Nonlinear System Identification using a Genetic Algorithm and Recurrent Artificial Neural Networks

Yuqing Zhu

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

April 2006

Your file Votre référence
ISBN: 978-0-494-20771-0
Our file Notre référence
ISBN: 978-0-494-20771-0

ABSTRACT

Nonlinear System Identification using a Genetic Algorithm and Recurrent Artificial Neural Networks

In this study, the application of Recurrent Artificial Neural Network (RANN) in nonlinear system identification has been extensively explored. Three RANN-based identification models have been presented to describe the behavior of the nonlinear systems. The approximation accuracy of RANN-based models relies on two key factors: architecture and weights. Due to its inherent property of parallelism and evolutionary mechanism, a Genetic Algorithm (GA) becomes a promising technique to obtain good neural network architecture. A GA is developed to approach the optimal architecture of a RANN with multiple hidden layers in this study. In order to approach the optimal architecture of Neural Networks in the sense of minimizing the identification error, an effective encoding scheme is in demand. A new Direct Matrix Mapping Encoding (DMME) method is proposed to represent the architecture of a neural network. A modified Back-propagation (BP) algorithm, in the sense of not only tuning NN weights but tuning other adjustable parameters as well, is utilized to tune the weights of RANNs and other parameters. The RANN with optimized or approximately optimized architecture and trained weights have been applied to the identification of nonlinear dynamic systems with unknown nonlinearities, which is a challenge in the control community. The effectiveness of these models and identification algorithms are extensively verified in the identification of several complex nonlinear systems such as a "smart" actuator preceded by hysteresis and friction-plague harmonic drive.

Acknowledgements

This thesis was carried out in the Mechanical and Industrial Engineering Department of Concordia University, Montreal, Canada. It was a great opportunity for me to have such experience in the Artificial Intelligent Control filed. I am grateful to all those who have contributed to this thesis and helped me:

Firstly, I would like to express my great thanks to my supervisor, Dr. W. F. Xie, for her effective support and guidance throughout the work of this thesis. Her hard-working deeply impressed me. Her broad knowledge on control and artificial intelligence area also encourages me to study deeply in this research field.

Secondly, I would like to thank the Technical Support and System Analyst Group of the Department of Mechanical and Industrial Engineering at Concordia University for their continuous support and help.

I am also very grateful to my classmates, friends and fellow students for their help and advice during my research.

Finally, I would also like to thank my families, for their encouragement, support and help in my study and life.

Yuqing Zhu

Montreal, Canada

Table of Contents

List of Figures

List of Tables

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
NN	Neural Network
ANN	Artificial Neural Network
RANN	Recurrent Artificial Neural Network
GA	Genetic Algorithm
BP	Back-propagation
DMME	Direct Matrix Mapping Encoding
SISO	single-input single-output
$\boldsymbol{C_{ij}}$	The connection matrix between the i^{th} and the j^{th} layer
$\boldsymbol{W_{ij}}$	The weight matrix between the i^{th} and the j^{th} layer
$\boldsymbol{M}$	The delays number of input signal
$\boldsymbol{N}$	The delays number of output signal
$\boldsymbol{N_h}$	The delays number of nonlinear element
$f_L(\cdot)$	The output of the linear part
$f_{NL}(\cdot)$	The output of the nonlinear element
$\boldsymbol{P}$	The total number of training samples
$\boldsymbol{E}$	The total mean average mean square error of the identification system
$E^{(m)}$	The average mean square error under m^{th} training

	sample
e	The difference between the real output and identified output
$y(k), y_d$	Real system output
$\hat{y}(k), y_I$	Identification system output
$h(\cdot)$	Output of nonlinear element
$u(k)$	Input signal
a_i, b_i	Linear part parameters
$\hat{a}_i, \hat{b}_i$	Approximated linear part parameters
net_i^l	Net output of the i^{th} neuron in the l^{th} layer
y_i^l	Neuron output of the i^{th} neuron in the l^{th} layer
w_{ij}^l	Associated weight between the j^{th} neuron in the l^{th} Layer and the i^{th} neuron in the $(l-1)^{th}$ layer
Δw_{ij}^l	Increment of w_{ij}^l
$\Delta \hat{a}_i$	Increment of $\hat{a}_i$
$\Delta \hat{b}_i$	Increment of $\hat{b}_i$
δ_{ij}^l	Back-propagated error of the j^{th} neuron in the l^{th} layer
$\sigma(\cdot)$	Activation function of hidden layers
$f(\cdot)$	Activation function of the output layer

η	Learning rate for parameters in BP algorithm
$\boldsymbol{F}$	Evaluation of fitness
$\boldsymbol{T}$	Sampling interval
τ_M	Parameter of maximum input time delays
τ_N	Parameter of maximum output time delays
$\boldsymbol{W_p}$	Weight vector of the first hidden layer
$\boldsymbol{W_I}$	Input vector of the RANN
F_C	Coulomb friction force
F_v	Viscous friction force
F_s	The magnitude of the Stribeck friction
$F_{s,a}$	The magnitude of the Stribeck friction at the end of the previous sliding period
$F_{s,\infty}$	The magnitude of the Stribeck friction after a long time at rest
v	The velocity
v_s	The characteristic velocity of the Stribeck friction
τ_L	Time constant of frictional memory
γ	The temporal parameter of the rising static friction
t_2	The dwell time, time at zero velocity
$\boldsymbol{J}$	Moment of inertia of Harmonic Drive

$\boldsymbol{B}$	Viscous damping constant of Harmonic Drive
k_m	The torque constant of Harmonic Drive

Chapter 1 Introduction

1.1 Motivation

Many plants and electrical devices used in control systems can be modeled by linear differential equations; however, usually, some parameters in the linear differential equations are not exactly known. In addition, nonlinearities, such as backlash, friction, dead zone, etc., exist in most plants or devices, and usually are unknown or partially unknown. Systems with unknown linear parameters or unknown nonlinear characteristics cannot be controlled optimally. Therefore, in order to design control systems better, to identify these unknown linear parameters and nonlinear characteristics is essential in control domain.

System identification is a process of estimating the architecture and parameters of a model from the input and output data. Since the description of a process is often a prerequisite to the analysis and controller design, the study of system identification techniques has become an established branch of control theory. For the linear time-invariant system, a plethora of identification methods have been developed based on well-established results in linear system theory. For the most practical systems that are nonlinear and time varying, the assumption of linearity may not hold. Thus, these methods are generally applicable and extensible to only a special class of nonlinear systems. In some cases, the identification techniques must have been custom developed for the selected class of nonlinear systems. To apply the results to general classes of nonlinear systems, restrictive *a priori* knowledge about the system is required. The comprehensive overview of nonlinear system identification and its mathematical principles can be found in (Voss, Timmer et al. 2004), (Juditsky, Hjalmarsson et al. 1995), (Piche 1994), (Sjoberg, Zhang et al. 1995). Some linear and nonlinear system identification models were explored in (Canelon, Shieh et al. 2004), (Teixeira and Zak 1999), (Reed and Hawksford 1996), (Hunt, DeGroat et al. 1993), (Bendat 1990), and the approaches for linear and nonlinear system identification were summarized in (Prasad and Bequette 2003). Since much recent research has shown that intelligent techniques, such as neural networks (NNs), are very robust and effective to identify complex nonlinear systems, the research in this thesis aims at identifying nonlinear system using artificial intelligent techniques: Genetic Algorithm (GA) and Recurrent Artificial Neural Networks (RANNs).

1.2 Literature review

1.2.1 The development of system identification

The statistics and time series communities revealed the early explorations of system identification. Gauss and Fisher did some fundamental research on system identification before 1920 (Gauss 1809), (Fisher 1918a). Many researchers followed the theory of stationary stochastic processes before 1970 (Barrett 1963), (Astrom and Bohlin 1965). Graphical methods by analyzing step, impulse or sinusoidal responses and cross-correlation and cross-spectral methods were two popular non-parametric methods before the year of 1965. The main shortages of these methods are lack of precision, no tools to estimate model errors, infinite dimensional models of step responses and frequency functions, and cannot deal with close loop data. Kalman proposed a model-based prediction and control theory, i.e., Kalman filter (replaces Wiener filter) and LQG control (Ho and Kalman 1965). State-space based control design methods turned on the research of estimation of state-space parametric models from data, such as the ARMAX model for input-output formulation (Astrom and Bohlin 1965). The subspace identification was proposed by Ho and Kalman (Ho and Kalman 1965). The stochastic realization theory was inspired by the combination of deterministic realization theory and innovations theory. The Hankel matrix factorization and identification of parametric I/O models were two significant achievements in the 1970s. Identification was viewed as approximation in (Anderson, Moore et al. 1978), (Ljung and Caines 1979). After the year of 1985, identification is viewed as a design problem (Gevers and Ljung 1986). In the 1990s, much research focused on: state space parametric identification based on subspace methods, identification for control, and nonlinear system identification. A new research topic, control oriented experiment design, was triggered by the identification for control recently (Hildebrand and Gevers 2003), (Hjalmarsson 2005).

In particular, Professor Lennart Ljung (Gevers and Ljung 1986) contributed greatly to the system identification in the following aspects:

- PE framework as well as PE framework for nonlinear identification
- Identifiability of closed-loop systems
- Nominal parameters θ^* and the bias expression of θ^*
- Asymptotic variance expression
- Books and the Matlab toolbox for system identification

1.2.2 Overview of system identification techniques

For linear systems, state-space models can be obtained from input-output data using subspace identification methods. For nonlinear systems, these methods can not be applied directly (Prasad and Bequette 2003). M. Enqvist (Enqvist 2005) did an extensive research on how to approximate a nonlinear system using an estimated linear model, and the prediction-error method is the main system identification method used by Enqvist.

Sometimes, the physical model of a nonlinear system is very difficult to obtain, or only a not very suitable high order model can be developed, hence, the techniques of identifying such kind of systems by empirical models obtained from input-output data are in demand. For linear systems, two approaches are used to obtain the linear models from input-output data. One is to obtain an autoregressive model, such as ARX, ARMAX models and the other one is to find a state-space model, such as the N4SID model (Overschee and Moor 1994). The main shortage of autoregressive method is the high dependence of the structure and parameterization of the identified model. Nonlinear autoregressive models, polynomial functions based models—Volterra functions, and artificial neural network based techniques are main approaches for nonlinear systems identification. For ANNs, since recurrent neural networks are much difficult to be trained, most researchers use feedforward neural networks instead although recurrent neural networks are explored to be more suitable to obtain a state-space model (Prasad and Bequette 2003).

Among neural network based techniques, if linear parameters are known, then nonlinear intelligent observers can be used to identify nonlinear characteristics (Strobl and Schröder 1998), (Frenz and Schröder 1997), (Xie, Křzeminski et al. 2002). Schroder, *et al* (Schroder, Hintz et al. 2001) gave an overview of these methods in 2001. In the cases of some parameters of the linear part are unknown, Bernhard T. Angerer (Angerer, Hintz et al. 2004) proposed a method of structured recurrent neural network embedded into the partially known function of a nonlinear system to identify nonlinear systems with *a priori* known system structure. Some heuristic intelligent optimization techniques, such as genetic algorithms, simulated annealing, ant colony optimization, tabu search, and artificial neural networks used in the control areas, especially for nonlinear system identification are well discussed in (Kalinli and Karaboga 2004). Recurrent neural networks can be used to represent nonlinear systems exactly, and are very effective to identify complex nonlinear systems with models are completely unknown (Yu 2004).

In his book (Liu 2001), G. P. Liu also presented couple of intelligent techniques involving genetic algorithms and neural networks for nonlinear system identification and control. The variable-structure neural network, based on radial basis functions, is used in this book to derive a continuous-time adaptive control schema, and Lyapunov stability theory based techniques are adopted to guarantee the stability of this method. Wavelet networks technique, based on feedforward neural networks, is another nonlinear system identification approach in this book. A recursive identification scheme, based on Volterra polynomial basis function networks, is also introduced to identify nonlinear discrete time systems. Genetic algorithms and multi-objective optimization techniques, such as approximation accuracy and model complexity, are proposed for model selection and parameter estimation. In this book, affine neural network predictors are used in constructing a nonlinear predictive control scheme and variable-structure control.

In (Sjoberg, Zhang et al. 1995), structure models in system identification are classified into three categories: White-box models, Grey-box models and Black-box models, and structure constructing techniques, such as neural networks, radial basis networks, wavelet networks, hinging hyperplanes, wavelet-transform, and fuzzy sets and rules are extensively discussed. Lennart Ljung (Ljung 2005) categorizes these models into two main types: Black-box models and Grey-box models. Ljung introduced three techniques of constructing Black-box models, i.e., one hidden layer artificial sigmoidal neural networks, Wavelets, and (Neuro) Fuzzy modeling techniques. The regressors for dynamic systems, as Ljung pointed out, can be selected as one of the following models:

- NFIR-models, with using past inputs
- NARX-models, with using past inputs and outputs
- NOE-models, with using past inputs and past simulated outputs
- NARMAX-models, with using inputs, outputs predicted outputs
- NBJ-models, with using all four regressor types

Furthermore, Ljung mentioned that recurrent neural networks must be used when NOE, NARMAX or NBJ is chosen. Figure 1-1 shows the simulation results of the example, system identification with Sigmoidal network, by Ljung (Ljung 2005). Where dash line represents the measured output and the solid line represents the model simulated output.

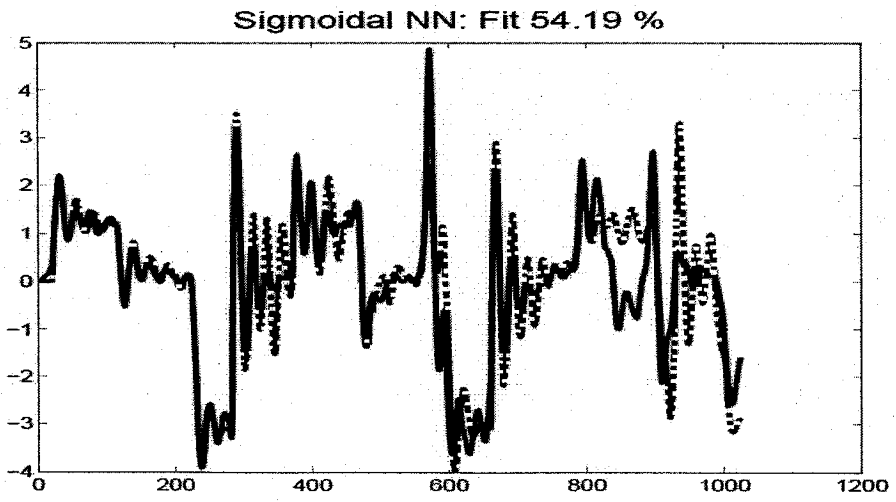


Figure 1-1 Example of using NN by Professor Ljung (Ljung 2005)

Ljung classified Grey-box models into four categories: Physical Modeling, Semi-physical Modeling, Block-models, and Local Linear Models, and explained in detail how to implement them. As a conclusion, Ljung pointed out that neural networks are effective techniques to obtain dynamic models, particularly for Black-box models. However, since local optima may exist when adjusting a parameterized model structure to data, it is difficulty to generate initial conditions when using neural network based techniques (Ljung 2005). In other words, to obtain good parameterized neural network architecture with appropriate parameters is very difficult.

1.2.3 The development of NN-based Nonlinear System Identification

Artificial Neural Networks (ANNs) are inspired by the structure and functions of the biological neural networks (BNNs) (Bose and P.Liang 1998), (Ham and Kostanic 2001), (Haykin 1999), (Widrow and Lehr 1990). The first logical neuron model was introduced by W. S. McCulloch and W. Pitts in 1943. Later, in 1949, D. O. Hebb introduced the neuron connection weights updating laws. These are fundamental of later research on ANNs. In 1958, F. Rosenblatt introduced the Perception model, which is the first complete ANN model.

Although there were lots of studies on the system identification models decades ago (Sagaspe 1979), (Rugh 1981), (Schetzen 1989), (Billings and Fakhouri 1977), (Asdente, Pascucci et al. 1976), (Barrett 1963), (Billings 1980), a lot of research has been done on describing and controlling nonlinear systems using ANN techniques since the 1980s. The neural networks with number of hidden layers have been explored to be nonlinear models that can approximate any function with arbitrary degree of accuracy (Hornik, Stinchcombe et al. 1989). Reviews of some relevant approaches are given in (Wang, Spronck et al. 2003), (Madár, Abonyi et al. 2005), (Canelon, Shieh et al. 2004), (Hunt, Sbarbaro et al. 1992) and (Agarwal 1997). Both linear models and nonlinear NN models were applied to the nonlinear discrete dynamic systems' control in (Canelon, Shieh et al. 2004). The use of the neural network model excludes the systems' reliance on its physical principle model, which was shown to bring direct benefits to the design of the controller: (a) the proposed approach and the design of the controller are based on the neural network obtained from data, not a physical principle dynamic system model; (b) the proposed approach avoids identifying physical principle nonlinear dynamic models, which is a very difficult work, and uses the same neural network model, so it is easy to implement the linearization approach of a network with low computational effort; (c) at every operating state, the design techniques of linear state-space control are available for the controller's design (Canelon, Shieh et al. 2004).

Recent rapid development of neural networks has provided invaluable tools to tackle the problem of nonlinear system identification. Past research has shown that Artificial Neural Networks are effective methods to model and control a broad category of complex nonlinear systems, especially to those systems whose mathematical models are extremely difficult to obtain. Three main characteristics of ANNs make them the prime candidates for the nonlinear system identification, (i) ability to learn from experience and adapt to different environmental conditions, (ii) generalization ability for untrained inputs and (iii) parallel and pipeline processing capability to perform different tasks more efficiently (Yazdizadeh and Khorasani 2002), (Gupta and Sinha 1999).

1.2.4 Category of Neural Networks

Neural networks can be categorized into two major types: feedforward networks and recurrent networks. In the literature, feedforward NNs are most popularly used for nonlinear system identification (Canelon, Shieh et al. 2004), (Hornik, Stinchcombe et al. 1989), (Cybenko 1989), (Funahashi 1989). A typical example is the Multilayer Perceptions (MLP), which is utilized to identify the nonlinear characteristics of a nonlinear system. The feedforward NNs suffer from two major pitfalls: sensitivity to the training data and ignorance of the local data structure information when updating weights. On the other hand, recurrent NNs can overcome these two disadvantages and have demonstrated strong dynamic nonlinear characteristics in a large amount of research works.

1.2.5 Types of NN-based Nonlinear System Identification

Nonlinear system identification using neural network techniques has become an interesting topic (Madár, Abonyi et al. 2005), (Becerra, Garces et al. 2005), (Canelon, Shieh et al. 2004), (Yu 2004), (Prasad and Bequette 2003), (Kiong, Rajeswari et al. 2003), (Liu, Kadirkamanathan et al. 1998), (Liu, Kadirkamanathan et al. 1996), (Polycarpou and Loannou 1991), (Billings and Chen 1992), (Chen, Billings et al. 1990), (Kadirkamanathan and Liu 1995), (Kuschewski, Hui et al. 1993), (Liu, Kadirkamanathan et al. 1998), (Qin, Su et al. 1992), (Willis, Montague et al. 1992). In general, the Neural Networks for identification of dynamic systems fall into three main categories (Yazdizadeh 1997). The first category is a static network with tapped delay lines. The delay elements are used to introduce delayed inputs and outputs that are then fed to a static network as the regressor vector so that the predicted NN output will simulate the target output. The second category is a dynamic network constructed by dynamic neurons. One of such dynamic NNs is time delay neural network (TDNN) with each weight associated with a delay. The third category is a Recurrent Artificial Neural Network (Funahashi and Nakamura 1993). Among various NNs, the RANN bears powerful capability to represent nonlinear systems effectively and robustly. This study uses a RANN to identify nonlinear systems.

1.2.6 Training algorithms for Neural Networks

Some NNs training methods were introduced in (Sjoberg, Zhang et al. 1995), (Piche 1994), (Huang, Koh et al. 1992), (Zurada 1992) and (Chandra and Singh 2004). In principle, the training of NNs aims at minimizing the difference between the target and the actual output by tuning adjustable parameters in NNs. Among various adaptive tuning techniques, the steepest descent algorithm is most widely used in the literature (Yu 2004), (Feng and Michel 1999), (Polycarpou and Ioannou 1992). The back-propagation (BP) algorithm, first introduced by Werbos in 1974, is a typical steepest descent approach. Sigmoid BP networks (Rumelhart, Hinton et al. 1986) were proved to bear arbitrary nonlinear characteristics as in (Funahashi 1989), (Gybenko 1989) and (Hornik, Stinchcombe *et al.* 1989). Wen Yu (Yu 2004) did some studies on the weights adjustment using the gradient descent law and the back-propagation like algorithm. PlA. Ioannou and J. Sun (Ioannou and Sun 1996) proposed another generalized method to improve the techniques. M.M. Polycarpou and P.A. Ioannou (Polycarpou and Ioannou 1992) concluded that the exact convergence of identification error can be guaranteed by back-propagation based algorithm. There are some other studies on modifying normal gradient or back-propagation algorithms to ensure the stability of the learning process (Jin and Gupta 1999), (Suykens, Vandewalle et al. 1997), (Kosmatopoulos, Ploycarpou et al. 1995), (Jagannathan and Lewis 1996). The major drawback of BP is that the identified weights tend to converge to local minima and the global optimality of the nonlinear identification cannot be guaranteed. In addition, BP only tunes the weights of a given NN to minimize the output error. How to optimize the architecture of NNs toward the global optimum remains a challenge for the researchers.

1.2.7 Key factors of NN-based applications

The successful application of NN for nonlinear system identification relies on two key factors: architecture and weights. Most research in the literature focuses on tuning the weights. The NN architecture, on the other hand, is designed via a tedious trial-and-error process with human intervention. The architecture design of Neural Network is crucial in the successful application of NN due to its significant impact on a network's information processing capabilities. Although there are some studies on the automatic design of architectures (Chester 1990), (Frean 1990), (Hirose, Yamashita et al. 1991), (Roy, Kim et al. 1993), (Weigend 1994), how to design a good architecture systematically and autonomously remains a challenging problem.

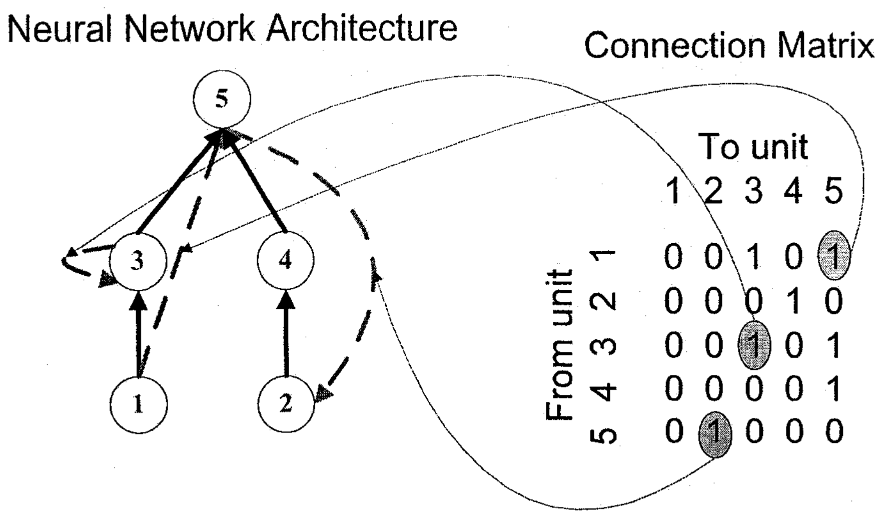
In general, model reduction should be considered in system identification (Prasad and Bequette 2003), (Zhou, Doyle et al. 1996), (Karnin 1990). Usually, a regulation operator is required for NN reduction during the training. The sensitivity analysis is one of the most popular methods (Prasad and Bequette 2003). The hidden neuron with the least sensitivity according to the output error will be deleted or assigned with a penalty term during the training process. The sensitivity analysis works with the learning rate, forces weights or other adjustable parameters towards zero, and it reduces the number of redundant parameters as well as the degree of over parameterization of the network (Prasad and Bequette 2003).

1.2.8 Neural network encoding methods

In order to optimize the architecture of NNs, an effective encoding scheme is in demand. There are two popular encoding approaches: Direct Encoding and Grammar Encoding. The Direct Encoding method takes into account all possible connections between neurons and encodes the parameters of the NNs, e.g. weights and/or connections, directly into a matrix.

In 1989, Miller *et al.* (Miller, Todd et al. 1989) used the Direct Encoding to evolve the connection topology for a feedforward neural network with a fixed number of units. A number of studies (Marin and Sandoval 1993), (Schaffer, Caruana et al. 1990), (Schiffmann, Joost et al. 1993), (Whitley, Starkweather et al. 1990), (Wilson 1990) also are focused on Direct Encoding.

Figure 1-2 shows a typical example of the Direct Encoding. The connection topology of the network is represented by a 5×5 matrix. Each entry in the matrix denotes the status of the connection - either "1" (connected) or "0" (unconnected). The connection matrix is transformed into the chromosome in the form of a binary string in the bottom of Figure 1-2. It can be seen that the binary string can be easily decoded into a NN's architecture as well.



$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 0 & 1 & 0 & 1 \\
2 & 0 & 0 & 0 & 1 & 0 \\
3 & 0 & 0 & 1 & 0 & 1 \\
4 & 0 & 0 & 0 & 0 & 1 \\
5 & 0 & 1 & 0 & 0 & 0
\end{array}$$

Chromosome: 00101 00010 00101 00001 01000

Figure 1-2 An example of Direct Encoding

As analyzed by Xin Yao (Yao 1999), a potential problem in Direct Encoding is the large size of connection matrix introducing heavy computational load. To cut down the size of the connection matrix for a large neural network requires sufficient domain knowledge and expertise (Yao 1999). Potential invalid connections (e.g. the connections between input neurons and output neurons) further increase non-trivial cost for fitness evaluation. Finally, most past studies on Direct Encoding are focused on designing a feedforward neural network.

A feedforward NN typically disallows connections between non-contiguous layers or feedbacks, as illustrated in dashed arrows in Figure 1-2. Hence the connection matrix and the chromosome have to be managed with high caution. It is known that in GA, the genetic content of any chromosome is subject to permutation or alteration due to various evolutionary operators, e.g. crossover and mutation. Although invalid connections may be explicitly banned, it is at the cost of a large amount of extra work for the validity check.

The Grammar Encoding was first proposed by Kitano (Kitano 1990). It uses complex grammar, which is a set of rules, to represent the connection status of an NN. Figure 1-3 is a typical example of the Grammar Encoding, given by Melanie (Melanie 1999). Comparing to the direct encoding method, the Grammar encoding is too complex with prohibitively high computational cost (Melanie 1999).

In this thesis, we propose a new approach based on the Direct Encoding method, to conquer the drawback of current encoding approaches, which will be detailed in Chapter 3.

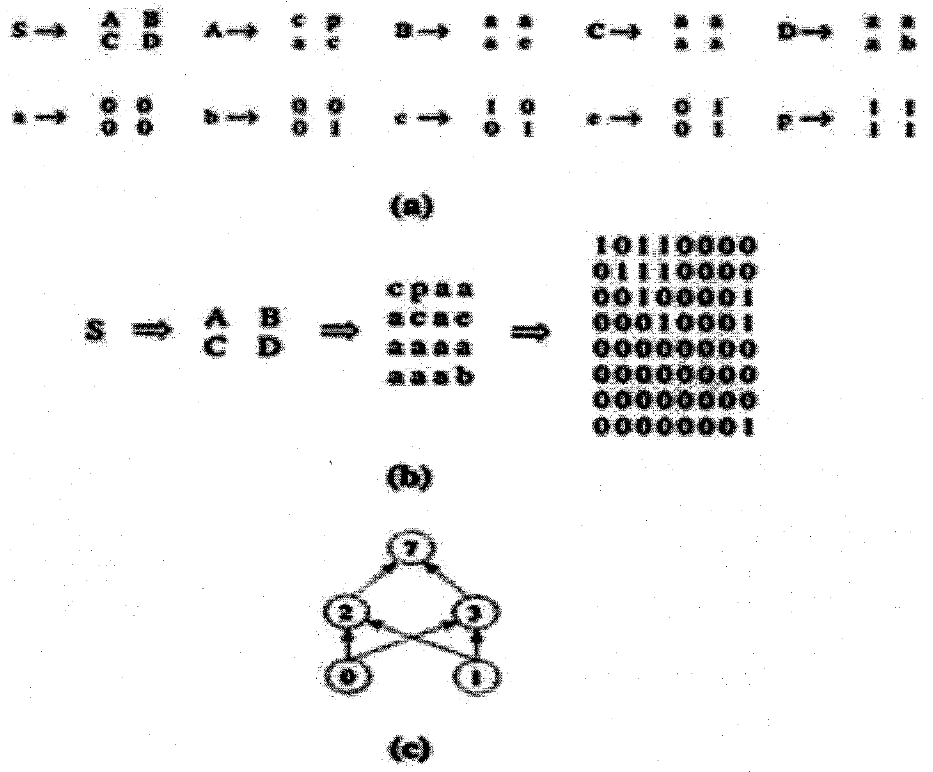


Figure 1-3 An example of Grammar Encoding

1.2.9 Genetic Algorithms

The Genetic Algorithm (GA) is a very powerful optimization algorithm due to its inherent property of implicit parallelism (Holland 1975). By exploring a large number of potential solutions in parallel, it is less likely to get stuck at a local optimum. The selection mechanism in a GA further ensures the good building blocks in chromosomes can be passed to next generations. In addition, GA works with a good balance between exploration and exploitation of the search space. The crossover operator recombines the obtained information to exploit promising areas in the search space. And the mutation operator generates new genes to ensure exploration of unvisited areas so that premature convergence of the population can be effectively avoided. In summary, GA is a promising technique to optimize the architecture of an NN.

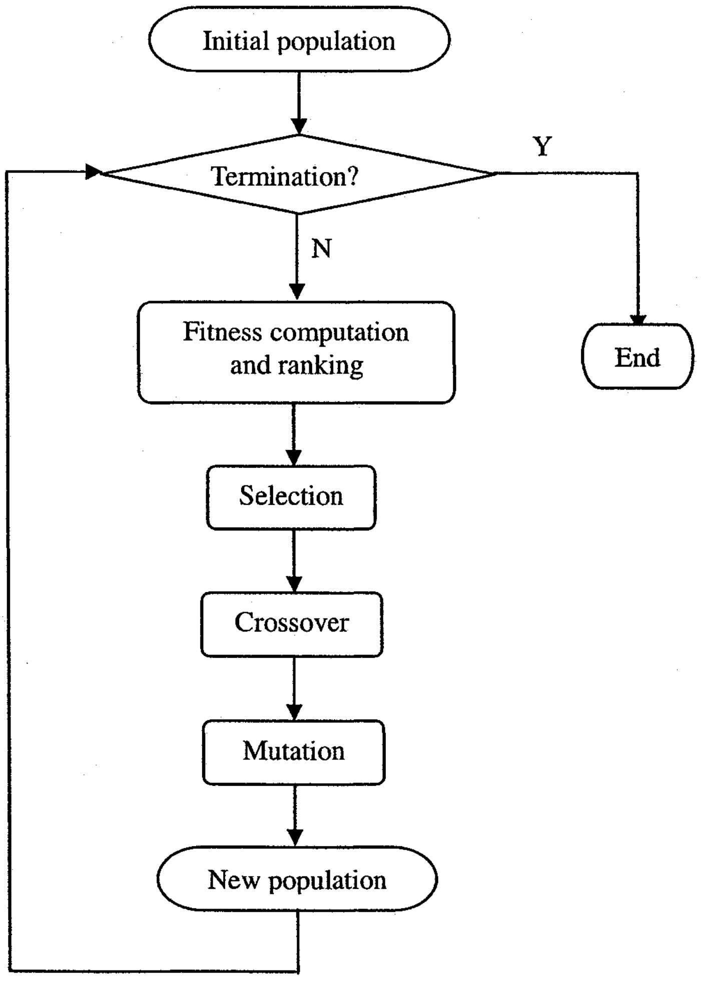


Figure 1-4 The work flow of a typical GA

The overall work flow of GA is shown in Figure 1-4. A GA starts with an initial population generated at random, each individual (chromosome) of which is assigned a fitness value (indicating the "goodness" of a potential solution). After ranking all the chromosomes with respect to their fitness, one keeps a proportion of the best candidates intact to the new generation (elitism). Pairs of parents are selected within the current population in such a way that a chromosome with better fitness has more reproductive opportunities. New offspring are generated, by exchanging the genetic information of their parents (crossover). Mutation is then applied to the whole population. A new population is formed after these operations. The program runs until a termination condition is satisfied.

Hence a typical GA essentially contains the following operators:

- Selection: select elites to be passed intact to the new population and parents to be mated;
- Crossover: recombine parental chromosomes to generate chromosomes;
- Mutation: randomly alter the genes of some chromosomes.

Some researches have used GA techniques to do nonlinear system identification (Madár, Abonyi et al. 2005), (Liu, Liu et al. 2004), (Rodriguez and Fleming 1998), (Gary, Smith et al. 1998), (Pham and Karaboga 1999). An overview of the applications of GAs in control fields is given in (Wang, Spronck et al. 2003). However, the GAs are only used either to find out the nonlinear function of the model or to figure out some parameters of the identification model. In this research, GA is used to approach the optimal structure of a neural network in the sense of minimizing the identification error.

1.3 Research objectives and main contributions of this thesis

1.3.1 Research objectives

The main research objectives of this thesis are:

1) To develop a new Genetic Algorithm to optimize the architecture of a Neural Network in the sense of minimizing the identification error. The optimized Neural Network should have strong and robust identification ability when embedded into nonlinear system identification models.

2) To develop a new Neural Network encoding method to encode Neural Networks more efficiently and effectively. The newly developed encoding method is expected to overcome two major problems of traditional encoding methods: high redundancy and computational cost.

3) To develop adaptation laws for adjustable parameters in the nonlinear models and searching algorithms to tune these parameters effectively.

4) To develop RANN-based nonlinear system identification models based on *a priori* knowledge of a nonlinear system and the relationship between the linear part and the nonlinear element of the nonlinear system, and verify the developed identification techniques by applying them to several complex nonlinear systems, including numerical complex nonlinear systems and real world nonlinear systems.

1.3.2 Main contributions

In this research, RANN is extensively studied to model dynamic nonlinear systems and the main contributions are summarized as:

- Three RANN-based models are established to describe the behaviour of nonlinear dynamic systems with different configurations.

- A novel GA-based strategy is developed to approach the optimal architecture of the RANN with multiple hidden layers in the sense of minimizing the identification error.
- The Direct Matrix Mapping Encoding (DMME) method, aiming at reducing the redundancy and computational cost of the NN architecture representation, is used to encode the architecture of a RANN.
- A modified back-propagation (BP) algorithm in the sense of tuning not only weights but all other parameters in the identification models as well is developed to tune the parameters of NNs.

The effectiveness of these models and identification algorithms are extensively verified in the identification of several nonlinear complex systems such as "smart" actuator preceded by hysteresis, and friction-plague harmonic drive.

1.4 Thesis Outline

The thesis is organized as follows. Chapter 2 introduces three classes of nonlinear systems and then analyzes the relationship between the input and the output of these three nonlinear systems, by integrating RANNs with linear mathematical models in different ways. In Chapter 3, a GA is discussed to optimize the architecture of the RANN in the sense of minimizing the identification error. Chapter 4 shows the modified BP algorithm that updates the adjustable parameters of the RANN and the linear mathematical model. Simulation results are presented in Chapter 5. Chapter 6 concludes the thesis with some possible future work. The detailed derivation of the adaptation laws is presented in Appendix A.

Chapter 2 Nonlinear systems and identification models

In this chapter, nonlinear systems are classified into three types based on the knowledge to the systems, and three corresponding RANN-based identification models are introduced to do nonlinear systems identification.

2.1 Three types of nonlinear systems

This research mainly focuses on identifying the input-output mapping of a single-input single-output (*SISO*) nonlinear system. A general *SISO* nonlinear system can be described by the following equation (Tan, Lee et al. 2001):

$$y(t+1) = f(y(t),\ldots,y(t-N+1),u(t),\ldots,u(t-M+1)) \qquad (2\text{-}1)$$

where M and N represent the maximum time delays of the input and output signals, respectively; and $M \leq N$. f is the nonlinear function of the system (detailed restrictions for the function were discussed in (Narendra and Parthasarathy 1990)). Based on the knowledge of the relationship between linear part and nonlinear elements in a *SISO* nonlinear system, we could categorize the *SISO* nonlinear systems into three types, for which three corresponding nonlinear system identification models are proposed in the following sections.

2.1.1 Type I: unknown nonlinear systems

When there is no *a priori* knowledge about a nonlinear system, the system is modeled as a completely unknown black-box (Billings 1980), (Sjoberg, Zhang et al. 1995) as shown in Figure 2-1.

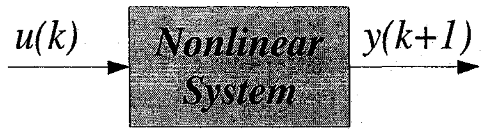


Figure 2-1 Unknown nonlinear systems

The discrete time nonlinear differential equation of this model is:

$$y(k+1) = f[y(k), y(k-1), \cdots, y(k-N+1), u(k), u(k-1), \cdots u(k-M+1)] \quad (2\text{-}2)$$

where M and N represent the time delays of the input and output signals, respectively with $M \leq N$; f represents a nonlinear function.

2.1.2 Type II: Serial-Linear nonlinear systems

In this type of nonlinear system, the relationship between linear part and nonlinear element is series as shown in Figure 2-2.

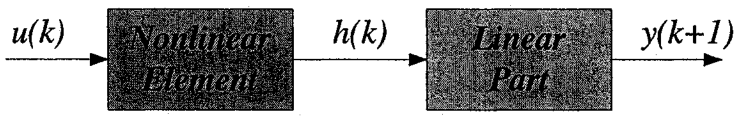


Figure 2-2 Serial-Linear nonlinear systems

The discrete time nonlinear differential equation of this nonlinear system is:

$$y(k+1) = \sum_{i=0}^{N-1} a_i y(k-i) + \sum_{i=0}^{M_h-1} b_i h(k-i) \quad (2\text{-}3)$$

where nonlinear element's output is:

$$h(k) = f[h(k-1), h(k-2), \cdots, h(k-N_h), u(k-1), u(k-2), \cdots u(k-M)] \quad (2\text{-}4)$$

where N and M_h represent the time delays of the output and input signals to linear part, respectively, and $M_h \leq N$, and N_h and M represent the time delays of the output and input signals to nonlinear element, respectively, and $M \le N_h$, here N_h is not the same parameter as N in Eq. (2-2).

2.1.3 Type III: Parallel-Linear nonlinear systems

In this type of nonlinear system, the relationship between linear part and nonlinear element is parallel as shown in Figure 2-3.

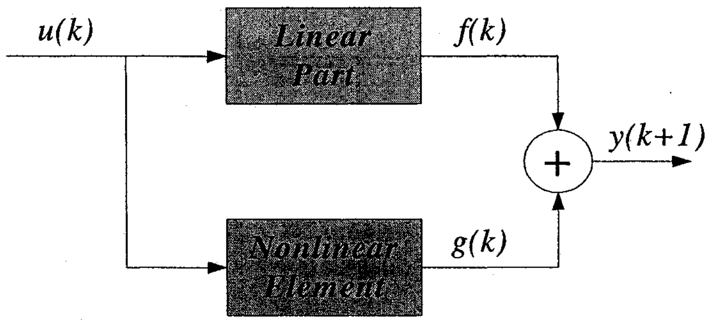


Figure 2-3 Parallel-Linear nonlinear systems

The discrete time nonlinear differential equation of this model is:

$$\begin{aligned} y(k+1) &= f(\boldsymbol{u},\boldsymbol{y}) + g(\boldsymbol{u},\boldsymbol{y}) \\ &= \sum_{i=0}^{N-1} a_i y(k-i) + \sum_{i=0}^{M-1} b_i u(k-i) + g(\boldsymbol{u},\boldsymbol{y}) \end{aligned} \tag{2-5}$$

where $f(\boldsymbol{u}, \boldsymbol{y})$ and $g(\boldsymbol{u}, \boldsymbol{y})$ are the outputs of linear part and nonlinear element, respectively, $\boldsymbol{y} = [y(k), y(k-1), \cdots, y(k-N+1)]$ is the vector of the delayed output signal, and $\boldsymbol{u} = [u(k), u(k-1), \cdots, u(k-M+1)]$ is the vector of the delayed input signal.

Any unknown nonlinear system can be treated as a black-box in system identification. If linear part of a nonlinear system is partially known, it can still be classified as either serial-linear or parallel-linear no matter how complex its nonlinearity is. Therefore, the three nonlinear system types introduced above can cover most of nonlinear systems and the corresponding three nonlinear system identification models can be used to describe most of nonlinear systems in principle.

2.2 Three RANN-based Identification Models for Nonlinear Systems

The objectives of the system identification in this study are to establish an off-line RANN-based model and develop algorithms to identify both the parameters and the architecture of the RANN model. In this section, three RANN-based models are described to demonstrate the input-output mapping of *SISO* nonlinear systems. The RANN has multi-delays on both input *x(k)* and output *y(k)* of a nonlinear system as shown in Figure 2-4.

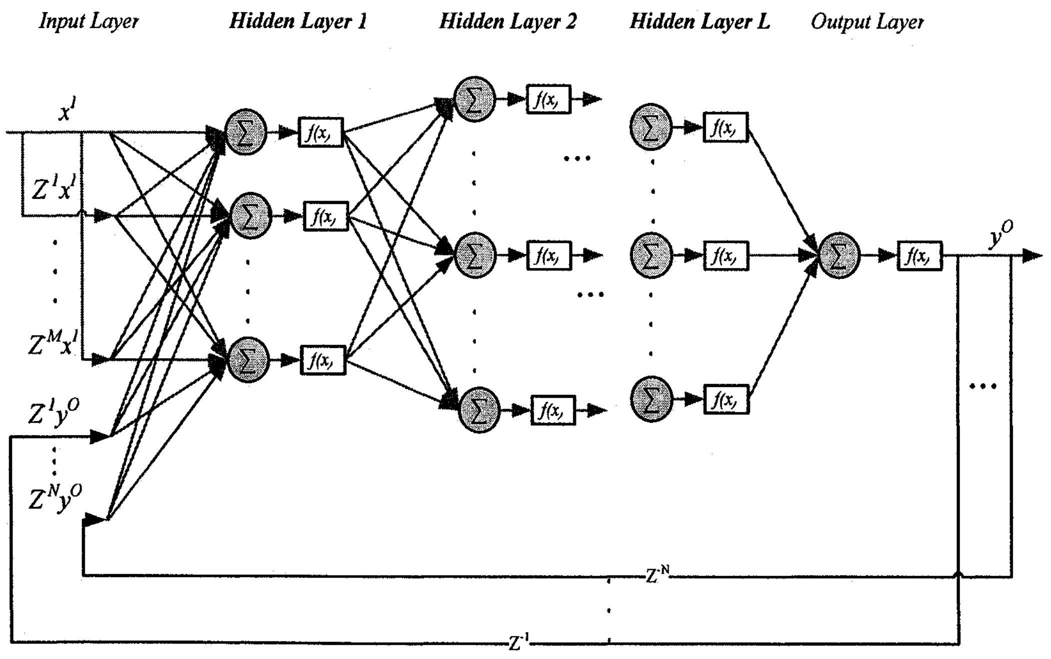


Figure 2-4 Delays of input and output neurons

The input vector to the input layer is formed by composing delayed input and output signals $[x_i^I(k), x_i^I(k-1), \cdots, x_i^I(k-M+1),\ y_j^O(k), y_j^O(k-1), \cdots, y_j^O(k-N+1)]^T \in R^{M+N}$, here *i=j=1.*

Based on *a prior* knowledge of a nonlinear system and the relationship between linear part and nonlinear element, three RANN-based nonlinear system identification models have been proposed.

2.2.1 Model I: Single RANN Model

When there is no *a priori* knowledge about a nonlinear system, One RANN model can be used to represent an unknown nonlinear system as shown in Figure 2-5. In this study, it is named as Single RANN Model.

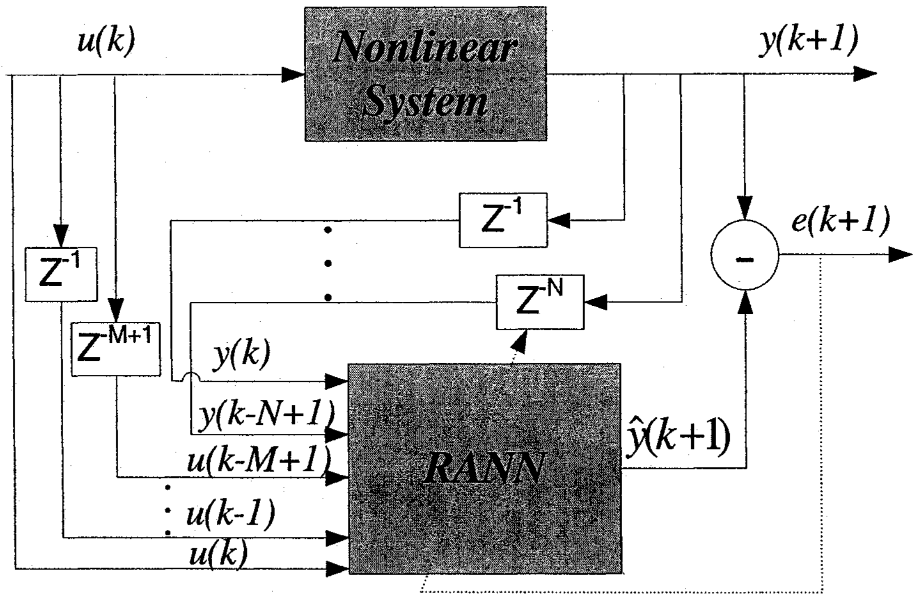


Figure 2-5 Single RANN Model for Nonlinear Systems Identification

In the Single RANN Model, the *M+N* inputs of the RANN are *u(k), u(k-1), ..., u(k-M+1), y(k), y(k-1), ..., y(k-N+1)*, and the only one output is $\hat{y}(k+1)$; where *N* and *M* represent the maximum steps of time delays of the output and the input signals, respectively. The RANN diagram of this model is shown in Figure 2-6.

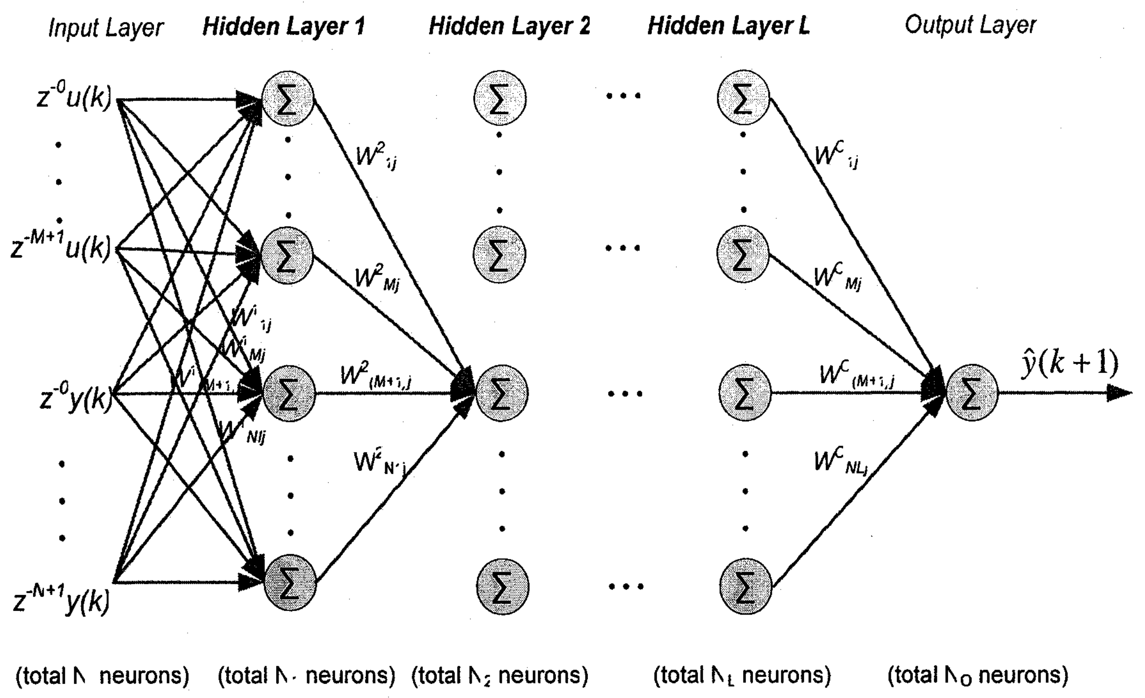


Figure 2-6 Diagram of the Single RANN Model

Since it is a *SISO* system, the RANN output $\hat{y}(k+1)$ can be obtained as:

$$\hat{y}(k+1) = f(net_1^O) = \sum_{i=1}^{N_L} w_{i1}^O x_1^O = \sum_{i=1}^{N_L} w_{i1}^O y_i^L \tag{2-6}$$

where $f(\cdot)$ is the activation function of the output layer O, net_1^O is the net output of the output neuron, L represents the L^{th} hidden layer, here we assume there are totally L hidden layers, N_L is the number of neurons in the hidden layer L, x_1^O is the input of the output neuron, and it equals to the output of the previous neuron, i.e., y_i^L, actually.

The output of the j^{th} neuron in the l^{th} hidden layer is:

$$y_j^l = \sigma(\sum_{i=1}^{N_{l-1}} w_{ij}^l y_i^{l-1}) \tag{2-7}$$

where $\sigma(\cdot)$ is the activation function of hidden neurons. Thus, the RANN output can be represented as:

$$\hat{y}(k+1) = f(net_1^O) = f(\sum_{i=1}^{N_L}[w_{i1}^O \cdots \sigma(\sum_{p=1}^{N_1} w_{pq}^2 \sigma(W_p \bullet V_I))]) \qquad (2\text{-}8)$$

where q is the number of neurons in the 2nd hidden layer, and $q \in [1, N_2]$, $W_p = [w_{1p}^1, w_{2p}^1, \ldots, w_{Mp}^1, w_{(M+1)p}^1, w_{(M+2)p}^1, \ldots, w_{(M+N)p}^1]^T$ is the weights vector of the first hidden layer, and, $V_I = [u(k), u(k-1), \ldots, u(k-M+1), y(k), y(k-1), \ldots, y(k-N+1)]^T$ is the input vector of the RANN.

The RANN's output $\hat{y}(k+1)$ is a nonlinear function (represents by the activation function in neural network) of $u(k-i), i = 0, 1, \cdots, (M-1)$ - the inputs of the RANN, and $y(k-i), i = 0, 1, \cdots, (N-1)$ - the delayed outputs of the plant. Hence the system can also be described as:

$$\hat{y}(k+1) = f_N[y(k), y(k-1), \cdots, y(k-N+1), u(k), u(k-1), \cdots u(k-M+1)] \qquad (2\text{-}9)$$

where $f_N()$ is a nonlinear function of the system.

For any nonlinear system (Eq. 2-2), the RANN identification model type I in Figure 2-5 describes its input-output mapping. Therefore, the Single RANN architecture can be used to represent an unknown dynamic nonlinear system (Funahashi and Nakamura 1993), (Yazdizadeh 1997).

2.2.2 Model II: Serial-Linear RANN Model

In some cases, linear part of a nonlinear system is known. The linear part can be integrated with a RANN in an either serial or parallel fashion. The RANN models the nonlinear elements in the system.

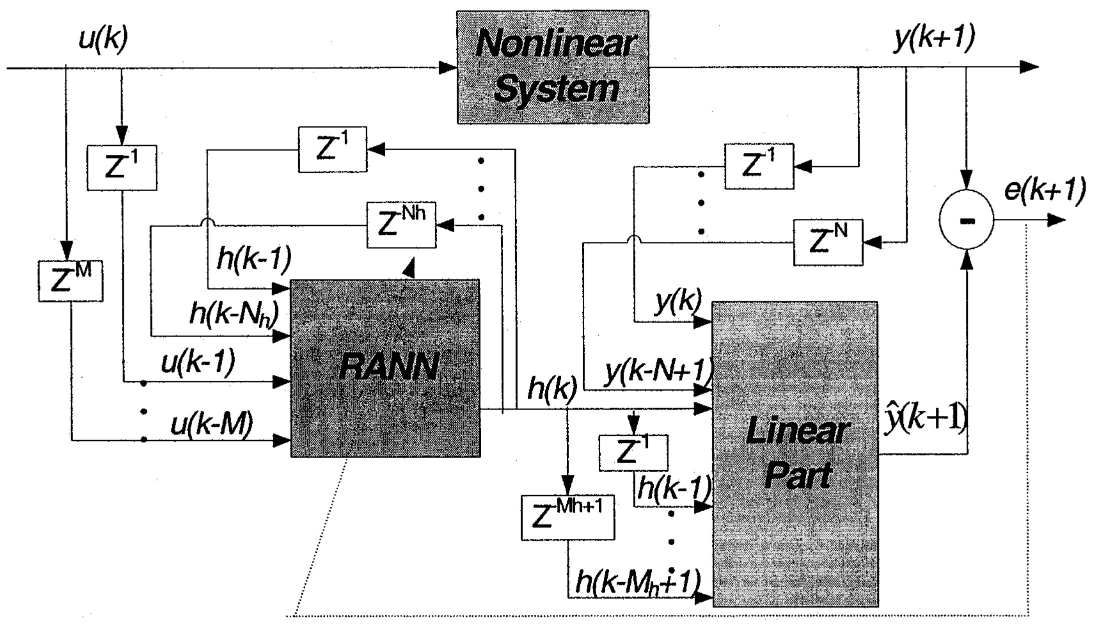


Figure 2-7 Serial-Linear Model for Nonlinear Systems Identification

In the Serial-Linear Model shown in Figure 2-7, the output of the identification system $\hat{y}(k+1)$ can be obtained by Eq. (2-10).

$$\hat{y}(k+1)=\sum_{i=0}^{N-1}a_i y(k-i)+\sum_{i=0}^{M-1}b_i h(k-i) \tag{2-10}$$

where q is the number of neurons in the 2nd hidden layer, and $q\in[1,N_2]$, $h(k)$ is the output of the RANN. Assume there are L hidden layers in the RANN, its output can be obtained by:

$$h(k)=f(net_1^O)=f(\sum_{i=1}^{N_L}[w_{i1}^O\cdots\sigma(\sum_{p=1}^{N_1}w_{pq}^2\sigma(W_p\bullet V_I))]) \tag{2-11}$$

where $W_p=[w_{1p}^1,w_{2p}^1,\cdots,w_{Mp}^1,w_{(M+1)p}^1,w_{(M+2)p}^1,\cdots,w_{(M+N)p}^1]^T$ is the weights vector of the first hidden layer, and, $V_I = [u(k), u(k-1), \cdots, u(k-M), h(k), h(k-1), \cdots, h(k-N_h)]^T$ is the input vector of the RANN.

For any nonlinear system of type II (Eq. 2-3, 2-4), the Serial-Linear RANN architecture above (Eq. 2-10, 2-11) can be used to describe its input-output mapping.

2.2.3 Model III: Parallel-Linear RANN Model

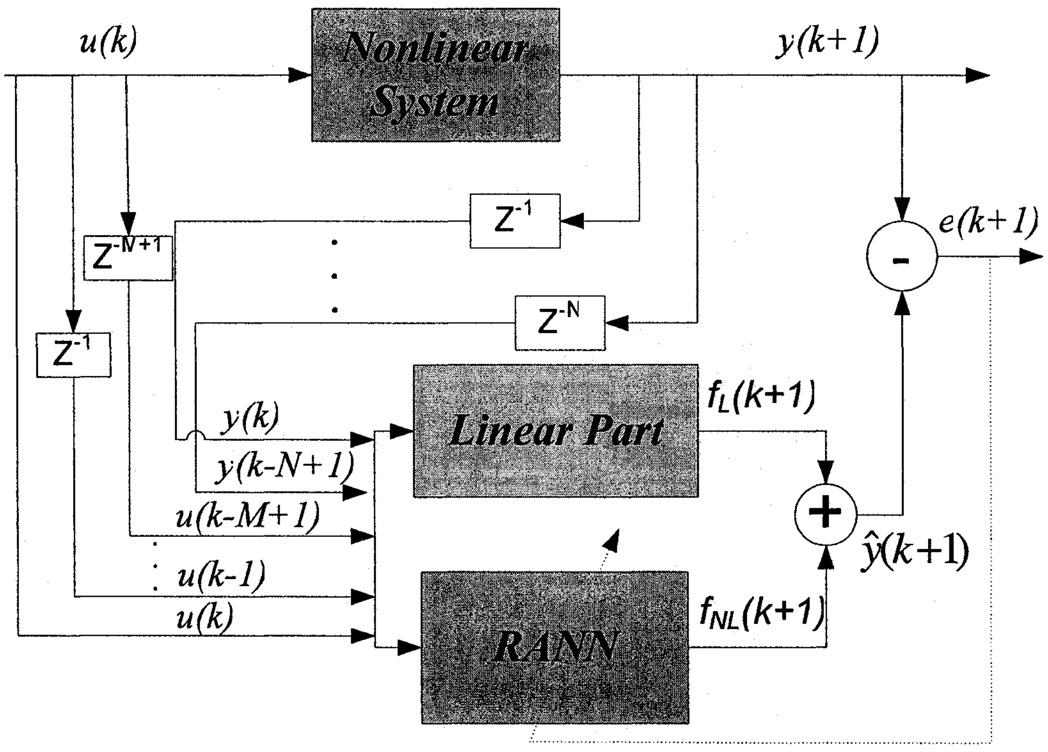


Figure 2-8 Parallel-Linear Model for Nonlinear System Identification

In the Parallel-Linear Model, the output of the identification system $\hat{y}(k+1)$ is:

$$\hat{y}(k+1) = f_L(k+1) + f_{NL}(k+1), \tag{2-12}$$

where $f_L(k+1)$ is the output of linear part:

$$f_L(k+1) = \sum_{i=0}^{N-1} a_i y(k-i) + \sum_{i=0}^{M-1} b_i u(k-i), \tag{2-13}$$

and $f_{NL}(k+1)$ is the output of nonlinear element:

$$f_{NL}(k+1) = f(net_1^O) = f(\sum_{i=1}^{N_L}[w_{i1}^O \cdots \sigma(\sum_{p=1}^{N_1} w_{pq}^2 \sigma(W_p \bullet V_I))]) . \quad (2\text{-}14)$$

Thus,

$$\begin{aligned}\hat{y}(k+1) &= f_L(k+1) + f_{NL}(k+1) \\ &= \sum_{i=0}^{N-1} a_i y(k-i) + \sum_{i=0}^{M-1} b_i u(k-i) + f(\sum_{i=1}^{N_L}[w_{i1}^O \cdots \sigma(\sum_{p=1}^{N_1} w_{pq}^2 \sigma(W_p \bullet V_I))])) \quad (2\text{-}15) \\ &= \sum_{i=0}^{N-1} a_i y(k-i) + \sum_{i=0}^{M-1} b_i u(k-i) + g[U,Y]\end{aligned}$$

where q is the number of neurons in the 2nd hidden layer, and $q \in [1, N_2]$, $W_p = [w_{1p}^1, w_{2p}^1, \cdots, w_{Mp}^1, w_{(M+1)p}^1, w_{(M+2)p}^1, \cdots, w_{(M+N)p}^1]^T$ is the weights vector of the first hidden layer, $V_I = [u(k), u(k-1), \cdots, u(k-M+1), y(k), y(k-1), \cdots, y(k-N+1)]^T$ is the input vector of the RANN, among which $Y = [y(k), y(k-1), \cdots, y(k-N+1)]$ is the output vector of the real system, and $U = [u(k), u(k-1), \cdots, u(k-M+1)]$ is the input vector of the system. $g(\mathrm{U}, \mathrm{Y})$ is the nonlinear function of the RANN.

The RANN identification model in Figure 2-8 can be used to represent a Parallel-Linear nonlinear system.

For most nonlinear systems, one of the three identification models can be used to describe their nonlinear behaviour based on *a priori* information. The architecture and weights of the RANNs may vary to approximate the output of the nonlinear system. The approximate accuracy of the RANN-based Models relies on two key factors: architecture and weights. How to find the approximate architecture and weights of RANNs will be discussed in the following chapters.

Chapter 3 Architecture approaching of RANNs

RANNs' architecture design and weights tuning are very crucial to the RANN-based nonlinear system identification and directly affect the approximation accuracy. Normally, the architecture of a RANN is pre-determined based on *a priori* knowledge of the nonlinear system or the designer's experience. The design process is basically a tedious trial-and-error process. The training of NNs focuses on tuning weights of NNs. Since GA is a powerful optimization algorithm (Holland 1975), in this chapter, a GA-based method is developed to approach the optimal architecture of RANNs in the sense of minimizing the identification error for the three types of RANN-based identification models discussed in Chapter 2. The purpose of this method is to find the optimal or approximately suboptimal feedforward network architecture.

3.1 The overall procedure

Figure 3-1 shows the framework of NN architecture optimization procedure. It can be seen that the GA works as the backbone and is "wrapped" around the modified BP, which tunes the weights of the RANNs, the adjustable linear parameters, as well as the time delays. The details of the modified BP algorithm will be elaborated in Chapter 4. The developed procedure of optimizing NN architecture using GA is similar to the general GA with adaptation laws to find the optimized architecture and weights of NNs. As illustrated in Figure 3-1, if the number of maximum generations is reached, the program exits from the GA cycles. Since one of the major advantages of NNs is their ability to generalize, a generalization process is carried out on the best RANN obtained from the GA.

Since in this research, *SISO* systems are studied, the proposed algorithm is mostly applied to the RANN-based *SISO* nonlinear systems identification.

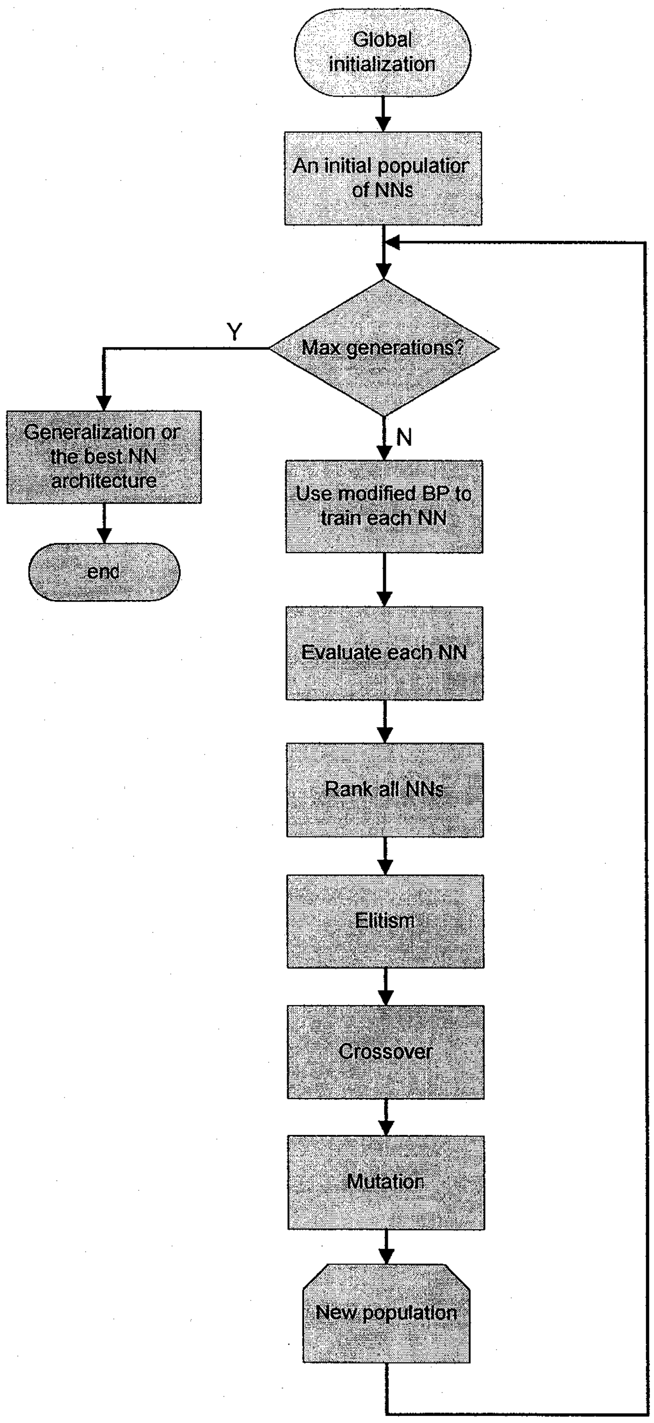


Figure 3-1 Framework of NN architecture optimization procedure

3.2 Direct Matrix Mapping Encoding method

In order to use GA to optimize NNs, the primary important thing is to choose an appropriate encoding method. In this study, an effective encoding method, called the Direct Matrix Mapping Encoding (DMME) method, is proposed.

DMME encodes an ANN with multiple hidden layers. The connections of neurons between two neighbouring layers are represented in a connection matrix. Only neurons in hidden layers are taken into account in the connection matrix. If there are L hidden layers, $(L+1)$ connection matrices will be generated. We assume that all connections in the output layer are fully connected.

The principle of this new method is illustrated as follows. The total number of neurons on the hidden layers is randomly selected from a given range $[0, N]$. N is the total number of neurons on the hidden layers and can be initialized as a large number for a particular application. For any hidden layer, its maximum number of neurons is N minus the sum of all neurons on its preceding hidden layers. For example, if the number of neurons on the 1st hidden layer is N_1, the maximum number of neurons on the 2nd hidden layer is $N - N_1$, and the number of neurons on this layer falls in the range of $[0, N - N_1]$. Further, if the number of neurons on the 2nd hidden layer is N_2, the maximum number of neurons on the 3rd hidden layer is $N - N_1 - N_2$, and the number of neurons on this layer falls in the range of $[0, N - N_1 - N_2]$, and so on.

A connection matrix C_{ij} describes all incoming connections of current layer i from its preceding neighbour layer j, the direction of which is from j to i. The mth row in C_{ij} denotes the mth neuron on the layer i. The nth column in C_{ij} denotes the nth neuron on the layer *j*. The value of each entry in C_{ij} is either "1" (with connection) or "*0*" (no connection).

As shown in Figure 3-2, the input layer of the NN is labelled as layer 0, the first and the second hidden layer is labelled as layer 1 and layer 2, respectively, until layer *L*. Hence there are altogether *L*+1 connection matrices denoted as C_{10}, C_{21}, :.., until $C_{(L+1)L}$.

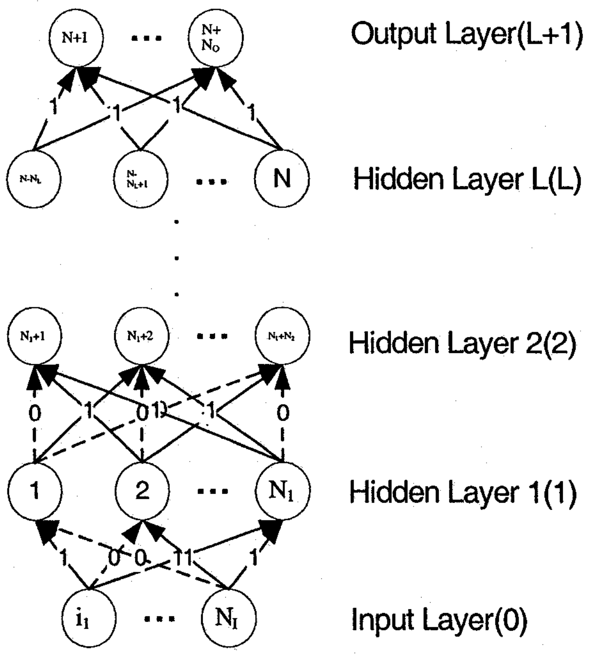


Figure 3-2 An example of neural network architecture

With solid arrows as valid connections (with the value "*1*"); and dashed arrows as empty connections (with the value "*0*") in Figure 3-2, these connection matrices can be written as:

$$C_{10} = \begin{bmatrix} c_{i_1 1} & \cdots & c_{N_I 1} \\ c_{i_1 2} & \cdots & c_{N_I 2} \\ & \cdots & \\ c_{i_1 N_1} & \cdots & c_{N_I N_1} \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 0 \\ 0 & \cdots & 1 \\ & \cdots & \\ 1 & \cdots & 1 \end{bmatrix}$$

$$C_{21} = \begin{bmatrix} c_{1(N_1+1)} & c_{2(N_1+1)} & \cdots & c_{N_1(N_1+1)} \\ c_{1(N_1+2)} & c_{2(N_1+2)} & \cdots & c_{N_1(N_1+2)} \\ & \cdots & \cdots & \\ c_{1(N_1+N_2)} & c_{2(N_1+N_2)} & \cdots & c_{N_1(N_1+N_2)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 1 \\ & \cdots & \cdots & \\ 0 & 1 & \cdots & 0 \end{bmatrix}$$

$$C_{(L+1)L} = \begin{bmatrix} c_{(N-N_L)(N+1)} & c_{(N-N_L+1)(N+1)} & \cdots & c_{N(N+1)} \\ & \cdots & \cdots & \\ c_{(N-N_L)N_O} & c_{(N-N_L+1)N_O} & \cdots & c_{NN_O} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ & \cdots & \cdots & \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

Note that we assume a full connection between the last hidden layer and the output layer, as indicated by the matrix $C_{(L+1)L}$. $L+1$ weight matrices W_{ij} with real-valued weights are also generated, in a one-to-one correspondence with C_{ij}, which act as the masking matrices of the former. The values of the weights are updated by a modified BP algorithm, before which each W_{ij} is multiplied by its corresponding C_{ij}.

For each W_{ij}, the weights and biases of a neuron are initially assigned a random value between -1.0 and $+1.0$ except 0, since a "0" simply means no connection at all, which is indicated in the connection matrix C_{ij}.

Therefore an NN of L hidden layers can be represented by its L+1 connection matrices C_{ij} and weight matrices W_{ij}. A chromosome in the GA is simply a group of these matrices.

3.3 Evolution of multi-hidden layer neural network architecture

For a specific NN, the numbers of neurons on the input and the output layers are fixed. Hence the GA optimization aims at finding out optimal or approximately optimal network architecture, with an optimal or approximately optimal number of neurons on each hidden layer and interconnections between neurons.

With the DMME method, each hidden layer is assigned a random number of neurons, which falls within a certain range, as discussed above in Section 3.2. Then the connections between neurons on neighbouring layers are randomly assigned either "0" (no connection) or "1" (with connection). A number of chromosomes are generated similarly and form the initial population.

The NN generated is evaluated by the average mean square error as shown follows between the output of the model and the plant in the training process.

$$E = \frac{1}{2P}\sum_{i=1}^{P} e_i^2 = \frac{1}{2P}\sum_{i=1}^{P}(y-\hat{y})^2 \tag{3-1}$$

where P is the number of training samples, e_i is the error between the target or real output y and the identification model output $\hat{y}$ for the i^{th} training sample.

The smaller the error is, the better the network architecture is. For each chromosome that corresponds to specific network architecture, a modified BP algorithm is used to train its weights, which is detailed in Chapter 4.

3.3.1 GA operators--Fitness Evaluation and Selection

Finding the optimized architecture C_{ij}^* of RANN using GA is a standard parametric optimization problem (Ramacher 1993). Since the minimum squares error algorithm is normally used to do the parametric identification (Ljung 1999), the network architecture will be optimized or approximately optimized based on the objective function F defined as:

$$\text{Minimize } F = \frac{1}{2P}\sum_{i_s=1}^{P} e_{i_s}^2 = \frac{1}{2P}\sum_{i_s=1}^{P}(y_{di_s} - \hat{y}_{li_s})^2 \tag{3-2}$$

To find C_{ij}^*, where connection matrix C_{ij}, $1 \le i \le (L+1)$, $0 \le j \le L$

Subject to:

Input $u(i_s)$ and output $y_d(i_s)$, $1 \le i_s \le P$,

The number of hidden layers $\le L$,

The total number of hidden neurons $\le N$,

where P is the number of training samples, e_{i_s} is the error between the target or real output y_{di_s} and the identification model output $\hat{y}_{li_s}$ for the i_s^{th} training sample.

To maintain a reasonable difference between relative fitness ratings of chromosomes and prevent a too-rapid takeover of some well performed chromosomes, ranking selection is used in this study. The ranking selection works as follows. Firstly, the population is sorted based on its fitness values from the smallest to the biggest, actually representing the chromosomes from the best to the worst. Then for a population with *n* chromosomes, the selection proportion for the k^{th} ranked chromosome is:

$$p_k = q_{\max} - \left(\frac{k-1}{n-1}\right)(q_{\max} - q_{\min}) \tag{3-3}$$

where $q_{\max}$ and $q_{\min}$ are the probabilities of the best and the worst chromosomes, respectively, and they satisfy:

$$q_{\max} + q_{\min} = \frac{2}{n} \tag{3-4}$$

where $q_{\max}$ and $q_{\min}$ can be set as: $q_{\max} = \frac{2}{n}$ and $q_{\min} = 0$. That means assigning the best chromosome with the highest proportion and no proportion for the worst chromosome.

Furthermore, in order to speed up the convergence, the elitism operator is adopted. The best 2% or at least 1 if it is less than 1, individual(s) within the population is (are) copied directly into the new generation.

3.3.2 GA operators--Crossover

Crossover occurs between a pair of randomly selected parents if a probability P, which is generated randomly, is smaller than a given bias (e.g. 0.7). Otherwise these parents are copied intact into the next generation.

A new crossover operator is defined specifically for this study. After two parents (chromosomes) are randomly selected from the population (with a crossover probability), the neurons to be manipulated are also randomly selected from each of them. For implementation, firstly a connection matrix is randomly selected. Subsequently, a row of this connection matrix is also randomly selected from each of the selected networks. The incoming connections of the two neurons (two selected rows) are then swapped.

Sometimes the dimensionality of these two matrices does not match. For example, suppose we have a matrix A of M_1 x N_1 and B of M_2 x N_2, where the number of columns $N_1 \neq N_2$, but $N_1 < N_2$. The rows to be swapped are a_1 in A and b_1 in B. In this case the entire a_1 and the first N_1 elements of b_1 are swapped.

Figure 3-3 gives out an example of crossover.

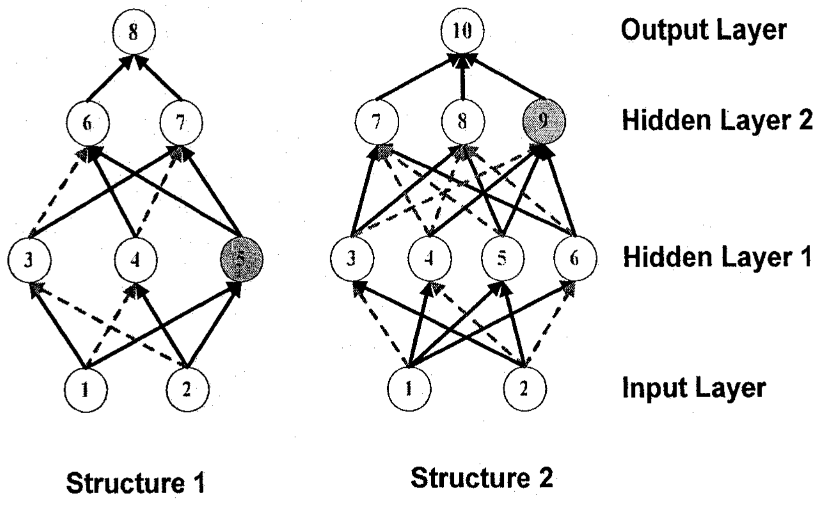


Figure 3-3 Two example network architecture before the crossover operator

The original matrices are:

Architecture 1:

$$C_{10}=\begin{bmatrix}1&0\\0&1\\1&1\end{bmatrix},\ C_{21}=\begin{bmatrix}0&1&1\\1&0&1\end{bmatrix},\ C_{32}=\begin{bmatrix}1&1\end{bmatrix}$$

Architecture 2:

$$C_{10}=\begin{bmatrix}0&1\\1&0\\1&1\\1&0\end{bmatrix},\ C_{21}=\begin{bmatrix}1&0&0&1\\1&0&1&0\\0&1&1&1\end{bmatrix},\ C_{32}=\begin{bmatrix}1&1&1\end{bmatrix}$$

Assume neuron 5 in architecture 1 and neuron 9 in architecture 2 are selected, that is, the 3rd row in C_{10} - [1 1], and the 3rd row in C_{21} - [0 1 1 1], respectively. According to the rule mentioned above, the first 2 elements of the latter are exchanged with the former ones. Hence the new rows become [0 1] and [1 1 1 1] respectively. The new connection matrices are:

Architecture 1:

$$C_{10} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix},\quad C_{21} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix},\quad C_{32} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

Architecture 2:

$$C_{10} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix},\quad C_{21} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix},\quad C_{32} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

The new network architecture is shown in Figure 3-4.

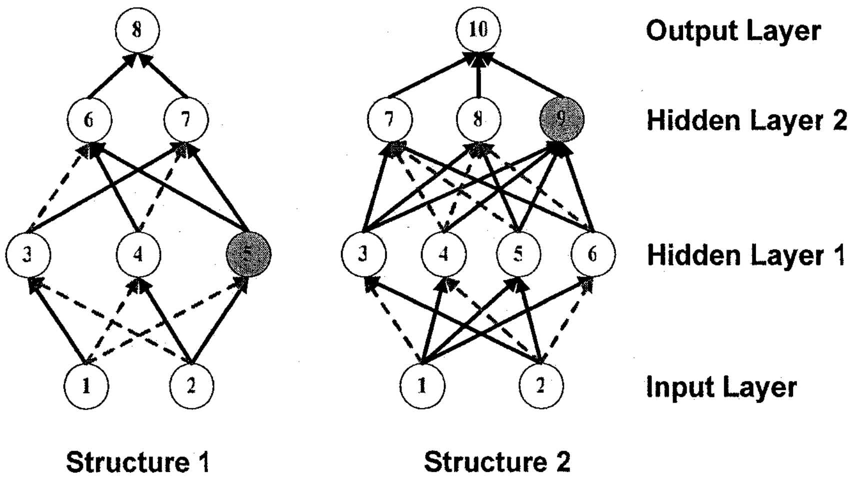


Figure 3-4 Two example network architecture after the crossover operator

3.3.3 GA operators--Mutation

The Mutation operator can prevent the loss of diversity by randomly altering some genes in a chromosome. Specifically in this study, a neuron to be mutated is randomly selected. All the connections from its preceding neurons to this neuron are then flipped.

Figure 3-5 shows an example of the mutation. Figure 3-5 (a) is the original NN. Assume neuron 4 is selected for mutation. Neurons 1 and 2 are its preceding neurons, with the connections [0 1] (i.e. no connection between 1 and 4, whereas a valid connection between 2 and 4). After the mutation, as shown Figure 3-5 (b), the connections become [1 0], that is, a valid connection between 1 and 4, and no connection between 2 and 4.

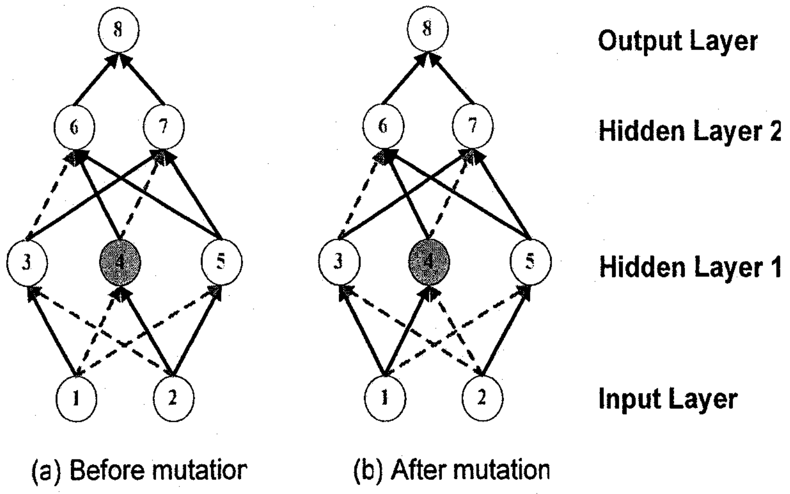


Figure 3-5 An example of mutation operation

The connection matrices before and after the mutation are shown below.

Before mutation:

$$C_{10} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad C_{21} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad C_{32} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

After mutation:

$$C_{10} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C_{21} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad C_{32} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

After the elitism, crossover and mutation operators, a new population is generated. After a sufficient number of generations, the optimal or approximately optimal architecture C_{ij}^{*} of an NN can be found by this GA algorithm.

In this chapter, a new neural network architecture optimization algorithm using GA is proposed. Detailed GA operators, such as selection, elitism, crossover, and mutation are presented. A new neural network encoding method - DMME is also introduced. The approximate optimal architecture of NN (C_{ij}^*) is obtained and is ready to be used in the RANN-based nonlinear system identification models.

Chapter 4 The modified BP algorithm and adaptation laws

The objective of adaptation laws is to find out the learning rules for the parameters in the identification models to approach their nominal values in the sense of minimizing the identification error. In this study, a modified BP algorithm is developed to tune the weights and other parameters of a given RANN. Known as the Generalized Delta Rule, BP is a predominant supervised training algorithm, which is based on the steepest gradient descent algorithm. The learning rate η, in the subsequent formulas, determines the rate of weight change. This is typically a number between 0 and 1. Usually the learning rate is chosen to be sufficiently small to increase the probability of finding out the optimum values in principle of the tuning parameters. However, the small learning rate means the slow convergence. Hence, in some cases, the learning rate is increased to speedup the convergence, with a side effect of increased oscillation. Usually a momentum term α is used to adjust the balance (McClelland and Rumelhart 1988). This term indicates how much the change of a previous weight should influence the change of a current weight. When the momentum term α is applied, the weights w_{ij} are updated as:

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij} + \alpha(w_{ij}(k) - w_{ij}(k-1)) \qquad (4\text{-}1)$$

where

$$\Delta w_{ij}(k+1) = \Delta w_{ij}(k) + \alpha(w_{ij}(k) - w_{ij}(k-1)) \qquad (4\text{-}2)$$

The modified BP differs from the traditional BP in that the traditional BP is mostly used to tune weights only; whereas the modified BP in this study is used to tune both weights and all other adjustable parameters, such as linear part's parameters and the number of delays of the input and the output of the system. The work flow of the modified BP is illustrated in Figure 4-1.

In this study, all RANNs obtained from the GA are trained off-line, that is, to use historical training data pairs to update parameters for epochs till a specified number of iterations or an acceptable error between the target and the actual output(s) have been reached (Prasad and Bequette 2003).

In the following sections, general formulas are used to describe the adaptation laws of parameters, e.g. weights. In case that some neurons in the previous layer are not connected to the current neuron; all the formulas can still hold. The only difference is that the number of the incoming connections is not the same as the number of neurons in the previous layer. In other words, the architecture optimization algorithm will not affect the adaptation of the RANN parameters.

4.1 The modified BP algorithm

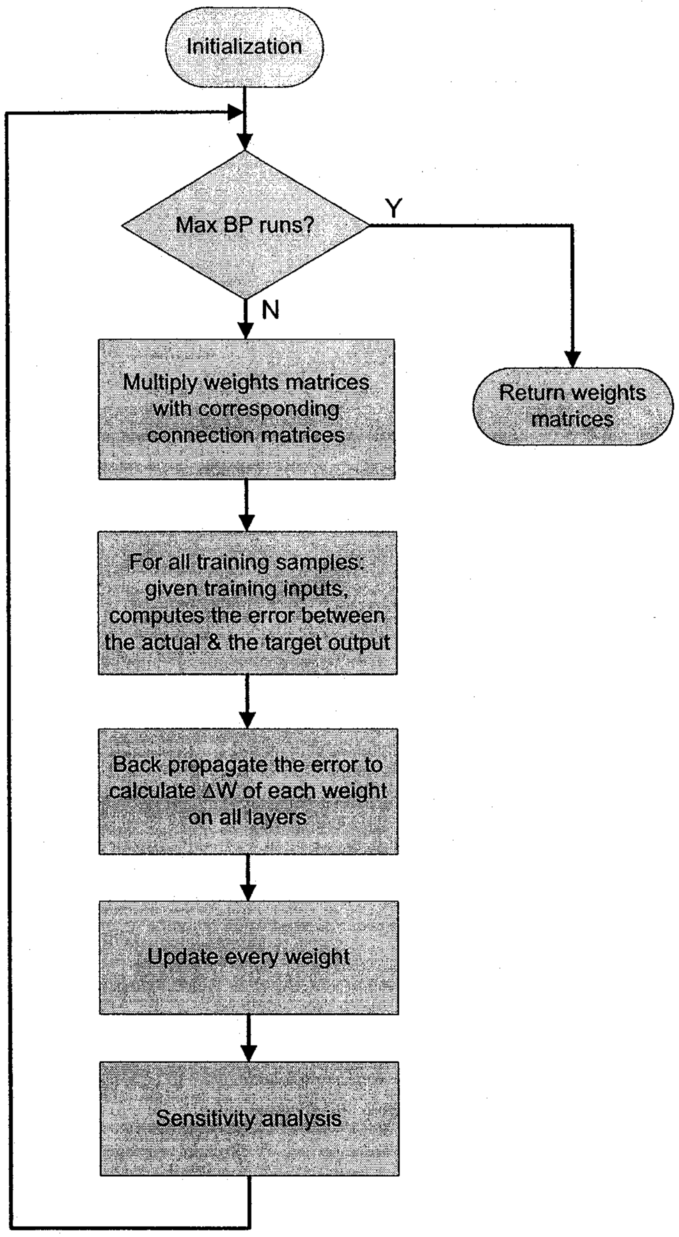

Initialization
Max BP runs?
Y
N
Multiply weights matrices with corresponding connection matrices
Return weights matrices
For all training samples: given training inputs, computes the error between the actual & the target output
Back propagate the error to calculate ΔW of each weight on all layers
Update every weight
Sensitivity analysis

Figure 4-1 The work flow of the modified BP algorithm

Figure 4-1 shows the work flow of the modified BP algorithm. After the weights are updated, a sensitivity analysis is performed to trim the NNs (Karnin 1990; Prasad and Bequette 2003). The weakest neuron on hidden layers is removed from the network to avoid redundancy and excessive growth of the network. When sensitivity analysis is used, to the input layer, the noise, such as Gaussian white noise (Barrett 1963), and redundancy in the input signal can be filtered out, and to the 1st hidden layer, the number of states in the NN models can be reduced. The sensitivity of weight w_{ij} can be derived by (Karnin 1990):

$$\widehat{S}_{ij} = \sum_{p=0}^{P-1} \frac{\partial E^{(p)}}{\partial w_{ij}} \Delta w_{ij}^{(p)} \frac{w_{ij}^F}{w_{ij}^F - w_{ij}^I} \tag{4-3}$$

where w_{ij}^F is the final weight, and w_{ij}^I is the initial weight, P is the total epochs of the training.

4.2 Adaptation laws for the Single RANN Model

In the Single RANN identification Model, there is only one RANN block to represent a completely unknown nonlinear system without any linear part accompanying with it. As shown in Figure 2-5, the inputs to the RANN are the last step control input, last step real output of the plant, and their up to M and N delayed values, respectively. And, since only *SISO* systems are considered here, there is only one output neuron in the RANN. The adaptation laws for this model are the weight adaptation laws only.

4.2.1 Weight adaptation laws

As an example, assume there are only two hidden layers in the architecture as shown below.

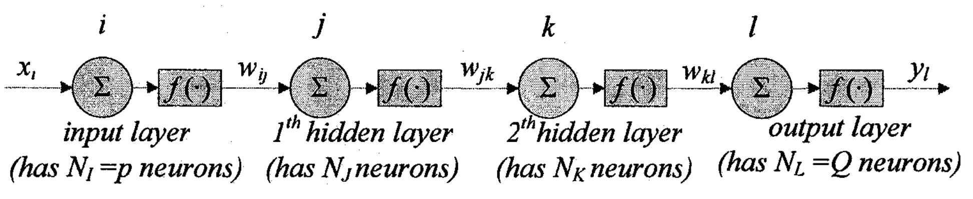


Figure 4-2 The structure of two hidden-layer NNs

Since the bipolar sigmoid function is used as the activation function in most simulations in this study, the weight adaptation laws under the bipolar sigmoid function are described here.

For the j^{th} neuron in any layer:

$$net_j = \sum_{i=1}^{M} w_{ij} x_j + \theta_j \tag{4-4}$$

$$y_j = f(net_j) = \frac{1 - e^{-net_j}}{1 + e^{-net_j}} = \frac{2}{1 + e^{-net_j}} - 1 \tag{4-5}$$

where w_{ij} is the incoming weights of the j^{th} neuron; x_i is the input of the j^{th} neuron, which equals to the output of the i^{th} neuron y_i on the previous layer, and θ_j is the bias of the j^{th} neuron.

Thus,

$$y'_j = f'(net_j) = \frac{1}{2}(1 - y_j^2) \tag{4-6}$$

The corresponding back-propagated errors in the output layer, the *2*nd hidden layer, and the *1*st hidden layer for the p^{th} training sample are shown as follows, respectively.

$$\delta_{kl}^{(p)} = -\frac{\partial E^{(p)}}{\partial net_l} = (y_d^{(p)} - y_l^{(p)}) f'(net_l) = \frac{1}{2}(y_d^{(p)} - y_l^{(p)})(1 - (y_l^{(p)})^2) \tag{4-7}$$

$$\delta_{jk}^{(p)} = -\frac{\partial E^{(p)}}{\partial net_k} = \delta_{kl}^{(p)} w_{kl} f'(net_k) = \frac{1}{2}\delta_{kl}^{(p)} w_{kl} (1 - (y_k^{(p)})^2) \tag{4-8}$$

$$\delta_{ij}^{(p)} = -\frac{\partial E^{(p)}}{\partial net_j} = \sum_{k=1}^{N_K} (\delta_{jk}^{(p)} w_{jk}) f'(net_j) = \frac{1}{2}\sum_{k=1}^{N_K} (\delta_{jk}^{(p)} w_{jk})(1 - (y_j^{(p)})^2) \tag{4-9}$$

For the incoming weight w_{ij} of the j^{th} neuron,

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \sum_{m=1}^{P} \frac{\partial E^{(m)}}{\partial w_{ij}} \tag{4-10}$$

where η is the learning rate, and, in general, $0 < \eta < 1$; P is the number of total training samples. Suppose there are Q output neurons in the output layer, the mean square error of a neural network for the p^{th} training sample is:

$$E^{(p)} = \frac{1}{2}\sum_{j=1}^{Q} e_j^2 = \frac{1}{2}\sum_{j=1}^{Q} (y_{dj}^{(p)} - y_{lj}^{(p)})^2 \tag{4-11}$$

where $y^{(p)}{}_{dj}$ and $y^{(p)}{}_{lj}$ represent the j^{th} output neuron's real/target output and its neural network output, respectively. For *SISO* systems, *Q=1*.

Based on the gradient descent learning algorithm, the weights adaptation laws for the output layer, the *2nd* hidden layer, and the *1st* hidden layer for the p^{th} training sample are derived:

$$w_{kl}(t+1) = w_{kl}(t) + \sum_{m=1}^{P} \Delta w_{kl}^{(m)} = w_{kl}(t) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} y_k^{(m)} \tag{4-12}$$

$$w_{jk}(t+1) = w_{jk}(t) + \sum_{m=1}^{P} \Delta w_{jk}^{(m)} = w_{jk}(t) + \eta \sum_{m=1}^{P} \delta_{jk}^{(m)} y_j^{(m)} \tag{4-13}$$

$$w_{ij}(t+1) = w_{ij}(t) + \sum_{m=1}^{P} \Delta w_{ij}^{(m)} = w_{ij}(t) + \eta \sum_{m=1}^{P} \delta_{ij}^{(m)} y_i^{(m)} \tag{4-14}$$

4.2.2 Number of time delays adaptation laws

Since the time delays are the parameters for the delayed input and output signals, they are the inputs of the neurons in the first hidden layer, as shown in the following figure.

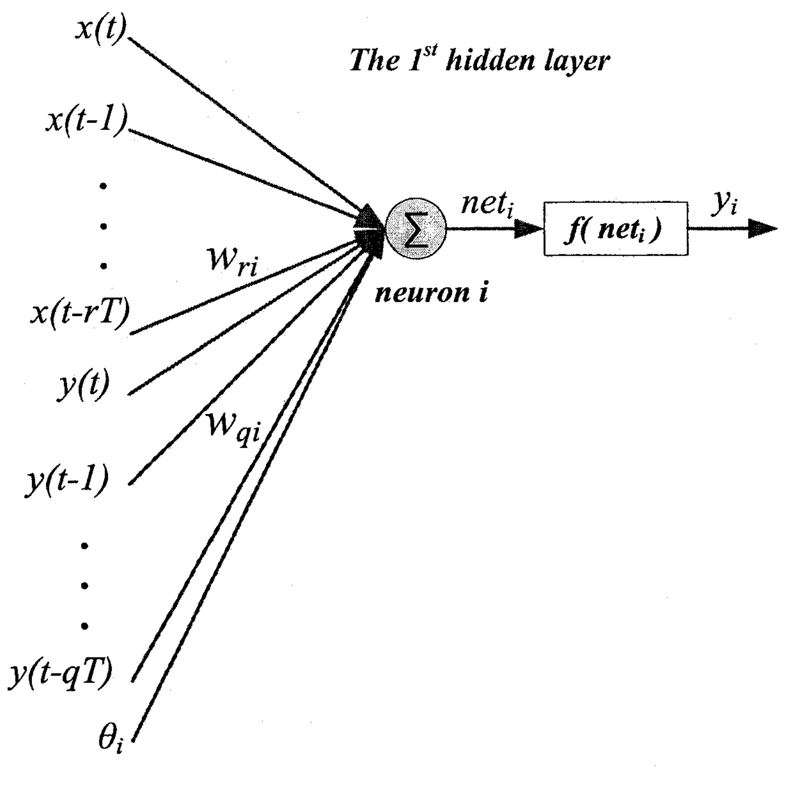


Figure 4-3 Time delays of the neural network

For the i^{th} hidden neuron on the first hidden layer:

$$net_i^{1(p)} = \sum_{r=0}^{M-1} w_{ri}^{(p)} u(t-rT) + \sum_{q=0}^{N-1} w_{qi}^{(p)} y(t-qT) \tag{4-15}$$

where r and q are the number of delays of input and output signals, respectively, and T is time interval.

Let $\tau_N = qT$. For the p^{th} training data set, the adaptation laws for the time delays τ_N are derived:

$$\Delta\tau_N^{(p)} = -\eta \frac{\partial E^{(p)}}{\partial \tau_N^{(p)}} = -\eta \frac{\partial E^{(p)}}{\partial net_1^{O(p)}} \frac{\partial net_1^{O(p)}}{\partial \tau_N^{(p)}} = \eta \delta_{kl}^{(p)} \frac{\partial net_1^{O(p)}}{\partial \tau_N^{(p)}} \tag{4-16}$$

where,

$$\frac{\partial net_1^{O(p)}}{\partial \tau_N^{(p)}} = \frac{\partial \sum_{k=1}^{N_L} w_{kl} y_k^{(p)}}{\partial \tau_N^{(p)}} = \sum_{k=1}^{N_L} w_{kl} \frac{\partial y_k^{(p)}}{\partial \tau_N^{(p)}} = \sum_{k=1}^{N_L} w_{kl} \left(\frac{\partial \sum_{j=1}^{N_{L-1}} w_{jk} y_j^{(p)}}{\partial \tau_N^{(p)}} \right) y_k'^{(p)}$$

$$= \sum_{k=1}^{N_L} w_{kl} y_k'^{(p)} [\sum_{j=1}^{N_{L-1}} w_{jk} y_j'^{(p)} (\sum_{i=1}^{N_{L-2}} w_{ij} y_i'^{(p)} (\frac{\partial}{\partial \tau_N^{(p)}} [\sum_{r=0}^{M} w_{ri} u^{(p)}(t-\tau_M^{(p)}) + \sum_{q=0}^{N} w_{qi} y^{(p)}(t-\tau_N^{(p)})]))]$$

$$\cong \sum_{k=1}^{N_L} w_{kl} y_k'^{(p)} [\sum_{j=1}^{N_{L-1}} w_{jk} y_j'^{(p)} (\sum_{i=1}^{N_{L-2}} w_{ij} y_i'^{(p)} (\sum_{q=0}^{N} w_{qi} \frac{y^{(p)}(t-\tau_N^{(p)}) - y^{(p)}(t-\tau_N^{(p)}-T)}{T}))] \tag{4-17}$$

For all training data pairs, the adaptation laws for the time delay τ_N are obtained:

$$\tau_N(t+1) = \tau_N(t) + \sum_{m=1}^{P} \Delta \tau_N^{(m)} \tag{4-18}$$

Let $\tau_M = rT$. For the p^{th} training data set, the adaptation laws for the time delays τ_M are derived as:

$$\Delta \tau_M^{(p)} = -\eta \frac{\partial E^{(p)}}{\partial \tau_M^{(p)}} = -\eta \frac{\partial E^{(p)}}{\partial net_1^{O(p)}} \frac{\partial net_1^{O(p)}}{\partial \tau_M^{(p)}} = \eta \delta_{kl}^{(p)} \frac{\partial net_1^{O(p)}}{\partial \tau_M^{(p)}} \tag{4-19}$$

where,

$$\frac{\partial net_1^{O(p)}}{\partial \tau_M^{(p)}} \cong \sum_{k=1}^{N_L} w_{kl} y_k'^{(p)} [\sum_{j=1}^{N_{L-1}} w_{jk} y_j'^{(p)} (\sum_{i=1}^{N_{L-2}} w_{ij} y_i'^{(p)} (\sum_{r=0}^{M} w_{ri} \frac{u^{(p)}(t-\tau_M^{(p)}) - u^{(p)}(t-\tau_M^{(p)}-T)}{T}))] \tag{4-20}$$

For all training data pairs, the adaptation laws for the time delays τ_M are obtained:

$$\tau_M(t+1) = \tau_M(t) + \sum_{m=1}^{P} \Delta \tau_M^{(m)} \tag{4-21}$$

4.3 Adaptation laws for the Serial-Linear Model

In the Serial-Linear Model, the output of the identification system $\hat{y}(k+1)$ is composed of both outputs of the RANN and the cascaded linear part, as described by Eq.

(2-10) and Eq. (2-11).

4.3.1 Weight adaptation laws

- **Output layer weight adaptation laws**

For the p^{th} training sample, based on Figure 2-7, the output layer weight adaptation laws are described as:

$$\Delta w_{kl}^{(p)} = -\eta \frac{\partial E^{(p)}}{\partial w_{kl}} = -\eta \frac{\partial E^{(p)}}{\partial \hat{y}(k+1)} \frac{\partial \hat{y}(k+1)}{\partial w_{kl}} = \eta \delta_{kl}^{(p)} \frac{\partial \hat{y}(k+1)}{\partial w_{kl}} \qquad (4\text{-}22)$$

where

$$\frac{\partial \hat{y}(k+1)}{\partial w_{kl}} = \sum_{i=0}^{N-1} \hat{a}_i \frac{\partial y(k-i)}{\partial w_{kl}} + \sum_{i=0}^{M-1} \hat{b}_i \frac{\partial h(k-i)}{\partial w_{kl}}. \qquad (4\text{-}23)$$

Since $y(k\text{-}i)$ is the real output of the plant, and has no relation with w_{kl}, the change of output $\hat{y}(k+1)$ with the weight w_{kl} are obtained as:

$$\frac{\partial \hat{y}(k+1)}{\partial w_{kl}} = 0 + \sum_{i=0}^{M-1} \hat{b}_i \frac{\partial h(k-i)}{\partial w_{kl}} = \sum_{i=0}^{M-1} \hat{b}_i f'(net_1^{O(k-i)}) y_k^{(k-i)} . \qquad (4\text{-}24)$$

Thus, the adaptation laws (Eq. 4-22) becomes:

$$\Delta w_{kl}^{(p)} = \eta \delta_{kl}^{(p)} \frac{\partial \hat{y}(k+1)}{\partial w_{kl}} = \eta (y_d^{(p)} - y_l^{(p)}) \sum_{i=0}^{M-1} \hat{b}_i f'(net_1^{O(k-i)}) y_k^{(k-i)} \qquad (4\text{-}25)$$

where $y_l^{(p)} = \hat{y}(k+1)$ is the output of the identification system, and $y_d^{(p)} = y(k+1)$ is the real output of the plant.

For all P training samples, the weights in the output layer are updated as follows.

$$\begin{aligned} w_{kl}(t+1) &= w_{kl}(t) + \sum_{m=1}^{P} \Delta w_{kl}^{(m)} = w_{kl}(t) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} \frac{\partial \hat{y}(k+1)}{\partial w_{kl}} \\ &= w_{kl}(t) + \eta \sum_{m=1}^{P} [(y_d^{(m)} - y_l^{(m)}) \sum_{i=0}^{M-1} \hat{b}_i f'(net_1^{O(k-i)}) y_k^{(k-i)}] \end{aligned} \tag{4-26}$$

- **Hidden layers weight adaptation laws**

For the p^{th} training sample, the hidden layers weight adaptation laws are:

$$\begin{aligned} \Delta w_{jk}^{(p)} &= -\eta \frac{\partial E^{(p)}}{\partial w_{jk}} \\ &= -\eta \frac{\partial E^{(p)}}{\partial \hat{y}(k+1)} \frac{\partial \hat{y}(k+1)}{\partial w_{jk}} \\ &= \eta \delta_{kl} \sum_{i=0}^{M-1} \hat{b}_i f'(net_1^{O(k-i)}) w_{kl} f'(net_k) y_j^{(p)}) \end{aligned} \quad . \tag{4-27}$$

For all P training samples, the weights in hidden layers are updated as follows.

$$w_{jk}(t+1) = w_{jk}(t) + \sum_{m=1}^{P} \Delta w_{jk}^{(m)} \tag{4-28}$$

4.3.2 Linear part parameters adaptation laws

Following the same principle of the steepest descent gradient algorithm, the updating of the coefficients of the linear mode is derived as follows.

$$\Delta \hat{a}_i^{(p)} = -\eta \frac{\partial E^{(p)}}{\partial \hat{a}_i} = -\eta \frac{\partial E^{(p)}}{\partial \hat{y}(k+1)} \frac{\partial \hat{y}(k+1)}{\partial \hat{a}_i} = \eta \delta_{kl}^{(p)} \frac{\partial \hat{y}(k+1)}{\partial \hat{a}_i} \tag{4-29}$$

where

$$\begin{aligned} \frac{\partial \hat{y}(k+1)}{\partial \hat{a}_i} &= \sum_{i=0}^{N-1} \hat{a}_i \frac{\partial y(k-i)}{\partial \hat{a}_i} + \sum_{i=0}^{M-1} \hat{b}_i \frac{\partial h(k-i)}{\partial \hat{a}_i} \\ &= y(k-i) + 0 = y(k-i) \end{aligned} \quad . \tag{4-30}$$

Therefore, Eq. 4-29 is rewritten as:

$$\Delta\hat{a}_i^{(p)} = \eta\delta_{kl}^{(p)}\frac{\partial\hat{y}(k+1)}{\partial\hat{a}_i} = \eta\delta_{kl}^{(p)}y(k-i) = \eta(y_d^{(p)} - y_l^{(p)})y(k-i). \quad (4\text{-}31)$$

For all P training samples, the coefficients of a_i are updated as:

$$\begin{aligned}\hat{a}_i(k+1) &= \hat{a}_i(k) + \sum_{m=1}^{P}\Delta\hat{a}_i^{(m)} \\ &= \hat{a}_i(k) + \eta\sum_{m=1}^{P}\delta_{kl}^{(m)}y(k-i) \\ &= \hat{a}_i(k) + \eta\sum_{m=1}^{P}(y_d^{(m)} - y_l^{(m)})y(k-i)\end{aligned} \quad . \qquad (4\text{-}32)$$

Similarly, the coefficients of b_i are updated as:

$$\begin{aligned}\hat{b}_i(k+1) &= \hat{b}_i(k) + \sum_{m=1}^{P}\Delta\hat{b}_i^{(m)} \\ &= \hat{b}_i(k) + \eta\sum_{m=1}^{P}\delta_{kl}^{(m)}h(k-i) \\ &= \hat{b}_i(k) + \eta\sum_{m=1}^{P}(y_d^{(m)} - y_l^{(m)})h(k-i)\end{aligned} \quad . \qquad (4\text{-}33)$$

4.3.3 Number of time delays adaptation laws

For the p^{th} training data set, the adaptation law for τ_N is:

$$\Delta\tau_N^{(p)} = -\eta\frac{\partial E^{(p)}}{\partial\tau_N^{(p)}} = -\eta\frac{\partial E^{(p)}}{\partial\hat{y}^{(p)}(k+1)}\frac{\partial\hat{y}^{(p)}(k+1)}{\partial\tau_N^{(p)}} = \eta\delta_{kl}^{(p)}\frac{\partial\hat{y}^{(p)}(k+1)}{\partial\tau_N^{(p)}} \qquad (4\text{-}34)$$

where,

$$\frac{\partial \hat{y}^{(p)}(k+1)}{\partial \tau_N^{(p)}}=\frac{\partial}{\partial \tau_N^{(p)}}[\sum_{i=0}^{N-1}\hat{a}_i y(k-i)+\sum_{i=0}^{M-1}\hat{b}_i h(k-i)]$$
$$=\sum_{i=0}^{N-1}\hat{a}_i\frac{\partial y(k-i)}{\partial \tau_N^{(p)}}+\sum_{i=0}^{M-1}\hat{b}_i\frac{\partial h(k-i)}{\partial \tau_N^{(p)}}$$
$$\cong\sum_{i=0}^{N-1}\hat{a}_i\frac{y^{(p)}(t-\tau_N^{(p)})-y^{(p)}(t-\tau_N^{(p)}-T)}{T}$$
$$+\sum_{i=0}^{M-1}\hat{b}_i(\sum_{k=1}^{N_L}w_{kl}y_k'^{(p)}[\sum_{j=1}^{N_{L-1}}w_{jk}y_j'^{(p)}(\sum_{i=1}^{N_{L-2}}w_{ij}y_i'^{(p)}(\sum_{q=0}^{N-1}w_{qi}\frac{y^{(p)}(t-\tau_N^{(p)})-y^{(p)}(t-\tau_N^{(p)}-T)}{T}))])$$

(4-35)

For all training data pairs, the adaptation law for τ_N is:

$$\tau_N(t+1)=\tau_N(t)+\sum_{m=1}^{P}\Delta\tau_N^{(m)} \tag{4-36}$$

For the p^{th} training data set, the adaptation law for τ_M is:

$$\Delta\tau_M^{(p)}=-\eta\frac{\partial E^{(p)}}{\partial \tau_M^{(p)}}=-\eta\frac{\partial E^{(p)}}{\partial \hat{y}^{(p)}(k+1)}\frac{\partial \hat{y}^{(p)}(k+1)}{\partial \tau_M^{(p)}}=\eta\delta_{kl}^{(p)}\frac{\partial \hat{y}^{(p)}(k+1)}{\partial \tau_M^{(p)}} \tag{4-37}$$

where,

$$\frac{\partial \hat{y}^{(p)}(k+1)}{\partial \tau_M^{(p)}}=\frac{\partial}{\partial \tau_M^{(p)}}[\sum_{i=0}^{N-1}\hat{a}_i y(k-i)+\sum_{i=0}^{M-1}\hat{b}_i h(k-i)]$$
$$=\sum_{i=0}^{M-1}\hat{b}_i\frac{\partial h(k-i)}{\partial \tau_M^{(p)}}$$
$$\cong\sum_{i=0}^{M-1}\hat{b}_i(\sum_{k=1}^{N_L}w_{kl}y_k'^{(p)}[\sum_{j=1}^{N_{L-1}}w_{jk}y_j'^{(p)}(\sum_{i=1}^{N_{L-2}}w_{ij}y_i'^{(p)}(\sum_{r=0}^{M}w_{ri}\frac{u^{(p)}(t-\tau_M^{(p)})-u^{(p)}(t-\tau_M^{(p)}-T)}{T}))])$$

.

(4-38)

For all training data pairs, the adaptation law for τ_N is:

$$\tau_M(t+1)=\tau_M(t)+\sum_{m=1}^{P}\Delta\tau_M^{(m)} \tag{4-39}$$

4.4 Adaptation laws for the Parallel-Linear Model

In the Parallel-Linear Model, the output of the identification system $\hat{y}(k+1)$ is the summation of the output of linear part $f_L(k+1)$ and the output of RANN $f_{NL}(k+1)$, as described in Eq. (2-15).

4.4.1 Weight adaptation laws

- **Output layer weight adaptation laws**

For the p^{th} training sample, base on Figure 2-8, the changes of the output weights are derived as:

$$\Delta w_{kl}^{(p)} = -\eta \frac{\partial E^{(p)}}{\partial w_{kl}} = -\eta \frac{\partial E^{(p)}}{\partial \hat{y}(k+1)} \frac{\partial \hat{y}(k+1)}{\partial w_{kl}}. \tag{4-40}$$

The back-propagated error of the output layer is defined as:

$$\delta_{kl}^{(p)} := -\frac{\partial E^{(p)}}{\partial \hat{y}(k+1)} = -\frac{\partial (\frac{1}{2}(y_d^{(p)} - y_l^{(p)})^2)}{\partial y_l^{(p)}} = (y_d^{(p)} - y_l^{(p)}) \tag{4-41}$$

where $y_l^{(p)} = \hat{y}(k+1)$ is the output of the identification system, and $y_d^{(p)} = y(k+1)$ is the real output of the plant.

and,

$$\frac{\partial \hat{y}(k+1)}{\partial w_{kl}} = \frac{\partial f_L(k+1)}{\partial w_{kl}} + \frac{\partial f_{NL}(k+1)}{\partial w_{kl}} \tag{4-42}$$

where

$$\frac{\partial f_L(k+1)}{\partial w_{kl}} = \sum_{i=0}^{N-1} a_i \frac{\partial y(k-i)}{\partial w_{kl}} + \sum_{i=0}^{M-1} b_i \frac{\partial u(k-i)}{\partial w_{kl}}. \tag{4-43}$$

Since both *y(k-i)* and *u(k-i)* are one of the samples of the real plant output and input, and are independent, the Eq. 4-43 becomes:

$$\begin{aligned}\frac{\partial f_L(k+1)}{\partial w_{kl}} &= \sum_{i=0}^{N-1} a_i \frac{\partial y(k-i)}{\partial w_{kl}} + \sum_{i=0}^{M-1} b_i \frac{\partial u(k-i)}{\partial w_{kl}} \\ &= 0+0=0\end{aligned} \tag{4-44}$$

Here $\frac{\partial y(k-i)}{\partial w_{kl}} = 0$ because *y(k-i)* is the real plant output, and it's not related to w_{kl}.

In case that *y(k-i)* is the output of the RANN, the computation of $\frac{\partial \hat{y}(k+1)}{\partial w_{kl}}$ becomes a recursive process (Piche 1994),

and

$$\frac{\partial f_{NL}(k+1)}{\partial w_{kl}} = \frac{\partial f(net_1^O)}{\partial w_{kl}} = \frac{\partial f(net_1^O)}{\partial net_1^O} \frac{\partial (\sum_{k=1}^{N_k} w_{kl} y_k^{(p)})}{\partial w_{kl}} = f'(net_1^O) y_k^{(p)}. \tag{4-45}$$

The Eq. 4-42 is rewritten as:

$$\begin{aligned}\frac{\partial \hat{y}(k+1)}{\partial w_{kl}} &= \frac{\partial f_L(k+1)}{\partial w_{kl}} + \frac{\partial f_{NL}(k+1)}{\partial w_{kl}} \\ &= 0 + f'(net_1^O) y_k^{(p)} \\ &= f'(net_1^O) y_k^{(p)}\end{aligned} \quad . \tag{4-46}$$

The change of output layer weight (Eq. 4-40) is shown:

$$\Delta w_{kl}^{(p)} = -\eta \frac{\partial E^{(p)}}{\partial w_{kl}} = \eta (y_d^{(p)} - y_l^{(p)}) f'(net_1^O) y_k^{(p)} . \tag{4-47}$$

For the p^{th} training sample, the adaptation laws for the output layer weights are:

$$w_{kl}(t+1) = w_{kl}(t) + \Delta w_{kl}^{(p)} = w_{kl}(t) + \eta (y_d^{(p)} - y_l^{(p)}) f'(net_1^O) y_k^{(p)} . \tag{4-48}$$

For all *P* training samples, the weights in the output layer are updated as:

$$w_{kl}(t+1) = w_{kl}(t) + \sum_{m=1}^{P} \Delta w_{kl}^{(m)} = w_{kl}(t) + \eta \sum_{m=1}^{P} (y_d^{(m)} - y_l^{(m)}) y_l'^{(p)} y_k^{(m)} \quad (4\text{-}49)$$

- **Hidden layers weight adaptation laws**

For the p^{th} training sample, the weights in hidden layers are updated as:

$$\begin{aligned} w_{jk}(t+1) &= w_{jk}(t) + \Delta w_{jk}^{(p)} = w_{jk}(t) - \eta \frac{\partial E^{(p)}}{\partial w_{jk}} \\ &= w_{jk}(t) - \eta \frac{\partial E^{(p)}}{\partial \hat{y}(k+1)} \frac{\partial \hat{y}(k+1)}{\partial w_{jk}} \\ &= w_{jk}(t) + \eta \delta_{kl} f'(net_1^O) w_{kl} f'(net_k) y_j^{(p)} \end{aligned} \quad . \quad (4\text{-}50)$$

For all P training samples, the weights in hidden layers are updated as follows.

$$\begin{aligned} w_{jk}(t+1) &= w_{jk}(t) + \sum_{m=1}^{P} \Delta w_{jk}^{(m)} \\ &= w_{jk}(t) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} f'(net_1^O) w_{kl} f'(net_k) y_j^{(m)} \end{aligned} \quad (4\text{-}51)$$

4.4.2 Linear part parameters adaptation laws

Based on the same principle of the steepest descent gradient algorithm, the coefficients of the linear mode are updated as:

$$\begin{aligned} \Delta \hat{a}_i^{(p)} &= -\eta \frac{\partial E^{(p)}}{\partial \hat{a}_i} = -\eta \frac{\partial E^{(p)}}{\partial \hat{y}(k+1)} \frac{\partial \hat{y}(k+1)}{\partial \hat{a}_i} = \eta \delta_{kl}^{(p)} \frac{\partial \hat{y}(k+1)}{\partial \hat{a}_i} \\ &= \eta \delta_{kl}^{(p)} \left(\frac{\partial f_L^{(p)}(k+1)}{\partial \hat{a}_i} + \frac{\partial f_{NL}^{(p)}(k+1)}{\partial \hat{a}_i} \right) \end{aligned} \quad (4\text{-}52)$$

where

$$\frac{\partial f_L^{(p)}(k+1)}{\partial \hat{a}_i} = \sum_{i=0}^{N-1} \frac{\partial[\hat{a}_i y(k-i)]}{\partial \hat{a}_i} + \sum_{i=0}^{M-1} \hat{b}_i \frac{\partial u(k-i)}{\partial \hat{a}_i}$$
$$= y(k-i) + 0 = y(k-i) \qquad (4\text{-}53)$$

Since $f_{NL}^{(p)}(k+1)$ is unrelated to a_i, the change of it with respect to a_i is:

$$\frac{\partial f_{NL}^{(p)}(k+1)}{\partial \hat{a}_i} = \frac{\partial f(net_1^O)}{\partial \hat{a}_i} = 0. \qquad (4\text{-}54)$$

The change of $\hat{y}(k+1)$ with respect to a_i becomes:

$$\frac{\partial \hat{y}(k+1)}{\partial \hat{a}_i} = \frac{\partial f_L^{(p)}(k+1)}{\partial \hat{a}_i} + \frac{\partial f_{NL}^{(p)}(k+1)}{\partial \hat{a}_i} = y(k-i). \qquad (4\text{-}55)$$

Therefore, one obtains the following update law:

$$\Delta \hat{a}_i^{(p)} = \eta \delta_{kl}^{(p)} \frac{\partial \hat{y}(k+1)}{\partial \hat{a}_i} = \eta \delta_{kl}^{(p)} y(k-i) = \eta (y_d^{(p)} - y_l^{(p)}) y(k-i). \qquad (4\text{-}56)$$

For all P training samples, the coefficient a_i is updated as:

$$\hat{a}_i(k+1) = \hat{a}_i(k) + \sum_{m=1}^{P} \Delta \hat{a}_i^{(m)} = \hat{a}_i(k) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} y(k-i)$$
$$= \hat{a}_i(k) + \eta \sum_{m=1}^{P} (y_d^{(m)} - y_l^{(m)}) y(k-i) \qquad (4\text{-}57)$$

Similarly, the coefficient b_i is updated as:

$$\hat{b}_i(k+1) = \hat{b}_i(k) + \sum_{m=1}^{P} \Delta \hat{b}_i^{(m)} = \hat{b}_i(k) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} u(k-i)$$
$$= \hat{b}_i(k) + \eta \sum_{m=1}^{P} (y_d^{(m)} - y_l^{(m)}) u(k-i) \qquad (4\text{-}58)$$

4.4.3 Number of time delays adaptation laws

In the parallel-linear model, the output of the identification model is composed of both outputs of linear part and RANN. The time delays of the input and output signals only affect the first hidden layer.

For the p^{th} training data set, the change of τ_N is:

$$\begin{aligned}
\Delta\tau_N^{(p)} &= -\eta\frac{\partial E^{(p)}}{\partial\tau_N^{(p)}} = -\eta\frac{\partial E^{(p)}}{\partial\hat{y}^{(p)}(k+1)}\frac{\partial\hat{y}^{(p)}(k+1)}{\partial\tau_N^{(p)}} = \eta\delta_{kl}[\frac{\partial\hat{y}_L^{(p)}(k+1)}{\partial\tau_N^{(p)}} + \frac{\partial\hat{y}_{NL}^{(p)}(k+1)}{\partial\tau_N^{(p)}}] \\
&= \eta\delta_{kl}[\frac{\partial(\sum_{i=0}^{N-1}[\hat{a}_i y^{(p)}(k-i)] + \sum_{i=0}^{M-1}[\hat{b}_i u^{(p)}(k-i)])}{\partial\tau_N^{(p)}} + \sum_{i=1}^{N_L}(\frac{\partial\hat{y}_{NL}^{(p)}(k+1)}{\partial net_1^{O(p)}}\frac{\partial net_1^{O(p)}}{\partial\tau_N^{(p)}})] \\
&= \eta\delta_{kl}[\sum_{i=0}^{N-1}\hat{a}_i\frac{y^{(p)}(t-\tau_N^{(p)}) - y^{(p)}(t-\tau_N^{(p)}-T)}{T} \\
&\quad + \sum_{k=1}^{N_L} y_l'^{(p)} w_{kl} y_k'^{(p)}[\sum_{j=1}^{N_{L-1}} w_{jk} y_j'^{(p)}(\sum_{i=1}^{N_{L-2}} w_{ij} y_i'^{(p)}(\sum_{q=0}^{N-1} w_{qi}\frac{y^{(p)}(t-\tau_N^{(p)}) - y^{(p)}(t-\tau_N^{(p)}-T)}{T}))]]
\end{aligned}$$

. (4-59)

For all training data pairs, the adaptation law of the number of delays is:

$$\tau_N(t+1) = \tau_N(t) + \sum_{m=1}^{P}\Delta\tau_N^{(m)}. \tag{4-60}$$

For the p^{th} training data set, the adaptation law is:

$$\begin{aligned}
\Delta\tau_M^{(p)} &= -\eta\frac{\partial E^{(p)}}{\partial\tau_M^{(p)}} = -\eta\frac{\partial E^{(p)}}{\partial\hat{y}^{(p)}(k+1)}\frac{\partial\hat{y}^{(p)}(k+1)}{\partial\tau_M^{(p)}} = \eta\delta_{kl}[\frac{\partial\hat{y}_L^{(p)}(k+1)}{\partial\tau_M^{(p)}} + \frac{\partial\hat{y}_{NL}^{(p)}(k+1)}{\partial\tau_M^{(p)}}] \\
&= \eta\delta_{kl}[\frac{\partial(\sum_{i=0}^{N-1}[\hat{a}_i y^{(p)}(k-i)] + \sum_{i=0}^{M-1}[\hat{b}_i u^{(p)}(k-i)])}{\partial\tau_M^{(p)}} + \sum_{i=1}^{N_L}(\frac{\partial\hat{y}_{NL}^{(p)}(k+1)}{\partial net_1^{O(p)}}\frac{\partial net_1^{O(p)}}{\partial\tau_M^{(p)}})] \\
&= \eta\delta_{kl}[\sum_{i=0}^{M-1}\hat{b}_i\frac{u^{(p)}(t-\tau_M^{(p)}) - u^{(p)}(t-\tau_M^{(p)}-T)}{T} \\
&\quad + \sum_{k=1}^{N_L} y_l'^{(p)} w_{kl} y_k'^{(p)}[\sum_{j=1}^{N_{L-1}} w_{jk} y_j'^{(p)}(\sum_{i=1}^{N_{L-2}} w_{ij} y_i'^{(p)}(\sum_{r=0}^{M-1} w_{ri}\frac{u^{(p)}(t-\tau_M^{(p)}) - u^{(p)}(t-\tau_M^{(p)}-T)}{T}))]]
\end{aligned}$$

(4-61)

For all training data pairs, the adaptation law for the number of delay τ_M is:

$$\tau_M(t+1) = \tau_M(t) + \sum_{m=1}^{P} \Delta\tau_M^{(m)} . \tag{4-62}$$

In this chapter, the modified BP algorithm is introduced in detail to tune all adjustable parameters in the three RANN-based nonlinear system identification models proposed in chapter 2. All the adaptation laws for the weights, linear parameters, and time delays are developed based on the steepest gradient descent algorithm. During the training process of the RANN, all these adjustable parameters are tuned, and their approximately optimal values in the sense of minimizing the identification error are obtained by the proposed modified BP algorithm. When the GA, discussed in chapter 3, and the modified BP algorithm work together, both the architecture of the RANN and all the parameters of the identification model are obtained to identify the nonlinear system successfully.

Chapter 5 Simulation results

In this chapter, seven examples and their results are demonstrated to evaluate the performance of the proposed GA and RANN-based nonlinear system identification. Two types of application systems are applied to each of the three RANN-based identification models. One is the numerical nonlinear systems, as used in (Yazdizadeh and Khorasani 2002) or (Narendra and Parthasarathy 1990), the other one is a real world nonlinear system. A special example of multi-inputs nonlinear system identification is demonstrated using the Single RANN model also.

Most research in literature identifies the input-output mapping of a system by using both input and output signals although there were few studies on using only the output signal (Rozario and Papoulis 1989) (relay on high order statistics of the output signals). In this research, all the simulations use both input and output signals. As described in (Yazdizadeh and Khorasani 2002), the reference input signals should be persistently exciting. Because the selection of the reference signals is out of the scope of this study, we simply select signals that are the same as or similar to those in (Yazdizadeh and Khorasani 2002) or (Narendra and Parthasarathy 1990).

Finally, for all the simulations, the configuration of the optimized RANN after GA is listed in Table 5-1 to 5-7, where the initial status of the RANN before GA is also listed for reference only.

5.1 Normalization of training and generalization data sets

The activation function used in neural networks could be any bounded, differentiable, and monotonically increasing nonlinear function (Chandra and Singh

2004). In general, the activation function for nonlinear identification is a smooth function, typical examples of which are the sigmoid function $\frac{1}{1+e^{-x}}$, as shown in Figure 5-1, and the hyperbolic tangent function $\frac{1-e^{-x}}{1+e^{-x}}$, as shown in Figure 5-2, with unipolar and bipolar outputs in the range of (0, 1) and (-1, 1), respectively.

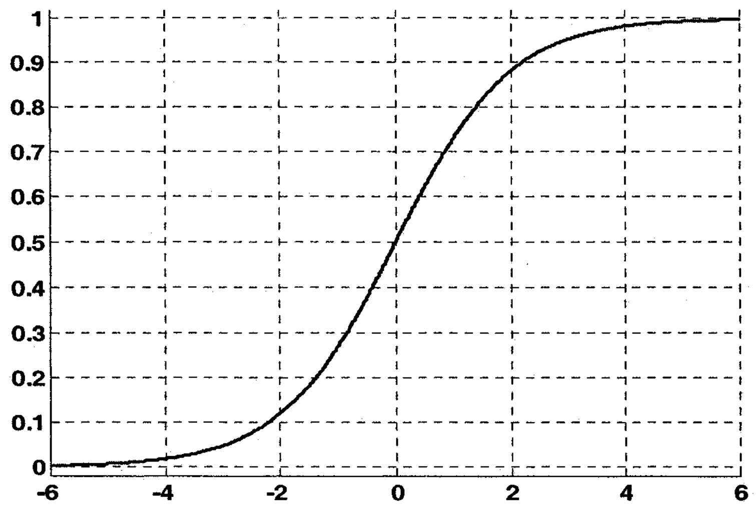


Figure 5-1 The simple sigmoid function

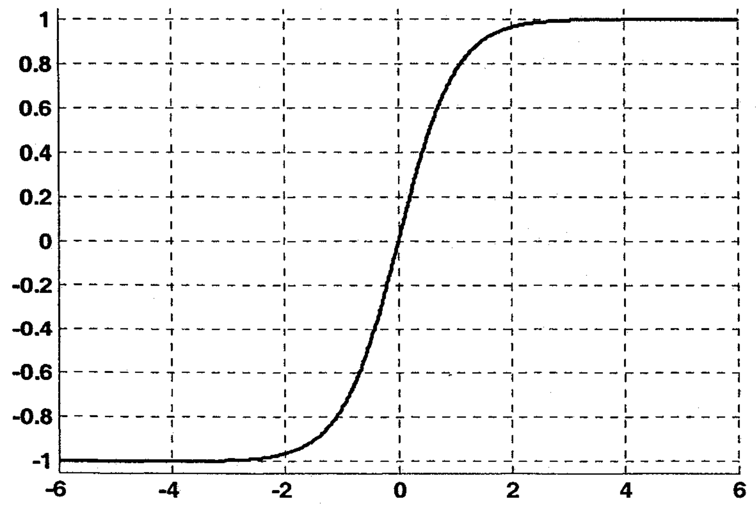


Figure 5-2 The hyperbolic tangent function

In order for the network to operate properly, both input and the output vectors must be normalized, by adding a normalization layer before the first hidden layer, or simply normalizing the input vector to [0,1] or [-1, 1] before it is fed to the neural network. In this study, the following general normalization formula proposed in (Canelon, Shieh *et al.* 2004) is used.

$$\theta = -1 + 2\times\frac{\Theta - \Theta_{\min}}{\Theta_{\max} - \Theta_{\min}} \tag{5-1}$$

where Θ and θ are the values before and after normalization, respectively. $\Theta_{\min}$ and $\Theta_{\max}$ are the lower and upper bounds of the original vector.

If the range of the input or the output is [0, 1] or [-1, 0], the following corresponding normalization formulas could be adopted.

$$\theta = \frac{\Theta - \Theta_{\min}}{\Theta_{\max} - \Theta_{\min}} \qquad \theta \in [0,1] \tag{5-2}$$

$$\theta = -1 + \frac{\Theta - \Theta_{\min}}{\Theta_{\max} - \Theta_{\min}} \qquad \theta \in [-1,0] \tag{5-3}$$

The hyperbolic tangent sigmoid function is used as the activation function in the modified BP algorithm, and the output of the plant is used as the feedback of the RANN both in the training and the generalization processes.

5.2 Applications of Single RANN Model

Example 1

The system's governing equation is:

$$y(t) = \frac{0.2y(t-1) + 0.6y(t-2)}{1 + y(t-1)^2} + \sin(u(t-1))\,. \tag{5-4}$$

The same reference input signal is adopted as in (Yazdizadeh and Khorasani 2002), i.e.:

$$u(t) = \sin(\frac{2\pi t}{10}) + \sin(\frac{2\pi t}{25}) \tag{5-5}$$

To test the robustness of the system, the input signal is modified as in (Yazdizadeh and Khorasani 2002): 50% amplitude reduction within 100-200 time steps, 50% frequency reduction within the 200-300 time steps, and 100% frequency increase within the 300-400 time steps.

The modified signal is illustrated in Figure 5-3 and is used in this example.

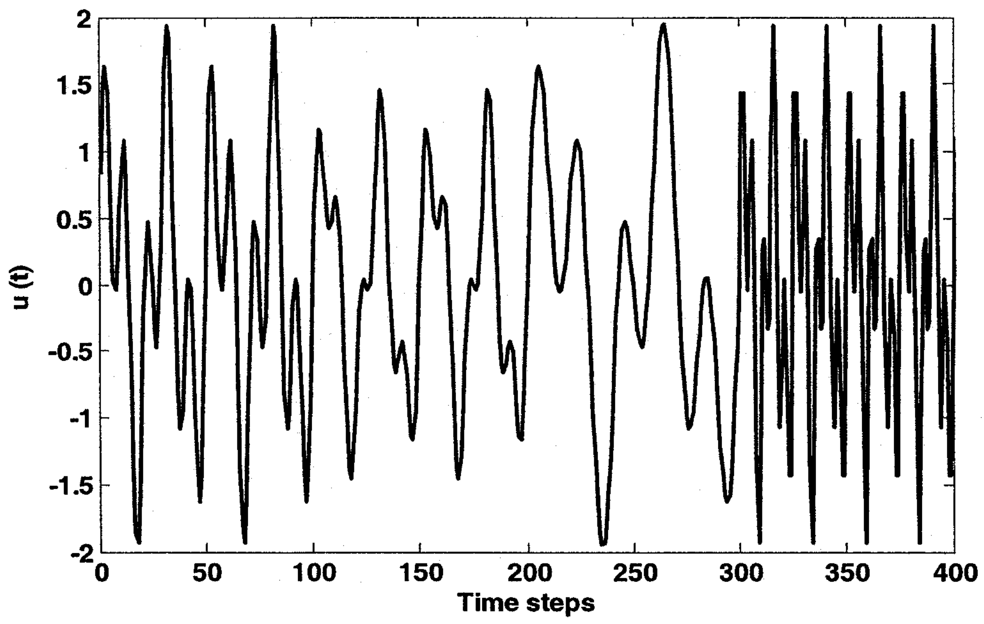


Figure 5-3 The modified input signal in Example 1

After tuning, the parameters of the GA and the modified BP are chosen as:

- Maximum hidden neuron number: 30
- Population size: 20
- Maximum GA generations: 800

- Crossover rate: 70%
- Mutation rate: 80%
- Initial connection rate in connection matrices: 80%
- Maximum BP runs: 200
- BP learning rate η: 0.016

The nonlinear identification model I was applied to the system, with the identification results shown in Figure 5-4.

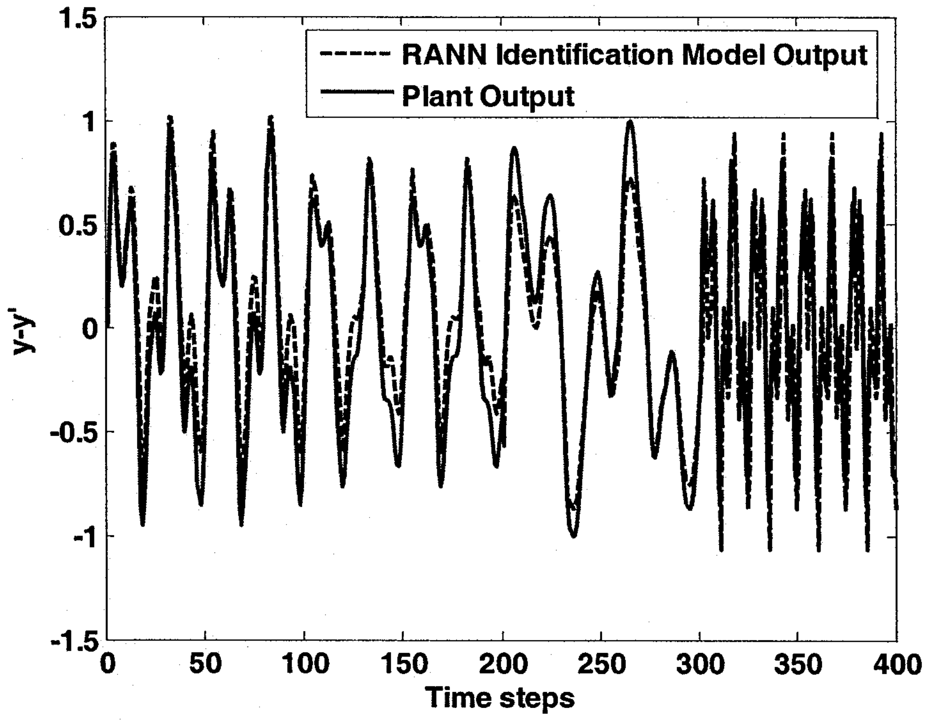


Figure 5-4 Identification results of Example 1

Figure 5-5 shows the minimal average square error of the best individual in the population in each generation.

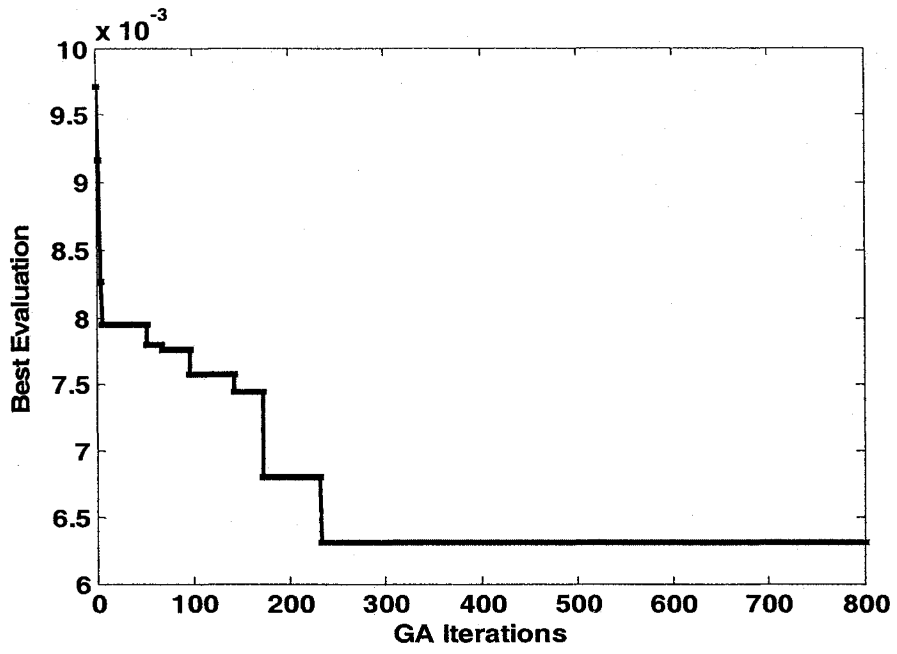


Figure 5-5 The performance of the architecture optimization GA in Example 1

The final average mean square error (Eq. 3-1) is 0.006309. Figure 5-4 shows that the output of the identification system simulates the target output well even when both the amplitude and the frequency of the input signal vary.

Table 5-1 RANN parameters comparison in Example 1

	Before GA	After GA
Hidden neurons	0-30	The 1st layer: 27, the 2nd layer: 1
Hidden layers	1 or 2	2
Connections	Random	Fixed

The connection matrix for the 1st hidden layer:

$$C_{10}^{*} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

The connection matrix for the 2nd hidden layer:

$$C_{21}^{*} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

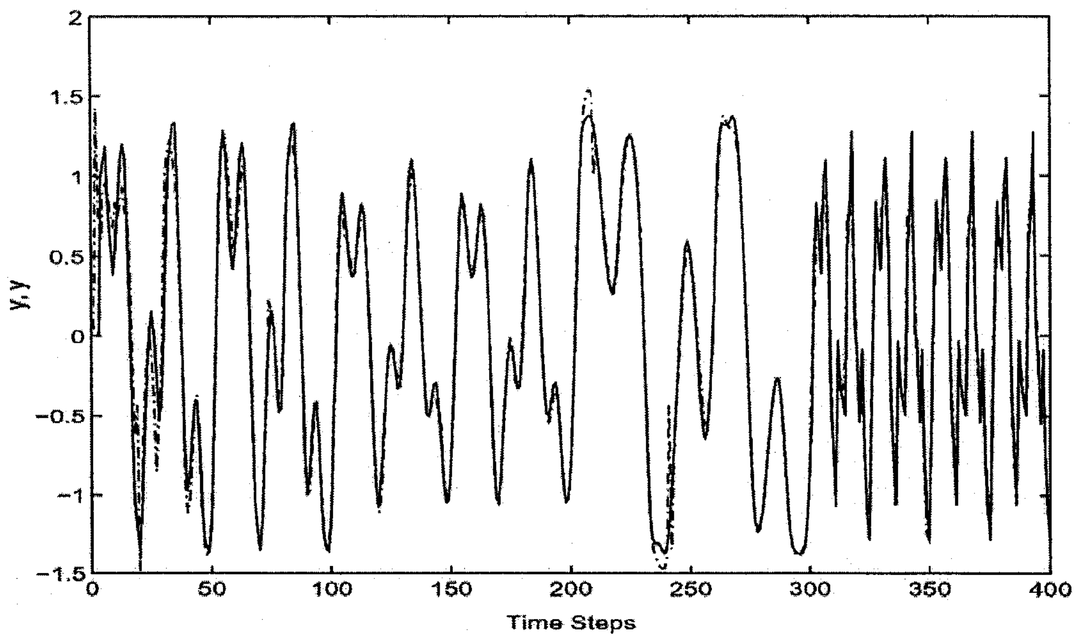


Figure 5-6 Identification results of Example 1 in (Yazdizadeh and Khorasani 2002)

Figure 5-6 shows the identification results of the same example in (Yazdizadeh and Khorasani 2002). It can be seen that in our example, although very small GA and BP parameters (population size: 20, generations: 800, BP iterations: 200) are used, and the obtained NN architecture is also very simple also (with only one neuron in the 2nd hidden layer), but the similar identification results could be researched as that in the reference paper.

Example 2

The governing equation of the system is:

$$y(t+1)=\frac{y(t)y(t-1)y(t-2)u(t-1)(y(t-2)-1)+u(t)}{1+y^2(t-2)+y^2(t-1)} \qquad (5\text{-}6)$$

The input signal, the same as in (Narendra and Parthasarathy 1990), is:

$$u(t)=\begin{cases}\sin\dfrac{2\pi t}{250} & t\le 500\\ 0.8\sin\dfrac{2\pi t}{250}+0.2\sin\dfrac{2\pi t}{25} & t>500\end{cases} \tag{5-7}$$

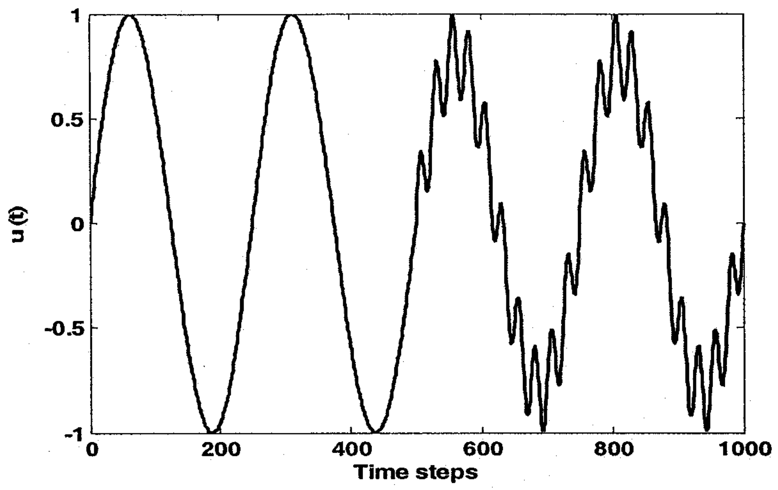


Figure 5-7 The input signal in Example 2

Here the total data sampling time step is 1000 and the time interval is 0.1. A subset of 200 input-output data pairs are used as the generalization data set.

After tuning, the parameters of GA architecture and modified BP training are chosen as:

- Maximum hidden neuron number: 30
- Population size: 20
- Maximum GA generations: 800
- Crossover rate: 70%
- Mutation rate: 80%
- Initial connection rate in connection matrices: 80%
- Maximum BP runs: 200

- BP learning rate η: 0.016

The nonlinear identification model I was applied to the system, with the identification results shown in Figure 5-8.

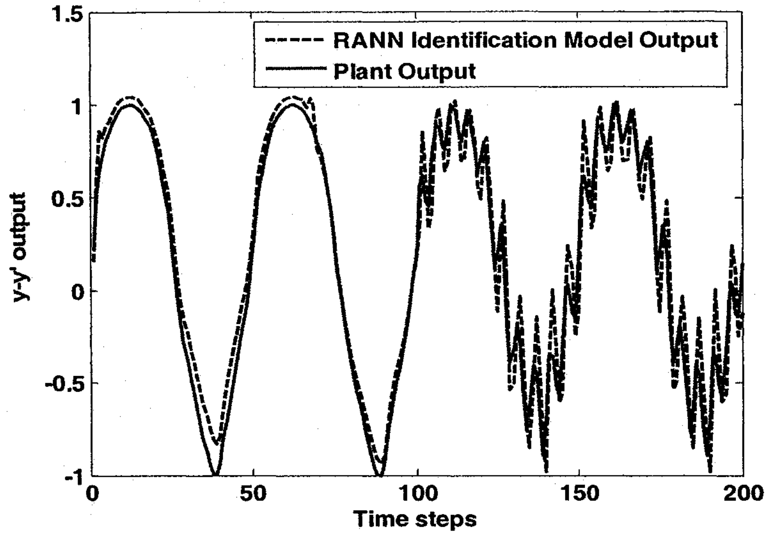


Figure 5-8 Identification results of Example 2

Figure 5-9 shows the minimal average square error of the best individual in the population in each generation.

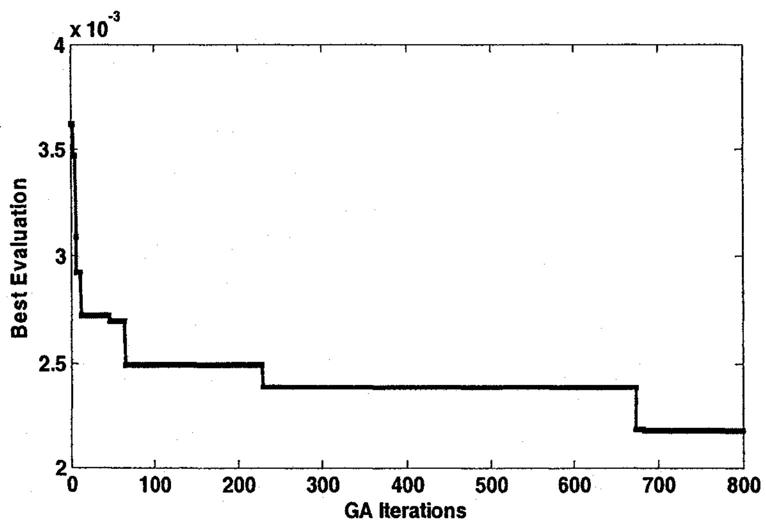


Figure 5-9 The performance of the architecture optimization GA in Example 2

The final average mean square error (Eq. 3-1) is 0.002178. It can be seen that our system achieves the same performance as that in (Narendra and Parthasarathy 1990), with a much smaller training data set.

Table 5-2 RANN parameters comparison in example 2

	Before GA	After GA
Hidden neurons	0-30	The 1st layer: 23, the 2nd layer: 2
Hidden layers	1 or 2	2
Connections	Random	Fixed

The connection matrix for the 1st hidden layer:

$$C_{10}^{*}=\begin{bmatrix}1&1&1&1&0\\1&0&1&1&1\\1&1&1&1&1\\0&0&1&1&1\\1&0&0&0&0\\0&0&1&0&0\\1&1&1&1&1\\0&0&0&0&1\\1&1&1&1&1\\0&1&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\1&1&1&1&1\\1&1&0&1&1\\0&1&0&0&0\\1&1&1&1&1\\1&0&1&1&1\\0&0&1&1&1\\1&1&1&1&1\\1&0&0&0&0\\1&1&0&0&0\\0&0&0&0&1\\0&0&1&0&0\end{bmatrix}$$

The connection matrix for the 2nd hidden layer:

$$C_{21}^{*} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

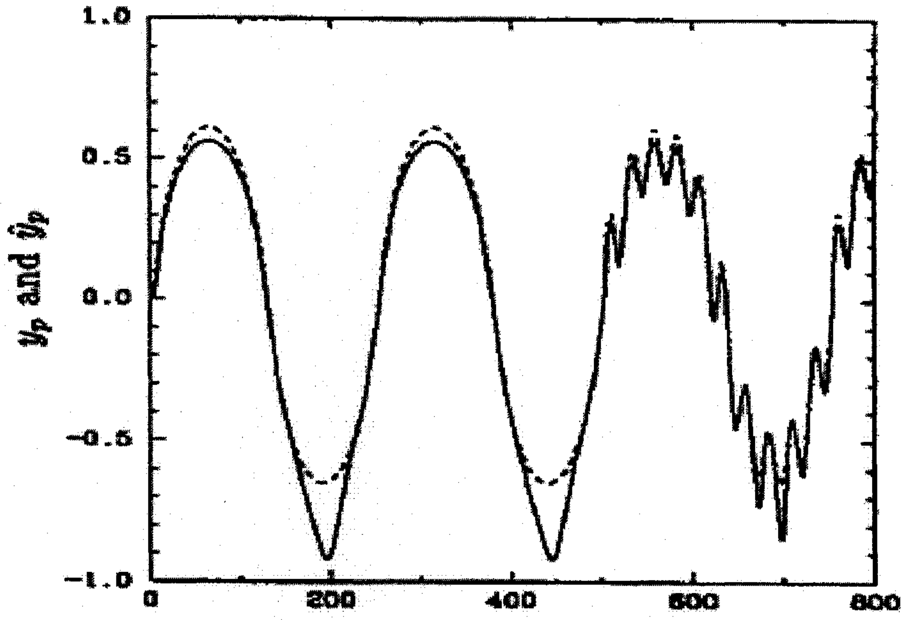


Figure 5-10 Identification results of Example 2 in (Narendra and Parthasarathy 1990)

As shown in Figure 5-10, for the same example in (Narendra and Parthasarathy 1990), although a large number of identification procedure (100 000) was carried out, a similar identification results was got as in our example with only 1000 identification procedure is carried out here.

Example 3

As a special study case, a VDCL (Voltage Dependent Current-reference Limiter) unit black-box nonlinear system (Narendra, Sood et al. 1995) is used to verify the proposed algorithms.

An HVDC (High Voltage Direct Current) converter fed into a weak AC system is prone to commutation failures. Under such conditions, an adaptive current reference can be useful to optimize the system recovery following a fault and alleviate the possibility of subsequent commutation failures. In an HVDC transmission system, a VDCL unit is traditionally used to generate an adaptive current reference for the converter controller. This current reference can be either adapted to the DC (transmission line) voltage or the AC voltage at the filter bus of the converter.

The traditional VDCL unit in the HVDC control system is of multiple-input single-output (*MISO*) type with a non-linear voltage-current characteristic. An intelligent VDCL unit was proposed in (Narendra, Sood *et al.* 1995) with an NN of eighty four neurons on the hidden layers. It demands a huge amount of computation, and may be not suitable for power systems, which need quick responses. According to Qi's research (Qi 2005), 40 neurons on the hidden layers could also get the same result, but it may not be the optimal solution. There may exist a better architecture of the neural network that can be found systematically. As a special example, the proposed Single RANN identification model I is used to solve this problem. The objective is to find a more effective neural network system with neurons less than 40.

After tuning, the parameters of GA architecture and modified BP training are chosen as:

- Maximum hidden neuron number: 30
- Population size: 30
- Maximum GA generations: 50
- Crossover rate: 40%

- Mutation rate: 10%
- Initial connection rate in connection matrices: 50%
- Maximum BP runs: 200
- BP learning rate η: 0.016

The GA and BP parameters are not the same in all simulation because, generally, we need to tune these parameters to get better or acceptable results; for instance, the mutation rate in this example is much smaller than in other simulations because if we use a small mutation rate in other simulations, it is much difficult to obtain better chromosomes. It shows that the mutation operator in other simulations pays a more important role than in this example. Figure 5-11 shows the average mean square error of the best individual in the population in each generation.

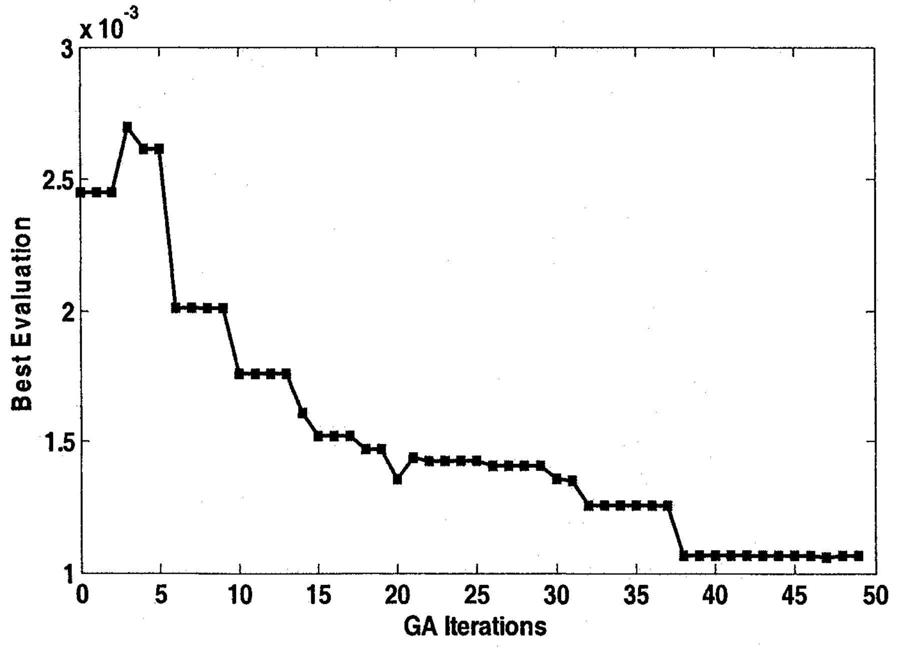


Figure 5-11 The performance of the architecture optimization GA in Example 3

Figure 5-12 shows the mean square error with respect to BP's time steps.

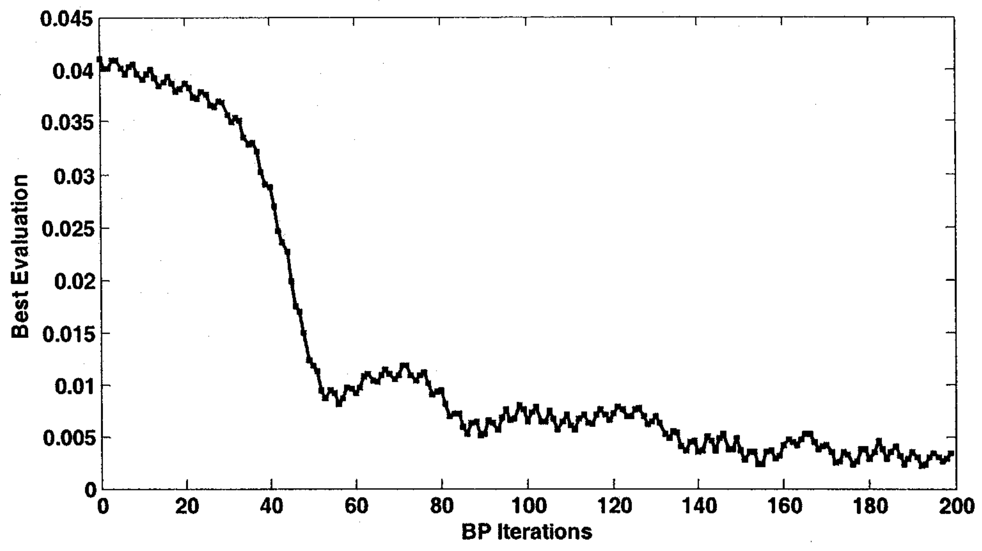


Figure 5-12 The convergence of BP algorithm in Example 3

It can be seen that the proposed algorithm is better than Qi's work in that it finds a RANN of 22 neurons in total, with 13 neurons and 9 neurons on the first and the second hidden layer, respectively. This is because the simple the neural network, the quicker the response of the VDCL unit (represented by the NN)

Table 5-3 RANN parameters comparison in example 3

	Before GA	After GA
Hidden neurons	0-30	The 1st layer: 13, the 2nd layer: 9
Hidden layers	1 or 2	2
Connections	Random	Fixed

The connection matrix for the 1st hidden layer:

$$C_{10}^{*} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

The connection matrix for the 2nd hidden layer:

$$C_{21}^{*} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

The three examples above indicate that the proposed unknown nonlinear identification model has a robust capability to identify completely unknown nonlinear systems.

5.3 Applications of Serial-Linear RANN Model

Example 4

The governing equation of the system is:

$$y(t) = 0.3y(t-1) + 0.6y(t-2) + \frac{0.6}{1+u(t-1)^2} \tag{5-8}$$

The same reference signal, as in (Yazdizadeh and Khorasani 2002), is:

$$u(t-1) = \sin\frac{2\pi(t-1)}{100} \tag{5-9}$$

The nonlinear identification model II was applied to the system, with the identification results shown in Figure 5-13.

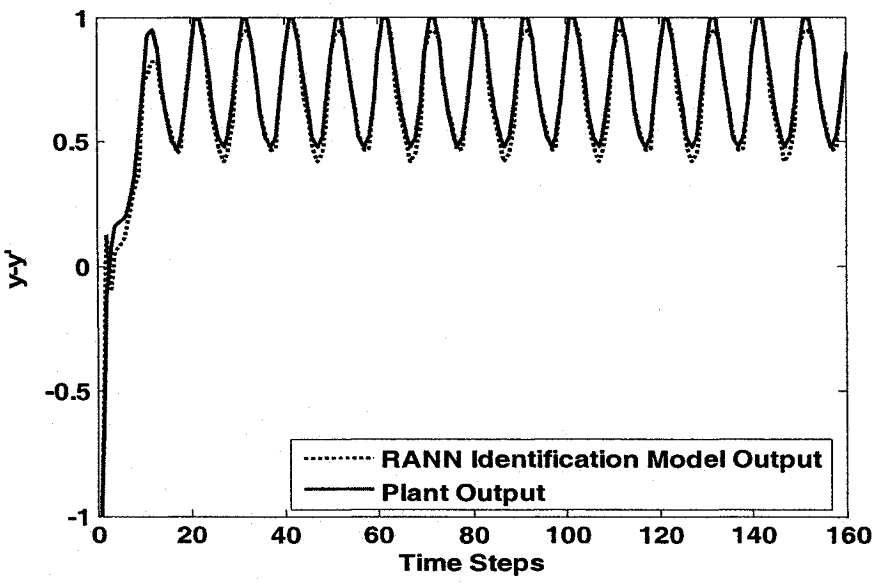


Figure 5-13 Identification results of Example 4

Figure 5-14 shows the minimal average square error of the best individual in the population in each generation.

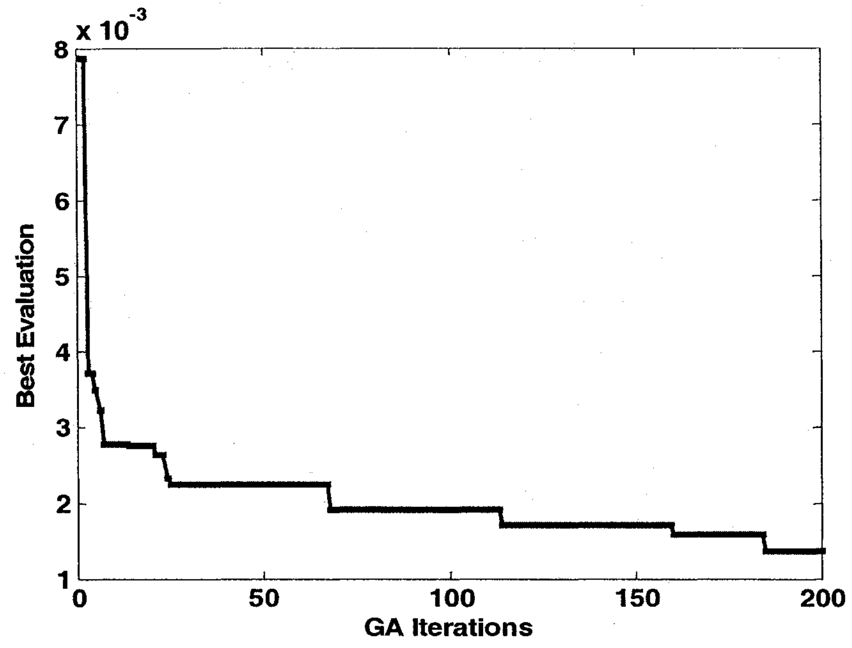


Figure 5-14 The performance of the architecture optimization GA in Example 4

After tuning, the parameters of GA architecture and modified BP training are chosen as:

- Maximum hidden neuron number: 30
- Population size: 20
- Maximum GA generations: 200
- Crossover rate: 70%
- Mutation rate: 80%
- Initial connection rate in connection matrices: 80%
- Maximum BP runs: 200
- BP learning rate η: 0.016

The final average mean square error (Eq. 3-1) in this example is 0.001359. It can be seen that although GA was only run for 200 generations, the performance of this system is very good, with a medium sized RANN, as shown in Table 5-4.

Table 5-4 RANN parameters comparison in example 4

	Before GA	After GA
Hidden neurons	0-30	The 1st layer: 13, the 2nd layer: 3
Hidden layers	1 or 2	2
Connections	Random	Fixed

The connection matrix for the 1st hidden layer:

$$C_{10}^{*} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The connection matrix for the 2nd hidden layer:

$$C_{21}^{*} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

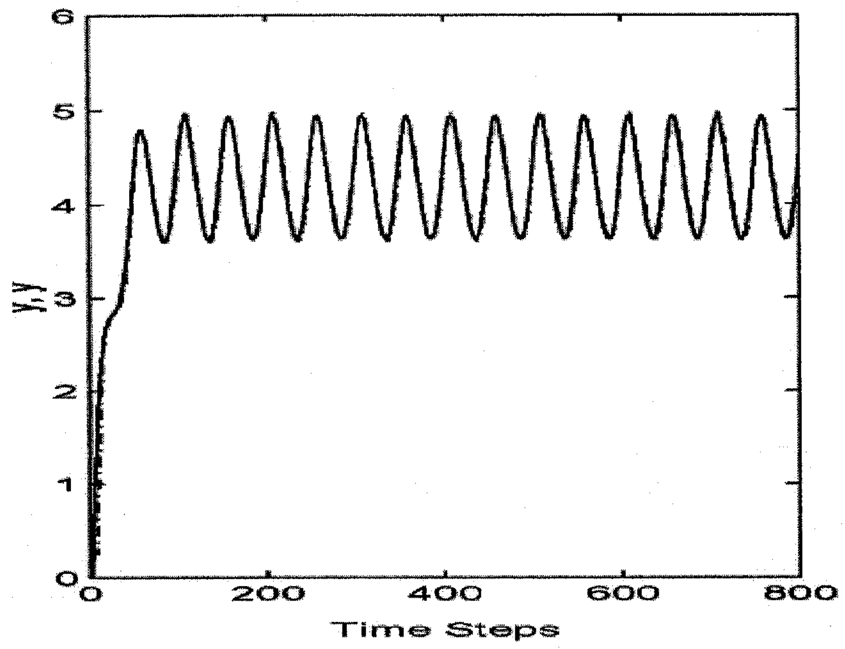


Figure 5-15 The simulation results of Example 4 in (Yazdizadeh and Khorasani 2002)

Figure 5-15 shows the identification results of the same example in (Yazdizadeh and Khorasani 2002). To obtain a similar result as in our example, a much larger training data set was used in the reference paper. In addition, the identification system order (4) was higher than the real system (2) in (Yazdizadeh and Khorasani 2002), but in our example, it is lower (1).

Example 5

In this example, the backlash-like hysteresis (Su, Stepanenko *et al.* 2000) is taken into account as the unknown nonlinear characteristic in “smart” actuator. As pointed by (Su, Stepanenko *et al.* 2000), hysteresis is a nonlinear characteristic exists in many control systems, such as electric servo actuators, harmonic drives, and electronic relay circuits. In this example, the same backlash-like hysteresis model is adopted. The description of the hysteresis model is:

$$h(t) = P(u(t))$$
$$= \begin{cases} c(u(t)-B), & \text{if } \dot{u}(t) > 0 \text{ and } h(t) = c(u(t)-B) \\ c(u(t)+B), & \text{if } \dot{u}(t) < 0 \text{ and } h(t) = c(u(t)+B) \\ h(t\_), & \textit{otherwise} \end{cases} \quad (5\text{-}10)$$

where *P()* is the function of the backlash-like hysteresis, *c>0* is the slope of the lines, and *B>0* is the backlash distance. The same values are used as in (Su, Stepanenko *et al.* 2000), i.e. *c=3.1635, B=0.345.*

Figure 5-16 shows the hysteresis while the input signal is chosen as $u(t) = \sin(2.3t)$.

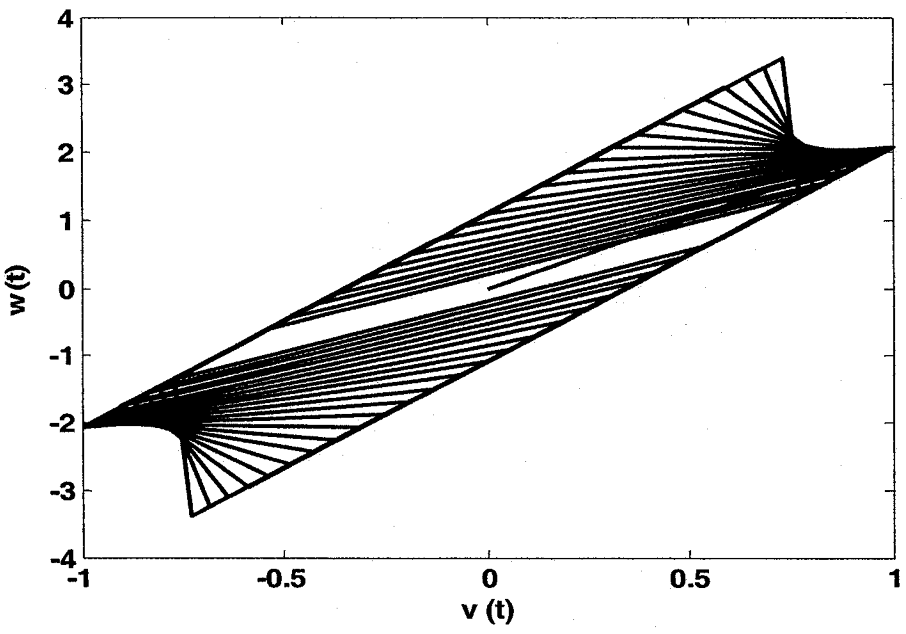


Figure 5-16 The hysteresis under the input $u(t) = \sin(2.3t)$

The "smart" actuator is piezoelectric actuator whose dynamics is identified as a second-order linear model coupled with a hysteresis (Jan and Hwang 2000).

$$m\ddot{y}(t) + b\dot{y}(t) + ky(t) = kh(t) \quad (5\text{-}11)$$

where $y(t)$ denotes the position of piezoelectric actuator, m, b and k denote the mass, damping, and stiffness.

The "smart" actuator preceded by backlash-like hysteresis is shown in the Figure 5-17.

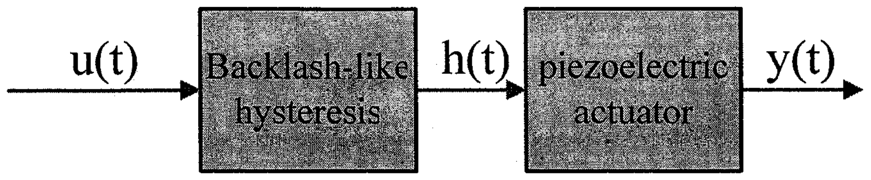


Figure 5-17 The representation of the "smart" actuator

Since in the representation model, the backlash-like hysteresis nonlinear element is serially preceding to the known linear model of the "smart" actuator, the proposed RANN identification model II is applied to this complex nonlinear system.

The following reference input signal is used:

$$u(t) = \sin(\frac{\pi t}{25}) \tag{5-12}$$

A similar modification in the example 3 of (Yazdizadeh and Khorasani 2002) is applied to the input signal: 50% amplitude reduction within 25-50 time steps, 100% amplitude increase within the 50-75 time steps, and 50% frequency reduction within the 75-100 time steps, and 100% frequency increase within the 100-150 time steps.

The modified input signal is illustrated below.

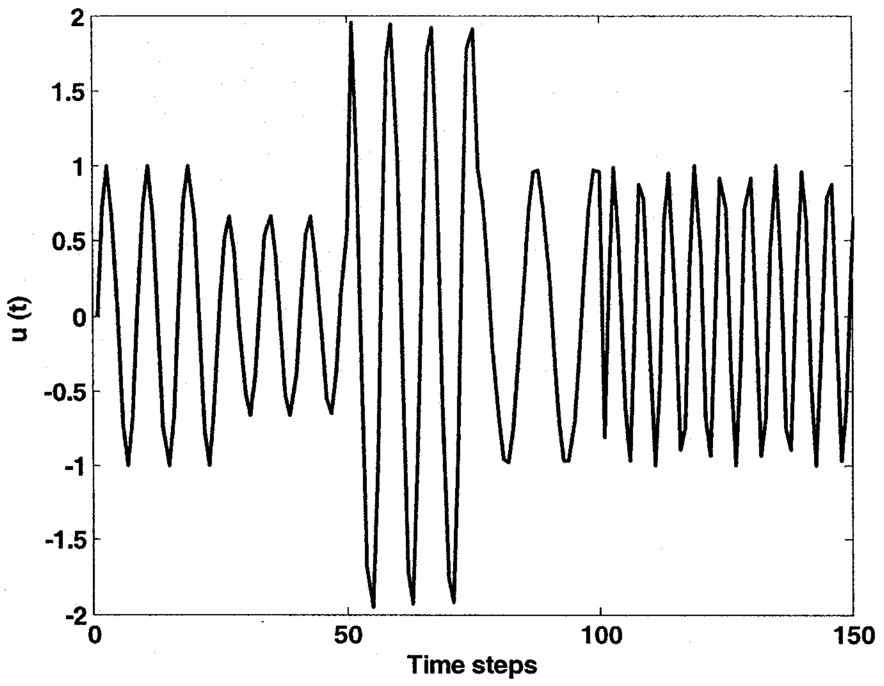


Figure 5-18 The modified input signal in Example 5

Identification results are shown in Figure 5-19.

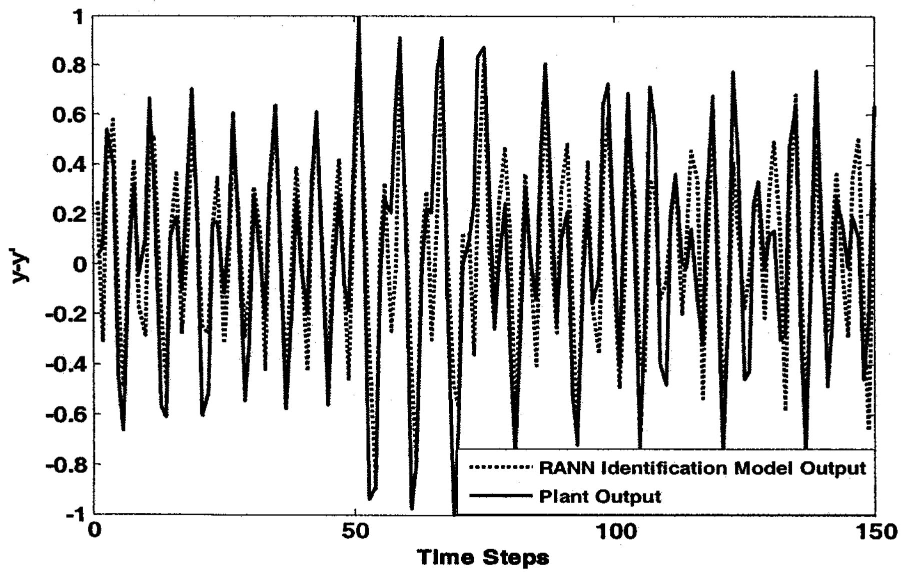


Figure 5-19 Identification results of Example 5

Figure 5-20 shows the minimal average square error of the best individual in the population in each generation.

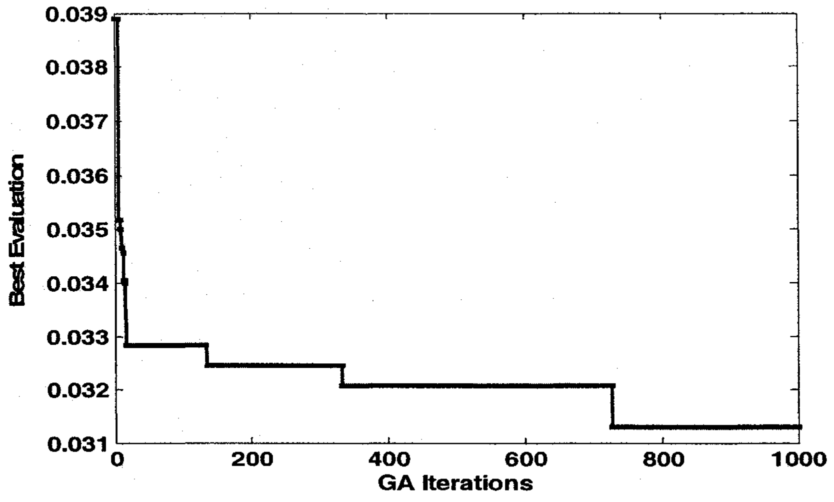


Figure 5-20 The performance of the architecture optimization GA in Example 5

After tuning, the parameters of GA architecture and modified BP training are chosen as:

- Maximum hidden neuron number: 30
- Population size: 20
- Maximum GA generations: 1000
- Crossover rate: 70%
- Mutation rate: 80%
- Initial connection rate in connection matrices: 80%
- Maximum BP runs: 200
- BP learning rate η: 0.016

The final average mean square error (Eq. 3-1) is 0.031290.

Table 5-5 RANN parameters comparison in example 5

	Before GA	After GA
Hidden neurons	0-30	The 1st layer: 13, the 2nd layer: 11
Hidden layers	1 or 2	2
Connections	Random	Fixed

The connection matrix for the 1st hidden layer:

$$C_{10}^{*} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The connection matrix for the 2nd hidden layer:

$$C_{21}^{*} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

5.4 Applications of Parallel-Linear RANN Model

Example 6

This example was taken from the example 3 in (Yazdizadeh and Khorasani 2002), but the system's governing equation is modified below:

$$
\begin{aligned}
y(t) = & \frac{y(t-1)y(t-2)(y(t-1)+2.5)}{1+y(t-1)^2+y(t-2)^2+0.03u(t-1)} \\
& +0.8u(t-1)+0.02u(t-2)+0.05y(t-1)+0.01y(t-2)
\end{aligned}
\quad . (5\text{-}13)
$$

The same reference input signal is used as in (Yazdizadeh and Khorasani 2002), i.e.:

$$
u(t) = \sin(\frac{2\pi t}{25}) \quad (5\text{-}14)
$$

The same modification as in (Yazdizadeh and Khorasani 2002) is applied to the input signal: 50% amplitude reduction within 100-200 time steps, 100% amplitude increase within the 200-300 time steps, and 50% frequency reduction within the 300-400 time steps, and 100% frequency increase within the 400-500 time steps.

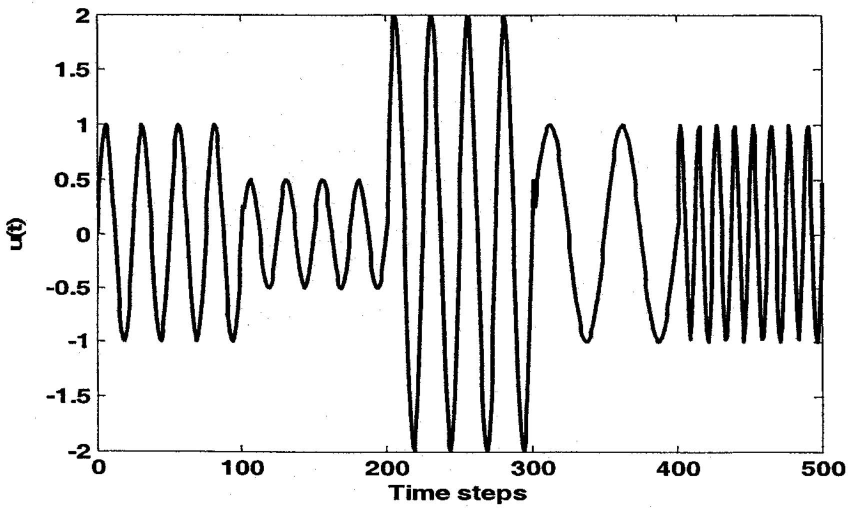


Figure 5-21 The modified input signal for Example 6

The proposed identification Model III was applied to the system. The identification results are shown in Figure 5-22.

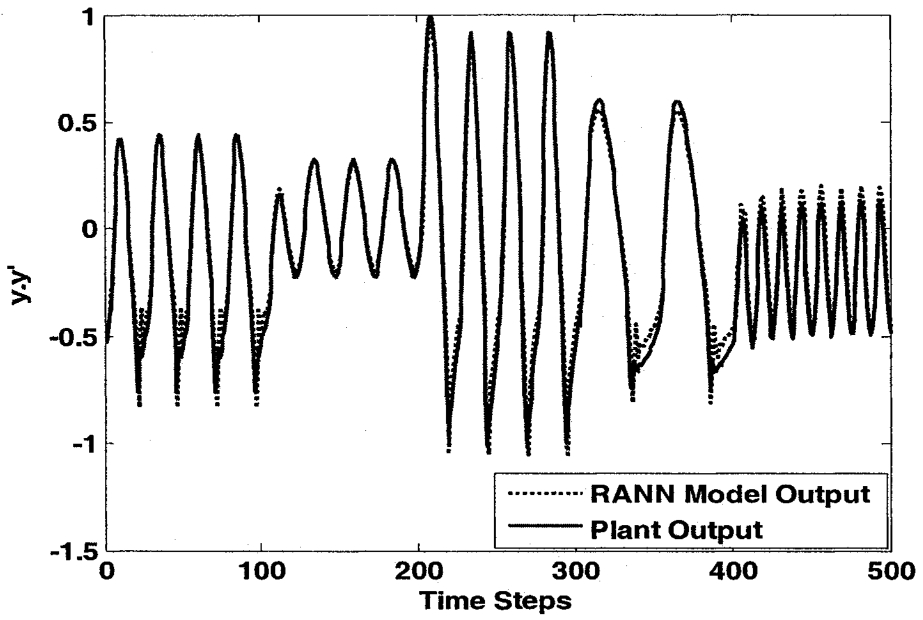


Figure 5-22 Identification results of Example 6

Figure 5-23 shows the minimal average square error of the best individual in the population in each generation.

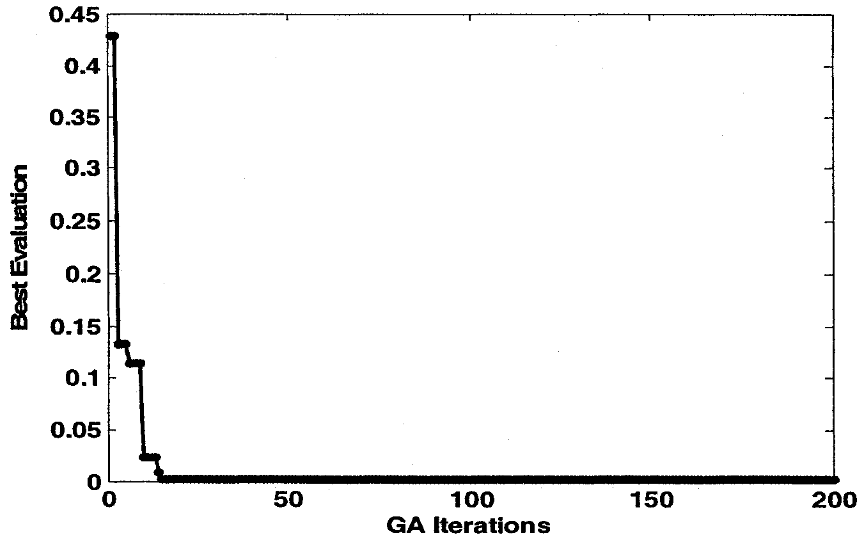


Figure 5-23 The performance of the architecture optimization GA in Example 6

After tuning, the parameters of GA architecture and modified BP training are chosen as:

- Maximum hidden neuron number: 30
- Population size: 20
- Maximum GA generations: 200
- Crossover rate: 70%
- Mutation rate: 80%
- Initial connection rate in connection matrices: 80%
- Maximum BP runs: 200
- BP learning rate η: 0.016

The final average mean square error (Eq. 3-1) is 0.001799. The proposed identification model III achieves equivalently good simulation effects with less complex neural network architecture than that in (Yazdizadeh and Khorasani 2002).

Table 5-6 RANN parameters comparison in example 6

	Before GA	After GA
Hidden neurons	0-30	The 1st layer: 11, the 2nd layer: 9
Hidden layers	1 or 2	2
Connections	Random	Fixed

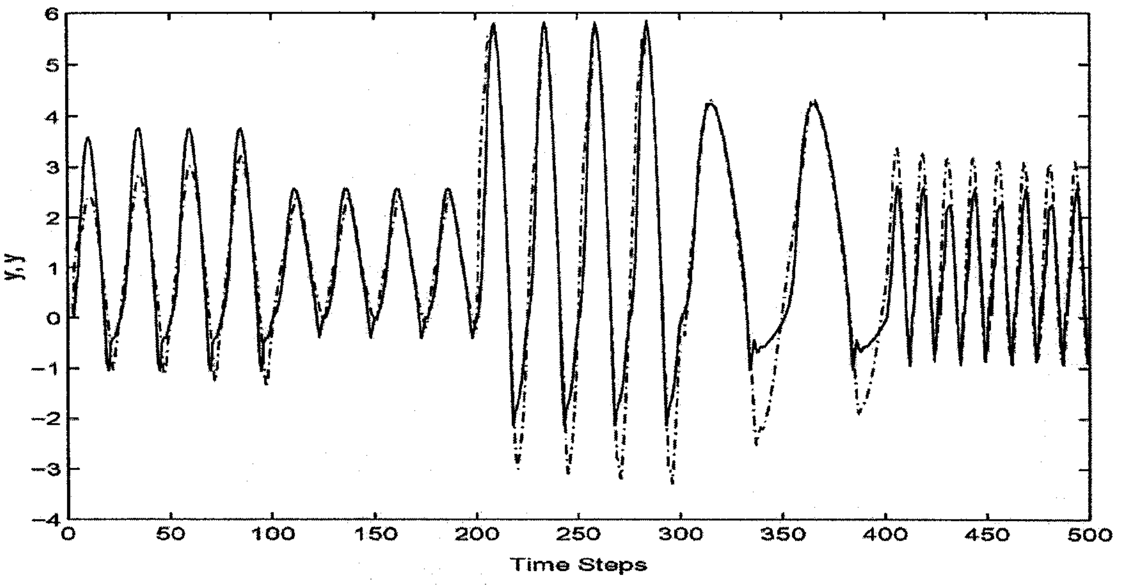


Figure 5-24 Identification results of Example 6 in (Yazdizadeh and Khorasani 2002)

Figure 5-24 shows the identification results of the similar example in (Yazdizadeh and Khorasani 2002). It can be seen that although our modified system is complex than the original one in the reference, with small population size and GA as well as BP iteration number, not worse results got in our example.

Example 7

In this example, friction is taken into account as the unknown nonlinear characteristic in a Harmonic Drive system, whose representation model is similar to that in (Xie, Krzeminski *et al.* 2002). A harmonic drive actuator is an integral package consisting of an encoder, servo motor and precision harmonic drive gearhead and is reported to be plagued by friction The same 7 parameters friction model as (Xie, Krzeminski *et al.* 2002) is used here, as described below.

$$F = F_c + F_v \bullet |v| + F_s(\gamma, t_2) \cdot \frac{1}{1 + \left(\frac{v \cdot (t - \tau_L)}{v_s} \right)^2} \qquad (5\text{-}15)$$

$$F_S(\gamma, t_2) = F_{s,a} + (F_{s,\infty} - F_{s,a}) \cdot \frac{t_2}{t_2 + \gamma} \qquad (5\text{-}16)$$

where,

F is the instantaneous friction force,

F_C is the Coulomb friction force,

F_v is the viscous friction force,

F_s is the magnitude of the Stribeck friction,

$F_{s,a}$ is the magnitude of the Stribeck friction at the end of the previous sliding period,

$F_{s,\infty}$ is the magnitude of the Stribeck friction after a long time at rest,

v is the velocity,

v_s is the characteristic velocity of the Stribeck friction,

τ_L is the time constant of frictional memory,

γ is the temporal parameter of the rising static friction,

t_2 is the dwell time, time at zero velocity.

The Harmonic drive considered with friction can be represented as shown in Figure 5-25.

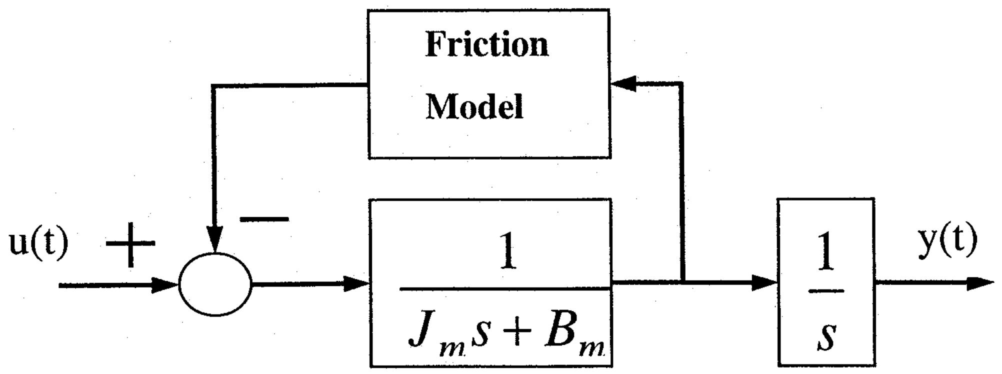


Figure 5-25 The representation of the harmonic drive

where J_m and B_m are J and B divided by K_m, respectively; J represents the harmonic drive moment of inertia, B represents the viscous damping constant, and K_m represents the torque constant of the harmonic drive.

The system, with a known linear model, is simulated in Matlab/Simulink. The identification model III is utilized for its identification. The same reference input signals as in example 5 are used.

Identification results are shown in Figure 5-26.

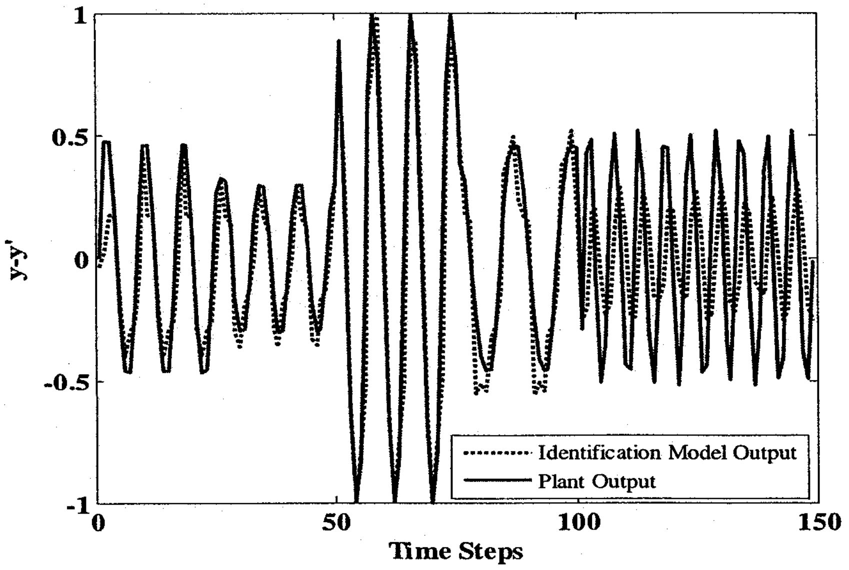


Figure 5-26 Identification results of Example 7

Figure 5-27 shows the minimal average square error of the best individual in the population in each generation.

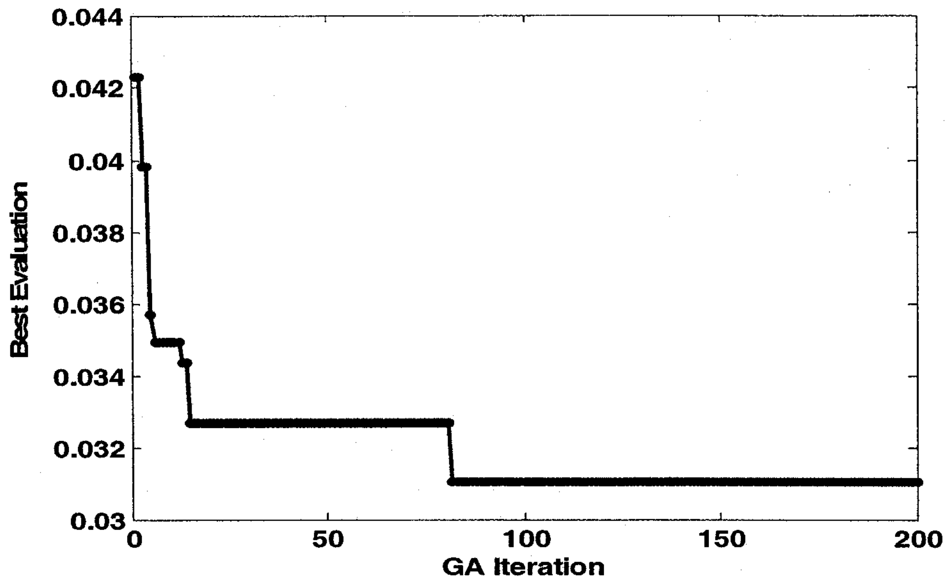


Figure 5-27 The performance of the architecture optimization GA in Example 7

The parameters of GA architecture and modified BP training are chosen as:

- Maximum hidden neuron number: 30
- Population size: 20
- Maximum GA generations: 200
- Crossover rate: 70%
- Mutation rate: 80%
- Initial connection rate in connection matrices: 80%
- Maximum BP runs: 200
- BP learning rate η: 0.016

The final average mean square error (Eq. 3-1) is 0.030041.

Table 5-7 RANN parameters comparison in example 7

	Before GA	After GA
Hidden neurons	0-30	The 1st layer: 16, the 2nd layer: 5
Hidden layers	1 or 2	2
Connections	Random	Fixed

The connection matrix for the 1st hidden layer:

$$C_{10}^{*} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

The connection matrix for the 2nd hidden layer:

$$C_{21}^{*} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

5.5 Conclusion

Seven simulations have shown that the three RANN-based identification models have strong approximate abilities and very good performance on nonlinear systems' identification. After running the GA for not larger than 1000 number of runs, all these simulations have achieved a RANN with a small amount of hidden neurons and no more than two hidden layers.

Besides that, the average square error of the best RANN in each generation progressively decreases, as shown in the figures of the architecture optimization GA in each example.

It can be concluded that the GA algorithm plays a very key role in finding good RANN architecture automatically, which is a significant advantage over that of trial-and-error process with human intervention. Although there are some other NN-based system identification methods, many of them use fixed network architecture or use one hidden layer and try to adjust the number of hidden neurons to get a good identification results. The proposed algorithm can evolve both the number of hidden layer and the number of hidden neurons on each hidden layer in architecture. It has been shown that it is a systematic way to obtain the architecture of networks.

Also, it can be seen that all the optimized RANNs have two hidden layers although they are initialized as either one or two hidden layers at random. We believe that the RANN with two hidden layers performs the best in terms of the mean square error.

Comparing to those reference examples as in (Yazdizadeh and Khorasani 2002) and (Narendra and Parthasarathy 1990), smaller GA and BP parameters, same or smaller training data set, more simple NN architecture (2 hidden layers), lower order of the identification models, same or more complex systems in our simulations, but very similar or better identification results are obtained here.

For the computational time, a light computation test of Example 6 (GA iterations: 200) and a heavy computation test of Example 7(GA iterations: 1000) are used to demonstrate the training time of the proposed algorithm. The results are shown in the following table.

Table 5-8 Tests of the computation time

	GA populations	GA iterations	BP iterations	Training data set	Computation time
Example 6	20	200	200	500	61 minutes
Example 7	20	1000	200	149	253 minutes
computer configurations	CPU: Intel Pentium III 999MHz; Memory: 512MB OS: Windows XP Professional, Version 2002, Service Pack 2				

Form Table 5-8, it can be seen that the computational time for the proposed algorithm is not long. Since the time recording is not a very scientific way, a more convincing time measurement could be a possible future work.

Chapter 6 Conclusions and Future work

In this study, genetic algorithms and recurrent artificial neural networks techniques are extensively studied for the identification of nonlinear dynamic systems. The main contributions of this research are summarized as:

1) Three nonlinear system identification models are adopted to represent three classes of nonlinear systems.
2) Recurrent neural networks are utilized in the proposed identification models for the identification of the nonlinear system.
3) A genetic algorithm is developed to optimize the architecture of RANNs in the sense of minimizing the identification error systematically, hence effectively avoiding the traditional trial-and-error method.
4) A novel DMME method is proposed to encode neural networks.
5) A modified BP has been proposed to tune both weights and all other adjustable parameters, such as linear part's parameters and number of time delays.
6) Extensive simulations have been carried out to evaluate the performance of the proposed GA and RANN-based nonlinear system identification.

Several examples including numerical and real world nonlinear systems are extensively studied using the proposed techniques. The identification results of all the examples show that the proposed algorithms are effective and robust in the identification of nonlinear systems. The simulation results also reveal that the GA used to optimize the NN architecture in the sense of minimizing the identification error pays a very important role. After GA, the identification error reduced largely, and a more simple NN architecture can be obtained. Hence, accompanying with the proposed Direct Matrix Mapping Encoding method, the computation load of RANN-based identification model also reduces. NNs with two hidden layers are found to perform better architecture than those with only one hidden layer. The reason could be that the simulations systems in this study are all very complex nonlinear dynamic systems. If a system is simple with low order, maybe only one hidden layer NNs can identify the system well enough. However, if the neurons number in the hidden layer increased largely, we may need to consider the using of two hidden layers with totally smaller hidden neurons instead.

Possible future work is as follows:

1) The Back-propagation algorithm is used in this study to train weights and other adjustable parameters. The major problem of the BP is its slow convergence and tendency to get trapped in local optima. Specifically in this study, two factors, i.e. the local optima and the initial weights, affect the performances of the systems to a large extent. For a given system with few local optima and appropriate initial weights, the BP algorithm usually works well. Otherwise its performances may degrade dramatically. In (Liu, Liu et al. 2004), a separated GA is utilized to evolve the initial weights, which are then fed into BP for further training. Alternatively, the convergence rate can be enhanced by adopting a dynamic parameter λ into the activation functions as: $f(x)=\frac{1}{1+e^{-\lambda x}}$ or $f(x)=\frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$. λ and other weights are then trained simultaneously.

2) GA is another interesting alternative to tune these weights due to its global search ability.

3) In this study, all nonlinear identification models are trained offline. The possible next step research is to apply these three identification models in online nonlinear system identification and control applications.

4) This research mainly focuses on *SISO* nonlinear systems identification, one possible future is to use the proposed algorithm to do multi-input and multi-output (*MIMO*) systems identification and the study of decentralization control.

Appendix A1: Weight adaptation laws for general NNs

For the j^{th} neuron in any layer, as shown in Figure A1-1, the net output of the neuron is:

$$net_j = \sum_{i=1}^{M} w_{ij} x_j + \theta_j \tag{A1-1}$$

Where w_{ij} is the connection weight between this j^{th} neuron and the i^{th} neuron in the previous layer; x_i is the input of the j^{th} neuron, which equals to the output of the i^{th} neuron y_i, and θ_j is the bias of this j^{th} neuron.

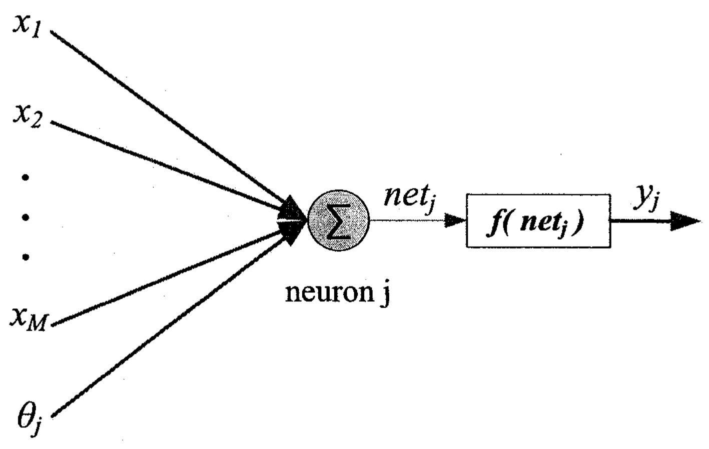


Figure A1-1 The structure of a neuron

The net output of this j^{th} neuron is:

$$y_j = f(net_j) \tag{A1-2}$$

where $f(\cdot)$ is the activation function of this j^{th} neuron.

For a given network, the error between an output neuron's output and its real/target output is:

$$e_j = y_{dj} - y_{lj} \tag{A1-3}$$

Where y_{dj} and y_{lj} represent the j^{th} output neuron's real/target output and its neural network output, respectively.

In general, suppose there are q output neurons in the output layer, thus, the mean square error of a neural network under the p^{th} training sample is:

$$E^{(p)} = \frac{1}{2}\sum_{j=1}^{q} e_j^2 = \frac{1}{2}\sum_{j=1}^{q}(y_{dj}^{(p)} - y_{lj}^{(p)})^2 \tag{A1-4}$$

Assume there are P training samples fed into a neural network, thus, the total mean square error of the network becomes:

$$\begin{aligned} E &= \sum_{m=1}^{P} E^{(m)} \\ &= \sum_{m=1}^{P}\frac{1}{2}\sum_{j=1}^{q}(y_{dj}^{(m)} - y_{lj}^{(m)})^2 \\ &= \frac{1}{2}\sum_{m=1}^{P}\sum_{j=1}^{q}(y_{dj}^{(m)} - y_{lj}^{(m)})^2 \end{aligned} \tag{A1-5}$$

Base on the gradient descent learning algorithms, the weights updating principles are as follows.

To the incoming weight w_{ij} of the j^{th} neuron,

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} = -\eta\sum_{m=1}^{P}\frac{\partial E^{(m)}}{\partial w_{ij}} \tag{A1-6}$$

Where η is the learning rate, and, in general, $0 < \eta < 1$.

Under the p^{th} training sample,

$$\begin{aligned} \Delta w_{ij}^{(p)} &= -\eta\frac{\partial E^{(p)}}{\partial w_{ij}} \\ &= -\eta\frac{\partial E^{(p)}}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \end{aligned} \tag{A1-7}$$

Define the back-propagated error as:

$$\delta_{ij}^{(p)} = -\frac{\partial E^{(p)}}{\partial net_j} \tag{A1-8}$$

The weights updated as:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) + \Delta w_{ij}^{(p)} \\ &= w_{ij}(t) + \eta \delta_{ij}^{(p)} \end{aligned} \tag{A1-9}$$

where t represents a time step.

For all training samples:

$$w_{ij}(t+1) = w_{ij}(t) + \sum_{m=1}^{P} \Delta w_{ij}^{(m)} \tag{A1-10}$$

Since, only *SISO* systems are studied here, thus there is only one output neuron in the output layer of the NN as shown Figure 4-2.

A1.1 Weights updating by using unipolar sigmoid function as activation function

For the output layer:

Under the p^{th} training sample, the output of a NN is the net output (activation function's output) of its output neuron, which is:

$$y_l^{(p)} = f(net_l) = \frac{1}{1+e^{-net_l}} \tag{A1-11}$$

Its derivative is:

$$
\begin{aligned}
y_l'^{(p)} &= f'(net_l) \\
&= \frac{-(-e^{-net_l})}{(1+e^{-net_l})^2} \\
&= \frac{1+e^{-net_l}-1}{(1+e^{-net_l})^2} \\
&= y_l^{(p)} - y_l^{(p)2} \\
&= y_l^{(p)}(1-y_l^{(p)})
\end{aligned}
\tag{A1-12}
$$

The weights are updated as:

$$
\begin{aligned}
w_{kl}(t+1) &= w_{kl}(t) + \Delta w_{kl}^{(p)} \\
&= w_{kl}(t) - \eta \frac{\partial E^{(p)}}{\partial w_{kl}} \\
&= w_{kl}(t) - \eta \frac{\partial E^{(p)}}{\partial net_l} \frac{\partial net_l}{\partial w_{kl}} \\
&= w_{kl}(t) - \eta \frac{\partial E^{(p)}}{\partial net_l} \frac{\partial (w_{kl} x_l^{(p)})}{\partial w_{kl}} \\
&= w_{kl}(t) - \eta \frac{\partial E^{(p)}}{\partial net_l} \frac{\partial (w_{kl} y_k^{(p)})}{\partial w_{kl}} \\
&= w_{kl}(t) - \eta \frac{\partial E^{(p)}}{\partial net_l} y_k^{(p)}
\end{aligned}
\tag{A1-13}
$$

Define the back-propagated error of the output layer as:

$$\begin{aligned}\delta_{kl}^{(p)} &= -\frac{\partial E^{(p)}}{\partial net_l} \\ &= -\frac{\partial E^{(p)}}{\partial y_l^{(p)}}\frac{\partial y_l^{(p)}}{\partial net_l} \\ &= -\frac{\partial(\frac{1}{2}(y_d^{(p)} - y_l^{(p)})^2)}{\partial y_l^{(p)}} y_l'^{(p)} \\ &= (y_d^{(p)} - y_l^{(p)})y_l^{(p)}(1 - y_l^{(p)})\end{aligned} \qquad \text{(A1-14)}$$

then,

$$\begin{aligned}\Delta w_{kl}^{(p)} &= -\eta\frac{\partial E^{(p)}}{\partial w_{kl}} \\ &= -\eta\frac{\partial E^{(p)}}{\partial net_l}\frac{\partial net_l}{\partial w_{kl}} \\ &= \eta\delta_{kl}^{(p)}\frac{\partial(\sum_{k=1}^{N_k} w_{kl}x_l^{(p)})}{\partial w_{kl}} \\ &= \eta\delta_{kl}^{(p)}\frac{\partial(\sum_{k=1}^{N_k} w_{kl}y_k^{(p)})}{\partial w_{kl}} \\ &= \eta\delta_{kl}^{(p)}y_k^{(p)}\end{aligned} \qquad \text{(A1-15)}$$

So, under the p^{th} training sample,

$$\begin{aligned}w_{kl}(t+1) &= w_{kl}(t) + \Delta w_{kl}^{(p)} \\ &= w_{kl}(t) + \eta\delta_{kl}^{(p)}y_k^{(p)} \\ &= w_{kl}(t) + \eta(y_d^{(p)} - y_l^{(p)})y_l'^{(p)}y_k^{(p)} \\ &= w_{kl}(t) + \eta(y_d^{(p)} - y_l^{(p)})y_l^{(p)}(1 - y_l^{(p)})y_k^{(p)}\end{aligned} \qquad \text{A1-16)}$$

Under all P training samples, the weights in the output layer are updated as follows.

$$\begin{aligned} w_{kl}(t+1) &= w_{kl}(t) + \sum_{m=1}^{P} \Delta w_{kl}^{(m)} = w_{kl}(t) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} y_k^{(m)} \\ &= w_{kl}(t) + \eta \sum_{m=1}^{P} (y_d^{(m)} - y_l^{(m)}) y_l'^{(m)} y_k^{(m)} \\ &= w_{kl}(t) + \eta \sum_{m=1}^{P} (y_d^{(m)} - y_l^{(m)}) y_l^{(m)} (1 - y_l^{(m)}) y_k^{(m)} \end{aligned} \tag{A1-17}$$

For the 2nd hidden layer:

under the p^{th} training sample,

$$\begin{aligned} w_{jk}(t+1) &= w_{jk}(t) + \Delta w_{jk}^{(p)} \\ &= w_{jk}(t) - \eta \frac{\partial E^{(p)}}{\partial w_{jk}} \\ &= w_{jk}(t) - \eta \frac{\partial E^{(p)}}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} \\ &= w_{jk}(t) - \eta \frac{\partial E^{(p)}}{\partial net_k} \frac{\partial (\sum_{j=1}^{N_J} w_{jk} x_k^{(p)})}{\partial w_{jk}} \\ &= w_{jk}(t) - \eta \frac{\partial E^{(p)}}{\partial net_k} \frac{\partial (\sum_{j=1}^{N_J} w_{jk} y_j^{(p)})}{\partial w_{jk}} \\ &= w_{jk}(t) - \eta \frac{\partial E^{(p)}}{\partial net_k} y_j^{(p)} \end{aligned} \tag{A1-18}$$

where N_J is the neuron number in the *1st* hidden layer.

Define the back-propagated error $\delta_{jk}^{(p)}$ in the *2nd* hidden layer under the p^{th} training sample as:

$$\begin{aligned}
\delta_{jk}^{(p)} &= -\frac{\partial E^{(p)}}{\partial net_k} \\
&= -\frac{\partial E^{(p)}}{\partial y_k^{(p)}} \frac{\partial y_k^{(p)}}{\partial net_k} \\
&= -\frac{\partial E^{(p)}}{\partial y_k^{(p)}} f'(net_k) \\
&= -\frac{\partial E^{(p)}}{\partial net_l} \frac{\partial net_l}{\partial y_k^{(p)}} f'(net_k) \\
&= \delta_{kl}^{(p)} \frac{\partial (\sum_{k=1}^{N_K} w_{kl} x_l^{(p)})}{\partial y_k^{(p)}} f'(net_k) \\
&= \delta_{kl}^{(p)} \frac{\partial (\sum_{k=1}^{N_K} w_{kl} y_k^{(p)})}{\partial y_k^{(p)}} f'(net_k) \\
&= \delta_{kl}^{(p)} w_{kl} f'(net_k) \\
&= \delta_{kl}^{(p)} w_{kl} y_k^{(p)} (1 - y_k^{(p)})
\end{aligned} \tag{A1-19}$$

where N_K is the neuron number in the 2nd hidden layer.

Thus,

$$\begin{aligned}
w_{jk}(t+1) &= w_{jk}(t) - \eta \frac{\partial E^{(p)}}{\partial net_k} y_j^{(p)} \\
&= w_{jk}(t) + \eta \delta_{jk}^{(p)} y_j^{(p)} \\
&= w_{jk}(t) + \eta \delta_{kl}^{(p)} w_{kl} y_k^{(p)} (1 - y_k^{(p)}) y_j^{(p)}
\end{aligned} \tag{A1-20}$$

Under all P training samples, the weights in the 2nd hidden layer are updated as follows.

$$\begin{aligned}
w_{jk}(t+1) = w_{jk}(t) + \sum_{m=1}^{P} \Delta w_{jk}^{(m)} &= w_{jk}(t) + \eta \sum_{m=1}^{P} \delta_{jk}^{(m)} y_j^{(m)} \\
&= w_{jk}(t) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} w_{kl} y_k^{(m)} (1 - y_k^{(m)}) y_j^{(m)}
\end{aligned} \tag{A1-21}$$

For the 1st hidden layer:

under the p^{th} training sample,

$$\begin{aligned}
w_{ij}(t+1) &= w_{ij}(t) + \Delta w_{ij}^{(p)} \\
&= w_{ij}(t) - \eta \frac{\partial E^{(p)}}{\partial w_{ij}} \\
&= w_{ij}(t) - \eta \frac{\partial E^{(p)}}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} \\
&= w_{ij}(t) - \eta \frac{\partial E^{(p)}}{\partial net_j} \frac{\partial (\sum_{i=1}^{N_I} w_{ij} y_i^{(p)})}{\partial w_{ij}} \\
&= w_{ij}(t) - \eta \frac{\partial E^{(p)}}{\partial net_j} y_i^{(p)}
\end{aligned} \qquad \text{(A1-22)}$$

Define the back-propagated error $\delta_{ij}^{(p)}$ in the *1st* hidden layer under the p^{th} training sample as:

$$\begin{aligned}
\delta_{ij}^{(p)} &= -\frac{\partial E^{(p)}}{\partial net_j} \\
&= -\frac{\partial E^{(p)}}{\partial y_j^{(p)}} \frac{\partial y_j^{(p)}}{\partial net_j} \\
&= -\frac{\partial E^{(p)}}{\partial y_j^{(p)}} f'(net_j) \\
&= -\sum_{k=1}^{N_K} (\frac{\partial E^{(p)}}{\partial net_k} \frac{\partial net_k}{\partial y_j^{(p)}}) f'(net_j) \\
&= \sum_{k=1}^{N_K} (\delta_{jk}^{(p)} \frac{\partial (\sum_{j=1}^{N_J} w_{jk} y_j^{(p)})}{\partial y_j^{(p)}}) f'(net_j) \\
&= \sum_{k=1}^{N_K} (\delta_{jk}^{(p)} w_{jk}) y_j^{(p)} (1 - y_j^{(p)})
\end{aligned} \qquad \text{(A1-23)}$$

Thus,

$$
\begin{aligned}
w_{ij}(t+1) &= w_{ij}(t) - \eta \frac{\partial E^{(p)}}{\partial net_j} y_i^{(p)} \\
&= w_{ij}(t) + \eta \delta_{ij}^{(p)} y_i^{(p)} \\
&= w_{ij}(t) + \eta \sum_{k=1}^{N_K} (\delta_{jk}^{(p)} w_{jk}) y_j^{(p)} (1 - y_j^{(p)}) y_i^{(p)}
\end{aligned}
\tag{A1-24}
$$

Or $w_{ij}(t+1)$ can be got directly by:

$$
\begin{aligned}
w_{ij}(t+1) &= w_{ij}(t) + \Delta w_{ij}^{(p)} \\
&= w_{ij}(t) - \eta \frac{\partial E^{(p)}}{\partial w_{ij}} \\
&= w_{ij}(t) - \eta \sum_{k=1}^{N_K} (\frac{\partial E^{(p)}}{\partial net_k} \frac{\partial net_k}{\partial w_{ij}}) \\
&= w_{ij}(t) + \eta \sum_{k=1}^{N_K} (\delta_{jk}^{(p)} \frac{\partial net_k}{\partial y_j^{(p)}} \frac{\partial y_j^{(p)}}{\partial w_{ij}}) \\
&= w_{jk}(t) + \eta \sum_{k=1}^{N_K} \delta_{jk}^{(p)} \frac{\partial (\sum_{j=1}^{N_J} w_{jk} y_j^{(p)})}{\partial y_j^{(p)}} \frac{\partial y_j^{(p)}}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} \\
&= w_{jk}(t) + \eta \sum_{k=1}^{N_K} \delta_{jk}^{(p)} w_{jk} f'(net_j) \frac{\partial (\sum_{i=1}^{N_I} w_{ij} y_i^{(p)})}{\partial w_{ij}} \\
&= w_{jk}(t) + \eta \sum_{k=1}^{N_K} \delta_{jk}^{(p)} w_{jk} y_j^{(p)} (1 - y_j^{(p)}) y_i^{(p)}
\end{aligned}
\tag{A1-25}
$$

Under all P training samples, the weights in the *1st* hidden layer are updated as follows.

$$
\begin{aligned}
w_{ij}(t+1) &= w_{ij}(t) + \sum_{m=1}^{P} \Delta w_{ij}^{(m)} = w_{ij}(t) + \eta \sum_{m=1}^{P} \delta_{ij}^{(m)} y_i^{(m)} \\
&= w_{ij}(t) + \eta \sum_{m=1}^{P} [\sum_{k=1}^{N_K} (\delta_{jk}^{(m)} w_{jk}) y_j^{(m)} (1 - y_j^{(m)})] y_i^{(m)}
\end{aligned}
\tag{A1-26}
$$

In summary, the weights updating laws in the output layer, the *2nd* hidden layer, and the *1st* hidden layer under all P training samples are as follows, respectively.

$$w_{kl}(t+1) = w_{kl}(t) + \sum_{m=1}^{P} \Delta w_{kl}^{(m)} = w_{kl}(t) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} y_k^{(m)} \qquad \text{(A1-27)}$$

$$w_{jk}(t+1) = w_{jk}(t) + \sum_{m=1}^{P} \Delta w_{jk}^{(m)} = w_{jk}(t) + \eta \sum_{m=1}^{P} \delta_{jk}^{(m)} y_j^{(m)} \qquad \text{(A1-28)}$$

$$w_{ij}(t+1) = w_{ij}(t) + \sum_{m=1}^{P} \Delta w_{ij}^{(m)} = w_{ij}(t) + \eta \sum_{m=1}^{P} \delta_{ij}^{(m)} y_i^{(m)} \qquad \text{(A1-29)}$$

where the corresponding back-propagated errors in different layers are as follows, respectively.

$$\delta_{kl}^{(p)} = (y_d^{(p)} - y_l^{(p)}) f'(net_l) = (y_d^{(p)} - y_l^{(p)}) y_l^{(p)} (1 - y_l^{(p)}) \qquad \text{(A1-30)}$$

$$\delta_{jk}^{(p)} = \delta_{kl}^{(p)} w_{kl} f'(net_k) = \delta_{kl}^{(p)} w_{kl} y_k^{(p)} (1 - y_k^{(p)}) \qquad \text{(A1-31)}$$

$$\delta_{ij}^{(p)} = \sum_{k=1}^{N_K} (\delta_{jk}^{(p)} w_{jk}) f'(net_j) = \sum_{k=1}^{N_K} (\delta_{jk}^{(p)} w_{jk}) y_j^{(p)} (1 - y_j^{(p)}) \qquad \text{(A1-32)}$$

Also, the hidden layer weight adaptation laws can be written by the following general formulas. There will not be for a specific hidden layer. For the *j*th hidden neuron in the *l*th hidden layer:

$$\Delta w_{ij}^{l(p)} = -\eta \frac{\partial E^{(p)}}{w_{ij}^{l(p)}} = -\eta \frac{\partial E^{(p)}}{net_j^{l(p)}} \frac{\partial net_j^{l(p)}}{\partial w_{ij}^{l(p)}} = \eta \delta_{ij}^{l(p)} \frac{\partial net_j^{l(p)}}{\partial w_{ij}^{l(p)}} \qquad \text{(A1-33)}$$

where,

$$\delta_{ij}^{l(p)} = -\frac{\partial E^{(p)}}{\partial net_j^{l(p)}} = \sum_{k=1}^{N_{l+1}} (\delta_{jk}^{(l+1)(p)} w_{jk}) f'(net_j^{l(p)}) \qquad \text{(A1-34)}$$

and,

$$\frac{\partial net_j^{l(p)}}{\partial w_{ij}^{l(p)}} = y_i^{(l-1)(p)} \tag{A1-35}$$

A1.2 Weights updating by using bipolar sigmoid function as activation function

For the j^{th} neuron in any layer,

$$net_j = \sum_{i=1}^{M} w_{ij} x_j + \theta_j \tag{A1-36}$$

Where w_{ij} is the incoming weights of this j^{th} neuron; x_i is the input of the j^{th} neuron, which equals to the output y_i of the i^{th} neuron in the previous layer, and θ_j is the bias of this j^{th} neuron.

The net output of this j^{th} neuron is:

$$\begin{aligned} y_j &= f(net_j) \\ &= \frac{1-e^{-net_j}}{1+e^{-net_j}} = \frac{2}{1+e^{-net_j}} - 1 \end{aligned} \tag{A1-37}$$

and

$$\begin{aligned} y_j' &= f'(net_j) \\ &= \frac{-2(e^{-net_j})'}{(1+e^{-net_j})^2} \\ &= \frac{1}{2}\frac{4e^{-net_j}}{(1+e^{-net_j})^2} \\ &= \frac{1}{2}\frac{(1+e^{-net_j})^2-(1-e^{-net_j})^2}{(1+e^{-net_j})^2} \\ &= \frac{1}{2}[1-(\frac{1-e^{-net_j}}{1+e^{-net_j}})^2] \\ &= \frac{1}{2}(1-y_j^2) \end{aligned} \tag{A1-38}$$

So, the corresponding back-propagated errors in the output layer, the *2nd* hidden layer, an in the *1st* hidden layer under the p^{th} training sample are as follows, respectively.

$$\delta_{kl}^{(p)} = (y_d^{(p)} - y_l^{(p)})f'(net_l) = \frac{1}{2}(y_d^{(p)} - y_l^{(p)})(1-(y_l^{(p)})^2) \tag{A1-39}$$

$$\delta_{jk}^{(p)} = \delta_{kl}^{(p)} w_{kl} f'(net_k) = \frac{1}{2}\delta_{kl}^{(p)} w_{kl}(1-(y_k^{(p)})^2) \tag{A1-40}$$

$$\delta_{ij}^{(p)} = \sum_{k=1}^{N_K}(\delta_{jk}^{(p)} w_{jk}) f'(net_j) = \frac{1}{2}\sum_{k=1}^{N_K}(\delta_{jk}^{(p)} w_{jk})(1-(y_j^{(p)})^2) \tag{A1-41}$$

weight updating laws in the output layer, the *2nd* hidden layer, an the *1st* hidden layer under the p^{th} training sample are as follows, respectively.

$$w_{kl}(t+1) = w_{kl}(t) + \sum_{m=1}^{P}\Delta w_{kl}^{(m)} = w_{kl}(t) + \eta\sum_{m=1}^{P}\delta_{kl}^{(m)} y_k^{(m)} \tag{A1-42}$$

$$w_{jk}(t+1) = w_{jk}(t) + \sum_{m=1}^{P}\Delta w_{jk}^{(m)} = w_{jk}(t) + \eta\sum_{m=1}^{P}\delta_{jk}^{(m)} y_j^{(m)} \tag{A1-43}$$

$$w_{ij}(t+1) = w_{ij}(t) + \sum_{m=1}^{P} \Delta w_{ij}^{(m)} = w_{ij}(t) + \eta \sum_{m=1}^{P} \delta_{ij}^{(m)} y_i^{(m)} \tag{A1-44}$$

A1.3 Weights updating by using linear activation function for output neurons

In many cased, the linear activation function $f(net) = net$ will be used for the output layer. Thus, the weights adaptation law in the output layer will be as follows.

since,

$$y_l'^{(p)} = f'(net_l) = (net_l)' = 1 \tag{A1-45}$$

and,

$$\begin{aligned} \delta_{kl}^{(p)} &= -\frac{\partial E^{(p)}}{\partial net_l} \\ &= -\frac{\partial E^{(p)}}{\partial y_l^{(p)}} \frac{\partial y_l^{(p)}}{\partial net_l} \\ &= -\frac{\partial (\frac{1}{2}(y_d^{(p)} - y_l^{(p)})^2)}{\partial y_l^{(p)}} y_l'^{(p)} \\ &= (y_d^{(p)} - y_l^{(p)}) \end{aligned} \tag{A1-46}$$

then,

$$\begin{aligned}
\Delta w_{kl}^{(p)} &= -\eta \frac{\partial E^{(p)}}{\partial w_{kl}} \\
&= -\eta \frac{\partial E^{(p)}}{\partial net_l} \frac{\partial net_l}{\partial w_{kl}} \\
&= \eta \delta_{kl}^{(p)} \frac{\partial (w_{kl} x_l^{(p)})}{\partial w_{kl}} \\
&= \eta \delta_{kl}^{(p)} \frac{\partial (w_{kl} y_k^{(p)})}{\partial w_{kl}} \\
&= \eta \delta_{kl}^{(p)} y_k^{(p)}
\end{aligned} \tag{A1-47}$$

So, under the p^{th} training sample,

$$\begin{aligned}
w_{kl}(t+1) &= w_{kl}(t) + \Delta w_{kl}^{(p)} \\
&= w_{kl}(t) + \eta \delta_{kl}^{(p)} y_k^{(p)} \\
&= w_{kl}(t) + \eta (y_d^{(p)} - y_l^{(p)}) y_k^{(p)}
\end{aligned} \tag{A1-48}$$

Under all P training samples, the weights in the output layer are updated as:

$$w_{kl}(t+1) = w_{kl}(t) + \sum_{m=1}^{P} \Delta w_{kl}^{(m)} = w_{kl}(t) + \eta \sum_{m=1}^{P} \delta_{kl}^{(m)} y_k^{(m)} \tag{A1-49}$$

References

Agarwal, M. (1997). "*A systematic classification of neural-network based control.*" IEEE Control Systems Magazine 17: 75-93.

Anderson, B., J. Moore, et al. (1978). "*Model approximation via prediction error identification.*" Atuomatica 14: 615-622.

Angerer, B. T., C. Hintz, et al. (2004). "*Online identification of a nonlinear mechatronic system.*" Control Engineering Practice 12: 1465-1478.

Asdente, M., M. C. Pascucci, et al. (1976). "*Modified Volterra-Wiener functional method for highly nonlinear systems.*" Alta frequenza 45(12): 369-380.

Astrom, K. J. and T. Bohlin (1965). "*Numerical identification of linear dynamic systems from normal operating records.*" Proc. 2nd IFAC Symposium on the Theory of Self-Adaptive Control Systems, Teddington, UK.

Barrett, J. F. (1963). "*The use of functionals in the analysis of nonlinear physical systems.*" Electron. & Contr. 15(6): 567-615.

Becerra, V. M., F. R. Garces, et al. (2005). "*An Efficient Parameterization of Dynamic Neural Networks for Nonlinear System Identification.*" IEEE Transactions on neural networks 16(4): 983-988.

Bendat, J. S. (1990). "*Nonlinear system analysis and identification from random data.*" New York, John Wiley and Sons.

Billings, S. A. (1980). "*Identification of Nonlinear Systems - a Survey.*" IEE Proceedings 127: 272-285.

Billings, S. A. and S. Chen (1992). "*Neural networks and system identification.*" Neural Networks for Systems and Control. K. W. e. al. London, Peter Peregrinus: 181-205.

Billings, S. A. and S. Y. Fakhouri (1977). "*Identification of nonlinear systems using the Wiener model.*" Electron. Lett. 13 (17): 502-504.

Bose, N. K. and P. Liang (1998). "*Neural Network Fundamentals with Graphs, Algorithms, and Applications.*" New Delhi, India: Tada, McGraw-Hill.

Canelon, J. I., L. S. Shieh, et al. (2004). "*A new approach for neural control of nonlinear discrete dynamic systems.*" Information Sciences.

Chandra, P. and Y. Singh (2004). "*Feedforward Sigmoidal Networks--Equicontinuity and*

Fault-Tolerance Properties." IEEE Transactions on neural networks 15(6): 1350- 1366.

Chen, S., S. A. Billings, et al. (1990). "*Nonlinear system identification using neural networks*." International Journal of Control 51(6): 1191-1214.

Chester, D. L. (1990). "*Why two hidden layers are better than one*." International Joint Conference on Neural Networks.

Cybenko, G. (1989). "*Approximation by superpositions of sigmoidal function*." Mathematics of Control, Signals and Systems 2: 303-314.

Enqvist, M. (2005). "*Linear models of nonlinear systems*." Department of Electrical Engineering, Linköpings University, Sweden. PhD.

Feng, Z. and A. N. Michel (1999). "*Robustness analysis of a class of discrete-time systems with applications to neural networks*." American Control Conference, San Diego.

Frean, M. (1990). "*The upstart algorithm: A method for constructing and training feedforward neural networks*." Neural Computation 2(2): 198-209.

Frenz, T. and D. Schröder (1997). "*Online identification and compensation of friction influence of feed drives of machine tools*." Proc. Electronics and Applications (EPE) Trondheim, Norway.

Funahashi, K. (1989). "*On the approximate realization of continuous mappings by neural networks*." Neural Networks 2(3): 183-192.

Funahashi, K. and Y. Nakamura (1993). "*Approximation of dynamical systems by continuous time recurrent neural networks*." Neural Networks 6: 801-806.

Gary, G., D. M. Smith, et al. (1998). "*Nonlinear model structure identification using genetic programming*." Control Engineering Practice 6: 1341-1352.

Gauss, C. F. (1809). "*Theoria motus corporum coelestium in sectionis conicis solem ambientum*." Hamburg: Perthes und Besser.

Gevers, M. and L. Ljung (1986). "*Optimal experiment designs with respect to the intended model application*." Automatica 22(5): 543-554.

Gupta, P. and N. K. Sinha (1999). "*An improved approach for nonlinear system identification using neural networks*." Journal of the Franklin Institute 336: 721-734.

Gybenko, G. (1989). "*Approximation by Superpositions of a Sigmoidal Function*." Math. Control, Signals, Syst. 2: 303-314.

Ham, F. M. and I. Kostanic (2001). "*Principles of Neurcomputing for Science and*

Engineering." Singapore, McGraw-Hill.

Haykin, S. (1999). "*Neural Networks: A Comprehensive Foundation.*" NJ, Prentice-Hal.

Hildebrand, R. and M. Gevers (2003). "*Identification for control: Optimal input design with respect to a worst case v-gap cost function.*" SIAM Journal on Control and Optimization 41(5): 1586-1608.

Hirose, Y., K. Yamashita, et al. (1991). "*Back-propagation algorithm which varies the number of hidden units.*" Neural Networks 4(1): 61-66.

Hjalmarsson, H. (2005). "*From experiment design to closed-loop control.*" Automatica 41: 393-438.

Ho, B. and R. Kalman (1965). "*Effective construction of linear state-variable models from input-output functions.*" Regelungstechnik 12: 545-548.

Holland, J. H. (1975). "*Adaptation in Natural and Artificial Systems.*" Ann Arbor, The University of Michigan Press.

Hornik, K., M. Stinchcombe, et al. (1989). "*Multilayer feedforward networks are universal approximators.*" Neural Networks 2: 359-366.

Huang, S. J., S. N. Koh, et al. (1992). "*Training algorithm based on newton's method with dynamic error control.*" International Joint Conference Neural Network, Baltimore.

Hunt, K. J., D. Sbarbaro, et al. (1992). "*Neural networks for control systems-a survey.*" Automatica 28: 1083-1112.

Hunt, L. R., R. D. DeGroat, et al. (1993). "*Identification of discrete-time nonlinear systems.*" The 32nd Conference on Decision and Control.

Ioannou, P. and J. Sun (1996). "*Robust Adaptive Control.*" New Jersey, Prentice-Hall.

Jagannathan, S. and F. L. Lewis (1996). "*Identification of nonlinear dynamical systems using multilayered neural networks.*" Automatica 32(12): 1707-1712.

Jan, C. and C.-L. Hwang (2000). "*Robust control design for a piezoelectric actuator system with dominant hysteresis.*" Industrial Electronics Society. IECON 2000. IEEE International Conference on Industrial Electronics, Control and Instrumentation.

Jin, L. and M. M. Gupta (1999). "*Stable dynamic Back-propagation learning in recurrent neural networks.*" IEEE Trans. Neural Networks 10(6): 1321-1334.

Juditsky, A., H. Hjalmarsson, et al. (1995). "*Nonlinear black-box models in system identification: mathematical foundations.*" Automatica 31(12): 1725-1750.

Kadirkamanathan, V. and G. P. Liu (1995). "*Robust identification with neural networks using multiobjective criteria.*" The 5th IFAC Symposium on Adaptive Systems in Control and Signal Processing.

Kalinli, A. and D. Karaboga (2004). "*Training recurrent neural networks by using parallel tabu search algorithm based on crossover operation.*" Engineering Applications of Artificial Intelligence 17: 529-542.

Karnin, E. D. (1990). "*A Simple Procedure for Pruning Back-Propagation Trained Neural Networks.*" IEEE trans. Neural Networks 1(2): 239-242.

Kiong, L. C., M. Rajeswari, et al. (2003). "*Nonlinear dynamic system identification and control via constructivism inspired neural network.*" Applied Soft Computing 3: 237-257.

Kitano, H. (1990). "*Designing neural networks using genetic algorithms with graph generation systems.*" Complex Systems 4: 461-476.

Kosmatopoulos, E. B., M. M. Ploycarpou, et al. (1995). "*High-order neural network structures for identification of dynamical systems.*" IEEE Trans. Neural Networks 6(2): 431-442.

Kuschewski, J. G., S. Hui, et al. (1993). "*Application of feedforward neural networks to dynamical system identification and control.*" IEEE Transactions on Control Systems Technology 1(1): 37-49.

Liu, G. P. (2001). "*Nonlinear Identification and Control: A Neural Network Approach.*" New York, Springer.

Liu, G. P., V. Kadirkamanathan, et al. (1996). "*Variable neural networks for adaptive control of nonlinear systems.*" 13th IFAC World Congress.

Liu, G. P., V. Kadirkamanathan, et al. (1998). "*Nonlinear predictive control using neural networks.*" International Journal of Control.

Liu, G. P., V. Kadirkamanathan, et al. (1998). "*Variable neural networks for adaptive control of nonlinear systems.*" IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews 29(1): 34-43.

Liu, Z., A. Liu, et al. (2004). "*Evolving neural network using real coded genetic algorithm (GA) for multispectral image classification.*" Future Generation Computer Systems 20: 1119-1129.

Ljung, L. (1999). "*System identification: Theory for the User.*" New Jersey, Prentice-Hall.

Ljung, L. (2005). "*Identification of Linear and Nonlinear Dynamical Systems.*" Berkeley.

Ljung, L. and P. Caines (1979). "*Asymptotic normality of prediction error estimators for approximate system models.*" Stochastics 3: 29-46.

Madár, J., J. Abonyi, et al. (2005). "*Genetic Programming for the Identification of Nonlinear Input-Output Models.*" Industrial and Engineering Chemistry Research 44 (9): 3178-3186.

Marin, F. J. and F. Sandoval (1993). "*Genetic synthesis of discrete time recurrent neural network.*" Int. Workshop Artificial Neural Networks.

McClelland, J. L. and D. E. Rumelhart (1988). "*Parallel Distributed Processing.*" Cambridge, London, England, The MIT Press.

Melanie, M. (1999). "*An Introduction to Genetic Algorithms.*" Cambridge, Massachusetts • London, England, The MIT Press.

Miller, G. F., P. M. Todd, et al. (1989). "*Designing neural networks using genetic algorithms.*" 3rd Int. Conf. Genetic Algorithms and Their Applications.

Narendra, K. G., V. K. Sood, et al. (1995). "*A Neuro-Fuzzy VDCL Unit to Enhance the Performance of an HVDC System.*" Canadian Conference on Electrical and Computer Engineering.

Narendra, K. S. and K. Parthasarathy (1990). "*Identification and control of dynamical systems using neural networks.*" IEEE Transactions on Neural Networks 1(1): 4-27.

Overschee, P. V. and B. D. Moor (1994). "*N4SID: subspace algorithms for the identification of combined deterministic-stochastic systems.*" Automatica 30: 75-93.

Pham, D. and D. Karaboga (1999). "*Training Elman and Jordan networks for system identification using genetic algorithms.*" Artificial Intelligence in Engineering 13(2): 107-117.

Piche, S. W. (1994). "*Steepest descent algorithms for neural network controllers and filters.*" IEEE Transaction on Neural Networks 5(2): 198-212.

Polycarpou, M. M. and P. A. Ioannou (1992). "*Learning and convergence analysis of neural-type structured networks.*" IEEE Trans. Neural Networks 3(1): 39-50.

Polycarpou, M. M. and P. A. Loannou (1991). "*Identification and control of nonlinear systems using neural network models: design and stability analysis.*" USA, Department of Electrical Engineering Systems, University of Southern California.

Prasad, V. and B. W. Bequette (2003). "*Nonlinear system identification and model reduction using artificial neural networks.*" Computers and Chemical Engineering 27: 1741-1754.

Qi, J. (2005). MASc. thesis. Electrical and Computer Engineering. Montreal, Concordia university. Master.

Qin, S. Z., H. T. Su, et al. (1992). "*Comparison of four net learning methods for dynamic system identification.*" IEEE Transactions on Neural Networks 3(1): 122-130.

Ramacher, U. (1993). "*Hamiltonian dynamics of neural networks.*" Neural Networks 6(4): 547-557.

Reed, M. J. and M. O. J. Hawksford (1996). "*Identification of discrete Volterra series using maximum length sequences.*" Proc. IEE, Circuits, Devices and Systems 143(5): 241-248.

Fisher, R. A. (1918a) "*The Correlation between Relatives on the Supposition of Mendelian Inheritance.*" Transactions of the Royal Society of Edinburgh, 52: 399 - 433.

Rodriguez, K.-V. and P. J. Fleming (1998). "*Multi-objective genetic programming for nonlinear system identification.*" Electronics Letters 34(9): 930-931.

Roy, A., L. S. Kim, et al. (1993). "*A polynomial time algorithm for the construction and training of a class of multilayer perceptions.*" Neural Networks 6(3): 535-545.

Rozario, N. and A. Papoulis (1989). "*The identification of certain nonlinear systems by only observing the output.*" Proc. of HOSA Workshop, Vail, Colorado.

Rugh, W. J. (1981). "*Nonlinear System Theory. The Volterra-Wiener Approach.*" Baltimore, John Hopkins University Press.

Rumelhart, D. E., G. E. Hinton, et al. (1986). "*Learning internal representations by error propagation.*" Parallel Distributed Processing: Explorations in the Microstructure of Cognition. D. Rumelhart and J. McClelland. Cambridge, The M.I.T. Press. 1.

Sagaspe, J. P. (1979). "*About nonlinear identification using Volterra model.*" 5th IFAC Symp. Identif. & Syst. Param. Estim, Darmstadt.

Schaffer, J. D., R. A. Caruana, et al. (1990). "*Using genetic search to exploit the emergent behavior of neural networks.*" Physica D 42(1-3): 244-248.

Schetzen, M. (1989). "*The Volterra and Wiener Theories of Nonlinear Systems.*" Malabar, Robert E. Krieger.

Schiffmann, W., M. Joost, et al. (1993). "*Genetic synthesis of discretive recurrent neural* *network.*" Int. Workshop Artificial Neural Networks.

Schroder, D., C. Hintz, et al. (2001). "*Intelligent modeling, observation, and control fro nonlinear system.*" IEEE/ASME Transactions on Neural Networks 2(6): 122-131.

Sjoberg, J., Q. Zhang, et al. (1995). "*Nonlinear black-box modelling in system identification: a unified overview.*" Automatica 31(12): 1691-1724.

Strobl, D. and D. Schröder (1998). "*Neural observers for the identification of backlash in electromechanical systems.*" Grenoble, France, IFAC Workshop on Motion Control.

Su, C.-Y., Y. Stepanenko, et al. (2000). "*Robust adaptive control of a class of nonlinear systems with unknown backlash-like hysteresis.*" IEEE Transactions on Automatic Control 45(12): 2427-2432.

Suykens, J. A. K., J. Vandewalle, et al. (1997). "*NLq theory: checking and imposing stability of recurrent neural networks for nonlinear modelling.*" IEEE Trans. Signal Process (special issue on neural networks for signal processing) 45(11): 2682-2691.

Tan, K. K., T. H. Lee, et al. (2001). "*Adaptive-Predictive Control of a Class of SISO Nonlinear Systems.*" Dynamics and Control 11(2): 151-174.

Teixeira, M. and S. Zak (1999). "*Stabilizing controller design for uncertain nonlinear systems using fuzzy models.*" IEEE Transactions on Fuzzy Systems 7: 133-142.

Voss, H. U., J. Timmer, et al. (2004). "*nonlinear dynamical system identification from uncertain and indirect measurements.*" International Journal of Bifurcation and Chaos 14(6): 1905-1933.

Wang, Q., P. Spronck, et al. (2003). "*An Overview of Genetic Algorithms Applied to Control Engineering Problems.*" Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi'an.

Weigend, A. (1994). "*On overfitting and the effective number of hidden units.*" Connectionist Models Summer School.

Whitley, D., T. Starkweather, et al. (1990). "*Genetic algorithms and neural networks: Optimizing connections and connectivity.*" Parallel Computing 14(3): 347-361.

Widrow, B. and M. A. Lehr (1990). "*30 years of adaptive Neural Networks: Perceptron, madaline, and backpropagation.*" Proc. IEEE 78: 1415-1442.

Willis, M. J., G. A. Montague, et al. (1992). "*Artificial neural networks in process estimation and control.*" Automatica 28(6): 1181-1187.

Wilson, S. W. (1990). "*Perceptron redux: Emergence of structure.*" Phys. D 42: 249-256.

Xie, W. F., M. Krzeminski, et al. (2002). "*Intelligent friction compensation (IFC) in a harmonic drive.*" Newfoundland Electrical and Computer Engineering Conference.

Yao, X. (1999). "*Evolving artificial neural networks.*" Proceedings of the IEEE 87(9): 1423-1447.

Yazdizadeh, A. (1997). "*Identification of nonlinear system using dynamic neural networks.*" Concordia University. Ph.D. .

Yazdizadeh, A. and K. Khorasani (2002). "*Adaptive time delay neural network structures for nonlinear system identification.*" Neurocomputing 47: 207-240.

Yu, W. (2004). "*Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms.*" Information Sciences 158: 131-147.

Zhou, K., J. Doyle, et al. (1996). "*Robust and Optimal Control.*" Prentice Hall.

Zurada, J. M. (1992). "*Introduction to Artificial Neural Systems.*" West Publishing Company.

Zhu, Y. Q, Xie, W. F and Wang, N. (2006), "*Nonlinear System Identification Using Genetic Algorithm based Recurrent Neural Networks.*" IEEE Canadian Conference on Electrical and Computer Engineering, Ottawa, Canada, May 2006.