{y}(k) \|^2$, is taken to be the best model. It was trained using the scaled conjugate gradient method. The error values in the cross validation stage are given in Table 6.1. 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.

Number of hidden units	1	2	3	4	5
Error value	0.0178	0.0170	1.4699×10^{-4}	3.6338×10^{-4}	7.0120×10^{-5}
Number of hidden units	6	7	8	9	10
Error value	1.474×10^{-4}	2.6237×10^{-4}	1.4149×10^{-4}	4.3071×10^{-4}	1.9413×10^{-4}

Table 6.1: Error values of the cross validation for the forward model of the sine function.

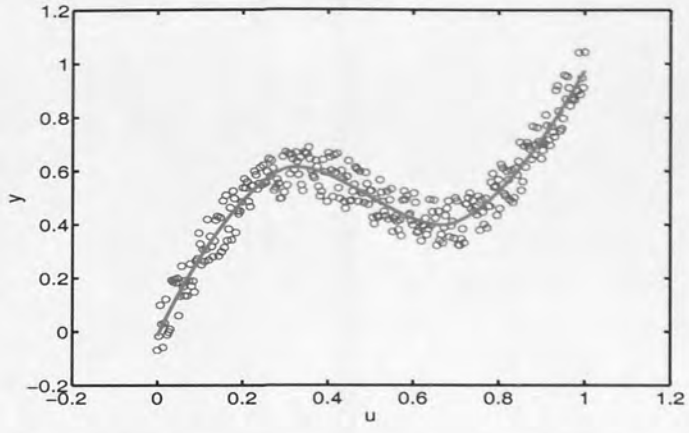


Figure 6.4: The forward model of the function $y(k) = u(k) + 0.3\sin(2\pi u(k)) + \epsilon$. The circles represent the samples generated from that function. The solid curve shows the result of training a thin plate spline radial basis function with 5 basis functions using a sum of square error function.

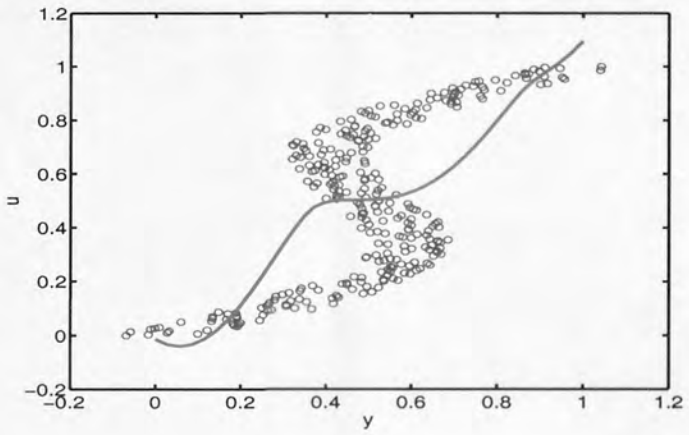


Figure 6.5: The inverse model of the function $y(k) = u(k) + 0.3\sin(2\pi u(k)) + \epsilon$. The circles represent the same data as in Fig 6.4. The solid curve shows the result of training a thin plate spline radial basis function with 15 basis functions using a sum of square error function.

6.4.1 Gaussian distribution model

We consider acquiring 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. Similarly an input-output model described by $\hat{u}(k) = f^{-1}(y(k))$ was chosen to find the inverse model of the plant. The network tries to approximate the conditional average of the target data, but this corresponds to a very poor representation of the process as can be seen from Fig 6.5. The network in this case had 15 thin plate spline basis functions and was trained using the scaled conjugate gradient optimisation method. This network has been connected in series with the plant to generate the control signal required to cause the plant to follow the desired output. The desired output has been considered to be given by $y_{ref}(k) = r(k) + 0.3\sin(2\pi r(k))$, where the input $r(k)$ has been chosen in such a way to generate data that have not been used in the training stage. The result is shown in Fig 6.6, where it can be seen that there is a large error between the desired output and the plant output.

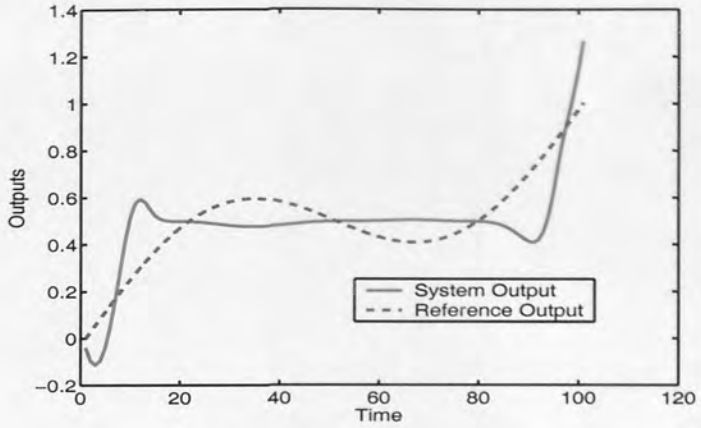


Figure 6.6: The control result from using the classical inverse controller, showing poor performance.

6.4.2 Mixture density network

In this section an MDN is applied 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 has been decided by applying the cross validation method. It was found that the best structure for the MDN consists of 7 thin plate spline basis functions with 9 outputs corresponding to 3 kernel functions. This best structure for the mixture density network is taken to be the structure which gives the minimum negative log likelihood value in the validation stage.

$$E = -\sum_{n} \ln\left\{ \sum_{j=1}^{M} \alpha_j(\mathbf{s}_n(k))\phi_j(u_n(k)|\mathbf{s}_n(k)) \right\}.$$

The MDN was trained using the scaled conjugate gradient optimisation method. The negative log likelihood values are given in Table 6.2. Once trained the MDN predicts the conditional probability density of the target data (regarded as the input to the plant u(k) in the inverse model) for each value of the input to the network (regarded as the output to the plant y(k) in the inverse model). Having obtained a good representation for the conditional density of the target data, we can in principle calculate any desired statistics from that distribution. In our control problem, since the conditional mean of the target data is a very poor result, we are interested in the evaluation of the centre of the most probable kernel according to Eq. (6.19), which gives the result shown in Fig 6.7. 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 Fig 6.8, where it can be seen that using the most probable value of the kernel functions has improved the performance of the controller significantly.

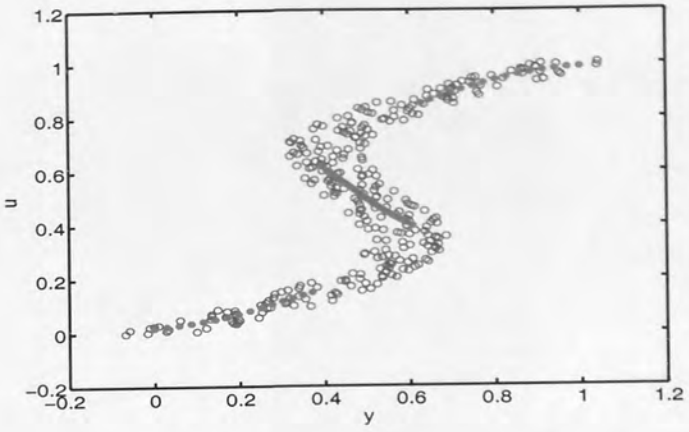


Figure 6.7: Plot of the central value of the most probable kernel as a function of y(k) from the Mixture Density Network.

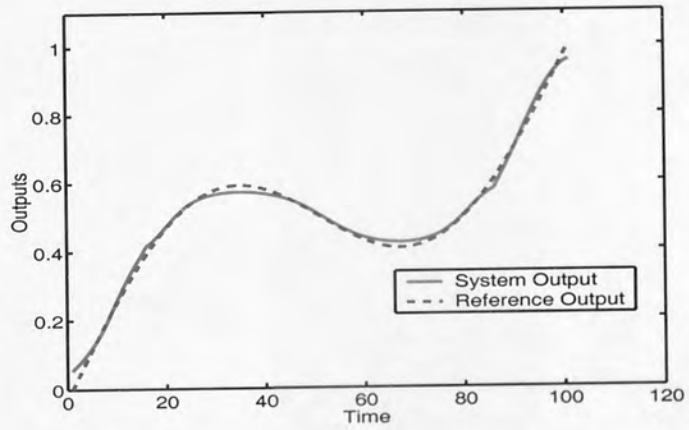


Figure 6.8: The control result from using most probable value of the Mixture Density Network as a control law, demonstrating improved performance.

	One component	Two components	Three components	Four components
One hidden unit	-5.6105	-125.3012	-150.4299	-135.0354
Two hidden units	-51.4993	-152.7491	-243.9469	-251.0456
Three hidden units	-7.2206	-143.0328	-244.6030	-230.1617
Four hidden units	-57.5423	-163.9040	-250.1389	-240.9337
Five hidden units	-59.3630	-154.7906	-244.1513	-250.9633
Six hidden units	-62.9161	-132.0105	-249.7488	-245.4352
Seven hidden units	-69.0090	-167.9405	-255.1973	-247.6768
Eight hidden units	-21.8316	-180.4569	-244.5906	-250.0194
Nine hidden units	-57.8991	-196.0582	-249.2017	-242.3674
Ten hidden units	-67.3336	-179.9004	-248.6118	-247.6496

Table 6.2: The negative log likelihood values of the cross validation for the inverse model of the sine function.

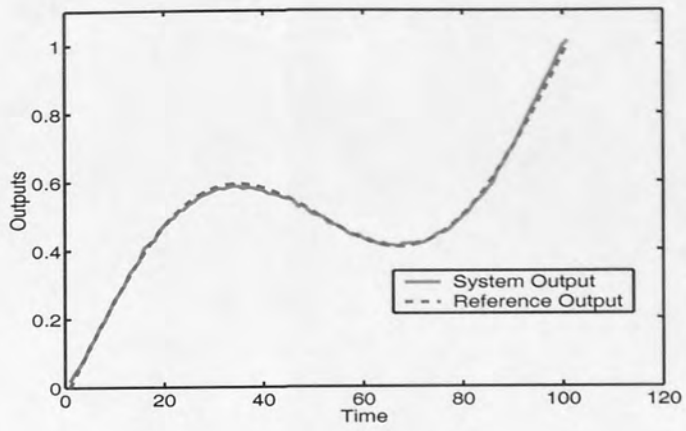


Figure 6.9: The control result from applying the proposed sampling approach from the mixture density network, demonstrating accurate tracking.

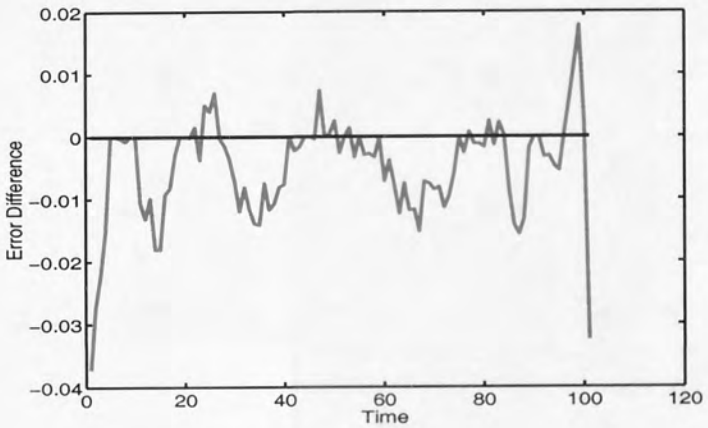


Figure 6.10: The Error Difference.

6.4.3 Proposed control approach

The final experiment that we have performed, is to sample from the control signal distribution (from the mixture density distribution). In the new proposed control approach the best control signal was found and forwarded to the plant, following the procedure presented earlier. Again 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 Fig 6.9. It can be seen from this figure that the proposed sampling approach performs even better than 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 proposed sampling approach, and the most probable value of the kernel function in the mixture density network is shown in Fig 6.10. From this figure one can see that the sampling approach has reduced the error significantly.

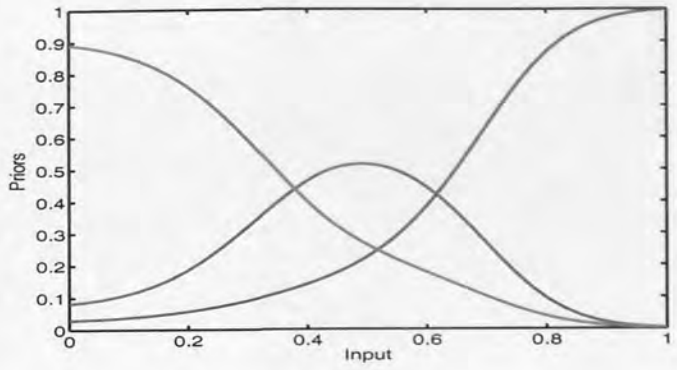


Figure 6.11: Plot of the priors $\alpha_j(\mathbf{s}(k))$ as a function of the input, $\mathbf{s}(k)$ for the three kernel function used in modeling the conditional distribution of the inverse of the sine function.

6.5 Outputs of the neural network in mixture density network architecture

It has been shown in Section 6.2.1 that the outputs of the neural network (the parameters of the kernel functions), are continuous single valued functions of the input variables. Although the mixture density network provides an estimation for the conditional distribution of the target data for problems where the mapping is multi-valued, the mixture density network is also able to produce a conditional density which is unimodal for some values of the input and multi-valued for other values. This can be achieved in the mixture density network by modulating the amplitudes of the mixing coefficients $\alpha_j(\mathbf{s}(k))$. This is demonstrated in Fig 6.11 which shows the plot for the three mixing coefficients as a function of the input for the static SISO sine function.

It can be seen that there is one significant kernel function at the input values $u(k) \leq 0.2$ and $u(k) \geq 0.8$. However, all three kernels have significant priors for input value equals to 0.5.

Moreover, the probability density function for the target data of the SISO sine function at different instant of times is shown in Fig 6.12. Figure 6.12.a shows the probability density function of the local models at time instant equals to 10s. It can be seen that two of the components in the mixture model have significant prior values. According to the mixture density network the centre of the component with the highest prior is taken to represent the output from the mixture density network; this is denoted by the $\times$ mark with an arrow points toward it in the figure. It is clear that a good estimation for the control signal could be obtained at this particular instant of time.

Time	prior1	prior2	prior3	centre1	centre2	centre3	σ_1^2	σ_2^2	σ_3^2	control signal
10	0.0709	0.247	0.6821	0.6507	0.7369	0.0925	0.0038	0.0047	0.0005	0.09
30	0.3434	0.4655	0.1912	0.8219	0.4163	0.2630	0.0011	0.0045	0.0017	0.2900
70	0.1473	0.4743	0.3783	0.7209	0.5962	0.1677	0.0022	0.0048	0.0010	0.6900
100	0.9921	0.0049	0.003	0.9821	-0.1024	0.3602	0.0003	0.0073	0.0339	0.9900

Table 6.3: Parameters of the kernel functions in the mixture density network for the SISO static sine function at different instants of time.

The centres of the kernel functions at time instant 30s is shown in Fig 6.12.b. Here the three

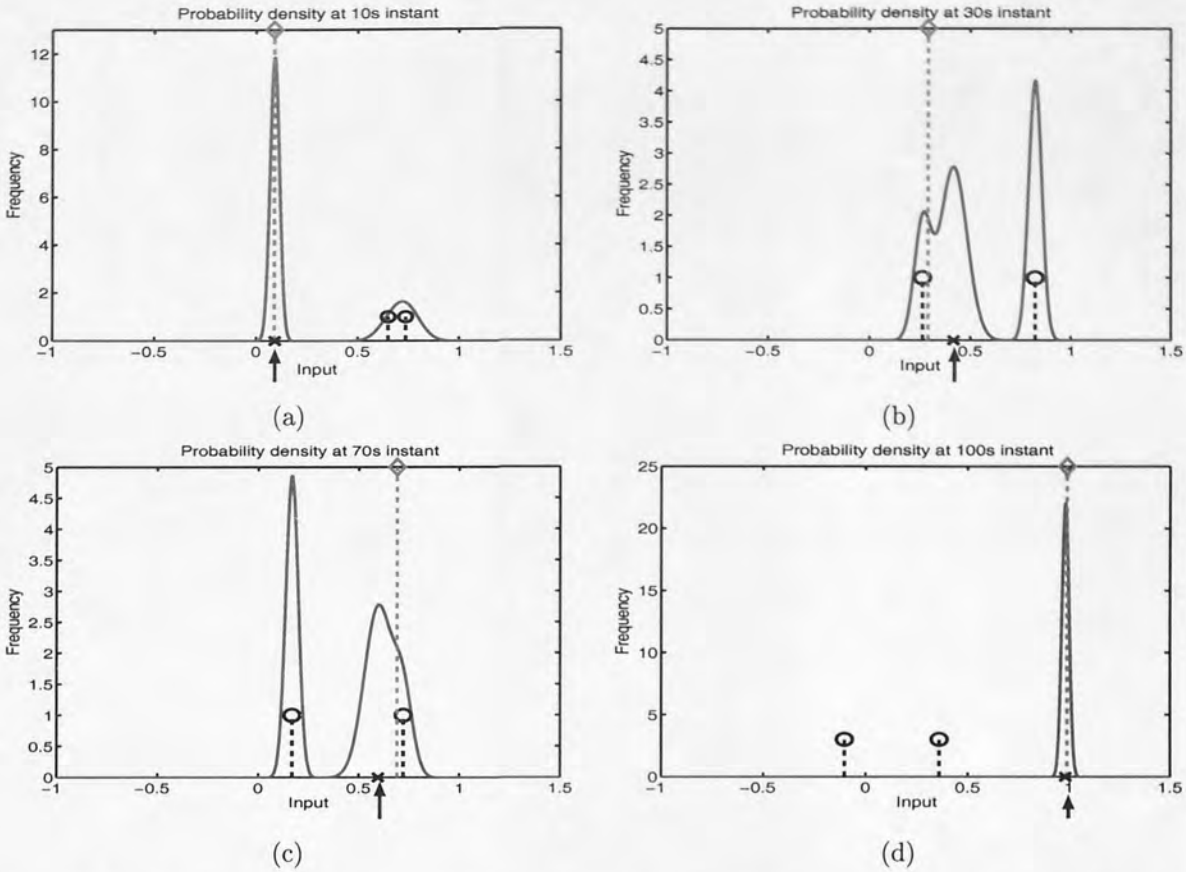


Figure 6.12: The plot of the conditional probability distribution at different instants of time for the SISO static sine function (shown by the solid curve). The centre of the component with the highest prior is plotted as the × with an arrow points toward it. The centres of the two other components are plotted by a line terminated with a circle. The actual value of the control signal however, is plotted as a dashed line terminated with ◊. These plots are for a mixture density network with 3 kernels and 7 hidden units in the RBF network: (a) Probability density function at 10s instant of time. (b) Probability density function at 30s instant of time. (c) Probability density function at 70s instant of time. (d) Probability density function at 100s instant of time.

kernel functions have shown a contribution towards the output of the mixture model. The component with the highest prior value is found to be component 2 and its centre is plotted as × with an arrow points toward it in the figure. The actual value of the control signal is found to be slightly far from the predicted value of the mixture model. The same can be said about the density function at time instant 70s. Here also the three components of the mixture model are found to have significant prior values. A good prediction for the control signal could be produced from the mixture model at this time.

At time 100s only one component of the mixture model was found to have a significant prior value. A very accurate prediction from the mixture model could also be produced. To summarise and for easy comparison, the parameters of the kernel functions, in addition to the actual control signal are given in Table 6.3.

6.6 Probability distribution modelling for MIMO dynamical system

In this section, the MIMO dynamical system of Section 5.5 described by the state equations

$$\begin{aligned}
x_1(k+1) &= 0.9x_1(k)\sin[x_2(k)] + \left[2 + 1.5\frac{x_1(k)u_1(k)}{1+x_1^2(k)u_1^2(k)}\right]u_1(k) + \left[x_1(k) + \frac{2x_1(k)}{1+x_1^2(k)}\right]u_2(k),\\
x_2(k+1) &= x_3(k)\{1+\sin[4x_3(k)]\} + \frac{x_3(k)}{1+x_3^2(k)},\\
x_3(k+1) &= \{3+\sin[2x_1(k)]\}u_2(k),\\
y_1(k) &= x_1(k),\\
y_2(k) &= x_2(k),
\end{aligned} \tag{6.24}$$

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

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

where $\mathbf{s}(k) = [\mathbf{y}(k+d), \mathbf{x}(k)]$, and $\mathbf{x}(k) = [\mathbf{y}(k), \mathbf{y}(k-1), \mathbf{y}(k-2), \mathbf{u}(k-1), \mathbf{u}(k-2)]$ is the same input vector as in Section 5.5. The same training data as for the Gaussian case in Section 5.5 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 optimal structure for the inverse model is a mixture density network with 2 components and 7 hidden units in the the multi-layer perceptron network.

However, the forward models for the first and the second outputs of the plant remain as before, modelled with a standard multi-layer perceptron network.

6.6.1 Mixture density network for MIMO system

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 follows the desired output. The same reference signals, $r_1(k)$ and $r_2(k)$, that are used in Section 5.5.2 are used here

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

The control signal, $u(k)$, from the mixture density network is taken to be the mean, $\mu_j(\mathbf{s}(k))$, 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 6.13. The average tracking error of the standard inverse control and the mixture density network is given in Table 6.4.

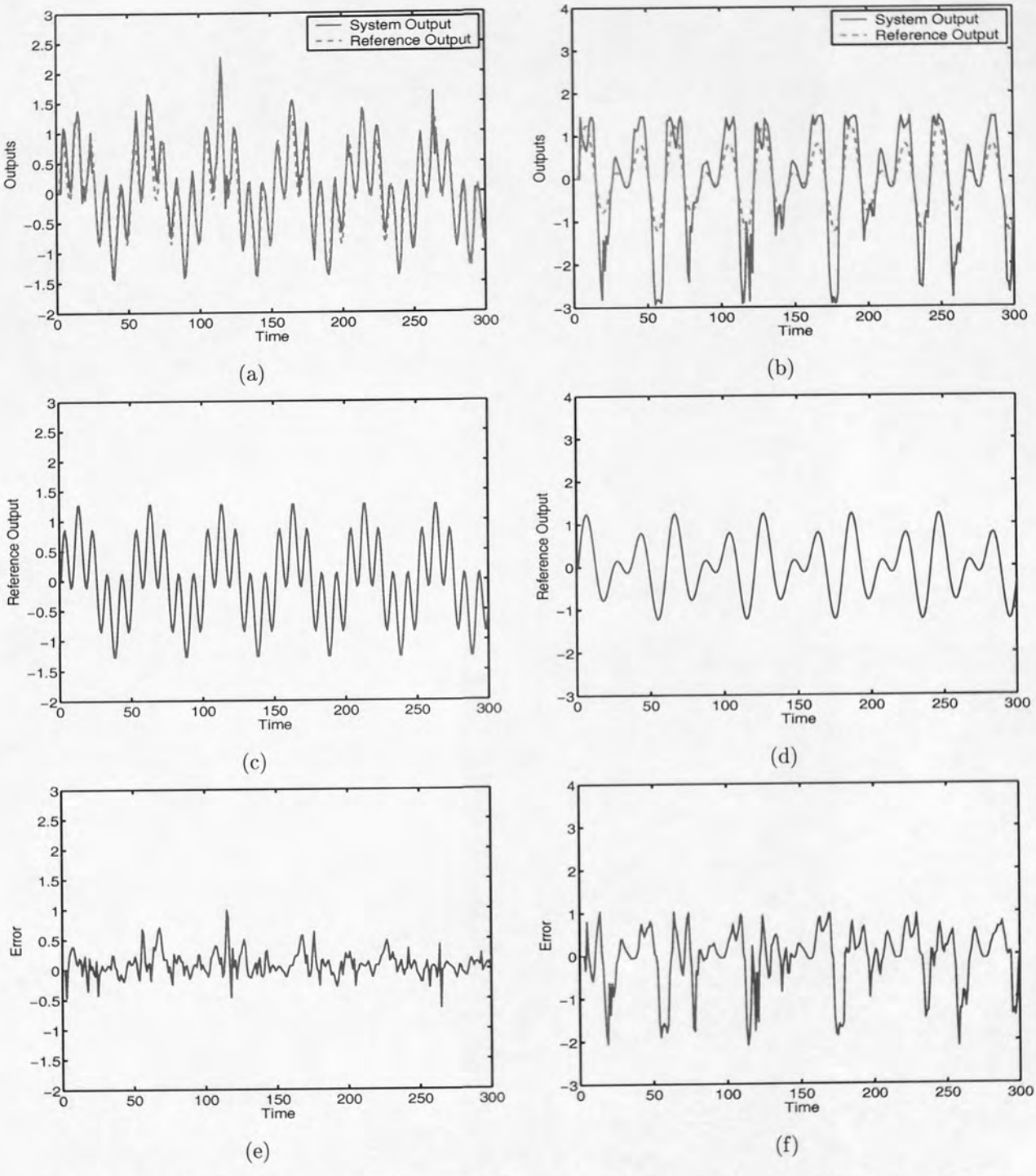


Figure 6.13: Control result using the most likely output of the mixture density network: (a) the actual and reference model outputs of the first output of the plant. (b) the actual and reference model outputs of the second output of the plant. (c) the reference model output for the first output of the plant. (d) the reference model output for the second output of the plant. (e) the tracking error of the first output of the plant. (f) the tracking error of the second output of the plant.

6.6.2 Sampling from conditional distribution of MIMO system

In this section the sampling method from an mixture density network for the MIMO control system is presented. Here the variances of the kernel functions in addition to the means are retrieved and used in the sampling method. Since each kernel is a Gaussian function, the random number generator 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 6.14. The number of samples used to search for the optimal control signal is taken to be 400. 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. The overall average tracking error values is given in Table 6.4 for comparison.

Standard inverse	Gaussian function and sampling	Mixture density	Mixture density and sampling
0.5711	0.0906	0.5350	0.0759

Table 6.4: The tracking error of the MIMO control system from the standard inverse and the mixture density networks.

6.7 Probability density function for the MIMO dynamical system

The mixture density network model can usually produce a conditional density which is unimodal for some values of the input, in which case only one of the kernels in the mixture density network will have a prior probability which differs significantly from zero, and multi-valued for other values of the input, here the kernels in the mixture density network will have comparable prior values [13].

In this section the prior values, the variances and the centres of each kernel function in the mixture density network of the MIMO dynamical system will be given at different instants of time to analyse whether the control signals distribution is multi-modal or not. In addition a plot for the probability densities of the inverse model will be provided at different instants of time.

Figure 6.15 shows the distribution $p(\mathbf{u}(k) \mid \mathbf{s}(k))$ resulting from using two kernel functions to model the conditional distribution of control signals, at 10, 120, 200, and 275 seconds. The values for the centres, the variances and the priors of each kernel function at these times are given in Table 6.5.

From Fig 6.15.a and Table 6.5, it is clear that the conditional probability density of the target data at time 10s is unimodal. The first component of the mixture density function is the dominant one with prior value equal to 0.9409. At time instant 120s the two components have significant prior values, with the centres of the two kernels being very close to each other. At time instant 200s, the second kernel is shown to be the dominant function with prior value equal to 0.8155. Figure 6.15.c shows the two kernel functions where the first kernel has the peak value centred at $[-0.1955, -0.3681]$ and the second kernel centred at $[-0.2833, -0.5738]$. However, at time instant 275 the system again is seen to be bimodal with the two kernel functions having significant prior values different from zero.

Unfortunately the true values for the target data cannot be provided (the values of the first and the second control signals), since this depends on the previous value of the state and the desired value of the system at each instant of time.

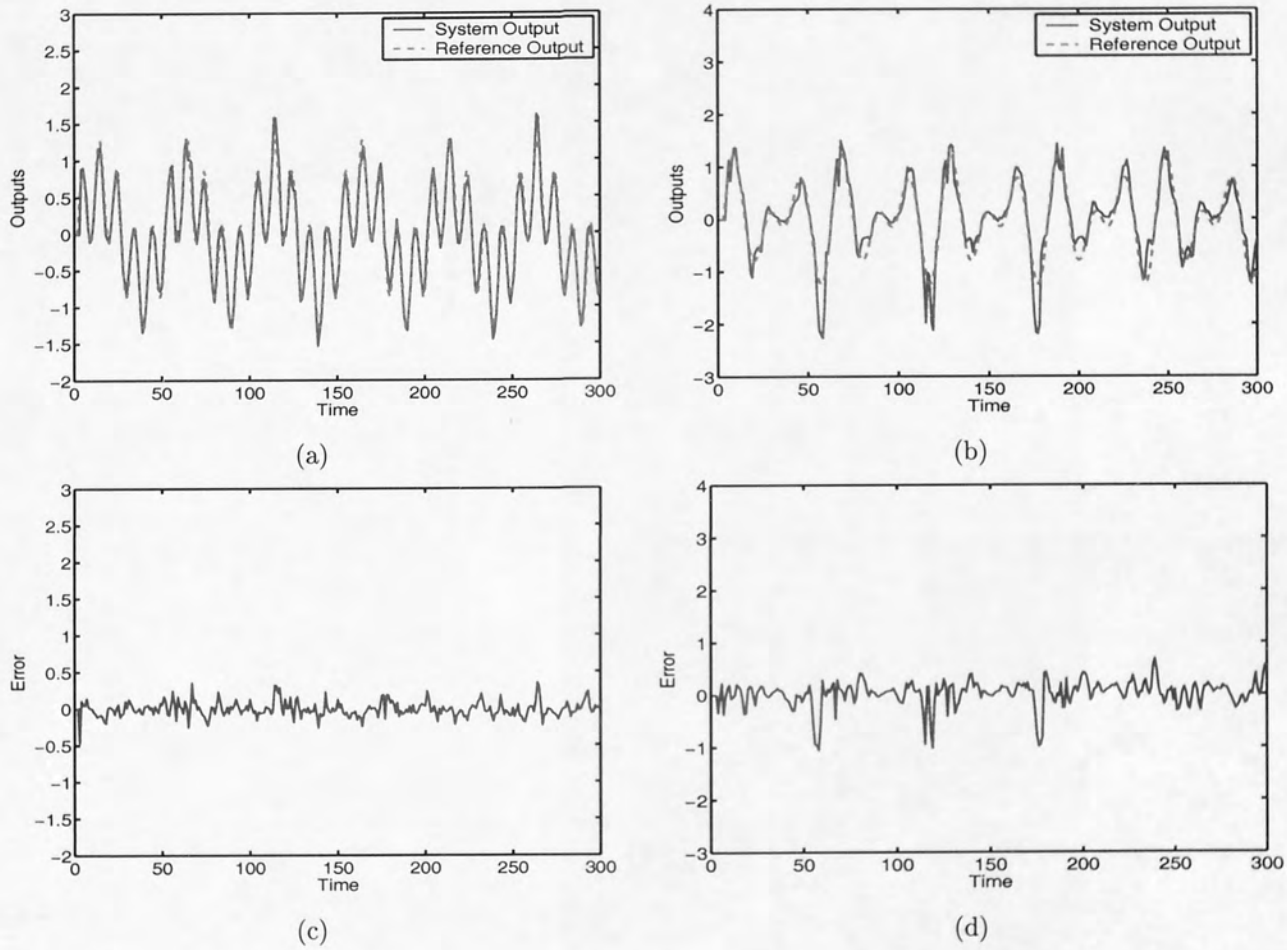


Figure 6.14: Performance of the proposed control approach of mixture density network for the dynamical MIMO system: (a) the actual and reference model outputs of the first output of the plant. (b) the actual and reference model outputs of the second output of the plant. (c) the tracking error of the first output of the plant. (d) the tracking error of the second output of the plant.

Time	prior1	prior2	centre1	centre2	σ_1^2	σ_2^2
10	0.9409	0.0591	[0.1229 0.4337]	[0.4575 0.8510]	0.0431	0.0119
120	0.3295	0.6705	[0.0057 -0.2200]	[0.0511 -0.3229]	0.002	0.0234
200	0.1845	0.8155	[-0.1955 -0.3681]	[-0.2833 -0.5738]	0.0014	0.0250
275	0.5296	0.4704	[0.1313 -0.0881]	[0.2657 -0.0797]	0.0038	0.0186

Table 6.5: Parameters of the kernel functions in the mixture density network for the MIMO dynamical system at different instant of times.

6.8 Discussion on sampling

In this section the problem of determining an appropriate number of samples is considered. In addition, a comparison of sampling from different distributions such as Gaussian, non-Gaussians, and moreover uniform will be considered.

Firstly several experiments have been performed with different number of samples from the Gaussian distribution. The dashed curve in Figure 6.16 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.

The result of determining the optimal control law by sampling from a non-Gaussian function and for different numbers of samples is given by the solid curve in Figure 6.16. A slight 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 6.16 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 cases.

6.9 Discussion

General inverse control can be considered to be a good control strategy if the model of the plant happens to be invertible and accurate. The main contribution of this chapter is that it extends the importance sampling approach from a Gaussian function by considering sampling from an arbitrary probability distribution function. Simulation experiments demonstrated the successful application of the importance sampling strategy from an arbitrary probability function to improve the controller performance for a class of nonlinear single input single output static systems in which the inverse

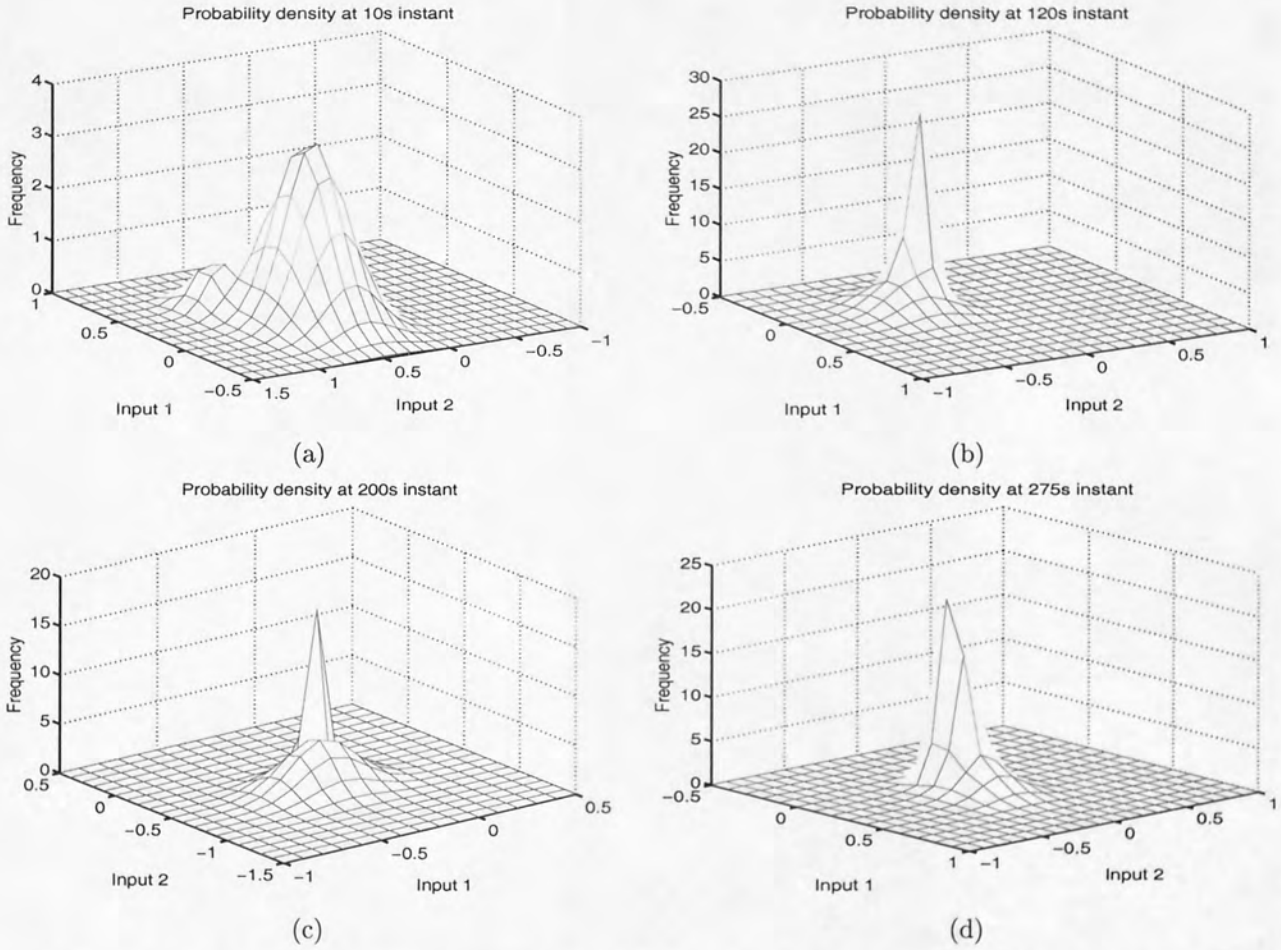


Figure 6.15: The plot for conditional probability distributions at different instants of time for the MIMO dynamical system. These plots are for a mixture density network with 2 kernels and 7 hidden units in the multi layer perceptron network: (a) Probability density function at 10s instant of time. (b) Probability density function at 120s instant of time. (c) Probability density function at 200s instant of time. (d) Probability density function at 275s instant of time

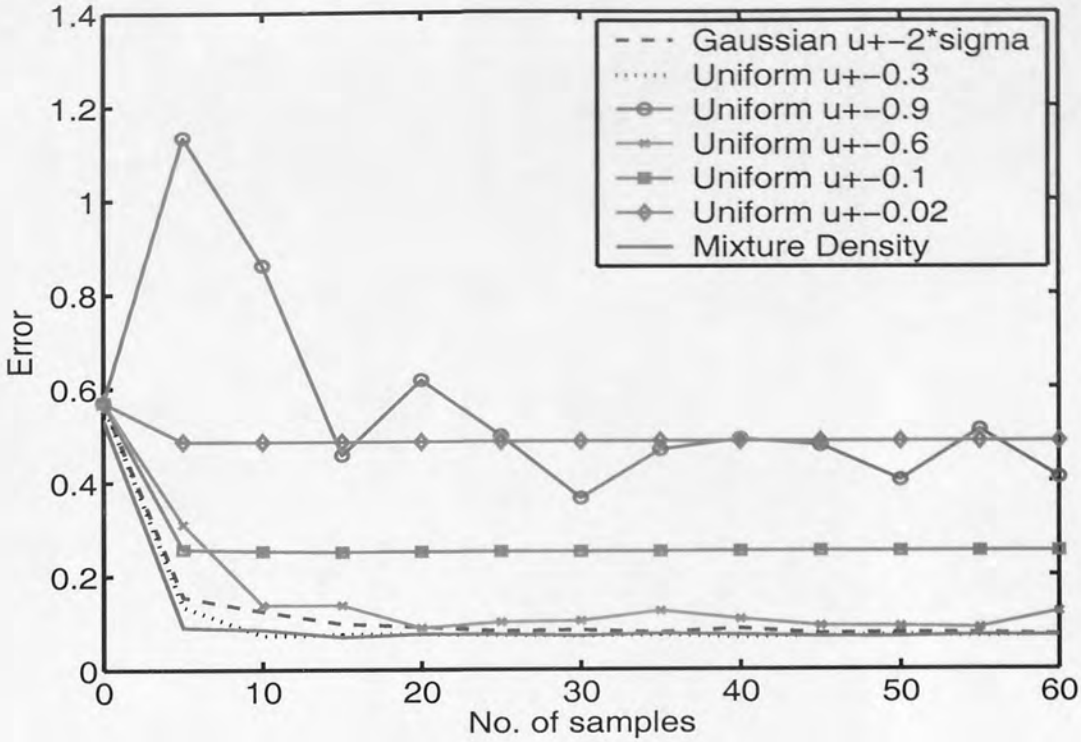


Figure 6.16: The error plot versus the number of samples for different kinds of distribution.

mapping is multi-valued. In addition, the use of the mixture density network to model the conditional distribution of control signals for a MIMO dynamical system was presented.

A comparison of sampling from different distributions of control signals is also provided. It has been shown that sampling from the right distribution provides the best result. In addition, it has been shown that the possibility of providing an estimate for the variance around the control signals should avoid the trial and error to estimate the upper and lower bounds of control signals and avoid the stability problems that could result from sampling from the wrong distribution.

The examples given in this chapter demonstrates 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.

Chapter 7

Incorporating Uncertainty in the Forward Model

In controlling real world problems, a certain amount of information on the plant needs to be provided or estimated. Mainly: (1) the desired response of the plant which can be provided directly in terms of the desired output of the plant, (2) constraints and characteristics of controllers such as the possibility of using linear controllers only, or when there is limitation on the total energy available for control purposes, (3) permissible interaction between the controller and the plant, such as the measurement equipment to be used, and (4) the plant equations which describe the dynamical characteristics of the physical system to be controlled. Moreover, a performance index which is necessary for calculating the optimal control law should be chosen.

In the intelligent control field, several control methods such as the direct inverse control, direct and indirect adaptive control, and the adaptive critic have been developed to obtain optimal control laws given the necessary information for the problem.

However, all these methods assume perfect information for the control problems to be solved. As an example, the indirect adaptive control and the adaptive critic methods assume the availability of accurate equations for the plant to be controlled.

The assumption of perfect information about the plant to be controlled however, can never be true in reality. This requires developing a new theory of control which allows firstly estimating the uncertainty of the plant models and then using uncertainty estimation to obtain a better control law.

Chapter 4 provided a new methodology for incorporating the estimated uncertainty in the controller (inverse model) to search for a better control law. The developed method in Chapter 4, assumed that the predicted output from the controller is uncertain, therefore, sampling from the distribution of control signals is suggested. To evaluate the effect of different control values on the system, a forward model of the plant is used. However, only the expected value of the forward model is considered when the samples from the inverse controller propagated through the forward model. Taking only the expected value of the forward model means ignoring the uncertainty in the forward model.

This chapter is intended to extend the proposed sampling method in Chapter 4 by allowing the uncertainty in the forward model to be incorporated when searching for a better control law.

To make the necessary link between the improved sampling method discussed in this chapter and the original sampling method which ignores the uncertainty in the forward method, we start the chapter by discussing in detail the optimal control law for a simple linear example, section 7.1. The

problem of uncertainty estimation for the forward model is presented in section 7.2. Incorporating the uncertainty of the forward model in the sampling approach is discussed in section 7.3. Section 7.4 discusses the algorithm for calculating the optimal control law. Simulation experiments to demonstrate theoretical development for incorporating uncertainty of the forward model in the sampling method are provided in section 7.5. Chapter conclusions are given in section 7.6.

7.1 Preliminary examples

Before introducing the detailed formulation for incorporating uncertainty in the forward model, a very simple example of optimal control problem is discussed in this section. The plant to be considered is described by the first order scalar difference equation

$$y(k+1) = ay(k) + bu(k). \tag{7.1}$$

The criterion function is taken to be

$$J(k) = \| y(k+1) - y_{ref}(k+1) \|^2 . \tag{7.2}$$

That is, an adaptive control problem where the system output should follow a predefined desired output is going to be considered. For the purpose of differentiating between different control policies under different assumptions of the system to be controlled, a deterministic system is firstly discussed in example 1. In example 2, the optimal control law where random disturbances affect the plant output will be derived. Example 3 is concerned with deriving the optimal control law where a random disturbance is added to the plant output. The optimal control law for systems with input dependent noise is discussed in example 4.

7.1.1 Example 1, deterministic control system

Here a deterministic control system described by the difference equation (7.1) is considered. The coefficients a and b of the system equation (7.1) are assumed to be known and observed exactly.

The optimal adaptive control law can be derived directly for the deterministic system. Since

$$\begin{aligned} J(k) &= \| y(k+1) - y_{ref}(k+1) \|^2, \\ &= \| ay(k) + bu(k) - y_{ref}(k+1) \|^2 . \end{aligned} \tag{7.3}$$

Taking the derivative of equation (7.3) with respect to the control signal $u(k)$ and setting the derivative equal to zero yields the optimal control law

$$u(k) = \frac{y_{ref}(k+1) - ay(k)}{b}. \tag{7.4}$$

Indeed the optimal control policy given in (7.4) is equivalent to the control policy of the proposed sampling method in Section 4.5 for the deterministic control problems. In Chapter 4 the optimal control policy in the proposed sampling method is derived based on the assumption that the forward model of the plant is accurate. Ignoring the uncertainty in the forward model and considering the uncertainty in the inverse model, sampling from the controller is proposed for searching a better control value. The performance criterion in the proposed sampling method is assumed to be given by (7.2).

To see the effect of the control values sampled from the distribution of control signals, an accurate forward model is assumed to be available. The forward model is taken to be a neural network which predicts the mean value of the output of the system. Although, the uncertainty in the forward model can be estimated, the assumption in Chapter 4 was to take the predicted mean value from the neural network as an accurate prediction for the system output.

The effect of ignoring the uncertainty in the forward model will be considered in further detail in example 4. It will be shown that a better control law can be found when considering the uncertainty, if there is any, in the forward model.

7.1.2 Example 2, stochastic control system: System with unknown noise

In this example the system in which the present value of the output is known, but the future value is affected by random forces is considered. In this case a random force, ξ is affecting the predicted output of the system. The system equation is given by

$$y(k+1) = ay(k) + bu(k) + \xi(k). \tag{7.5}$$

The probability distribution of the random variable ξ is assumed to be $p[\xi(k)]$. The random variables are assumed to be independent from one time instant to the next.

The predicted output $y(k+1)$, cannot be determined completely from the current output $y(k)$ and the current control $u(k)$, only the probability distribution for the predicted output can be determined. Therefore, the expected value of the performance index needs to be optimised rather than the performance index itself

$$J(k) = \underset{\xi}{E}[\| y(k+1) - y_{ref}(k+1) \|^2]. \tag{7.6}$$

To evaluate the optimal control policy, the random variable needs to be quantised to obtain a discrete probability distribution. For a given output and a given control each quantised value of ξ is substituted and then multiplied by its probability value. The expected value of the cost function can then be obtained by adding the quantities resulting from each quantised random variable. Given that the quantised values of $\xi(k)$ are $\xi^q(k)$,$q = 1, 2, 3, ..., Q$, the expected value is then given by

$$\begin{aligned} J(k) &= \underset{\xi}{E}[\| y(k+1) - y_{ref}(k+1) \|^2], \\ J(k) &= \sum_{q=1}^{Q} [(ay(k) + bu(k) + \xi^q(k) - y_{ref}(k+1))^2] p(\xi^q(k)). \end{aligned} \tag{7.7}$$

Differentiating (7.7) with respect to $u(k)$, the optimal control law can then be given by

$$u(k) = \frac{[y_{ref}(k+1) - ay(k)] \sum_{q=1}^{Q} p(\xi^q(k)) - \sum_{q=1}^{Q} \xi^q(k) p(\xi^q(k))}{b \sum_{q=1}^{Q} p(\xi^q(k))}. \tag{7.8}$$

The control law derived in (7.8) is equivalent to the control law derived in Section 4.5, equation (4.21) for the stochastic control problems which are driven by a stochastic component. Again, this control law assumes that the system equation is accurate and do not account for the uncertainty in the forward model. The possibility of including the uncertainty in the forward model in deriving the optimal control law for deterministic and stochastic system is discussed in section 7.1.4.

7.1.3 Example 3, stochastic control system: System with additive plant noise

The system to be considered in this example is the control system described by the difference equation (7.1), with a random disturbances added to the plant output:

$$y(k+1) = ay(k) + bu(k) + \xi(k), \tag{7.9}$$

where $\xi(k)$ are independent with

$$E(\xi(k)) = 0, \tag{7.10}$$

$$E(\xi^2(k)) = \sigma^2(k). \tag{7.11}$$

The criterion function is still the same given by equation (7.2). Since the system output is a random variable now, an optimal control policy is a control policy which minimises the expected value of the performance index J, $E(J)$ [3]. Defining the utility function as

$$M(k) = ay(k) + bu(k) + \xi(k) - y_{ref}(k+1), \tag{7.12}$$

the expected value of the performance index is given by

$$\begin{aligned} E[J(k)] &= E[\| (y(k+1) - y_{ref}(k+1)) \|^2], \\ &= E[\| ay(k) + bu(k) + \xi(k) - y_{ref}(k+1) \|^2], \\ &= \| ay(k) + bu(k) - y_{ref}(k+1) \|^2 + \sigma^2(k), \end{aligned} \tag{7.13}$$

since

$$E[M(k) \mid u(k), y(k)] = ay(k) + bu(k) - y_{ref}(k+1), \tag{7.14}$$

and

$$\text{var}[M(k) \mid u(k), y(k)] = \sigma^2(k). \tag{7.15}$$

Equations (7.14) and (7.15) can be obtained directly from the distribution of the random variable $\xi(k)$. This can be achieved by setting $\xi(k) = M(k) - ay(k) - bu(k) + y_{ref}(k+1)$

Proceeding in the same way as example 1, but by minimising the expected value of $J(k)$, the optimal control law is given by

$$u(k) = \frac{y_{ref}(k+1) - ay(k)}{b}, \tag{7.16}$$

since $\sigma^2(k)$ is constant independent of the control variable $u(k)$. Comparing (7.4) and (7.16), the optimal control policy for the deterministic and stochastic systems is the same. This is due to the fact that the mean of the random variable $\xi(k)$ is zero. This means that the deterministic control system can be obtained from the stochastic control system by replacing $\xi(k)$ by its mean. This is called the certainty equivalence principle to the system [3].

However, the control law derived in equation (7.16) is optimal if the variance $\sigma^2(k)$ of the random variable is independent of the control variable $u(k)$. In example 4, the control law for systems where the variance of the random variable $\xi(k)$ is dependent on the input $u(k)$ will be derived.

7.1.4 Example 4, Stochastic control system: system with input dependent output noise

As another example of stochastic nature, consider the stochastic control system with input dependent output noise

$$y(k+1) = ay(k) + bu(k) + \xi(u(k), y(k)), \tag{7.17}$$

where $\xi(u(k), y(k))$ is an independently identically distributed random disturbance added to the system output, $y(k+1)$ with

$$E[\xi(k+1)] = 0, \tag{7.18}$$

$$E[\xi^2(k+1)] = \sigma^2(k+1), \tag{7.19}$$

and where $\sigma^2(k+1)$ is input dependent and given by the difference equation $\sigma^2(k+1) = cy(k)+du(k)$.

The formulation in (7.17) is equivalent to the case where a neural network is used to model the system equations. Two outputs from the neural network can be predicted. The first one is the mean value, $\hat{y}(k+1)$, of the target data $y(k+1)$, and the second output is the variance, $\sigma^2_{y(k+1)}$ of the target data.

In the developed methods such as the indirect adaptive control, where a model for the forward characteristics of the system is required to derive optimal control laws, only the predicted mean from the neural network is considered. This means that the certainty equivalence principle is assumed.

Replacing the system equation $y(k+1) = ay(k) + bu(k) + \xi(k+1)$ by the model $y(k+1) = \hat{a}y(k) + \hat{b}u(k) + \hat{\xi}(k+1)$, assuming the certainty equivalence principle means that the optimal control law is obtained by considering the deterministic plant

$$\hat{y}(k+1) = \hat{a}y(k) + \hat{b}u(k). \tag{7.20}$$

From (7.4), the optimal control law is given by

$$u(k) = \frac{y_{ref}(k+1) - \hat{a}y(k)}{\hat{b}}. \tag{7.21}$$

With the control law in (7.21), the conditional expected value of the performance index $J(k)$, i.e., the expected value of the performance index from the last control value, is given by

$$\begin{aligned} E[J(k) \mid u(k), y(k)] &= E\left[\left(\hat{a}y(k) + \hat{b}\frac{y_{ref}(k+1) - \hat{a}y(k)}{\hat{b}} - y_{ref}(k+1) + \hat{\xi}(k+1)\right)^2 \mid y(k)\right], \\ &= \sigma^2(k+1). \end{aligned} \tag{7.22}$$

Since the random forces $\hat{\xi}(k+1)$ are dependent on the input $u(k)$, and the variance of $\hat{\xi}(k+1)$, $\sigma^2(k+1)$ is assumed to be given by the following difference equation

$$\sigma^2(k+1) = \hat{c}y(k) + \hat{d}u(k). \tag{7.23}$$

The use of (7.23), and (7.21) allows us to write the conditional expected value of the performance index as

$$E[J(k) \mid u(k), y(k)] = \frac{\hat{c}\hat{b} - \hat{a}\hat{d}}{\hat{b}}y(k) + \frac{\hat{d}}{\hat{b}}y_{ref}(k+1). \tag{7.24}$$

However, if the following control variable is taken to be the optimal control law

$$u(k) = \frac{y_{ref}(k+1) - \hat{a}y(k)}{\hat{b}} - \frac{\hat{d}}{2\hat{b}^2}, \tag{7.25}$$

then with this control law, the expected value of the performance index is given by

$$\begin{aligned} E[J(k) \mid u(k), y(k)] &= \left(\left\{ \hat{a}y(k) + \hat{b}\left(\frac{y_{ref}(k+1) - \hat{a}y(k)}{\hat{b}} - \frac{\hat{d}}{2\hat{b}^2} \right) - y_{ref}(k+1) \right\}^2 \right. \\ &+ \left. \left\{ \hat{c}y(k) + \hat{d}\left(\frac{y_{ref}(k+1) - \hat{a}y(k)}{\hat{b}} - \frac{\hat{d}}{2\hat{b}^2} \right) \right\} \right), \\ &= \frac{\hat{c}\hat{b} - \hat{a}\hat{d}}{\hat{b}} y(k) + \frac{\hat{d}}{\hat{b}} y_{ref}(k+1) - \frac{\hat{d}^2}{4\hat{b}^2}. \end{aligned} \tag{7.26}$$

Comparing (7.24) and (7.26), it is clear that the optimal control for the deterministic system (7.20) is not optimal, since a better control law (7.25) is obtained. The derivation of the control law (7.25) is given in the next section.

System with input dependent noise

In the previous section, it has been shown that the control law for a stochastic system which is derived using the certainty equivalence principle is not optimal. In this section the optimal control law for the system with input dependent output noise is derived assuming a stochastic model for the plant. The derivation will be computed under the assumption that the random forces $\xi(k+1)$ are independently and identically distributed Gaussian variables, with

$$E[\xi(k+1)] = 0, \tag{7.27}$$

$$E[\xi^2(k+1)] = \sigma^2(k+1). \tag{7.28}$$

Hence assuming that the forward characteristic model of the the plant is given by,

$$\hat{y}(k+1) = \hat{a}y(k) + \hat{b}u(k) + \hat{\xi}(u(k), y(k)), \tag{7.29}$$

the expected value of the performance index is

$$E[J(k)] = \| \hat{a}y(k) + \hat{b}u(k) - y_{ref}(k+1) \|^2 + \sigma^2(k+1), \tag{7.30}$$

where the variance, $\sigma^2(k+1)$, of the random variable, $\xi(k+1)$, is an input dependent variance, given by

$$\sigma^2(k+1) = \hat{c}y(k) + \hat{d}u(k). \tag{7.31}$$

Differentiating the expected value of the performance index (7.30) with respect to $u(k)$, yields the optimal control law $u(k)$

$$u(k) = \frac{y_{ref}(k+1) - \hat{a}y(k)}{\hat{b}} - \frac{\hat{d}}{2\hat{b}^2}. \tag{7.32}$$

7.2 Uncertainty estimation of the forward model

Accepting the fact that the predicted output of the controller is uncertain, sampling from the control signal distribution was proposed in Chapter 4. To see the effect of various samples from the control signal distribution, an accurate forward model was assumed to be available. Ignoring the uncertainty in the forward model, the proposed sampling method searches for a better control law which minimises the difference between the predicted mean value of the forward model and the desired output of the system. This however is shown to be suboptimal as discussed in Section 7.1.4. The control law derived

so as to minimise a performance index for a deterministic system is shown to be suboptimal if the added noise to the system output is input dependent.

Therefore, it is expected that a better control law can be obtained from the proposed sampling method if the uncertainty in the forward model could be estimated and then incorporated in the performance measure criterion.

In Chapter 4, the predictive error bar method is used to estimate uncertainty of the inverse method. The predictive error bar will be used here as well to provide an estimate for the uncertainty in the forward model.

In this work only input-output data is used to estimate forward and inverse models. The forward and inverse models are taken to be an input-output model of the form

$$\begin{aligned} \hat{y}(k+d) &= N_f(y(k), y(k-1), ..., y(k-q+1), u(k), ..., u(k-p+1)), \\ \hat{u}(k) &= N_f^{-1}(y(k+d), y(k), y(k-1), ..., y(k-q+1), u(k-1), ..., u(k-p+1)). \end{aligned} \quad (7.33)$$

Considering the input output models for the forward and inverse problems, the output from the neural network approximates the conditional mean of the target data. For the forward model of the plant, this means $N_{f_{opt}}(u(k), \mathbf{x}(k)) =< y(k+d) \mid u(k), \mathbf{x}(k) >$. Here $\mathbf{x}(k) = [y(k), ..., y(k-q+1), u(k-1), ..., u(k-p+1)]$. Hence the local variance can be estimated by $\|y(k+d) - N_{f_{opt}}(u(k), \mathbf{x}(k))\|^2$.

Again, in implementing the predictive error bars, two neural networks are required. If an RBF network is used, both networks (the network which predicts the mean of the target data and the network which predicts the variance of the target data) share the hidden layer parameters. Otherwise, for multi layer perceptron networks two networks with different parameters and the same inputs are needed. The parameters in the first network are optimised so as to predict the target value of the system output. Once the parameters of the first network are optimised, the variance can be calculated. The parameters in the second network can then be optimised so as to predict the calculated variance.

7.3 Incorporating 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 optimised. Defining the performance index as before

$$J(k) = \underset{u \in U}{Min}\ M^2(k) = \underset{u \in U}{Min}\ [(g(u(k), \mathbf{x}(k)) - y_{ref}(k+d))^2], \quad (7.34)$$

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

$$g(u(k), \mathbf{x}(k)) = \hat{g}(u(k), \mathbf{x}(k), W) + \xi(u(k), \mathbf{x}(k), \tilde{W}), \quad (7.35)$$

where $\xi(u(k), \mathbf{x}(k), \tilde{W})$ is assumed to be a random noise with zero mean and σ_ξ^2. The uncertainty $\xi(u(k), \mathbf{x}(k), \tilde{W})$ in the forward model can be modelled as described before in section 7.2 assuming a Gaussian distribution of the error.

Substituting (7.35) into (7.34) yields

$$J(k) = \underset{u \in U}{Min}\ [(\hat{g}(u(k), \mathbf{x}(k), W) + \xi(u(k), \mathbf{x}(k), \tilde{W}) - y_{ref}(k+d))^2]. \quad (7.36)$$

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

performance index is given by

$$\begin{aligned} < J(k) >_{\xi} &=< (\hat{g}(u(k),\mathbf{x}(k),W) + \xi(u(k),\mathbf{x}(k),\tilde{W}) - y_{ref}(k+d))^2 >_{\xi}, \\ &= (\hat{g}(u(k),\mathbf{x}(k),W) - y_{ref}(k+d))^2 + \sigma_{\xi}^2, \end{aligned} \tag{7.37}$$

since

$$< M(k) \mid u(k),\mathbf{x}(k) >= \hat{g}(u(k),\mathbf{x}(k),W) - y_{ref}(k+d), \tag{7.38}$$

and

$$var(M(k) \mid u(k),\mathbf{x}(k)) = \sigma_{\xi}^2. \tag{7.39}$$

It is clear from (7.37) that the conditional probability density $p(M(k) \mid u(k),\mathbf{x}(k))$ is given by that of $\xi(u(k),\mathbf{x}(k),\tilde{W})$ with $\xi(u(k),\mathbf{x}(k),\tilde{W}) = M(k) - \hat{g}(u(k),\mathbf{x}(k),W) + y_{ref}(k+d)$. From (7.37), the optimal control law is given by

$$u(k) = arg \underset{u \in U}{Min} [(\hat{g}(u(k),\mathbf{x}(k),W) - y_{ref}(k+d))^2 + \sigma_{\xi}^2], \tag{7.40}$$

where U is the set of samples from the control signal distribution, $U = [u_1, u_2, ...u_N]$, and N is the number of samples generated from the control signal distribution. Therefore, instead of minimising the gap between the average of the forward model and the desired trajectory as in (4.18), 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 minimised.

7.4 Optimal control law

Since taking the mean value of the forward model to search for a better control value in the proposed sampling method is shown to be suboptimal, the algorithm is slightly modified to allow for uncertainty in the forward model.

Based on estimates of the distributions of control signal values, and model output values, the algorithm in Section 4.5.1, is slightly modified and repeated here. The block diagram of this algorithm is shown in figure 7.1.

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

$$p(y(k+d) \mid u(k),\mathbf{x}(k)) = \frac{1}{(2\pi\sigma_{y(k+d)}^2)^{\frac{1}{2}}} \exp\left(-\frac{(y(k+d) - \hat{y}(k+d))^2}{2\sigma_{y(k+d)}^2}\right), \tag{7.41}$$

   where $\sigma_{y(k+d)}^2 = \sigma_{\xi}^2$.

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

$$p(u(k) \mid y(k+d),\mathbf{x}(k)) = \frac{1}{(2\pi\sigma_{u(k)}^2)^{\frac{1}{2}}} \exp\left(-\frac{(u(k) - \hat{u}(k))^2}{2\sigma_{u(k)}^2}\right), \tag{7.42}$$

   or a mixture of Gaussians given by

$$p(u(k) \mid y(k+d),\mathbf{x}(k)) = \sum_{j=1}^{M} \alpha_j(y(k+d),\mathbf{x}(k))\phi_j(u(k) \mid y(k+d),\mathbf{x}(k)). \tag{7.43}$$

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 following:

i. For the Gaussian function:
Generate a random sample from a Gaussian distribution.

ii. For the mixture density network:
Since Gaussian kernel functions are used, the samples can be generated randomly from each kernel function. 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.

The vector of samples is considered as the 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 minimises the following cost function.

$$J(k) = \underset{u \in U}{\text{Min}} \underset{\bar{\nu}}{E}[(\hat{y}(k+d) - y_{ref}(k+d))^2 + \sigma_\xi^2], \tag{7.44}$$

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 $\bar{\nu}$, 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 function in (7.44) can be optimised directly.

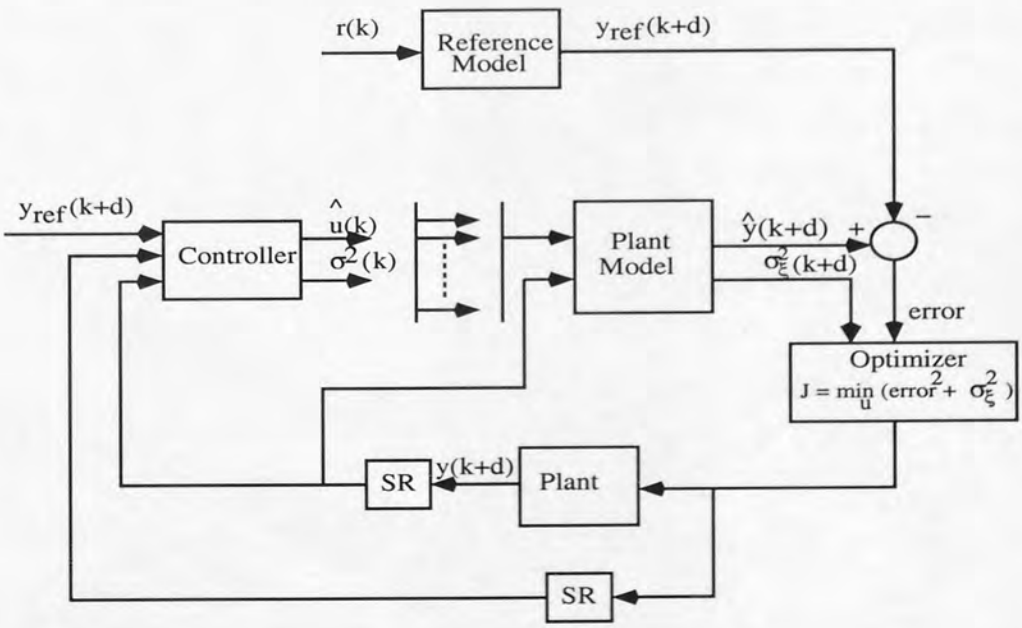


Figure 7.1: The block diagram of the modified proposed optimisation method.

7.5 Simulation experiments

The same simulation experiments that are used to demonstrate theoretical development for incorporating uncertainty in the inverse model, will be repeated in this chapter to illustrate the development for incorporating uncertainty in the forward model. The system equations are:

Simulation experiment 1,

$$\begin{aligned} y(k) &= 0.9722y(k-1) + 0.3578u(k-1) - 0.1295u(k-2) - 0.3103y(k-1)u(k-1) \\ &- 0.04228y^2(k-2) + 0.1663y(k-2)u(k-2) - 0.3513y^2(k-1)u(k-2) \\ &+ 0.3084y(k-1)y(k-2)u(k-2) + 0.1087y(k-2)u(k-1)u(k-2) \\ &- \bar{v}y^2(k-1)y(k-2). \end{aligned} \tag{7.45}$$

Simulation experiment 2,

$$y(k+1) = y(k)\sin(u_1(k)) + u_2(k) - 0.1u_1(k-1)y(k-1) + 0.5u_2^2(k-1) + \bar{v}(k). \tag{7.46}$$

Simulation experiment 3,

$$\begin{aligned} x_1(k+1) &= 0.9x_1(k)\sin[x_2(k)] + \left[2 + 1.5\frac{x_1(k)u_1(k)}{1+x_1^2(k)u_1^2(k)}\right]u_1(k) + \left[x_1(k) + \frac{2x_1(k)}{1+x_1^2(k)}\right]u_2(k), \\ x_2(k+1) &= x_3(k)\{1+\sin[4x_3(k)]\} + \frac{x_3(k)}{1+x_3^2(k)}, \\ x_3(k+1) &= \{3+\sin[2x_1(k)]\}u_2(k), \\ y_1(k) &= x_1(k), \\ y_2(k) &= x_2(k). \end{aligned} \tag{7.47}$$

Here, two cases will be considered. The first case is where the inverse model is taken to be a Gaussian function. The second case is where the inverse model is modelled using a mixture density network.

Simulation experiment 4,

$$y(k) = u(k) + 0.3\sin(2\pi u(k)) + \epsilon. \tag{7.48}$$

The algorithm in Section 7.4 for the sampling method is implemented in the above simulation experiments to search for a better control law. Table 7.1 shows the overall average tracking error (A.T.E), for each simulation experiment using different control approaches.

$$\text{A.T.E} = \frac{\sum_{i=1}^{N}\{y(i) - y_{ref}(i)\}^2}{N}, \tag{7.49}$$

where y is the actual system output, y_{ref} is the desired output, and N is the number of samples in the testing set.

From Table 7.1, one can see that the average tracking error for the proposed sampling method when the uncertainty in the forward model is ignored is always less than that of the direct inverse control in all of the experiments. Moreover, the proposed sampling method shows the ability to stabilise the output of the system when it approaches sensitive regions. When the uncertainty in the forward model is included in the performance measure criterion, the average tracking error of the sampling method is found to be comparable to and sometimes less than that when ignoring the uncertainty in the forward model.

Indeed, this was expected since it has been shown in Section 7.1.4 that a lower value for the performance measure can be obtained when using the control law derived based on the assumption

that the system is stochastic rather than when using the control law derived based on the assumption that the system is deterministic.

For simulation 3 (Gaussian inverse) and simulation 4 the average tracking error from the sampling method including uncertainty in the forward model is found to be slightly higher than that of the sampling method ignoring the uncertainty in the forward model. This can be due to several reasons. Firstly the added noise to the system output in the simulation 4 experiment is generated from a uniform distribution, which violates the assumption that the modelled noise has a Gaussian distribution. Secondly, the estimated variance around the predicted outputs of the controller and inverse model is only an estimate. This means that if the estimates for the uncertainties in the forward and inverse models are inaccurate, then the expected result can be inaccurate as well. However, in all the experiments performed in this work, good results have been obtained, which indicates that the estimates for the forward and inverse models distributions have been accurate enough.

	Number of samples	A.T.E direct inverse	A.T.E, sampling: ignoring uncertainty in the forward model	A.T.E, sampling: including uncertainty in the forward model
Simulation 1	maximum, 27	unstable	0.0035	0.0029
Simulation 2	20	0.1559	0.1298	0.1187
Simulation 3 Gaussian inverse	20	0.5711	0.0906	0.0921
Simulation 3 MDN inverse	20	0.5350	0.0759	0.0713
Simulation 4	20	2.1010×10^{-4}	6.5078×10^{-5}	7.8139×10^{-5}

Table 7.1: The average tracking error resulting from implementing different control methods.

7.6 Conclusion

This chapter considered the possibility of incorporating uncertainty of the forward model in the sampling method proposed in this work. Using a simple linear equation, it was shown that the performance index has a lower value when the derived control law accounts for the added noise to the system equation rather than when the derived control law ignores the added noise.

The previous sampling method was modified in such a way that searching for a better control law from the control signal distribution is performed such that the expected value of the cost function over the uncertainty in the forward model is minimised rather than minimising the cost function by ignoring the uncertainty in the forward model.

Simulation experiments demonstrated the successful application of the modified sampling method which accounts for the uncertainty in the forward model.

Chapter 8

Conclusions and Directions for Future Research

In this thesis a new control architecture for the improvement of controller performance under uncertain environments is developed. As has been shown in Chapter 3, uncertainty around the predicted output of the forward model and the controller arises as a result of:

1. The uncertainty in the model parameters.
2. The nondeterministic relationship between the input and the output variables.
3. The incomplete information we may have about the process.
4. The inappropriate cost function used to optimise the model or the controller parameters.

Two important requirements for the developed control architecture are to estimate a representative model of the plant dynamics and to train a feed-forward controller. These tasks were briefly outlined in a neural network context in Chapter 2. Different control architectures were also reviewed. To achieve the objective of the thesis we chose the direct inverse control approach to be used in the thesis. The reason for using direct inverse control is that it provides an explicit way for estimating the controller uncertainty.

As an important tool for the development of the control architecture in this thesis, we reviewed and studied in Chapter 3 several methods for estimating the uncertainty around the predicted outputs of the neural network models. The study is based on the use of the RBF network as a function approximator. In the first method we studied, the Bayesian method, two different cases are considered for estimating the error bars of an RBF network. The first is to estimate the error bars when the parameters of the output layer of the RBF network are optimised only. The second is to estimate the error bars when all the parameters of the RBF network are optimised. Our key finding was that unrealistic Bayesian error bars can be obtained by optimising the parameters of the output layer only. Moreover, it has been shown that a full covariance matrix for the uncertainty of the predicted output could be estimated using the Bayesian methods. Significant computational time is required for Gaussian processes, the second method. In conclusion we found that predictive error bar method is most suitable in control applications, where fast estimate of the neural network output and error bars is required and where online adaptation is essential.

Based on using the direct inverse control approach and the predictive error bar method for estimating uncertainty of the controller, we developed in Chapter 4 a new control architecture incorporating uncertainty directly. The developed control architecture is based on importance sampling from the distribution of control signal. Using a stochastic SISO simulation example we have demonstrated that the developed control architecture can yield an improved control result compared to the direct inverse control method. Moreover it is proved that the proposed sampling method can stabilise the plant output when it goes unstable. However, since the proposed sampling method requires online adaptation for the control law, implementing this approach is found to be computationally more expensive than the direct inverse control method. Nevertheless, this should not stop us from implementing the new sampling architecture if we believe that the precision of control comes at the cost of extra calculations and that real time control is still possible. Convergence of the output error as a result of online adaptation of control signal was proved by using a suitable Lyapunov function.

Training an adaptive inverse controller is well understood task for SISO nonlinear plants [7] and has been studied for MIMO nonlinear plants [60]. In Chapter 5 we saw how to extend the developed sampling method to improve the performance of the controller when applied to MISO and MIMO nonlinear plants. Simulation results show the successful application of the sampling method. An improved performance compared to the direct inverse control has been achieved.

In the sampling method developed in Chapter 4 the predicted distribution of control signals is assumed to be Gaussian. However it should be noted that the Gaussian assumption of the control signal distribution is not always appropriate. The reason is that the mapping for inverse problems can often be multi-valued, with input values having several valid values for the outputs. This is known as an ill-posed problem [13]. A more general approach to model the conditional distribution of the control signal for multi-valued problems is presented in Chapter 6. It is based on the use of the mixture density network. In Chapter 6 the formulation of the mixture density network was developed for dynamic systems. Moreover, the sampling method developed in Chapter 4 was extended to sampling from the distribution of the density network. Static and dynamic simulation examples were used to demonstrate the use of the mixture density network as an inverse controller. This was compared to the standard neural network, where a Gaussian distribution for the inverse controller is assumed. Simulation results showed the poor performance of the standard inverse controller compared to the density network. Moreover, sampling from the distribution of the density network was proved to give better control results compared to taking the most probable value of the mixture density network.

With the design completed to this point, only uncertainty in the inverse model (the controller) was considered. Uncertainty in the forward model, which is one of the basic elements of the developed sampling method was ignored. In Chapter 7, it has been shown that ignoring the uncertainty in the forward model will yield a sub-optimal control law. Based on this finding, an improved sampling method which accounts for the uncertainty in the forward model was developed. Simulation examples demonstrated the successful application of the improved sampling method. It has been demonstrated in most of the worked examples that a better control result compared to the sampling method where the uncertainty in the forward model is ignored can be obtained.

8.1 Directions for future research

1. **Mixture density network:**
   In this thesis the mixture density network was used to solve the ill-posed class of multi-modality, where the solution for the inverse problem is not unique. Other classes of multi-modality, such as the spatial and the temporal multi-modalities discussed in Chapter 6 have not been considered. The mixture density may be applied to the spatial or temporal multi-modalities. Since the spatial and temporal multi-modalities differ in concept to the ill-posed problem, other control architectures may need to be developed. Take as an example the spatial multi-modality, where the forward dynamics of the plant exhibit different characteristics over different operating zones, the following points can be considered:

   - A mixture density network can be used to obtain the forward dynamics of the plant.
   - Since different operating zones characterise the dynamics of the plant, another mixture density network can be used to model the inverse of the plant and act as a controller. This network can be trained in different ways:
     - Supervised learning, where the actual control signal is assumed to be known. This is the direct inverse control. The mixture density network of the inverse model can be designed to predict the means and the variances of the mixture model only. The same prior values as the forward model can be used. Using the same priors for the inverse and the forward models makes the inverse and the forward models tightly coupled both during training and control.
     - As in the multiple model approach, the indirect adaptive control can be used. The forward model is firstly identified using the mixture density network. The conditional average of the output of the forward model can then be obtained from $\sum_j \alpha_j \mu_j$. In the indirect adaptive control the utility function is defined as the difference between the actual output and the desired output of the plant, $J = \| y(k+d) - y_{ref}(k+d) \|^2$. Replacing the output of the system by the conditional mean of the forward model output, the controller parameters can be obtained by back-propagating the error through the forward model.

2. **Incorporating uncertainty in the forward model:**
   This thesis considers the direct inverse control architecture only. The concept of incorporating the forward model uncertainty may be applied to other control architectures. Take as an example the indirect adaptive control, the forward model uncertainty can be incorporated as follows: As in the conventional indirect adaptive control, the plant is firstly parameterised using one of the input output models described in Chapter 2 and the parameters of the neural network model are updated using the identification error, $e = \| y(k+d) - \hat{y}(k+d) \|^2$. As discussed in Chapter 7, once the forward model is obtained another neural network with the same inputs as the forward model can be trained to predict the uncertainty around the predicted output of the forward model. Considering the uncertainty of the forward model, the utility function to be minimised is defined as $J = \| \hat{y}(k+d) - y_{ref}(k+d) \|^2 + \sigma^2_{y(k+d)}$. This is compared with the conventional indirect adaptive control, where the gap between the actual output and the desired output, $J = \| \hat{y}(k+d) - y_{ref}(k+d) \|^2$, is minimised. Taking the modified definition for the utility

function, the controller parameters in turn can be adjusted by back-propagating the derivative of the utility function through the identified forward model and the model of uncertainty.

3. **Full covariance matrix of the predicted output:**
   The predictive error bar method for estimating the variance of the predicted output distribution is applicable to the case of m-dimensional outputs. The predictive error bar method can provide a general stochastic model. In this thesis the covariance matrix $< \mathbf{e}\mathbf{e}^T >$, is assumed to be diagonal. The Bayesian method can also provide an estimate for the full covariance matrix as explained in chapter 3. It would be interesting to extend the sampling methods proposed in this thesis to the case where a full covariance matrix can be estimated.

4. **Global search for the optimal control law using the estimated uncertainty:**
   In this work a local search for the optimal control law at each instant of time was suggested. Following the dynamic programming method a global search for the optimal control law can be performed. To achieve the global search for an optimal control law an input dependent discretisation for the output (state) space can be performed. Specifically, the uncertainty in the inverse model can be used to generate a set of possible control values. The output of the model corresponding to each control value generated from the control signal distribution can then be calculated. Using this discretisation method has an advantage over the dynamic programming approach, since a higher number of points in the desired region of the state space can be obtained. However, global search for optimal control can be seen to be computationally more expensive than the local search method proposed in this work. On the other hand it would be interesting to see how much improvement could result from the global search compared to the local search method.

Appendix A

Bayesian Error Bars

A.1 Standard Bayesian error bars

For a regression problem with a set of inputs $\mathbf{s}(k) = [u_1(k), ..., u_N(k), y_1(k), ..., y_N(k), y_1(k-q+1), ..., y_N(k-q+1), u_1(k-1), ..., u_N(k-1)...., u_1(k-p+1), ..., u_N(k-p+1)]$, and a corresponding set of targets $y(k+d) = y_1(k+d), ..., y_N(k+d)$, then $D = \{y(k+d), \mathbf{s}(k)\}$ forms a data set from which inference about the relationship between $u(k)$ and $y(k+d)$ can be made. Given the deterministic function $\hat{g}[\mathbf{s}(k), \boldsymbol{W}]$ produced by a network with weights $\boldsymbol{W}$ together with additive Gaussian noise of zero mean and inverse variance β then [13]

$$p(y(k+d)|\mathbf{s}(k), D) = \left(\frac{\beta}{2\pi}\right)^{1/2} \exp\left(-\frac{\beta}{2}\{y(k+d) - \hat{g}[\mathbf{s}(k), \boldsymbol{W}]\}^2\right). \tag{A.1}$$

Given that the data D is conditionally independent of $y(k+d)$ given $\boldsymbol{W}$, the conditional distribution of the predicted target can be written as

$$p(y(k+d)|\mathbf{s}(k), D) = \int p(y(k+d)|\mathbf{s}(k), \boldsymbol{W}) p(\boldsymbol{W}|D) d\boldsymbol{W}. \tag{A.2}$$

From Bayes' rule $p(\boldsymbol{W}|D)$ is given by

$$\begin{aligned} p(\boldsymbol{W}|D) &= \frac{1}{p(D)} p(D|\boldsymbol{W}) p(\boldsymbol{W}), \\ &= \frac{1}{Z_S} \exp(-\beta E_D - \alpha E_{\boldsymbol{W}}), \\ &= \frac{1}{Z_S} \exp(-S(\boldsymbol{W})), \end{aligned} \tag{A.3}$$

where Z_S is a normalising constant given by the integral. Provided that the data is independent the term E_D can be written as

$$\begin{aligned} p(D|\boldsymbol{W}) &= \prod_{i=1}^{N} p(y_i(k+d)|\mathbf{s}_i(k), \boldsymbol{W}), \\ &= \frac{1}{Z_D(\beta)} \exp\left(-\frac{\beta}{2} \sum_{i=1}^{N} \{y_i(k+d) - \hat{g}[\mathbf{s}_i(k), \boldsymbol{W}]\}^2\right), \end{aligned} \tag{A.4}$$

where the normalisation constant Z_D is then the product of N independent Gaussian integrals and is given by $Z_D(\beta) = (2\pi/\beta)^{(N/2)}$. The term $E_{\boldsymbol{W}}$ represents the probability distribution for the weights.

This distribution is usually chosen to be Gaussian with zero mean and inverse variance α.

$$p(\boldsymbol{W}) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W),$$
$$= \frac{1}{Z_W(\alpha)} \exp\Big(-\frac{\alpha}{2} \parallel \boldsymbol{W} \parallel^2 \Big), \tag{A.5}$$

where $Z_{\boldsymbol{W}}(\alpha)$ is a normalising constant and is given by $Z_{\boldsymbol{W}}(\alpha) = (2\pi/\alpha)^2$.

To solve the integral given by Eq. (A.2) some simplifying approximations are required. This can be done using a Gaussian approximation for the posterior distribution. This is obtained by taking the second order Taylor expansion of $S(\boldsymbol{W})$ around its minimum

$$S(\boldsymbol{W}) = S(\boldsymbol{W}_{MP}) + \frac{1}{2}(\boldsymbol{W} - \boldsymbol{W}_{MP})^T \boldsymbol{A} (\boldsymbol{W} - \boldsymbol{W}_{MP}), \tag{A.6}$$

where $\boldsymbol{A} = \nabla_{\boldsymbol{W}} \nabla_{\boldsymbol{W}} S(\boldsymbol{W}_{MP})$.

Substituting Eqs. (A.1) and (A.6) into Eq. (A.2) gives the following relationship

$$p(y(k+d)|\boldsymbol{s}(k), D) \alpha \int \exp\Big(-\frac{\beta}{2}\{y(k+d) - \hat{g}[\boldsymbol{s}(k), \boldsymbol{W}]\}^2 \Big) \times \exp\Big(-\frac{1}{2} \Delta \boldsymbol{W}^T \boldsymbol{A} \Delta \boldsymbol{W} \Big) d\boldsymbol{W}. \tag{A.7}$$

Assuming that the width of the posterior distribution (given by the Hessian matrix $\boldsymbol{A}$) is sufficiently narrow, the function $\hat{g}[\boldsymbol{s}(k), \boldsymbol{W}]$ can be approximated by its linear expansion around $\boldsymbol{W}_{MP}$

$$\hat{g}[\boldsymbol{s}(k), \boldsymbol{W}] \approx \hat{g}[\boldsymbol{s}(k), \boldsymbol{W}_{MP}] + \boldsymbol{G}^T \Delta \boldsymbol{W}, \tag{A.8}$$

where $\boldsymbol{G} = \nabla_{\boldsymbol{W}} \hat{g}[\boldsymbol{s}(k), \boldsymbol{W}]$ evaluated at $\boldsymbol{W}_{MP}$.

Substituting Eq. (A.8) into Eq. (A.7) and evaluating the integral gives

$$p(y(k+d)|\boldsymbol{s}(k), D) = \frac{1}{(2\pi\sigma^2_{y(k+d)})^{(1/2)}} \exp\Big(-\frac{(y(k+d) - \hat{g}[\boldsymbol{s}(k), \boldsymbol{W}_{MP}])^2}{2\sigma^2_{y(k+d)}} \Big). \tag{A.9}$$

This distribution is Gaussian distribution with mean y_{MP} and variance

$$\sigma^2_{y(k+d)} = \frac{1}{\beta} + \boldsymbol{G}^T \boldsymbol{A}^{-1} \boldsymbol{G}. \tag{A.10}$$

A.2 Bayesian error bars with input noise

In the case of noisy input data the predicted distribution $p(y(k+d)|\boldsymbol{z}(k), D)$ can be written in terms of the integral [132]

$$p(y(k+d)|\boldsymbol{z}(k), D) = \int p(y(k+d)|\boldsymbol{s}(k), D) p(\boldsymbol{s}(k)|\boldsymbol{z}(k)) d\boldsymbol{s}(k), \tag{A.11}$$

where $\boldsymbol{s}(k)$, is the noiseless input, $\boldsymbol{z}(k)$ is the noisy input, and $y(k+d)$ is the target value. The first term of the right hand side of Eq. (A.11) is the posterior over the model. This can be expanded in terms of the model variables $\boldsymbol{W}$. Allowing for the independence of $\boldsymbol{W}$ on $\boldsymbol{s}(k)$ this gives

$$p(y(k+d)|\boldsymbol{s}(k), D) = \int p(y(k+d)|\boldsymbol{W}, \boldsymbol{s}(k)) p(\boldsymbol{W}|D) d\boldsymbol{W}. \tag{A.12}$$

Now by expanding $D = [y^n(k+d), \boldsymbol{z}^n(k)]$ and marginalising over $\boldsymbol{s}^n(k) = \boldsymbol{s}_1(k), ...\boldsymbol{s}_N(k)$, the $p(\boldsymbol{W}|D)$ term can be written as

$$p(\boldsymbol{W}|D) = \int p(\boldsymbol{W}, \boldsymbol{s}^n(k)|y^n(k+d), \boldsymbol{z}^n(k)) d\boldsymbol{s}^n(k). \tag{A.13}$$

Using Bayes' rule and conditional independence of $y^n(k+d)$ and $\mathbf{z}^n(k)$ Eq.(A.13) can be rewritten in the following form

$$p(\boldsymbol{W}|D) = \frac{p(\boldsymbol{W})}{p(y^n(k+d)|\mathbf{z}^n(k))} \int p(y^n(k+d)|\boldsymbol{W}, \mathbf{s}^n(k)) p(\mathbf{s}^n(k)|\mathbf{z}^n(k)) d\mathbf{s}^n(k). \tag{A.14}$$

Finally substituting equations (A.12, A.13, A.14) into Eq. (A.11) and using Bayes' rule to express the conditional distribution over $\mathbf{s}(k)$ and $\mathbf{z}(k)$ in terms of the generative noise distribution gives

$$\begin{aligned} p(y(k+d)|\mathbf{z}(k), D) = \frac{1}{Z_{tz}} \int & p(y(k+d)|\boldsymbol{W}, \mathbf{s}(k)) p(\mathbf{z}(k)|\mathbf{s}(k)) p(\mathbf{s}(k)) p(y^n(k+d)|\boldsymbol{W}, \mathbf{s}^n(k)) \\ & p(\mathbf{z}^n(k)|\mathbf{s}^n(k)) p(\mathbf{s}^n(k)) p(\boldsymbol{W}) d\boldsymbol{W} d\mathbf{s}^n(k) d\mathbf{s}(k), \end{aligned} \tag{A.15}$$

where $Z_{tz} = \int p(\mathbf{z}(k)) p(y^n(k+d), \mathbf{z}^n(k)) dy^n(k+d) d\mathbf{z}^n(k)$ is a normalising constant. The Laplace approximation can be used to estimate the integral given in Eq. (A.15). To do this the integrals over $\mathbf{s}^n(k)$ and $\mathbf{s}(k)$ in Eq. (A.15). are of the form

$$I_{\mathbf{s}(k)} = \int p(y(k+d)|\boldsymbol{W}, \mathbf{s}(k)) p(\mathbf{z}(k)|\mathbf{s}(k)) p(\mathbf{s}(k)) d\mathbf{s}(k). \tag{A.16}$$

Because in this work it is assumed that the noise process is additive Gaussian $\mathcal{N}(0, \Sigma_{\mathbf{s}(k)})$ then the integral simplifies to

$$I_{\mathbf{s}(k)} = \int \exp\Big(-\frac{\beta}{2}(y(k+d) - \hat{g}(\mathbf{s}(k), \boldsymbol{W}))^2 - (\mathbf{s}(k) - \mathbf{z}(k))^T \Sigma^{-1}_{\mathbf{s}(k)} (\mathbf{s}(k) - \mathbf{z}(k)) \Big) d\mathbf{s}(k). \tag{A.17}$$

Here it is assumed that the prior $p(\mathbf{s}(k))$ is a uniform distribution. Given this assumption and assuming that the noise process is small, linearisation of $\hat{g}(\mathbf{s}(k), \boldsymbol{W})$ around $\mathbf{z}(k)$ is now possible. The Taylor expansion of the first order gives

$$\hat{g}(\mathbf{s}(k), \boldsymbol{W}) = \hat{g}(\mathbf{z}(k), \boldsymbol{W}) + \mathbf{h}^T \delta\mathbf{z}(k), \tag{A.18}$$

where, $\mathbf{h} = \nabla_{\mathbf{s}(k)} \hat{g}(\mathbf{s}(k), \boldsymbol{W})|_{\mathbf{s}(k)=\mathbf{z}(k)}$ and $\delta\mathbf{z}(k) = \mathbf{s}(k) - \mathbf{z}(k)$. The integral $I_{\mathbf{s}(k)}$ now becomes

$$\begin{aligned} I_{\mathbf{s}(k)} &= \int \exp\Big(-\frac{\beta}{2}(y(k+d) - \hat{g}(\mathbf{z}(k), \boldsymbol{W}) - \mathbf{h}^T \delta\mathbf{z}(k))^2 - \delta\mathbf{z}^T(k) \Sigma^{-1}_{\mathbf{s}(k)} \delta\mathbf{z}(k) \Big) d\delta\mathbf{z}(k), \\ &= \frac{1}{Z_z} \exp\Big(-\frac{\beta'}{2}(y(k+d) - \hat{g}(\mathbf{z}(k), \boldsymbol{W}))^2 \Big), \end{aligned} \tag{A.19}$$

where Z_z is a normalising constant and

$$\frac{1}{\beta'} = \frac{1}{\beta} + \mathbf{h}^T \Sigma_{\mathbf{s}(k)} \mathbf{h}. \tag{A.20}$$

Now this result can be used to compute the integral over $\mathbf{s}(k)$ and $\mathbf{s}^n(k)$ in Eq.(A.15). Using a quadratic prior over the weights of the form $E_{\boldsymbol{W}} = \frac{1}{2} \| \boldsymbol{W} \|^2$, the predictive distribution can be written as

$$\begin{aligned} p(y(k+d)|\mathbf{z}(k), D) = \frac{1}{Z_x} \int & \exp\Big(-\frac{\beta'}{2}(y(k+d) - \hat{g}(\mathbf{z}(k), \boldsymbol{W}))^2 \Big) \\ & \prod_{i=1}^{N} \exp\Big(-\frac{\beta'}{2}(y_i(k+d) - \hat{g}(\mathbf{z}_i(k), \boldsymbol{W}))^2 - \frac{\alpha}{2} \| \boldsymbol{W} \|^2 \Big) d\boldsymbol{W}, \end{aligned} \tag{A.21}$$

where Z_x is a normalising constant. Using the Laplace approximation leads to:

$$p(y(k+d)|\mathbf{z}(k), D) \propto \int \exp\Big(-\frac{\beta'}{2}(y(k+d) - \hat{g}(\mathbf{z}(k), \boldsymbol{W}))^2 \Big) \times \exp\Big(-\frac{1}{2} \Delta\boldsymbol{W}^T \boldsymbol{A} \Delta\boldsymbol{W} \Big) d\boldsymbol{W}. \tag{A.22}$$

Where $\mathbf{A}$ is the Hessian matrix

$$\mathbf{A} = \nabla_{\boldsymbol{W}} \nabla_{\boldsymbol{W}} S(\boldsymbol{W}_{MP}). \tag{A.23}$$

Taking the linear expansion of $\hat{g}(\mathbf{z}(k), \boldsymbol{W})$ about $\boldsymbol{W}_{MP}$ the integral over $\boldsymbol{W}$ can be calculated and a Gaussian approximation of the predictive distribution can be written as

$$p(y(k+d)|\mathbf{z}(k), D) = \frac{1}{(2\pi\sigma^2_{y(k+d)})^{1/2}} \exp\left(- \frac{(y(k+d) - \hat{g}(\mathbf{z}(k), \boldsymbol{W}_{MP}))^2}{2\sigma^2_{y(k+d)}} \right). \tag{A.24}$$

And the variance of the distribution is given by

$$\begin{aligned} \sigma^2_{y(k+d)} &= \frac{1}{\beta} + \mathbf{G}^T \mathbf{A}^{-1} \mathbf{G}, \\ &= \frac{1}{\beta} + \mathbf{h}^T \Sigma_{\mathbf{s}(k)} \mathbf{h} + \mathbf{G}^T \mathbf{A}^{-1} \mathbf{G}. \end{aligned} \tag{A.25}$$

Appendix B

Weight Updates for the Radial Basis Function

B.1 Weights updating for the linear output layer

In Section 3.2.1, it has been asserted that the formula for updating the weights using direct matrix inversion is given by

$$\boldsymbol{W} = (\beta\phi\phi^{T} + \alpha I)^{-1}(\beta\phi^{T})y(k+d). \tag{B.1}$$

This can be derived from the fact that the following quadratic error function should be minimised

$$S(\boldsymbol{W}) = \frac{\beta}{2}(y(k+d) - \hat{y}(k+d))^2 + \frac{\alpha}{2} \parallel \boldsymbol{W} \parallel^2, \tag{B.2}$$

where $\hat{y}(k+d)$ is the output of the RBF network and is given by the following formula

$$\hat{y}(k+d) = \boldsymbol{W}\phi, \tag{B.3}$$

where $\boldsymbol{W}$ are the parameters of the RBF network, and ϕ is the output vector of the hidden units. Given that W is the total number of weights and biases in the network

$$\parallel \boldsymbol{W} \parallel^2 = \sum_{i=1}^{W} w_i^2. \tag{B.4}$$

Taking the derivative of Eq. (B.2) with respect to the weights and setting the derivative to zero, the following equation is obtained

$$(\beta(\phi^{T}\phi) + \alpha I)\boldsymbol{W} = \beta\phi^{T}y(k+d). \tag{B.5}$$

Now solving Eq. (B.5) for W

$$\boldsymbol{W} = (\beta\phi\phi^{T} + \alpha I)^{-1}(\beta\phi^{T})y(k+d). \tag{B.6}$$

B.2 Weights updating with respect to all the parameters of the RBF network

Derivation of the formula for updating the weights of the RBF when all the parameters of the RBF are changed using the Levenberg-Marquardt training algorithm can be obtained by defining the required

objective function which in this work is given by

$$\begin{aligned} F(\boldsymbol{W}) &= \frac{\beta}{2}\sum_{i=1}^{N} f_i{}^2(\boldsymbol{W}) + \frac{\alpha}{2} \parallel \boldsymbol{W} \parallel^2, \\ &= \frac{\beta}{2}\underline{f}^T(\boldsymbol{W})\underline{f}(\boldsymbol{W}) + \frac{\alpha}{2}\boldsymbol{W}\boldsymbol{W}^T, \end{aligned} \tag{B.7}$$

where $\underline{f}$ is the difference between the output of the network and the target values, $\underline{f} = \hat{y}(k+d) - y(k+d)$. By taking the derivative of Eq.(B.7) with respect to the weights, the following equation is obtained

$$\begin{aligned} [\nabla F(\boldsymbol{W})]_j &= \beta \sum_{i=1}^{N} f_i(\boldsymbol{W}) \frac{\partial f_i(\boldsymbol{W})}{\partial W_j} + \alpha \boldsymbol{W}, \\ &= \beta \boldsymbol{G}^T(\boldsymbol{W})\underline{f}(\boldsymbol{W}) + \alpha \boldsymbol{W}, \end{aligned} \tag{B.8}$$

where $\boldsymbol{G}$ is the weight gradient of the neural network. Since the Levenberg-Marquardt method used the approximation of the Hessian matrix the derivative of Eq. (B.8) needs to be computed, which should in turn give the second derivative for Eq. (B.7)

$$[\nabla^2 F(\boldsymbol{W})]_{kj} = \beta \sum_{i=1}^{N} \left(\frac{\partial f_i(\boldsymbol{W})}{\partial W_k} \frac{\partial f_i(\boldsymbol{W})}{\partial W_j} + f_i \frac{\partial^2 f_i}{\partial W_k \partial W_j} \right) + \alpha I. \tag{B.9}$$

Now let $T_i(\boldsymbol{W}) = \nabla^2 f_i(\boldsymbol{W})$, then Eq.(B.9) can be rewritten as

$$[\nabla^2 F(\boldsymbol{w})] = \beta \boldsymbol{G}^T(\boldsymbol{W})\boldsymbol{G}(\boldsymbol{W}) + \beta \sum_{i=1}^{N} f_i(\boldsymbol{W}) T_i(\boldsymbol{W}) + \alpha I. \tag{B.10}$$

Let $S(\boldsymbol{W}) = \sum_{i=1}^{N} f_i(\boldsymbol{W}) T_i(\boldsymbol{W})$, then Eq. (B.10) can be rewritten in the following form

$$[\nabla^2 F(\boldsymbol{W})] = \beta \boldsymbol{G}^T(\boldsymbol{W})\boldsymbol{G}(\boldsymbol{W}) + \beta S(\boldsymbol{W}) + \alpha I. \tag{B.11}$$

Now recall Newton's method

$$\begin{aligned} \Delta \boldsymbol{W} &= -[\nabla^2 F(\boldsymbol{W})]^{-1} \nabla F(\boldsymbol{W}), \\ &= -[\beta \boldsymbol{G}^T(\boldsymbol{W})\boldsymbol{G}(\boldsymbol{W}) + \beta S(\boldsymbol{W}) + \alpha I]^{-1} [\beta \boldsymbol{G}^T(\boldsymbol{W})\underline{f}(\boldsymbol{W}) + \alpha \boldsymbol{W}]. \end{aligned} \tag{B.12}$$

Ignoring $S(\boldsymbol{W})$ yields the Levenberg-Marquardt training algorithm.

Appendix C

Mean and Variance of Gaussian Processes

In this appendix the derivation of the mean and the variance of the Gaussian process predictive distribution is analysed in detail. For this purpose, consider the following target vectors in the control problem

$$\mathbf{y}(k+d) = \begin{bmatrix} y^1(k+d) \\ \vdots \\ y^N(k+d) \end{bmatrix}, \quad y^{N+1}(k+d) = [y^{N+1}(k+d)], \quad \mathbf{y}^t(k+d) = \begin{bmatrix} y^{N+1}(k+d) \\ \mathbf{y}(k+d) \end{bmatrix}, \tag{C.1}$$

and similarly for the mean and the variance

$$E[\mathbf{y}(k+d)] = \mu, \qquad E[y^{N+1}(k+d)] = \mu^{N+1}, \tag{C.2}$$

$$\begin{aligned} \mathbf{K} &= E[(y^{N+1}(k+d) - \mu^{N+1})(\mathbf{y}(k+d) - \mu)^T], \\ \mathbf{C_N} &= E[(\mathbf{y}(k+d) - \mu)(\mathbf{y}(k+d) - \mu)^T], \\ \kappa &= E[(y^{N+1}(k+d) - \mu^{N+1})(y^{N+1}(k+d) - \mu^{N+1})^T], \end{aligned} \tag{C.3}$$

with these definitions, the following result will be proved.

If the distribution of $\mathbf{y}^t(k+d)$ *is Gaussian, then the predictive distribution for* $y^{N+1}(k+d)$ *given* $\mathbf{y}(k+d)$ *is Gaussian with mean* $\mu^{N+1} + \mathbf{K}^T\mathbf{C}_N^{-1}[\mathbf{y}(k+d) - \mu]$ *and variance matrix* $\kappa - \mathbf{K}^T\mathbf{C}_N^{-1}\mathbf{K}$ [85].

Introducing the following definitions $O^1 = y^{N+1}(k+d) - \mathbf{K}^T\mathbf{C}_N^{-1}\mathbf{y}(k+d)$ and $O^2 = \mathbf{y}(k+d)$. Then $Q = \begin{bmatrix} O^1 \\ O^2 \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{K}^T\mathbf{C}_N^{-1} \\ 0 & I \end{bmatrix} .\mathbf{y}^t(k+d)$. Since the vector Q is a nonsingular transform of $\mathbf{y}^t(k+d)$, it has a Gaussian distribution with the following mean and variance:

$$\begin{aligned} E\begin{bmatrix} O^1 \\ O^2 \end{bmatrix} &= \begin{bmatrix} 1 & -\mathbf{K}^T\mathbf{C}_N^{-1} \\ 0 & I \end{bmatrix} E[\mathbf{y}^t(k+d)], \\ &= \begin{bmatrix} 1 & -\mathbf{K}^T\mathbf{C}_N^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} \mu^{N+1} \\ \mu \end{bmatrix}, \\ &= \begin{bmatrix} \mu^{N+1} - \mathbf{K}^T\mathbf{C}_N^{-1}\mu \\ \mu \end{bmatrix}. \end{aligned} \tag{C.4}$$

Introducing the definition $E[Q] = M$, the mean values from (C.4) are

$$\begin{aligned} E[O^1] &= [\mu^{N+1} - \mathbf{K}^T\mathbf{C}_N^{-1}\mu] = M^1, \\ E[O^2] &= \mu = M^2. \end{aligned} \tag{C.5}$$

The variance matrix can then be obtained from

$$\mathrm{E}[(Q-M)(Q-M)^{\mathsf{T}}] = \begin{bmatrix} \mathrm{E}[(O^1-M^1)(O^1-M^1)^{\mathsf{T}}] & \mathrm{E}[(O^1-M^1)(O^2-M^2)^{\mathsf{T}}] \\ \mathrm{E}[(O^2-M^2)(O^1-M^1)^{\mathsf{T}}] & \mathrm{E}[(O^2-M^2)(O^2-M^2)^{\mathsf{T}}] \end{bmatrix}, \quad \text{(C.6)}$$

$$= \begin{bmatrix} \kappa - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{K} & 0 \\ 0 & \mathbf{C}_N \end{bmatrix}. \quad \text{(C.7)}$$

Equation (C.7) can be easily obtained. Compute as an example the first element of the matrix in (C.6) yields the first element in (C.7)

$$\begin{aligned} \mathrm{E}[(O^1-M^1)(O^1-M^1)^{\mathsf{T}}] = \mathrm{E}[&(y^{N+1}(k+d)-\mu^{N+1}) - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}(\mathbf{y}(k+d)-\mu)] \\ &[(y^{N+1}(k+d)-\mu^{N+1})^{\mathsf{T}} - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}(\mathbf{y}(k+d)-\mu)^{\mathsf{T}}], \\ &= \kappa - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{K} - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{K} + \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{C}_N\mathbf{C}_N^{-1}\mathbf{K}, \\ &= \kappa - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{K}. \end{aligned}$$

The computation of the other three elements can be done in the same way. Equation (C.7) shows that the two random variables O^1 and O^2 are independent, therefore the joint density of O^1 and O^2 is the multiplication of the marginal distribution of O^1 and the marginal distribution of O^2

$$p(O^1, O^2) = N(O^1 \mid \mu^{N+1} - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mu, \kappa - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{K}).N(O^2 \mid \mu, \mathbf{C}_N). \quad \text{(C.8)}$$

Now substituting $y^{N+1}(k+d) - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{y}(k+d)$ for O^1 and $\mathbf{y}(k+d)$ for O^2 in (C.8) and multiplying by the Jacobian of the transformation which is one, yields the joint density of $y^{N+1}(k+d)$ and $\mathbf{y}(k+d)$

$$\begin{aligned} p(y^{N+1}(k+d), \mathbf{y}(k+d)) = &(2\pi)^{-\frac{1}{2}} \mid \sigma_{112} \mid^{-\frac{1}{2}} \\ &\times \exp\{-\frac{1}{2}[(y^{N+1}(k+d)-\mu^{N+1}) - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}(\mathbf{y}(k+d)-\mu)]^{\mathsf{T}} \\ &\times \sigma_{112}^{-1}[(y^{N+1}(k+d)-\mu^{N+1}) - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}(\mathbf{y}(k+d)-\mu)]\} \\ &\boxed{\times (2\pi)^{-\frac{N}{2}} \mid \mathbf{C}_N \mid^{-\frac{1}{2}} \exp\{-\frac{1}{2}(\mathbf{y}(k+d)-\mu)^{\mathsf{T}}\mathbf{C}_N^{-1}(\mathbf{y}(k+d)-\mu)\}}, \quad \text{(C.9)} \end{aligned}$$

where $\sigma_{112} = \kappa - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{K}$.

The conditional density of $y^{N+1}(k+d)$ given $\mathbf{y}(k+d)$ can be obtained using Bayes theorem, hence

$$p(y^{N+1}(k+d) \mid \mathbf{y}(k+d)) = \frac{p(y^{N+1}(k+d), \mathbf{y}(k+d))}{p(\mathbf{y}(k+d))}, \quad \text{(C.10)}$$

the marginal distribution of $p(\mathbf{y}(k+d))$ is simply the factor in the box in (C.9), therefore the conditional distribution of $y^{N+1}(k+d)$ given $\mathbf{y}(k+d)$ is

$$\begin{aligned} p(y^{N+1}(k+d) \mid \mathbf{y}(k+d)) = &(2\pi)^{-\frac{1}{2}} \mid \sigma_{112} \mid^{-\frac{1}{2}} \\ &\times \exp\{-\frac{1}{2}[(y^{N+1}(k+d)-\mu^{N+1}) - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}(\mathbf{y}(k+d)-\mu)]^{\mathsf{T}} \\ &\times \sigma_{112}^{-1}[(y^{N+1}(k+d)-\mu^{N+1}) - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}(\mathbf{y}(k+d)-\mu)]\}. \quad \text{(C.11)} \end{aligned}$$

The conditional distribution in (C.11) is a normal distribution with mean $\mu^{N+1} + \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}[\mathbf{y}(k+d)-\mu]$ and variance matrix $\kappa - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{K}$.

Since in the Gaussian processes the assumption was that the marginal density functions have zero means yields the final result that the predictive distribution for $y^{N+1}(k+d)$ given $\mathbf{y}(k+d)$ is a normal distribution with mean $\mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{y}(k+d)$ and variance $\kappa - \mathbf{K}^{\mathsf{T}}\mathbf{C}_N^{-1}\mathbf{K}$.

Appendix D

Recurrence Relation in Dynamic Programming

To derive the iterative procedure for dynamic programming [69] the minimum cost function is firstly defined. Starting at any admissible state $\mathbf{x}$, $\mathbf{x} \in X$, the minimum cost function resulting from using all admissible control for the remainder of the process at any stage $k, 0 \leq k \leq K$ can be written as

$$J(k) = \min_{\substack{\mathbf{u}(j)\in U \\ j=k,\ldots,K}} \sum_{j=k}^{K} l[\mathbf{x}(j), \mathbf{u}(j), j]. \tag{D.1}$$

By splitting the summation into two parts, and defining $\mathbf{x}(k) = \mathbf{x}$

$$J(k) = \min_{\substack{\mathbf{u}(j)\in U \\ j=k,\ldots,K}} \left(l[\mathbf{x}, \mathbf{u}(k), k] + \sum_{j=k+1}^{K} l[\mathbf{x}(j), \mathbf{u}(j), j] \right). \tag{D.2}$$

Again by splitting the minimisation in Eq. (D.2) into two parts, a minimisation over $\mathbf{u}(k)$, and a minimisation over the controls $\mathbf{u}(k+1), \mathbf{u}(k+2), .., \mathbf{u}(K)$,

$$J(k) = \min_{\mathbf{u}(k)\in U} \min_{\substack{\mathbf{u}(j)\in U \\ j=k+1,\ldots,K}} \left(l[\mathbf{x}, \mathbf{u}(k), k] + \sum_{j=k+1}^{K} l[\mathbf{x}(j), \mathbf{u}(j), j] \right). \tag{D.3}$$

The first term in brackets in Eq. (D.3) depends only on $\mathbf{u}(k)$, therefore

$$\min_{\mathbf{u}(k)\in U} \min_{\substack{\mathbf{u}(j)\in U \\ j=k+1,\ldots,K}} (l[\mathbf{x}, \mathbf{u}(k), k]) = \min_{\mathbf{u}(k)\in U} (l[\mathbf{x}, \mathbf{u}(k), k]). \tag{D.4}$$

The second term in brackets in Eq. (D.3), however depends on the state $\mathbf{x}(k+1)$ which is a function of the control $\mathbf{u}$ at stage k

$$\mathbf{x}(k+1) = g[\mathbf{x}, u(k), k]. \tag{D.5}$$

Using this relation the second term in brackets in Eq. (D.3) can then be written as

$$\min_{\mathbf{u}(k)\in U} \min_{\substack{\mathbf{u}(j)\in U \\ j=k+1,\ldots,K}} \left(\sum_{j=k+1}^{K} l[\mathbf{x}(j), \mathbf{u}(j), j] \right) = \min_{\mathbf{u}(k)\in U} \left(J[g(\mathbf{x}, \mathbf{u}(k), k), k+1] \right). \tag{D.6}$$

Substituting Eq. (D.4), (D.6) into Eq. (D.3) and suppressing the index k on $\mathbf{u}(k)$, the functional equation can be written as

$$J(k) = \min_{\mathbf{u}\in U} \left(l[\mathbf{x}, \mathbf{u}, k] + J[g(\mathbf{x}, \mathbf{u}, k), k+1] \right). \tag{D.7}$$

Equation (D.7) describes an iterative relation for determining $J(k)$ from knowledge of $J(k+1)$. The optimal control is then defined as

$$\mathbf{u}^*(k) = \arg \min_{\mathbf{u}} \Big(l[\mathbf{x}, \mathbf{u}, k] + J[g(\mathbf{x}, \mathbf{u}, k), k+1] \Big). \tag{D.8}$$

Appendix E

The Derivatives of the Error Function of an MDN

To optimise the parameters of the mixture density network, the derivative of the error function (negative log likelihood) of the MDN with respect to each parameter needs to be calculated. The detail of the derivation of the gradient of the negative log likelihood of an MDN network is provided in this appendix.

E.1 The error function of an MDN

The error function of the mixture density network for the n^{th} input pattern $\mathbf{s}_n(k)$ and the n^{th} output pattern $u_n(k)$ is defined as the negative log likelihood

$$E^n = -\ln\left\{\sum_{j=1}^{M} \alpha_j(\mathbf{s}_n(k))\phi_j(u_n(k) \mid \mathbf{s}_n(k))\right\}. \tag{E.1}$$

In this work the j^{th} kernel ϕ_j for the n^{th} input output pattern is taken to be a spherical Gaussian

$$\phi_j(u(k)|\mathbf{s}(k)) = \frac{1}{(2\pi)^{c/2}\sigma_j^c(\mathbf{s}_n(k))} \exp\left(-\frac{\| u(k) - \mu_j(\mathbf{s}_n(k)) \|^2}{2\sigma_j^2(\mathbf{s}_n(k))}\right). \tag{E.2}$$

The total error is then defined as the summation of the error of each pattern

$$E = \sum_{n=1}^{N} E^n. \tag{E.3}$$

In the mixture density network architecture the parameters of the kernel function are assumed to be produced by the outputs of a feedforward neural network. To optimise the parameters of the kernel function, the parameters of the feedforward network need to be updated in order to minimise the negative log likelihood function. Minimising the error function requires calculating the derivatives of the error E with respect to the weights in the feedforward network. Given that the derivatives of the error E with respect to the outputs of the feedforward network can be obtained, the derivatives of the error E with respect to the feedforward network parameters can be calculated using the static backpropagation method [13].

Therefore, the derivatives of the error function with respect to the output of the feedforward network are evaluated in this appendix. Since the total error is the sum of the error of each pattern,

Eq. (E.3), the analysis is on a per-pattern basis. For clarity the references to the input and output data set is dropped from (E.1) and (E.2). Using the terminology of Bishop [13] the derivatives need to be calculated are:

1. The partial derivative of the error function with respect to the outputs corresponding to the mixing coefficients z_j^α
$$\frac{\partial E_n}{\partial z_j^\alpha}. \tag{E.4}$$
2. The partial derivative of the error function with respect to the outputs corresponding to the mixing variances z_j^σ
$$\frac{\partial E_n}{\partial z_j^\sigma}. \tag{E.5}$$
3. The partial derivative of the error function with respect to the outputs corresponding to the centres of the kernel functions z_{jk}^μ
$$\frac{\partial E_n}{\partial z_{jk}^\mu}. \tag{E.6}$$

Since ϕ_j can be regarded as conditional density function with prior probabilities α_j, the posterior probability of a pattern using Bayes theorem is introduced in [13] to simplify the analysis of evaluating the derivative of the error function with respect to the outputs of the network. Hence using Bayes theorem

$$\pi_j = \frac{\alpha_j \phi_j}{\sum_{l=1}^{M} \alpha_l \phi_l}, \tag{E.7}$$

$$\sum_{j=1}^{M} \pi_j = 1, \tag{E.8}$$

$$0 \leq \pi_j \leq 1 \qquad \forall j. \tag{E.9}$$

E.2 Evaluating the derivatives of the mixing coefficients

The derivative of the error function with respect to the feedforward network outputs corresponding to the mixing coefficient can be calculated using the chain rule

$$\frac{\partial E_n}{\partial z_j^\alpha} = \sum_k \frac{\partial E_n}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial z_j^\alpha}. \tag{E.10}$$

The first term in the right hand side of (E.10) can be calculated from (E.1)

$$\begin{aligned} \frac{\partial E_n}{\partial \alpha^k} &= -\left(\frac{\phi_k}{\sum_{j=1}^{M} \alpha_j \phi_j}\right) \frac{\alpha_k}{\alpha_k}, \\ &= -\frac{\alpha_k \phi_k}{\sum_{j=1}^{M} \alpha_j \phi_j} \frac{1}{\alpha_k}. \end{aligned} \tag{E.11}$$

Using (E.7), yields

$$\frac{\partial E_n}{\partial \alpha^k} = -\frac{\pi_k}{\alpha_k}. \tag{E.12}$$

Since the mixing coefficients represent probabilities, their sum should be equal to one, that is $\sum_{j=1}^{M} \alpha_j = 1$. This is achieved by choosing α_j to be related to the corresponding feedforward network by a softmax function

$$\alpha_j(\mathbf{s}(k)) = \frac{\exp(z_j^\alpha)}{\sum_{l=1}^{M} \exp(z_l^\alpha)}. \tag{E.13}$$

Using the softmax transformation makes the value of α_j depend on all of the feedforward network outputs which are corresponding to the mixing coefficients. Therefore differentiating α_j with respect to the feeforward network outputs which contribute to the mixing coefficient can be achieved using the quotient rule for differentiation by considering the two cases for $j = k$ and $j \neq k$. Hence,

- for $j = k$

$$\begin{aligned}\frac{\partial \alpha_k}{\partial z_j^\alpha} &= \frac{\sum_{l=1}^{M} \exp(z_l^\alpha).\exp(z_j^\alpha) - \exp(z_j^\alpha).\exp(z_k^\alpha)}{[\sum_{l=1}^{M} \exp(z_l^\alpha)]^2}, \\ &= \frac{\exp(z_j^\alpha)}{\sum_{l=1}^{M} \exp(z_l^\alpha)} - \left(\frac{\exp(z_j^\alpha)}{\sum_{l=1}^{M} \exp(z_l^\alpha)}\right)^2, \\ &= \alpha_k - \alpha_k^2, \qquad j = k. \end{aligned} \tag{E.14}$$

- for $j \neq k$

$$\begin{aligned}\frac{\partial \alpha_k}{\partial z_j^\alpha} &= \frac{\sum_{l=1}^{M} \exp(z_l^\alpha) \times 0 - \exp(z_j^\alpha).\exp(z_k^\alpha)}{[\sum_{l=1}^{M} \exp(z_l^\alpha)]^2}, \\ &= -\alpha_j \alpha_k, \qquad j \neq k. \end{aligned} \tag{E.15}$$

Using the Kronecker delta function, δ_{jk}, equations (E.14) and (E.15) can be written as

$$\frac{\partial \alpha_k}{\partial z_j^\alpha} = \delta_{jk}\alpha_k - \alpha_k \alpha_j. \tag{E.16}$$

The final derivative can then be obtained by substituting (E.12) and (E.16) into (E.10). This yields

$$\begin{aligned}\frac{\partial E_n}{\partial z_j^\alpha} &= \sum_k -\frac{\pi_k}{\alpha_k}\left(\delta_{jk}\alpha_k - \alpha_k \alpha_j\right), \\ &= \sum_k -\frac{\pi_k}{\alpha_k}\alpha_k \delta_{jk} + \sum_k \frac{\pi_k}{\alpha_k}\alpha_k \alpha_j. \end{aligned} \tag{E.17}$$

Since $\sum_k \pi_k = 1$, and $\sum_k \pi_k \delta_{jk} = \pi_j$,

$$\frac{\partial E_n}{\partial z_j^\alpha} = -\pi_j + \alpha_j. \tag{E.18}$$

E.3 Evaluating the derivatives of the variances

In the mixture density network, the variances σ_j^2 of the kernel functions are taken to be the exponentials of the corresponding feedforward network outputs, z_j^σ. Therefore, the derivative of the error function will be evaluated with respect to the variance σ_j^2. Again using the chain rule

$$\frac{\partial E_n}{\partial z_j^\sigma} = \frac{\partial E_n}{\partial \sigma_j^2}\frac{\partial \sigma_j^2}{\partial z_j^\sigma}. \tag{E.19}$$

Differentiating (E.1) with respect to σ_j^2 yields,

$$\frac{\partial E_n}{\partial \sigma_j^2} = -\left(\frac{1}{\sum_{l=1}^{M} \alpha_j \phi_j}\frac{\partial(\alpha_j \phi_j)}{\partial \sigma_j^2}\right), \tag{E.20}$$

since the mixing coefficients α_j is not a function of the variances σ_j^2, the derivative of $(\phi_j \alpha_j)$ with respect to the variances can be obtained as

$$\frac{\partial(\alpha_j \phi_j)}{\partial \sigma_j^2} = \alpha_j \frac{\partial \phi_j}{\partial \sigma_j^2}. \tag{E.21}$$

This means that the derivative of ϕ_j with respect to the variances σ_j^2 needs to be calculated only. The result can then be multiplied by the mixing coefficients. Deriving ϕ_j with respect to the variances yields,

$$\begin{aligned}
\frac{\partial(\phi_j)}{\sigma_j^2} &= \left[-c \frac{1}{2\sigma_j^2} \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\| u_n - \mu_j \|^2}{2\sigma_j^2} \right) \right. \\
&+ \left. \frac{\| u_n - \mu_j \|^2}{2\sigma_j^4} \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\| u_n - \mu_j \|^2}{2\sigma_j^2} \right) \right], \\
&= \underbrace{\frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\| u_n - \mu_j \|^2}{2\sigma_j^2} \right)}_{\text{equation (E.2)}} \left[-\frac{c}{2\sigma_j^2} + \frac{\| u_n - \mu_j \|^2}{2\sigma_j^4} \right], \\
&= \frac{\phi_j}{2} \left[-\frac{c}{\sigma_j^2} + \frac{\| u_n - \mu_j \|^2}{\sigma_j^4} \right].
\end{aligned} \tag{E.22}$$

Hence, substituting (E.22) into (E.21), yields

$$\frac{\partial(\alpha_j \phi_j)}{\partial \sigma_j^2} = \frac{\alpha_j \phi_j}{2} \left[-\frac{c}{\sigma_j^2} + \frac{\| u_n - \mu_j \|^2}{\sigma_j^4} \right]. \tag{E.23}$$

Merging (E.20) and (E.23), gives

$$\frac{\partial E_n}{\partial \sigma_j^2} = -\left[\frac{\alpha_j \phi_j}{2 \sum_{l=1}^{M} \alpha_l \phi_l} \left(-\frac{c}{\sigma_j^2} + \frac{\| u_n - \mu_j \|^2}{\sigma_j^4} \right) \right]. \tag{E.24}$$

The second term in equation (E.19) can be computed directly

$$\begin{aligned}
\frac{\partial \sigma_j^2}{\partial z_j^\sigma} &= \exp(z_j^\sigma), \\
&= \sigma_j^2.
\end{aligned} \tag{E.25}$$

Finally substituting (E.24) and (E.25) into (E.19) yields the required derivative

$$\frac{\partial E_n}{\partial z_j^\sigma} = -\frac{\pi_j}{2} \left[\frac{\| u_n - \mu_j \|^2}{\sigma_j^2} - c \right]. \tag{E.26}$$

E.4 Evaluating the derivatives of the kernel centres

Since the centres μ_j of the kernel functions are given directly by the feedforward network outputs z_{jk}^μ, the derivative of the error function with respect to the kernel centres can be calculated directly as

$$\frac{\partial E_n}{\partial z_{jk}^\mu} = -\left[\frac{1}{\sum_{l=1}^{M} \alpha_l \phi_l} \frac{\partial(\alpha_j \phi_j)}{\partial z_{jk}^\mu} \right]. \tag{E.27}$$

Again since α_j is not a function of the kernel centres μ_{jk}

$$\frac{\partial(\alpha_j \phi_j)}{\partial z_{jk}^\mu} = \alpha_j \frac{\partial \phi_j}{\partial z_{jk}^\mu}. \tag{E.28}$$

Deriving ϕ_j with respect to the kernel centres μ_{jk} yields

$$\begin{aligned}\frac{\partial \phi_j}{\partial z^{\mu}_{jk}} &= \left[\frac{u_k - \mu_{jk}}{\sigma_j^2}\right] \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_j^c} \exp\left(-\frac{\| u_n - \mu_j \|^2}{2\sigma_j^2}\right), \\ &= \phi_j \left(\frac{u_k - \mu_{jk}}{\sigma_j^2}\right). \end{aligned} \tag{E.29}$$

Now make use of (E.28), yields

$$\frac{\partial(\alpha_j \phi_j)}{\partial z^{\mu}_{jk}} = \phi_j \alpha_j \left(\frac{u_k - \mu_{jk}}{\sigma_j^2}\right). \tag{E.30}$$

Then substituting (E.30) into (E.27) gives the final result

$$\begin{aligned}\frac{\partial E_n}{\partial z^{\mu}_{jk}} &= -\left[\frac{\alpha_j \phi_j}{\sum_{l=1}^{M} \alpha_l \phi_l} \frac{u_k - \mu_{jk}}{\sigma_j^2}\right], \\ &= \pi_j \left[\frac{\mu_{jk} - u_k}{\sigma_j^2}\right]. \end{aligned} \tag{E.31}$$

Bibliography

[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] 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.

[3] M. Aoki. *Optimisation of Stochastic Systems.* Academic Press Inc., New York. London, 1967.

[4] C. G. Atkeson and D. J. Reinkensmeyer. Using associative content-addressable memories to control robots. In *Proceedings of the IEEE Conference on Decision Control*, pages 792–797, Austin, TX, December 1988.

[5] S. N. Balakrishnan and V. Biega. Adaptive-critic-based neural networks for artificial optimal control. *Journal of Guidance, Control, and Dynamics*, 19(4):893–898, July-August 1996.

[6] A. G. Barto. Connectionist learning for control. In R. S. Sutton W. T. Miller and P. J. Werbos, editors, *Neural Networks for Control*, chapter 1, pages 5–58. MIT Press, Cambridge, Massachusetts, London, England., 1990.

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

[8] R. E. Bellman. *Adaptive Control Processes.* Princeton University Press, Princeton, NJ, 1961.

[9] R. E. Bellman. *Applied Dynamic Programming.* Princeton University Press, Princeton, NJ, 1962.

[10] B. Bhat and T. J. McAvoy. Use of neural nets for dynamic modeling and control of chemical process systems. *Comput. Chem. Eng*, 14:573–583, 1990.

[11] S. Billings and S. Chen. Neural networks and system identification. In K. Warwick, G. W. Irwin, and K. J. Hunt, editors, *Neural Networks for Control Systems*, pages 181–205. Peter Peregrinus Ltd, London, 1992.

[12] C. M. Bishop. Mixture density networks. Technical report NCRG 4288, United Kingdom, Birmingham, Aston University, Neural Computing Research Group, 1994.

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

[14] M. A. Botto, B. Wams, van den Boom, and J. M. G. Sà 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.

[15] M. A. Botto, B. Wams, van den Boom, and J. M. G. Sà da Costa. Stability analysis of nonlinear plants under approximate feedback linearization. In *J. Maciejowski (eds.); CONTROL 2000 proceedings. UKACC International Conference on Control*, pages 1–6, Cambridge, UK, September 2000. project code: ET00-06.

[16] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[17] S. M. Carroll and B. W. Dickinson. Construction of neural nets using the radon transform. In *IEEE International Joint Conference on Neural Networks - IJCNN'89*, pages 607–611, 1989.

[18] F. C. Chen and H. K. Khalil. Adaptive control of a class of nonlinear discrete-time systems using neural networks. *IEEE Transactions on Automatic Control*, 40(5):791–801, May 1995.

[19] Fu-Chuang Chen. Back-propagation neural networks for nonlinear self-tuning adaptive control. *IEEE Control System Magazine*, pages 44–48, 1990.

[20] D. L. Chester. Why two hidden layers are better than one. In *IEEE International Joint Conference on Neural Networks - IJCNN'90*, pages 265–268, 1990.

[21] G. Cybenko. Approximation by superpositions of sigmoidal functions. *Math. Control Signals Systems*, 2:303–314, 1989.

[22] T. J. A. de Vries, L. J. Idema, and W. J. R. Velthuis. Parsimonious learning feed-forward control. In *ESANN'1998 Proceedings- European Syposium on Artificial Neural Networks*, pages 85–90, Bruges(Belgium), April 1998.

[23] T. J. A. de Vries, W. J. R. Velthuis, and J van Amerongen. Learning feed-forward control: A survey and historical note. In *Proceedings* 1st *IFAC Conference on Mechatronic Systems*, pages 949 –954, Germany, September 2000.

[24] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[25] D. J. Evans. *A Pragmatic Bayesian Approach to Wind Field Retrieval*. Phd, Aston University, August 2001.

[26] 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.

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

[28] S. Feteih and D. Sadhukhan. Training strategy for back-propagation neural networks using input weighting. In *Proceedings of the American Control Conference*, volume 2, pages 1384–1385, June 1994.

[29] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[30] F. Gao, F. Wang, and M. Li. A simple nonlinear controller with diagonal recurrent neural network. *Chemical Engineering Science*, 55:1283–1288, 2000.

[31] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[32] F. Girosi and T. Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, 1:465–469, 1989.

[33] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

[34] H. Gomi and M. Kawato. Recognition of manipulated objects by motor learning with modular architecture networks. *Neural Networks*, 6:485–497, 1993.

[35] A. M. Graybiel, T. Aosaki, A. W. Flaherty, and M. Kimura. The basal ganglia and adaptive motor control. *Science*, 265:1826–1831, 23 September 1994.

[36] J. A. Gustafson and P. S. Maybeck. Flexible spacestructure control via moving-bank multiple model algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 30(3):750–757, July 1994.

[37] S. Haykin. *Neural Networks, A Comprehensive Foundation*. Macmillan, 1994.

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

[39] R. Herzallah and Y. Al-assaf. Control of non-linear and time-variant dynamic systems using neural networks. In *Fifth International Conference on Prod. Eng. and Design For Development (PEDD5)*, pages 53–62, Ain Shams University, Egypt, April 1998.

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

[41] 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.

[42] 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*, volume 2, pages 387–392, Faro, Portugal, April 2003.

[43] R. Herzallah and D. Lowe. Probability distribution modelling to improve stability in nonlinear MIMO control. In *IEEE Conference on Control Applications, CCA*, volume 2, pages 954–959, Istanbul, Turkey, June 2003.

[44] R. Herzallah and D. Lowe. Robust control of nonlinear stochastic systems by modelling conditional distributions of control signals. *Neural Computing and Applications*, 2003. Accepted.

[45] J. P. and D. Liberzon Hesbanha and A. S. Morse. Logic-based switching control of a nonholonomic system with parametric modelling uncertainty. *System and Control Letters (Special Issue on Hyprid Systems)*, 38:167–177, November 1999.

[46] K. Hornik. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[47] 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*, volume 2, pages 1555–1560, Anchorage, Alaska, USA, May 2002.

[48] N. Hovakimyan, F. Nardi, and A. J. Calise. A novel observer based adaptive output feedback approach for control of uncertain systems. In *Proceedings of the American Control Conference*, volume 3, pages 2444–2449, Arlington, VA, USA, June 2001.

[49] R. A. Howard. *Dynamic Programming and Markov Processes.* The Massachusetts Institute of Technology and John Wiley and Sons, Inc., New York. London, 1960.

[50] 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.

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

[52] A. Isidori. *Nonlinear Control Systems, An Introduction.* Springer-Verlag, Berlin, Heidelberg, 1985.

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

[54] T. A. Johansen and B. A. Foss. Identification of non-linear system structure and parameters using regime decomposition. *Automatica*, 31(2):321–326, 1995.

[55] M. I. Jordan. Supervised learning and systems with excess degrees of freedom. Coins technical report 88-27, 1-41, Amherst, University of Massachusetts, Department of Computer Science and Information, 1988.

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

[57] 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, November 18 1993.

[58] V. D. and D. Popovic Kalanovic and N. T. Skaug. Feedback error learning neural network for trans-femoral prosthesis. *IEEE Transactions on Rehabilitation Engineering*, 8(1):71–80, March 2000.

[59] M. Kavchak and H. Budman. Adaptive neural network structures for non-linear process estimation and control. *Computers and Chemical Engineering*, 23:1209–1228, 1999.

[60] 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.

[61] M. and K. Furukawa Kawato and R. Suzuki. A hierarchical neural network model for control and learning of voluntary movement. *Biological Cybernetics*, 57:169–185, 1987.

[62] K. Kim and E. B. Bartlett. Error estimation by series association for neural network systems. *Neural Computation*, 7:799–808, 1995.

[63] D. Kincaid and W. Cheney. *Numerical Analysis*. Brooks/Cole Publishing Company, 1996. Second Edition.

[64] A. N. Kolmogorov. On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR*, 114:953–956, 1957.

[65] M. Krstič, J. Sun, and P. V. Kokotovi'. Control of feedback linearizable systems with input unmodeled dynamic. In *Proceedings of the 33rd Conference on Decision and Control*, pages 1633–1638, December 1994.

[66] K. Kuhnen and H. Janocha. Adaptive inverse control of piezoelectric actuators with hysteresis operators. In *Proceedings of the European Control Conference*, Karlsruhe, August 1999.

[67] M. Kuperstein. Neural model of adaptive hand-eye coordination for single postures. *Science*, 239:1308–1311, 1988.

[68] D. G. Lainiotis. Partitioninig: A unifying framework for adaptive systems, 1: Estimation. *Proceedings of the IEEE*, 64(8):1126–1143, August 1976.

[69] 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.

[70] R. E. Larson. *Principles of Dynamic Programming*, volume 7 of *Control and Systems Theory*. M. Dekker, New York, 1978.

[71] S. Lefschetz. *Stability of Nonlinear Control Systems*. Academic Press Inc., New York. London, 1965.

[72] D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Control System Magazine*, 19:59–70, October 1999.

[73] R. P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4:4–22, April 1987.

[74] Y. Liu. Robust parameter estimation and model selection for neural network regression. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advance in Neural Information Processing Systems*, volume 6, pages 192–199, San Mateo, CA: Morgan Kaufmann, 1994.

[75] D. Lowe. Characterising complexity by the degrees of freedom in a radial basis function network. *Neurocomputing*, 19:199–209, 1998.

[76] 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.

[77] D. J. C. Mackay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[78] D. J. C. Mackay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.

[79] D. J. C. Mackay. Gaussian processes a replacement for supervised neural networks. Lecture notes for NIPS, United Kingdom, Cambridge University, Department of Physics, November 1997.

[80] P. S. Maybeck and R. D. Stevens. Reconfigurable flight control via multiple model adaptive control methods. *IEEE Transactions on Aerospace and Electronic Systems*, 27(3):470–480, May 1991.

[81] M. B. McFarland and A. J. Calise. Robust adaptive control of uncertain nonlinear systems using neural networks. In *Proceedings of the American Control Conference*, volume 3, pages 1996–2000, June 1997.

[82] W. T. Miller. Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE Journal of Robotics and Automation*, RA-3(2):157–165, April 1987.

[83] W. T. Miller, F. H. Glanz, and L. G. Kraft. Application of a general learning algorithm to the control of robotic manipulators. *The International Journal of Robotics research*, 6(2):84–98, 1987.

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

[85] V. Mises. *Mathematical Theory of Probability and Statistics.* Academic Press, New York., 1964.

[86] 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.

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

[88] 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.

[89] K. S. Narendra. Adaptive control using neural networks. In D. A. White and D. Sofge, editors, *Handbook of Intelligent Control*, chapter 5, pages 141–183. Multiscience Press, Inc, New York, 1992.

[90] K. S. Narendra and J. Balakrishnan. Improving transient response of adaptive control systems using multiple models and switching. *IEEE Transactions on Automatic Control*, 39(9):1861–1866, September 1994.

[91] K. S. Narendra and J. Balakrishnan. Adaptive control using multiple models. *IEEE Transactions on Automatic Control*, 42(2):171–187, February 1997.

[92] K. S. Narendra, J. Balakrishnan, and M. K. Ciliz. Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems Magazine*, 15:37–51, June 1995.

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

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

[95] K. S. Narendra and K. Parthasarathy. *Neural Networks in Control Systems.* 1990.

[96] K. S. Narendra and K. Parthasarathy. Gradient methods for the optimization of dynamical systems containing neural networks. *IEEE Transactions on Neural Networks*, 2(2):252–262, March 1991.

[97] R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical report no. 9702, University of Toronto Department of Statistics, January 1997.

[98] R. Neuneier, F. Hergert, W. Finnof, and D. Ormoneit. Estimation of conditional densities: a comparison of approaches. In M. Mrinaro and P. G. Morasso, editors, *International Conference on Artificial Neural Networks, ICANN94*, volume 1, pages 689–692. Springer-Verlag, 1994.

[99] K. Ogata. *Discrete-Time Control Systems.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[100] G. L. Plett. *Adaptive Inverse Control of Plants with Disturbances.* Phd, Stanford University, May 1998.

[101] T. Poggio and F. Girosi. Networks for approximation and learning. *IEEE*, 78(9):1481–1497, September 1990.

[102] M. Pottmann and D. E. Seborg. A radial basis function control strategy and its application to a pH neutralization process. In *Proceedings of the European Control Conference*, Groningen, the Netherlands, June 1993.

[103] D. V. Prokhorov and L. A. Feldkamp. Primitive adaptive critics. In *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks (ICNN'97)*, pages 2263–2267, Houston, TX, June 1997.

[104] D. V. Prokhorov, R. A. Santiago, and D. C. Wunsh II. Adaptive critic designs: A case study for neurocontrol. *Neural Networks*, 8(9):1367–1372, 1995.

[105] D. V. Prokhorov and D. C. Wunsch. Adaptive critic designs. *IEEE Transactions on Neural Networks*, 8(5):997–1007, September 1997.

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

[107] C. S. Qazaz. *Bayesian Error Bars for Regression.* Phd, Aston University, November 1996.

[108] H. E. Rauch. Autonomous control reconfiguration. *IEEE Control Systems Magazine*, 15:37–48, December 1995.

[109] W. J. Runggaldier. Concepts and methods for discrete and continuous time control under uncertainty. *Insurance: Mathematics and Economics*, 22:25–39, 1998.

[110] J. La Salle and S. Lefschetz. *Stability by Liapunov's Direct Method.* Academic Press Inc., New York. London, 1961.

[111] R. M. Sanner and J. E. Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3(6):837–863, 1992.

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

[113] M. A. Shah and P. H. Mecki. Using saturation detection to shorten the training duration for Gaussian ANNs. In *Proceedings of the American Control Conference*, volume 2, pages 1366–1367, June 1994.

[114] T. T. Shannon. Partial, noisy and qualitative models for adaptive critic based neurocontrol. In *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks (ICNN'99)*, Washington D.C., 1999.

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

[116] Y. Takahashi. Generalization and approximation capabilities of multilayer networks. *Neural Computation*, 5:132–139, 1993.

[117] H. A. B. te Braake, M. A. Botto, H. J. L. van Can, José Sá da Costa, and H. B. Verbruggen. Linear predictive control based on approximate input-output feedback linearisation. *IEE Proceedings - Control Theory and Applications*, 146(4):295–300, 1999.

[118] R. Tibshirani. A comparison of some error estimates for neural network models. *Neural Computation*, 8:152–163, 1996.

[119] B. and M. Stankovi'c Todorović and C. Moraga. Extended Kalman filter trained recurrent radial basis function network in nonlinear system identification. In *International Conference on Artificial Neural Networks, ICANN*, pages 819–824, Madrid, Spain, August 2002.

[120] T. J. J. van den Boom. Robust nonlinear predictive control using feedback linearization and linear matrix inequalities. In *Proceedings of the American Control Conference*, volume 5, pages 3068–3072, Albuquerque New Mexico, June 1997.

[121] Xiaotong Wang, Chin-Chen Chang, and Fang Du. Achieving a more robust neural network model for control of a MR damper by signal sensitivity analysis. *Neural Computing and Applications*, 10:330–338, 2002.

[122] P. E. Wellestead and M. B. Zarrop. *Self-Tuning Systems Control and Signal Processing.* Wiley, Great Britain., 1987.

[123] P. J. Werbos. Approximate dynamic programming for real-time control and neural modeling. In D. A. White and D. A. Sofge, editors, *Handbook of Intelligent Control*, pages 493–525. Multiscience Press, Inc, New York, N.Y., 1992.

[124] D. A. White and M. I. Jordan. Optimal control: A foundation for intelligent control. In D. A. White and D. Sofge, editors, *Handbook of Intelligent Control*, chapter 6, pages 185–214. Multiscience Press, Inc, New York, N.Y., 1992.

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

[126] B. Widrow and G. L. Plet. Nonlinear adaptive inverse control. In *Proceedings of the 36th Conference on Decision and Control*, pages 1032–1037, San Diego, California, USA, December 1997.

[127] C. K. I. Williams. Computation with infinite neural networks. Techreport NCRG/97/025, United Kingdom, Aston University, Neural computing Research Group, September 1997.

[128] C. K. I. Williams. Prediction with Gaussian processes from linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning in Graphical Models*, chapter 9, pages 599–621. MIT Press, Cambridge, Massachusetts, London, England., 1999.

[129] P. H. Winston. *Artificial Intelligence*. Addison-Wesley Publishing Company, 1992.

[130] D. H. Wolpert. Stacked generalisation. *Neural Networks*, 5:241–259, 1992.

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

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

[133] G. G. Yen and Liang-Wei Ho. Fault tolerant control: An intelligent sliding mode control strategy. In *Proceedings of the American Control Conference*, volume 6, pages 4204–4208, Chicago, Illinois, June 2000.