Neural Networks For Modelling and Control of a Non-Linear Dynamic System

Roderick Murray-Smith, Dietmar Neumerkel,
Daimler-Benz Research Group, Alt-Moabit 91B, W–1000 Berlin 21, Germany.

Daniel Sbarbaro-Hofer,
Control Group, Mechanical Engineering, Glasgow University, Glasgow G12 8QQ, Scotland

Abstract

Artificial neural nets can provide a general framework for non-linear modelling and control. This paper describes the use of neural nets to model and control a non-linear second order electromechanical model of a drive system with varying time constants and saturation effects. A Model Predictive Control structure, with neural network model is used. This is compared with a PI controller with regards to performance and robustness against disturbances. Two types of network architecture were compared: Multi-layer Perceptrons and Radial Basis Function Nets. The problems involved in the transfer of connectionist theory to practice are discussed.

Key words: Artificial Neural Nets, Modelling, Model Predictive Control, Drive Systems, Non-linear Dynamic Systems.

Notation used

T^*	– Torque reference
T_L	– Load torque
T_{el}	– Electrical torque
T_{me}	– mechanical torque
T_{max}	– T^* saturation level
T_{max0}	– initial saturation level
τ_{init}	– initial motor time constant
τ	– motor time constant
J	– inertia
ω	– shaft speed
ω_T	– threshold speed
t_s	– sampling time
t_h	– Prediction horizon
u_k	– input at time k
y_k	– output at time k
y_{ref}	– target output
y_{model}	– model output
y_{opt}	– model prediction used by optimisation routine
$d_{1\&2}$	– plant and measurement disturbances
d'	– estimated disturbance
E	– modelling error
E_{max}	– worst modelling error in training set
I	– Index of the controller quality
K_p	– Proportional constant in PI controller
K_i	– Integral constant in PI controller

Neural Networks in Modelling and Control

The conventional method of modelling and controlling dynamic processes is to try to form a physically based mathematical model of the system being studied which approximates the input-output relationship observed from the real system. For the sake of practicality, this usually involves the engineer making simplifying assumptions during the modelling phase, which results in poorer control performance, and requires a great deal of experience. Non-linear systems present particular problems since there is no simple and unifying body of theory available for this case.

It is often claimed that learning systems such as neural nets offer an alternative, faster, better solution to such problems. The networks are inherently non-linear and multi-variable. They can *theoretically* learn arbitrary input-output mappings of data which can be both quantitative and qualitative. Hunt and Sbarbaro have suggested the use of conventional control structures which are ideal for neural networks [4].

This paper describes an implementation of some of these ideas, with the intention of evaluating their applicability and any difficulties, on the basis of a simple example.

Modelling & Control of Non-linear Dynamic Systems

The problem considered involves the speed control of a second order system based on a rough model of an electromechanical drive system, which can be controlled using normal control techniques, but which also provides some challenging problems. Five models of varying complexity have been analysed (Systems A–E).

The basic model (System E) is shown in Figure 1. The model has internal feedback effects, which are introduced by making the saturation limit on the control input inversely proportional to ω, when $\omega > \omega_T$. The variable ω is available for feedback.

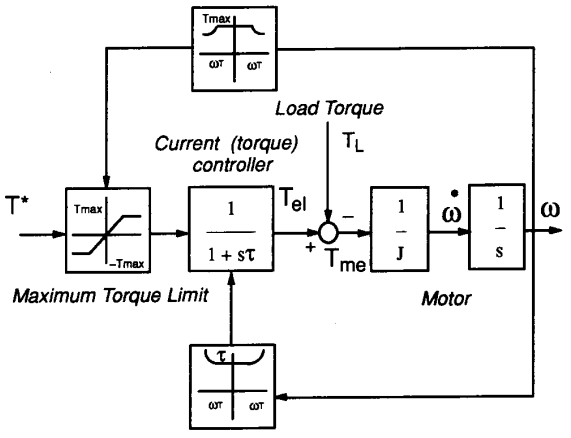

Figure 1 : The model with varying time constant and saturation (Syst. E)

The saturation and time constant varying as shown below:

$$T_{max} = \frac{T_{max0}}{\left(1 + \left(\frac{|\omega| - \omega_T}{\omega_T}\right)\right)}$$

$$\tau = \tau_{init} \left(1 + \left(\frac{|\omega| - \omega_T}{\omega_T}\right)\right)$$

Systems A to D are varying configurations of this model, described later.

Interesting areas in this problem are:

1. Modelling a non-linear system using neural nets.
2. Modelling a dynamic system
3. Use of neural nets in standard control structures for more robust control.
4. The load torque was viewed as an unmeasurable disturbance.

Modelling the Plant

A major proportion of any control project is spent trying to form an adequate model of the plant. This is where it is claimed that neural networks have a great advantage over classical techniques because they can be 'trained' with observed data from the real plant (or an existing model of the plant), to reproduce the characteristics of the plant, as a black box model. The neural networks are used as non-linear, multi-variable function approximation tools.

The 'training' involves the optimisation of the weights in the network until the knowledge distributed among them is sufficient to provide a black-box model of the plant. There are different methods of adjusting the weights (learning algorithms), and different neural architectures.

Dynamic Models

The construction of a dynamic model with neural nets is a more difficult task than that of a static model. To ensure that past inputs are considered, the problem can be transformed from a temporal one to spatial one by supplying the past values of inputs and outputs as new input dimensions. An alternative to this is the use of recurrent networks, which contain internal feedback connections.

Two feed-forward network types, the Multi-Layer Perceptron (MLP) and Radial Basis-Function Nets (RBF), were used to model the system. As this system is second order, the discretised form of the function can be described by an equation of the form $y_k = f(u_{k-1}, u_{k-2}, y_{k-1}, y_{k-2})$. The use of neural networks to describe a non-linear dynamic system is based on the assumption that the system can be represented by a non-linear auto-regressive moving average model with exogenous inputs (NARMAX) over the whole operation envelope. The validity of this assumption is discussed by Leontaritis and Billings in [6].

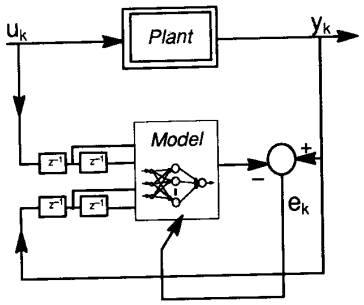

Figure 2 : Training a neural model of a dynamic plant

Creating the Training Data

A training set of input output pairs is needed to provide sufficient information to model the non-linear plant. An input signal must be chosen which will excite all the dynamic modes of the system and cover the whole amplitude range of interest, otherwise the validity of the model will become highly input sensitive. The aim of this work was to see how feasible it is to try and form a non-linear dynamic model using a training set obtained by measuring the inputs to, and outputs from the real plant, using a minimum of *a priori* knowledge.

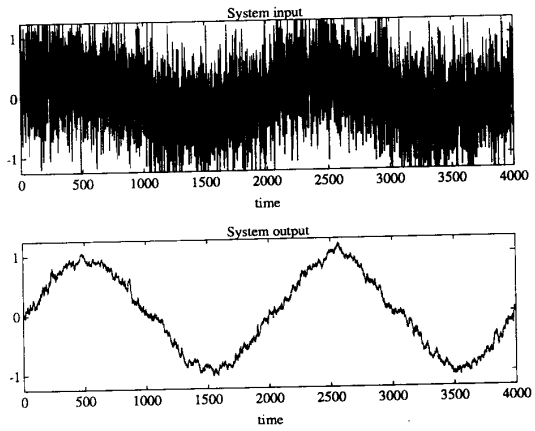

Figure 3 : Training data for the neural model of the plant

The data used for training was obtained by putting the model in a simple proportional feedback control structure. The choice of an exciting reference trajectory for a non-linear dynamic system is not easy and intuition is not always optimal [2].The reference signal used during the data acquisition phase was a low frequency sinusional input which was disturbed by white noise. The sinusoidal basis ensured that the training examples were taken from the whole input space. The white noise ensured that throughout the input space the network had been exposed to the system's dynamics by making abrupt changes to the reference inputs. It is important that the basic sine wave is of a low enough frequency that the plant can easily keep up with it. If a plant spends most of its time at a particular operating

point, it is also sensible to provide more training examples in this area.

The quality index used to determine the model performance over the training set was the average absolute error from each pattern p:

$$E = \sum_{p=1}^{total} \left\| \frac{y_{ref}(p) - y_{model}(p)}{total} \right\|$$

also important is the worst error found in the data set.

$$E_{max} = \max\left(\left\| y_{ref}(p) - y_{model}(p) \right\|\right)$$

The creation of test and training sets by stimulating non-linear dynamic systems is a complex area which requires further study. As the model quality is dependent on the training set quality, an important research goal is to produce a general, clear procedure for the production of a training set, as for the techniques necessary to test the trained model and controller.

Measurement Noise and Disturbances

Many systems are prone to unmeasureable disturbances. If possible, these should be kept to a minimum during training, as they provide conflicting training data. This does not mean that the system cannot learn, just that the results are usually poorer. The unmeasurable factor in this experiment was the load, T_L. During the collection of the training data, this was maintained at a constant operating point. Performance can be improved by modelling the disturbances and using this to improve control.

In a real system, the measurements of the training data are likely to be subject to noise, and the system output is going to be influenced by unpredictable disturbances. The training set will, therefore, have contradictory examples and an exact fit to the training set will not be possible. The job of the neural net is therefore; to find a generalisation which represents the underlying system, while ignoring the noise, or which finds the underlying characteristics of the disturbances.

Simulation Environment

All experiments were carried out in a MATLAB/SIMULAB programming environment. This enabled the combination of neural net ideas with standard control and signal processing toolboxes.

Modelling Results

All results in this section refer to the same sets of training data. This consisted of 4000 input–output pairs. The figures for average and worst error shown in Tables 1 & 2, refer to the results of a test run of 1000 input–output pairs not presented during the training phase.

The radial basis function (RBF) network used was a standard RBF Net, with Gaussian units, as described in [7][8]. The optimisation of the weights in these networks is linear, and fast, if the centres are fixed beforehand.

The simplest method of placing the centres is simply to cover the input space with a regular pattern of Gaussian units, in a similar technique to that of Albus [1]. We experimented with this method, but the problem with was that to achieve an adequate resolution to approximate the transfer function required a large number of units with very restricted receptive fields. This then lead to the requirement of even more training data, and long training times, which proved to be impractical.

Moody and Darken [7] used a fully supervised method which attempted to optimise weights, centres and radii together, but this was not particularly successful. They then developed a hybrid method, where the centres are chosen using k-means clustering and the radii using a global first nearest neighbours statistic. The weights were then optimised using LMS optimisation.

A similar solution to Moody and Darken's hybrid one was chosen. The basis function centres were placed using a clustering technique, which reduced the number of units required dramatically. The unit radii were set according to the proximity of the neighbouring centres. The weights were then adapted using one-pass training by the calculation of the pseudo-inverse.

The Back-Propagation algorithm (BP) [3] was used to train multilayer perceptrons (MLP) for comparison, as it is the most widely used learning algorithm.

The ability of the two neural architectures to model the varying degrees of non-linearity is compared in Tables 2 and 3.

System parameters were:
$J = 1\ kgm^2$, $\tau = 0.1$ s, $T_L = 0.0$ Nm, $T_{max} = 1$ Nm,
$\omega_T = 0.5\ rads^{-1}$.

The sampling time was $t_s = 0.05$ s. A discrete representation of the plant was used. The inputs and outputs were all normalised to lie within the range 0 to 1. The various systems (A–E) are described in Table 1 below.

System	Description
A	$T_{max}=\infty$, fixed τ.
B	Fixed T_{max}, fixed τ.
C	Varying T_{max}, fixed τ.
D	Fixed T_{max}, varying τ.
E	As in Figure 1, Varying T_{max}, varying τ.

Table 1 : Systems A–E with varying complexity.

The results shown in Table 2 are for the RBF Net. The numbers in brackets are the results of a test set of data, which had not been used during network training.

System	No. units	Av. error	Max error	Max $\Delta\omega$	Ave. $\Delta\omega$
A	118	0.08 (0.11)	0.32 (1.18)	2.80	0.59
B	119	0.08 (0.11)	0.35 (1.55)	2.06	0.59
C	114	0.07 (0.08)	0.28 (2.35)	2.02	0.57
D	120	0.07 (0.08)	0.28 (2.35)	2.11	0.56
E	121	0.07 (0.08)	0.31 (1.77)	1.91	0.50
F	114	0.12 (0.33)	0.95 (4.62)	2.12	0.42

Table 2 : Modelling results for the RBF Net

System	No. hidden units	Av. error	Max error	Max $\Delta\omega$	Ave. $\Delta\omega$
A	20	0.11 (0.11)	0.4 (0.5)	2.80	0.59
B	20	0.15 (0.16)	1.1 (1.1)	2.06	0.59
C	20	0.15 (0.15)	1.1 (1.3)	2.02	0.57
D	20	0.11 (0.12)	1.1 (1.2)	2.11	0.56
E	20	0.17 (0.18)	1.3 (1.5)	1.91	0.50

Table 3 : Modelling results using MLP trained with BP.

The most basic form of BP, with no 'improvements' was used, for simplicity. The standard three-layered network, with one input layer, one hidden layer and one output layer, was used. The hidden layer contained 20 units. Different sizes of networks were tested, but increasing the size did not bring any improvement in performance.

A comparison of the model accuracies described in the above tables shows a consistantly higher modelling accuracy from the RBF nets. This is consitent with claims that RBF nets have the better approximation ability, compared to multi-layer perceptrons[8].

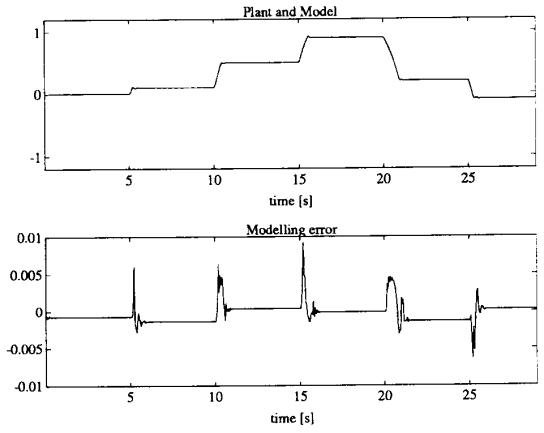

Figure 4 : The output of the RBF model of System E, operating in parallel with the plant.

A plot of the model working in parallel with the plant is shown in Figure 4. Note that only one line can be seen in the upper plot, as the model and plant have such similar behaviour.

It is also interesting to note that for both BP and RBF, the different types of non-linearities in the various systems have little effect on the accuracy of the final model.

Model Predictive Control

Model Predictive Control (MPC) is one of many conventional control structures suitable for use with neural networks.

If it is possible to model the system dynamics adequately with a neural network, the trained network can be used to provide predictions of the plant response over a specified prediction horizon t_h [9]. These predictions can then be used by an optimisation routine to produce the optimum control output for the current time step, by minimising the index:

$$I = \sum_{i=k}^{t_h+k} \left(y_{ref}(i) - y_{model}(i)\right)^2 + \lambda_i(\Delta u(i))^2$$

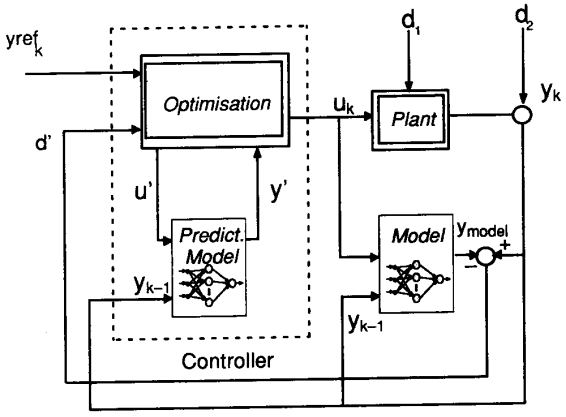

Figure 5 : Predictive control using a neural network for the model

Optimising the Controller

In this experiment, the index we used was as above, with λ_i set to zero. The control effort u was optimised by using a gradient descent algorithm to minimise the index. In other words, to adjust u, use $\frac{\partial y_{model}}{\partial u}$ to calculate the desired change in u (Δu), and apply this iteratively, until the convergence criteria are met.

$$E = \left(y_{ref} - y_{model}\right)^2$$

$$\Delta u \propto -\frac{\partial E}{\partial u}$$

$$\Delta u \propto -\frac{\partial E}{\partial y_{model}} \frac{\partial y_{model}}{\partial u}$$

$$\Delta u \propto -\left(y_{ref} - y_{model}\right) \frac{\partial y_{model}}{\partial u}$$

If the model is a neural network the numerical derivation is theoretically easy. In practice, using gradient descent is not quite so straightforward. The step size is important, local minima can occur, the convergence criteria will vary from problem to problem, and the model

must be accurate. The models produced by the Back-Propagation learning algorithm were too inaccurate to produce good control using gradient descent optimisation.

This method can also be used to produce a training set for a neural controller, where the optimal value for each starting point in the optimisation process is found. Training another neural network on this data will result in improved run-time efficiency, as the on-line optimisation is computationally expensive.

Control Results compared to PI Control

The two control algorithms; Proportional+Integral (PI), and Model Predictive Control (MPC) were tested with the reference inputs shown below, controlling System E. A PI controller was chosen, not only because of its simplicity, but also because it is still a very widely used controller for drive systems [5].

To prevent controller windup of the PI controller in the saturation area, the control variable was limited between $-T_{max}$ and T_{max}. The control law is therefore:

$$u(k) = \text{limit}(u(k-1) + K_p e(k) + (K_i - K_p)e(k-1), [-T_{max}\ T_{max}]),$$

The PI constants were tuned using the Ziegler-Nichols method, giving $K_p = 6.75$ & $K_i = 19.62$.
The RBF Model for System E was used for the Model Predictive Control. The prediction horizon t_h was set to be two steps ahead of the current time.

The minimum expectation is that the controller should have a steady state error of zero (i.e. matches the integral part of the classical PI controller). For step changes in the reference input, the percent overshoot of the actual output should not be more than 5%. The settling time to within 2% of the desired value should be minimised. The load cannot be measured and can change instantaneously.

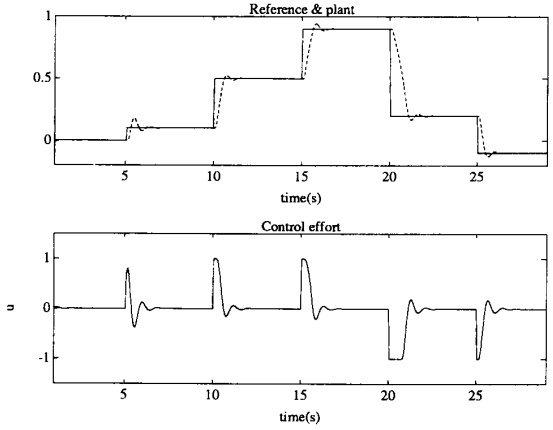

Figure 6 : Controlling system E with a PI Controller.

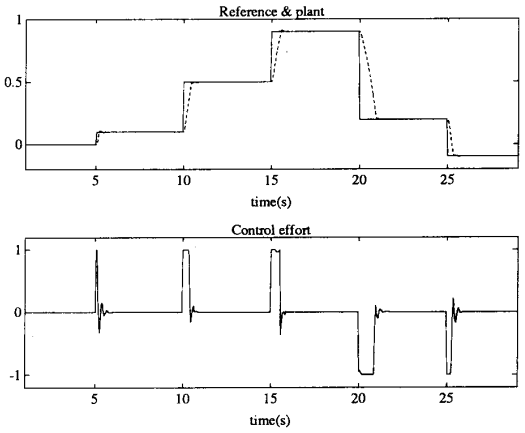

Figure 7 : Model Predictive Controller with a neural model.

Controller	Index
PI	6.710
RBF MPC	5.733

Table 4 : Performance index for controllers, without disturbances.

Comparing the above plots, it is obvious that the MPC approach provides better control. There is less overshoot and oscillation, which is to be expected, taking the simplicity of the PI controller into account. The design of a linear controller for a non-linear plant will always have to make trade-offs of performance at various operating points. This is reflected in the indices in Table 4.

Optimising Control Effort, with Disturbances

The difference between y_{model} and y_k is effectively an estimate of the disturbance acting on the plant, as well as the modelling error. This can be included in the predictive model, during optimisation at time k, as shown in the equations below. d' is taken to be a step-like disturbance. y_{opt} is the predicted motor speed used by the optimisation routine.

$$y_{opt}(i+1) = y'(i+1) + d'(i)$$

$$d'(i+1) = \begin{cases} d'(i), & i > k \\ y(i) - y_{model}(i), & i = k \end{cases}$$

This can be clarified by imagining a disturbance caused by an increase in the load. If the estimated disturbance were not to be taken into account, the optimisation routine would produce a control effort inadequate for the new situation, being modelled on the systems response with a lower load.

Controlling Disturbances

The controller was applied to a system with disturbances. The load was increased and decreased, as shown in Figure 8.

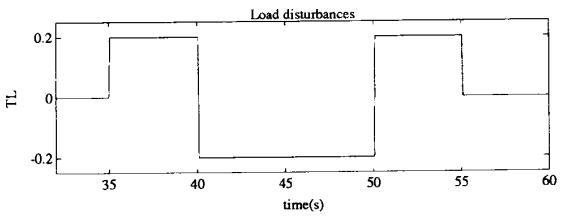

Figure 8 : Load disturbances

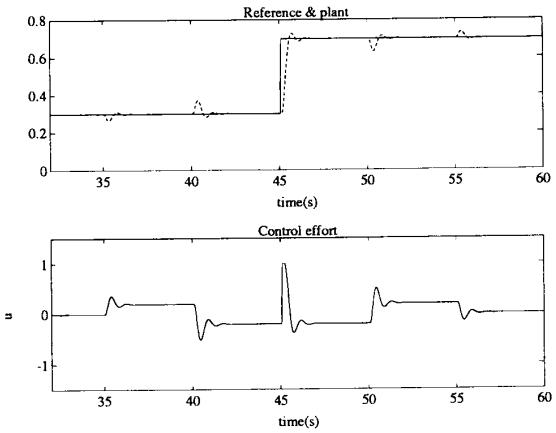

Figure 9 : PI Controller with disturbances

The Model Predictive Controller (MPC) coped with the disturbances well, returning to the reference value, then producing a control signal which exactly balanced out the disturbance, without overshoot, or oscillation, unlike the PI controller. The relative performance can also be observed by comparing the indices.

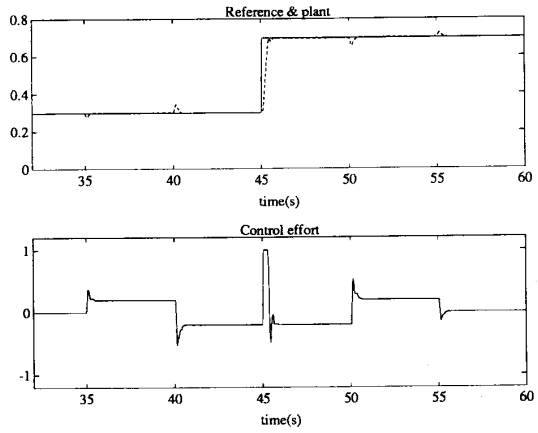

Figure 10 : Model Predictive Control with disturbances

Controller	Index
PI	0.788
RBF MPC	0.578

Table 5 : Performance index for both controllers, with disturbances.

Conclusions

An accurate model of a non-linear second order system was created by exciting the system, measuring the inputs and outputs, and using this data to train a neural model. RBF nets produced more accurate models of the system than the MLP architecture trained with Back-Propagation.

The system was then controlled by using a standard model predictive control structure with the trained neural network model, achieving better performance than a PI controller with anti-windup. This provides a stepping stone to more imaginative use of such learning algorithms on more realistic systems.

The experience gained during this work also indicates that more widespread use of neural networks in practical applications requires generally reliable engineering strategies for the implementation of these new techniques. Such strategies are far from complete.

A priori knowledge about the system being modelled is not disregarded when neural nets are used, contrary to claims made by many neural net researchers. It plays a vital role implicitly; the choice of input variables to the model, the limits on these variables, the construction of the training data, and the choice of control structure are all factors which are heavily influenced by an understanding of the system. Neural networks must be seen as an extension to the conventional modelling and control methods, rather than a replacement.

Acknowledgements

We would like to thank Dr. J. Böcker from AEG Automation for suggesting this control problem, and providing useful help and advice.

References

[1] J. S. Albus, *Data Storage in the Cerebellar Model Articulation Controller (CMAC)*,Trans. ASME, Journal of Dynamic Systems, Measurement & Control, 97, pp220–227, 1975.

[2] Brian Armstrong, *On Finding 'Exciting' Trajectores for Identification Experiments involving Systems with Non-Linear Dynamics*, IEEE Robotics and Automation 87, Raleigh, pp1131-1139, Vol. 2.

[3] Robert Hecht-Nielsen, *Theory of the Backpropagation Neural Network*, IJCNN, Vol. I, pp593–605, San Diego, June 1990.

[4] K. J. Hunt, D. Sbarbaro, *Neural networks for control and systems*, Eds. K. Warwick, G. W. Irwin, K. J. Hunt, IEE Control Eng. series 46, Peter Peregrinus Ltd., 1992.

[5] Masato Koyama, Masao Yano, *Two degrees of freedom speed controller using reference system model for motor drives*, 4th European Conf. on Power Electronics & Appl., Vol. 3, pp 60-65, Firenze 1991.

[6] I. J. Leontaritis, S. A. Billings, *Input-Output Parametric Models for non-linear systems*, Int. Journal of Control, Vol. 41, No. 2, pp329-344, 1985.

[7] J. Moody, C. Darken, *Fast Learning in Networks of Locally Tuned Processing Units*, Neural Computation 1, pp281–294, 1989.

[8] Thomas Poggio, Federico Girosi, *Networks for Approximation and Learning*, Proc. IEEE, Vol 78, No. 9, Sept. 1990.

[9] Daniel Sbarbaro, Ken Hunt, *A non-linear Receding Horizon Controller, based on Connectionist Models*, 30th IEEE Conf. Decision & Control, Proceedings WI-6, Brighton, Britain, 1991.