Neural networks for nonlinear internal model control

K.J. Hunt
D. Sbarbaro

Indexing terms: Networks, modelling

Abstract: A novel technique, directly using artificial neural networks, is proposed for the adaptive control of nonlinear systems. The ability of neural networks to model arbitrary nonlinear functions and their inverses is exploited. The use of nonlinear function inverses raises questions of the existence of the inverse operators. These are investigated and results are given characterising the invertibility of a class of nonlinear dynamical systems. The control structure used is internal model control. It is used to directly incorporate networks modelling the plant and its inverse within the control strategy. The potential of the proposed method is demonstrated by an example.

1 Introduction

This paper focuses on the use of artificial neural networks for the identification and control of non-linear dynamic systems. Such networks go by other names: parallel distributed processing, or connectionist networks. The term neural networks comes from the similarity of these networks to models of brain activity.

Artificial neural networks provide a distinctive computational paradigm and have proved effective for a range of practical problems where conventional computation techniques have not succeeded.

An artificial neural network consists of many simple processing elements each having a number of inputs and a single output. The output of each element is determined as a nonlinear function of a weighted sum of the inputs. A range of activation functions can be used; in this work we consider Gaussian activation functions because of their function representation properties (discussed in Section 5.2). The basic elements are interconnected using variable strength links, or weights. The strengths of the individual links determine the overall behaviour of the network. The weights of a particular network can be chosen to achieve a desired input-output relationship. The weights are adjusted during a training period when the network is presented with a range of input patterns and its output is compared to the desired output each time. The aim is to successively drive the weights to values which make the network output equal to the target output.

Paper 8183D (C8), first received 17th September 1990 and in revised form 29th April 1991

The authors are with the Control Group, Department of Mechanical Engineering, The University of Glasgow, Glasgow G12 8QQ, United Kingdom

In this work we are primarily concerned with using artificial neural networks for dynamical systems control. The characteristics of neural networks suggest that they may be useful in certain classes of control problems. In particular, the ability of neural networks to represent arbitrary nonlinear mappings encourages the study of neural networks for complex nonlinear control problems. Relevant features of neural networks in the control context are

(i) the ability to represent arbitrary non-linear relations

(ii) adaptation and learning in uncertain systems, provided through both off-line and online weight adaptation

(iii) information is transformed to internal representation allowing data fusion, with both quantitative and qualitative signals

(iv) parallel distributed processing architecture allowing fast processing for large-scale dynamical systems

(v) architecture providing a degree of robustness through fault tolerance and graceful degradation

Each current practical design method based on linear control theory can address a subset of these problems. Although the present emphasis here is on nonlinear control problems, we note that the characteristics of artificial neural networks allow them, potentially, to address all these problems simultaneously.

The ability of artificial neural networks to represent arbitrary non-linear relations is now well established [1, 2]. This has lead to the investigation of non-linear dynamic systems modelling using neural networks [3, 4].

The ability of neural networks to represent nonlinear relations leads to the idea of using networks directly in a model-based control strategy. The idea followed here is based on the possibility of training networks to learn both a system's input/output relationship and the corresponding inverse relationship. A suitable control strategy which directly incorporates the plant model (and the corresponding inverse model) is provided by internal model control (IMC) [5, 6]. The applicability of IMC to nonlinear systems control has been demonstrated by Economou *et al.* [7]. The inverse of the nonlinear operator modelling the plant was shown to play a crucial role in the implementation of nonlinear IMC. These authors studied analytical and numerical methods for the necessary construction of nonlinear operator inverses. In the present work artificial neural networks are used for the construction of plant models and their inverses, and it is intended that they are used directly within the IMC control structure.

Results characterising the invertibility of a class of non-linear dynamic systems are presented. Architectures and algorithms for the training of neural networks for

modelling dynamical inverses are described and convergence results for these algorithms are established.

A further novel feature of our approach is the use of radial basis functions, and the Gaussian function in particular, as the nonlinear function in the network hidden units [8]. There is recent theoretical support for the 'best representation' property of such networks [9]. This approach stands in contrast to the widespread use of sigmoidal nonlinearities. To the best of our knowledge Gaussian networks have not yet been used by others in the study of dynamical systems identification and control, although this type of network has been studied in the related problem of time series prediction [10, 11].

The idea of using neural networks for nonlinear IMC has been considered by Bhat and McAvoy [12]. However, they did not investigate invertibility conditions for nonlinear dynamic systems neither did they propose learning algorithms for building the inverse models. Further, they do not appear to have implemented a neural network based nonlinear IMC controller.

In this paper we describe our work relating to these three issues. We have implemented the nonlinear IMC algorithm and performed extensive simulations, with very encouraging results. A representative sample of the results is given below.

2 Internal model control

The IMC structure is now well known and has been shown to underlie a number of control design techniques of apparently different origin [5]. IMC has been shown to have a number of desirable properties; a detailed analysis has been given by Morari and Zafiriou [6]. Here, we briefly summarise these properties.

The nonlinear IMC structure is shown in Fig. 1 (in this Section we follow Economou *et al.* [7]). The nonlinear operators denoted by P, M and C represent the plant, the plant model, and the controller, respectively. The operator F denotes a filter, discussed later in the paper. The double lines used in the block diagram emphasise that the operators are nonlinear and that the usual block diagram manipulations do not hold.

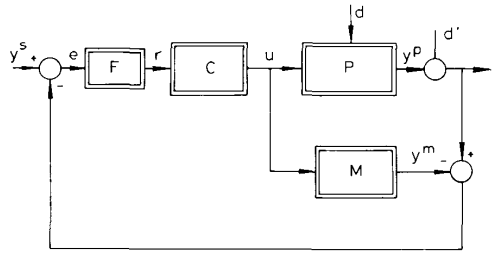

Fig. 1 *Nonlinear IMC structure*

The important characteristics of IMC are summarised with the following properties:

Property P1: Assume that the plant and controller are input–output stable and that the model is a perfect representation of the plant. Then the closed-loop system is input–output stable.

Property P2: Assume that the inverse of the operator describing the plant model exists, that this inverse is used as the controller, and that the closed-loop system is input–output stable with this controller. Then the control will be perfect, i.e. $y = y^s$.

Property P3: Assume that the inverse of the steady state model operator exists, that the steady state controller operator is equal to this, and that the closed-loop system is input–output stable with this controller. Then offset free control is attained for asymptotically constant inputs.

The IMC structure provides a direct method for the design of nonlinear feedback controllers. According to the above properties, if a good model of the plant is available, the closed-loop system will achieve exact set-point following despite unmeasured disturbances acting on the plant.

Discussion so far has considered only the idealised case of a perfect model, leading to perfect control. In practice, however, a perfect model can never be obtained. In addition, the infinite gain required by perfect control would lead to sensitivity problems under model uncertainty. The filter F is introduced to alleviate these problems. By suitable design, the filter can be selected to reduce the gain of the feedback system, thereby moving away from the perfect controller. This introduces robustness into the IMC structure. A full treatment of robustness and filter design for IMC is given by Morari and Zafiriou [6].

The IMC is significant because the stability and robustness properties of the structure can be analysed and manipulated in a transparent manner, even for nonlinear systems. Thus IMC provides a general framework for nonlinear systems control. Such generality is not apparent in alternative approaches to nonlinear control.

A second role of the filter is to project the signal e into the appropriate input space for the controller.

3 Nonlinear IMC using neural networks

We propose a two step procedure for using neural networks directly within the IMC structure. The first step involves training a network to represent the plant response. This network is then used as the plant model operator M in the control structure of Fig. 1. The architecture shown in Fig. 2 provides the method for training a network to represent the plant. Here, the error signal used to adjust the network weights is the difference between the plant output and the network output. Thus,

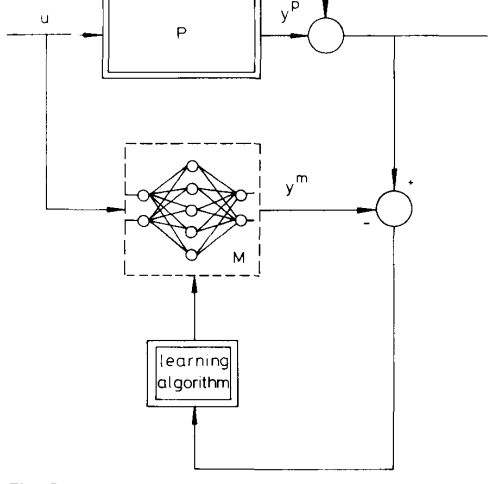

Fig. 2 *Plant identification*

the network is forced towards copying the plant dynamics. Full details of the learning law used here are given in Section 6.1.

Following standard IMC practice (guided by property P2 above) we select the controller as the plant inverse model. The second step in the procedure is to train a second network to represent the inverse of the plant. To do this we use the architecture shown in Fig. 3. Here, for

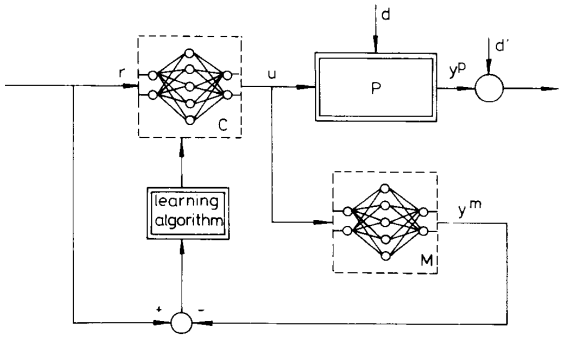

Fig. 3 *Specialised learning structure*

reasons explained in Section 6.2, we employ the plant model (obtained in the first learning step) in the inverse learning architecture rather than the plant itself. For inverse modelling the error signal used to adjust the network is defined as the difference between the (inverse modelling) network input and the plant model output. This tends to force the transfer function between these two signals to unity; i.e. the network being trained is forced to represent the inverse of the plant model. Having obtained the inverse model, this network is then used as the controller block C in the control structure of Fig. 1.

The final IMC architecture incorporating the trained networks is shown in Fig. 4.

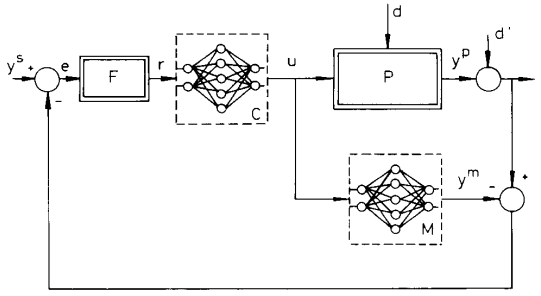

Fig. 4 *General IMC architecture*

4 Invertibility of discrete nonlinear systems

The inversion of nonlinear operators plays a central role in the development of nonlinear IMC. Here we explore the invertibility of nonlinear dynamic systems. Consider a general analytic system Σ governed by the following nonlinear difference equation

$$\Sigma: y^p(k+1) = f(y^p(k), \ldots, y^p(k-n); u(k), \ldots, u(k-m)) \quad y^p \in \mathscr{R} \quad u \in \mathscr{R} \quad (1)$$

Here, $n+1$ is the order of the system, with $m \leqslant n$. For the moment we consider single-input single-output systems; the approach can be generalised to multivariable systems.

The system Σ is called invertible at $[y^p(k), \ldots, y^p(k-n), u(k-1), \ldots, u(k-m)]^T$ if there is a subset A of $\mathscr{R}^{m+n+1}$, such that for $[y^p(k), \ldots, y^p(k-n), u(k-1), \ldots, u(k-m)]^T \in A$

$$f(y^p(k), \ldots, y^p(k-n); u^1(k), \ldots, u(k-m)) \neq f(y^p(k), \ldots, y^p(k-n); u^2(k), \ldots, u(k-m))$$

for any distinct inputs $u^1(k)$, $u^2(k)$.

The system Σ is singular if for any $[y^p(k), \ldots, y^p(k-n), u(k-1), \ldots, u(k-m)]^T \in A$ and for any distinct inputs $u^1(k)$, $u^2(k)$, the resulting outputs are equal:

$$f(y^p(k), \ldots, y^p(k-n); u^1(k), \ldots, u(k-m)) = f(y^p(k), \ldots, y^p(k-n); u^2(k), \ldots, u(k-m))$$

Proposition 1: If $f(y^p(k), \ldots, y^p(k-n); u(k), \ldots, u(k-m))$ is monotonic with respect to $u(k)$ then the system is invertible at $[y^p(k), \ldots, y^p(k-n), u(k-1), \ldots, u(k-m)]^T$.

Proof: If $f(y^p(k), \ldots, y^p(k-n); u(k), \ldots, u(k-m))$ is monotonic and $u^1(k) > u^2(k)$ then for the same point $[y^p(k), \ldots, y^p(k-n), u(k-1), \ldots, u(k-m)]^T$,

$$f(y^p(k), \ldots, y^p(k-n); u^1(k), u(k-1), \ldots, u(k-m)) > f(y^p(k), \ldots, y^p(k-n); u^2(k), u(k-1), \ldots, u(k-m))$$

In the same way,

$$f(y^p(k), \ldots, y^p(k-n); u^1(k), u(k-1), \ldots, u(k-m)) < f(y^p(k), \ldots, y^p(k-n); u^2(k), u(k-1), \ldots, u(k-m))$$

if $u^1(k) < u^2(k)$. The system is therefore invertible.

5 Network architecture

The characteristics of any network are defined by the particular nonlinear activation function of the elements and by the way in which the elements are interconnected. In this Section we define the network architecture used in the present plant and plant inverse modelling studies.

5.1 Basic elements

The simple basic element comprises differently weighted inputs that are processed by a certain nonlinear function, called the activation function. Some of the most frequently used activation functions are shown in Fig. 5. Each

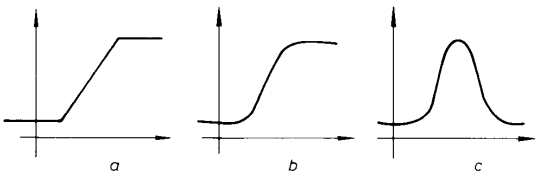

Fig. 5 *Activation functions*
a saturation
b sigmoid
c gaussian

representation of the basic unit has advantages and disadvantages. The choice of a specific unit depends on the problem being studied.

5.2 Gaussian network

The basic elements by themselves are not very powerful in terms of computation or representability but their

interconnection allows us to encode relations between the variables, giving different powerful processing capabilities. The connection of several layers, as shown in Fig. 6, gives the possibility of nonlinear mapping between the inputs and the outputs. This capability can be used to represent complex nonlinear relations among the inputs and outputs.

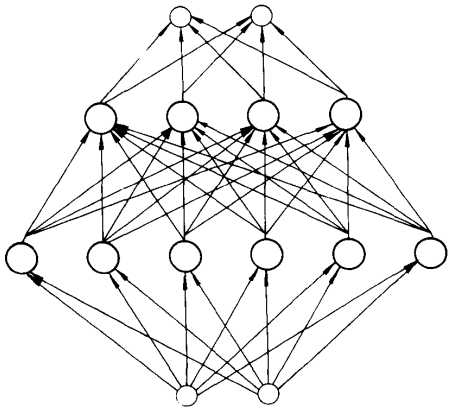

Fig. 6 *Multilayer network*
○ processing units
o fan in–out units

The basic structure having one hidden layer with sigmoid units has been shown to be powerful enough to produce an arbitrary mapping among variables [2]. The standard approach to training this type of network is the back-propagation algorithm. However, this method is known to have drawbacks in terms of speed of convergence and the attainment of a unique global solution to the optimisation problem. These problems arise mainly because the technique involves the solution of a nonlinear problem with local minima.

The use of Gaussian units, on the other hand, has the advantage of simplifying the problem, because in a natural way it produces a partition of the input space. There is theoretical support for representation and consistence problems with these networks [1]. It has been shown that this kind of network has the best approximation property [9].

In this work we now consider networks consisting of many inputs and a single output (by duplication of this structure the approach can easily be generalised to multi-output systems), as shown in Fig. 7. The number of input units n_i corresponds to the dimension of the network input vector, denoted by $x \in \mathscr{R}^{n_i}$. Each linear input unit is given an activation value corresponding to the relevant value of the input vector. The linear output unit is fully connected to the hidden units; the network output y (a nonlinear function, $g(.)$, of the input vector x) is a weighted sum of the activation levels of the N hidden units:

$$y = g(x) = \sum_{i=1}^{N} c_i K_i \tag{2}$$

Here, c_i denotes the connection weight between hidden unit i and the network output. K_i is the output (the activation value) of hidden unit i.

The activation level of a hidden unit in a Gaussian network depends only on the distance between the input vector and the centre of the Gaussian function of that unit. The centre of the hidden unit i is denoted by $x_i \in \mathscr{R}^{n_i}$. More precisely, the activation level K_i of the hidden unit i is defined as

$$K_i = e^{-d_i(x, x_i, \Delta)} \tag{3}$$

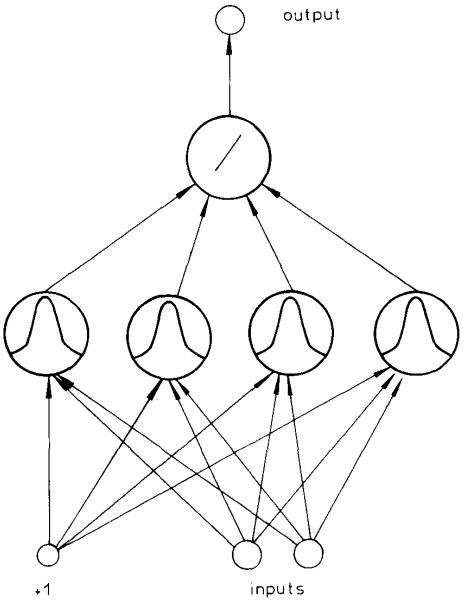

Fig. 7 *Basic Gaussian network*

Here, the distance function d_i for hidden unit i represents the scaled distance between the centre of the function x_i for that unit and the input vector x:

$$d_i(x, x_i, \Delta) = (x - x_i)^T \Delta (x - x_i) \tag{4}$$

The matrix Δ represents the bandwidth of the hidden unit functions. We consider a constant diagonal matrix Δ whose diagonal entries are equal. Having such a Δ constrains the representation but simplifies the learning algorithm. We refer to such a network as a *regular* Gaussian network. Fig. 7 shows the general architecture of this network.

6 Learning algorithms

Section 3 outlined the architectures used for plant modelling and plant inverse modelling. These are necessary in the IMC formulation. The learning laws used for training the networks are considered below.

6.1 Modelling the plant

The plant is modelled using a network described by

$$y^m(k+1) = \sum_{i=1}^{N^m} c_i^m K_i^m \tag{5}$$

where

$$K_i^m = \exp\left(-d_i^m(x^m(k), x_i^m, \Delta^m)\right) \tag{6}$$

Here, the m superscript indicates a variable related with the plant model. We choose the structure of the plant model to be the same as that of the plant (eqn. 1), i.e. the model output is a nonlinear function of the present and

past plant outputs, and the present and past plant inputs. The model input vector $x^m(k)$ is thus given as

$$x^m(k) = [y^p(k), \ldots, y^p(k-n), u(k), \ldots, u(k-m)]^T$$

We denote the centre of the Gaussian function of hidden unit i as

$$x_i^m = [y_{i,0}, \ldots, y_{i,n}, u_{i,0}, \ldots, u_{i,m}]^T$$

The parameters x_i^m and Δ^m are fixed to meet the interpolation conditions, i.e. the x_i^m are distributed uniformly over the input space and Δ^m is adjusted such that $\sum_{i=1}^{N^m} K_i = 1$ over the input space. There are other possibilities for this [13].

The parameter vector c_i^m is adjusted to minimise the mean square error between the real plant and the model; i.e.

$$c_i^m(k+1) = c_i^m(k) + \alpha K_i(y^p(k+1) - y^m(k+1)) \tag{7}$$

Here, α is a gain parameter. Fig. 2 represents the structure used. Using standard linear systems theory it can be shown that if the plant can be modelled as eqn. 5, the least mean square solution can be found by eqn. 7 [8].

6.2 Inverse model identification

If the model of the plant is invertible then the inverse of the plant can be approximated in a similar way to the plant. This model is then used as the controller. For reasons described in Section 6.2.1 we choose to use the plant model inverse rather than the inverse of the real plant. We utilise a second network described by

$$u(k) = \sum_{i=1}^{N^C} c_i^C K_i^C \tag{8}$$

where

$$K_i^C = \exp(-d_i^C(x^C(k), x_i^C, \Delta^C)) \tag{9}$$

Here, the C superscript indicates a variable related with the controller. The inverse of the function f in eqn. 1 (required to obtain $u(k)$) depends on the future plant output value $y^p(k+1)$. In order to obtain a realisable approximation we replace this value by the controller input value r. Finally, since we need to approximate the inverse of the plant model (as opposed to the plant itself), we define the controller network input vector $x^C(k)$ as

$$x^C(k) = [y^m(k), \ldots, y^m(k-n), r(k+1), u(k-1), \ldots, u(k-m)]^T$$

Here, the future value $r(k+1)$ is obtained at time k by suitable definition of the IMC filter F (see Section 7). The centre of the Gaussian function of hidden unit i is given by

$$x_i^C = [y_{i,0}, \ldots, y_{i,n}, r_i, u_{i,1}, \ldots, u_{i,m}]^T$$

6.2.1 Non-iterative methods: To adjust c_i^C the architecture shown in Fig. 3 is used. This architecture is similar to the specialised learning architecture presented by Psaltis *et al.* [14] (the difference being that here we use the plant model, rather than the plant itself). The parameters in c_i^C are adjusted to minimise the mean square error between the output of the model and the input of the controller. This leads to the following learning algorithm:

$$c_i^C(k+1) = c_i^C(k) + \alpha K_i^C(r(k+1) - y^m(k+1)) \frac{\partial y^m(k+1)}{\partial u(k)} \tag{10}$$

Here, if the real plant were used in the learning procedure (as in [14]) then $\partial y(k+1)/\partial u(k)$ would require to be estimated. This can be done using first-order differences [14] changing each input to the plant slightly at the operating point and measuring the change at the output. By using the plant model, however, the derivatives can be calculated explicitly. From eqn. 5 one obtains

$$\frac{\partial y^m(k+1)}{\partial u(k)} = -2\Delta^m \sum_{i=1}^{N^m} c_i^m K_i^m(u(k) - u_{i,0}) \tag{11}$$

Proposition 2: The learning algorithm defined by eqn. (10) converges to a global minimum of the index defined by

$$J = \sum_j (r(j+1) - y^m(j+1))^2 \tag{12}$$

if the system is monotonically increasing with respect to $u(k)$.

Proof: Consider the mean square error defined over all the possible inputs $r(k)$ described by eqn. 12. Its change ΔJ under an adaptation step, eqn. 10 resulting from an input r^* is

$$\Delta J(r^*) = -2 \sum_j (r(j+1) - y^m(j+1)) \sum_i \frac{\partial y^m(j+1)}{\partial u(j)} \times K_i^C(x^C(j), x_i^C, \Delta^C)\, \Delta c_i^C$$

Inserting eqn. 10 and averaging over all inputs r_k^*

$$\langle \Delta J \rangle = -2\alpha \sum_l \sum_j (r(j+1)) - y^m(j+1)) \times (r^*(l+1) - y^{m*}(l+1)) \times \sum_i \frac{\partial y^m(j+1)}{\partial u(j)} K_i^C(x^C(j), x_i^C, \Delta^C) \times \frac{\partial y^{m*}(l+1)}{\partial u(l)} K_i^C(x^{C*}(l), x_i^C, \Delta^C)$$

$$\langle \Delta J \rangle = -2\alpha \sum_i \left(\sum_l \frac{\partial y^m(l+1)}{\partial u(l)} K_i^C(x^C(l), x_i^C, \Delta^C) \times (r(l+1) - y^m(l+1)) \right)^2 \tag{13}$$

Clearly, this quantity can only be negative or zero. For a monotonic function it cannot be zero. From eqn. 13, if the function is not monotonic the algorithm can reach a local minimum when $\partial y/\partial u$ is zero.

Another approach involves the use of a synthetic signal [15]. This leads to the so called general learning architecture [14] as shown in Fig. 8. In this case the adaptation law for the weights does not depend on the derivatives of the plant

$$c_i^C(k+1) = c_i^C(k) + \alpha K_i^C(S_s - u(k)) \tag{14}$$

Here, S_s is the synthetic signal.

Proposition 3: If the system is invertible the algorithm defined by eqn. 14 converges to the best approximation of the inverse in the least square sense.

Proof: If the system is invertible then there exists an injective mapping which represents the inverse. Thus, from linear systems theory the algorithm defined by eqn. 14 converges to the least squares error [8].

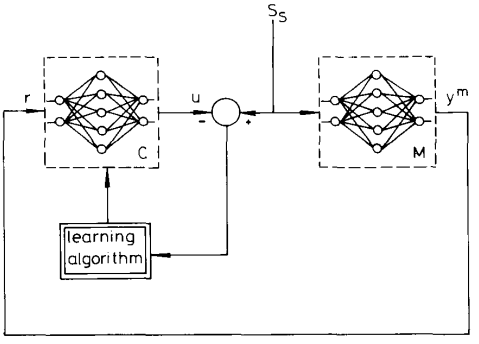

Fig. 8 *Use of synthetic signal, general learning structure*

As Psaltis *et al.* [14] indicated the specialised method allows the training of the inverse network in a region in the expected operational range of the plant. On the other hand, the generalised training procedure produces an inverse over the whole operating space. Psaltis *et al.* suggest a hybrid training procedure in which the specialised and generalised architectures are combined.

6.2.2 Iterative methods: Iterative methods use a plant model to calculate the inverse. In this case a recursive method is used to find the inverse of the model in each operating point. This method is useful in singular systems which only satisfy the invertibility conditions outlined earlier locally, and not over the whole operating space. This approach can also be used when it is necessary to have small networks due to memory or processing limitations. In this case the restricted accuracy of the trained network can be enhanced by using the network to provide stored initial values for the iterative method, establishing a compromise between speed of convergence and storing capacities.

At time k, the objective is to find an input u which will produce a model output $y^m(k+1)$ equal to $r(k+1)$. It is possible to use the method of successive substitution

$$u^{n+1}(k) = u^n(k) + \gamma(r(k+1) - y^m(k+1))$$

where γ is a weight to be chosen. A functional block diagram is shown in Fig. 9.

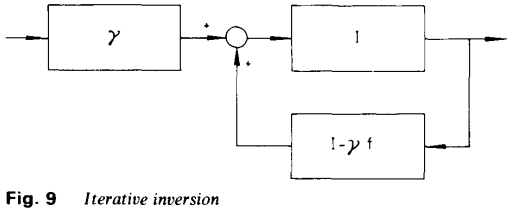

Fig. 9 *Iterative inversion*

According to the small gain theorem [16], the inverse operator is input–output stable if the product of the operator gains in the loop is less than 1

$$\|I\| \, \|I - \gamma f\| \leqslant 1$$

The initial value $u^0(k)$ can be stored in a connectionist network.

7 Examples

The plant to be considered is described by

$$y^p(k+1) = f_1(y^p(k)) + f_2(u(k))$$

$$= \frac{y^p(k)}{1 + y^p(k)^2} + u(k)^3 \tag{15}$$

This system was previously considered by Narendra and Parthasarathy [3]. The system is monotonic with respect to $u(k)$ and therefore invertible. For all the simulations a simple first-order linear filter F was used. In this case the main objective of the filter was to map the error into the input space defined for the controller.

7.1 Separable case

First, we use the prior information to decompose the system into two parts

$$y^m(k+1) = N_1(y^p(k)) + N_2(u(k))$$

where N_1 and N_2 represent connectionist networks with 40 and 20 units, which represent f_1 and f_2, respectively, over a defined interval. Note that y^p is used for training the network; when training is complete the network can be used independent of the plant with $y^m(k)$ as input to N_1. This approach is followed below.

The structure of the inverse of the model can be deduced from the above equations. We require the input of the controller r to equal the model output

$$r(k+1) = N_1(y^m(k)) + N_2(u(k))$$

Thus, the control signal u is obtained as

$$u(k) = N_2^{-r}(r(k+1) - N_1(y^m(k))$$

where N_2^{-r} is a network with 40 units, representing the right inverse of N_2. The structure of this system is shown in Fig. 10 [3]. Fig. 11 shows the functions and their estimates after training. The response to reference changes for the IMC structure is shown in Fig. 12. The response to disturbances and references changes is illustrated in Fig. 13, the system in this case is capable of eliminating the step disturbance.

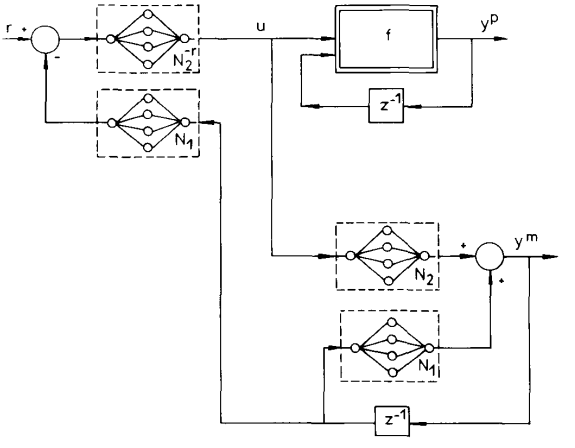

Fig. 10 *Separable system*

7.2 Single network

Alternatively, we may ignore the structure of the nonlinearity. In this case a single connectionist network rep-

resents the system

$$y^m(k+1) = N(y^p(k), u(k))$$

The inverse is represented by

$$u(k) = N^{-r}(y^m(k), r(k+1))$$

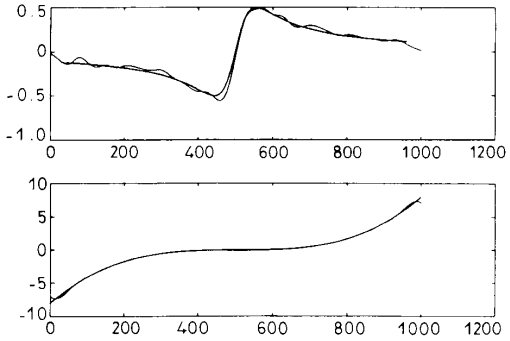

Fig. 11 *Functions and estimates*

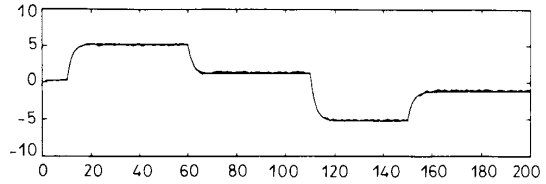

Fig. 12 *Response to changes in reference signal*

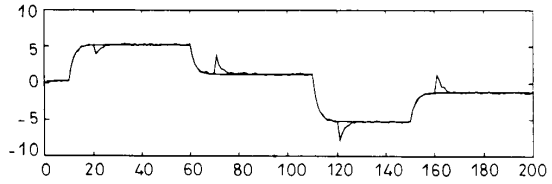

Fig. 13 *Response to changes in reference signal with a perturbation*

This represents the right inverse of N. Fig. 14 shows the response using the IMC structure to changes in the reference signal using a mesh of 100 units, and Fig. 15 shows the relation between the input to the inverse and the output of the model, that is $N[y^m(k), N^{-r}(y^m(k), r(k+1))]$. Fig. 16 shows the response to changes in the reference signal using a mesh of 400 units; in this case the error is much smaller compared to the 100 units case. Finally, Fig. 17 shows the relation between the input to the network that represents the inverse of the model and the output of the model, for this case. Theoretically it is possible to increase the number of units to obtain the desired behaviour.

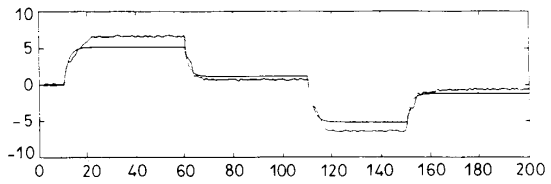

Fig. 14 *Response to changes in reference signal using 100 units*

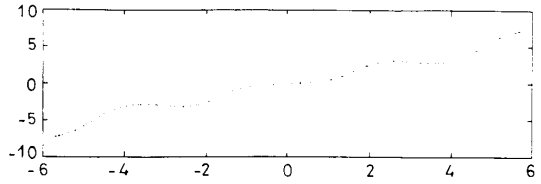

Fig. 15 *Input to controller against output of model*

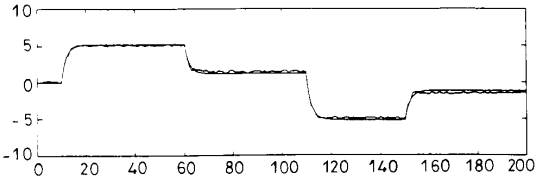

Fig. 16 *Response to changes in reference signal using 400 units*

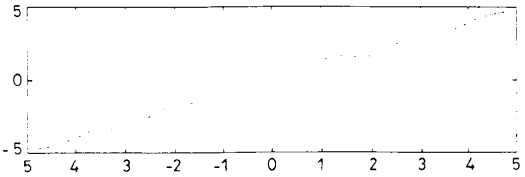

Fig. 17 *Input to controller against output of model*

7.3 Iterative calculation

The fidelity of the approximation is limited by the number of units. In order to increase the accuracy without increasing the number of units it is possible to add a refinement procedure over the first approximation using the iterative procedure outlined in Section 6.2.4.

The method used in the refinement, as described above, is the contraction mapping. Fig. 18 shows the results obtained with this procedure as a complement to a network with 100 units. The increased accuracy as a result of the improvement in the inverse model should be noted. Fig. 19 shows the relation between the input to the inverse and the output of the model for this case.

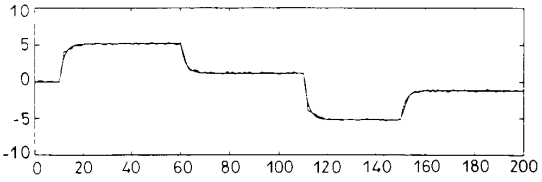

Fig. 18 *Response to changes in reference signal using iterative scheme*

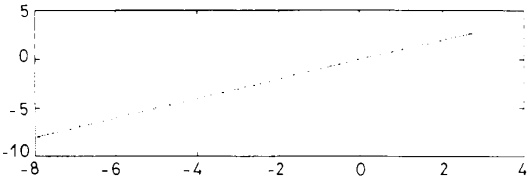

Fig. 19 *Input to controller against output of model*

8 Conclusions

The use of artificial neural networks for non-linear IMC has been explored. In doing so, we studied the invertibility of a class of non-linear dynamical systems and derived an invertibility characterisation.

We proposed architectures and algorithms for training networks to represent nonlinear dynamic relations, and the inverse of these relations. In addition, proof of convergence for the proposed algorithm is provided. The use

of neural networks within the IMC structure was demonstrated by example.

In a companion paper [17] we investigate the relationship between IMC and adaptive inverse control [15]. In that work the neural network based IMC is interpreted as a nonlinear adaptive inverse control; the neural networks are viewed as nonlinear adaptive filters.

Future work will focus on the extension of the approach to multivariable systems, and on the robustness of the controller.

9 References

1 GEORGIEV, A.A.: 'Fitting of multivariate functions', *Proc. IEEE*, 1987, **75**, pp. 970–971

2 CYBENKO, G.: 'Approximation by superpositions of a sigmoidal function', *Math. Control Signal Systems*, 1989, **2**, pp. 303–314

3 NARENDRA, K.S., and PARTHASARATHY, K.: 'Identification and control of dynamic systems using neural networks', *IEEE Trans. Neural Networks*, 1990, **1**, pp. 4–27

4 CHEN, S., BILLINGS, S.A., and GRANT, P.M.: 'Non-linear system identification using neural networks', *Int. J. Control*, 1990, **51**, pp. 1191–1214

5 GARCIA, C.E., and MORARI, M.: 'Internal model control — 1. A unifying review and some new results', *Ind. Eng. Chem. Process Des. Dev.*, 1982, **21**, pp. 308–323

6 MORARI, M., and ZAFIRIOU, E.: 'Robust process control' (Prentice-Hall, 1989)

7 ECONOMOU, C.G., MORARI, M., and PALSSON, B.O.: 'Internal model control. 5. extension to nonlinear systems', *Ind. Eng. Chem. Process Des. Dev.*, 1986, **25**, pp. 403–411

8 SBARBARO, D.G., and GAWTHROP, P.J.: 'Self-organization and adaptation in gaussian networks'. 9th IFAC/IFORS symposium on identification and system parameter estimation. Budapest, Hungary, 1991

9 GIROSI, F., and POGGIO, T.: 'Networks and the best approximation property', *Biol. Cybernetics*, 1990, **63**, pp. 169–176

10 WEIGAND, A.S., HUBERMAN, B.A., and RUMELHART, D.E.: 'Predicting the future: A connectionist approach'. Report PARC-SSI-90-20, Stanford University, 1990

11 CASDAGLI, M.: 'Non-linear system prediction of chaotic time series', *Physica D*, 1989, **35**, pp. 335–356

12 BHAT, N., and MCAVOY, T.J.: 'Use of neural nets for dynamical modelling and control of chemical process systems', *Comput. Chem. Eng.*, 1990, **14**, (4–5), pp. 573–583

13 LEE, S., and KIL, R.M.: 'Multilayer feedforward potential function network', *Neural Networks*, 1990

14 PSALTIS, D., SIDERIS, A., and YAMAMURA, A.A.: 'A multilayered neural network controller', *IEEE Control System Mag.*, 1988, **8**, pp. 17–21

15 WIDROW, B.: 'Adaptive inverse control'. Preprints 2nd IFAC workshop on Adaptive Systems in Control and Signal Processing, Lund, Sweden, 1986, pp. 1–5

16 DESOER, C.A., and VIDYASAGAR, M.: 'Feedback systems: input–output properties' (Academic Press, London, 1975)

17 HUNT, K.J., and SBARBARO, D.: 'Adaptive filtering and neural networks for realisation of internal model control' (submitted for publication).