The University of Sydney

Department of Civil Engineering
Sydney NSW 2006
AUSTRALIA

http://www.civil.usyd.edu.au/

Environmental Fluids/Wind Group

Self-Organizing Polynomial Neural Network for Modelling Complex Hydrological Processes

Research Report No R861

X. Wang, BE ME PhD
L. Li, BE MPhil PhD
D. Lockington, BE PhD
D. Pullar, BE PhD
D-S Jeng, BE ME PhD

December 2005

ISSN 1833-2781

The University of Sydney

Department of Civil Engineering
Environmental Fluids/Wind Group
http://www.civil.usyd.edu.au/

Self-Organizing Polynomial Neural Network for Modelling Complex Hydrological Processes

Research Report No R861

X. Wang, BE ME PhD
L. Li, BE MPhil PhD
D. Lockington, BE PhD
D. Pullar, BE PhD
D-S Jeng, BE ME PhD

December 2005

Abstract:

Artificial neural networks (ANNs) have been used increasingly for modelling com-plex hydrological processes. In this paper, we present a self-organizing polynomial neural network (SOPNN) algorithm, which combines the theory of bio-cybernetic self-organizing polynomial (SOP) with the artificial neural network (ANN) approach. With the SOP feature of seeking the best combination of polynomial model parame-ters through optimal reduction of the partial polynomial nodes in the network and the ANN function of seeking minimum error at each layer of the network, the algorithm possesses superiority in nonlinear modelling of dynamic systems. The developed al-gorithm is applied to model a real-time rainfall-runoff process. The results demon-strate the capability of the SOPNN approach in addressing difficult issues of ANN modelling: selection of appropriate model inputs, optimization of the network struc-ture and error minimization. The comparison of modelling results shows that the SOPNN algorithm performs better in complex hydrological modelling than two other nonlinear approaches: the group method of data handling (GMDH) algorithm and the traditional back-propagation network (BPN) algorithm

Keywords:

Self-organizing polynomial; artificial neural network; GMDH algorithm, rainfall-runoff modelling

Copyright Notice

Department of Civil Engineering, Research Report R861
Self-Organizing Polynomial Neural Network for Modelling Complex Hydrological Processes

d.jeng@civil.usyd.edu.au

ISSN 1833-2781

Published by:
Department of Civil Engineering
The University of Sydney
Sydney NSW 2006
AUSTRALIA

December 2005

This report and other Research Reports published by The Department of Civil Engineering are available on the Internet:

http://www.civil.usyd.edu.au

Contents

1 Introduction[1]

Modelling of complex hydrological processes such as rainfall-runoff and stream flow has been an active area of research in water resource management, and will remain an interesting topic in the foreseeable future (Kumar et al., 2005). However, hydrological phenomena present difficult challenges for modellers due to large spatial-temporal variability of meteorological and hydrological characteristics. Although a range of conceptual and process-based models have been developed to predict the phenomena, most of them involve either over-simplification or over-parameterisation of the hydrological system (Thirumalaiah and Deo, 2000; Moradklhani et al., 2004). In both cases, model applications are hinged by uncertainties of model input parameters. Inductive modelling provides an alternative approach to simulating complex hydrological processes.

In the last decade, ANN has become an attractive inductive approach to modelling nonlinear hydrological systems because of its flexibility in building models without explicit physical representations which may not be well described in most complex hydrological environments, and its learning peculiarity in abstracting systematic nonlinear characteristics from inputs which consist of pattern, noise and irrelevant data (Thirumalaiah and Deo, 2000). A number of investigations have been conducted to explore the ability of ANNs in mapping nonlinear relationships of hydrological systems (Hsu et al., 1995, 2002; Zealand et al., 1999; Tokar and Markus, 2000; Imrie amd Korre, 2000; Islam and Kothart, 2000; Coulibaly et al., 2001; Moradklhani et al., 2004; Wu et al., 2005). However, the selection of an optimal network structure (layers and nodes) and training algorithms still remains a difficult issue in ANNs applications (Maier and Dandy, 2000). Attempts have been made to use a cascade correlation algorithm (CCA) to rebuild an optimum network structure by adding nodes in the hidden layers during the training process (Imrie amd Korre, 2000; Mahmoud, et al, 2004); however, hidden units added to the network based on the correlation measure can cause problems with computer resources and may also danger the network stability with highly uncertain hydrological data. On the other

[1] Part of this report forms the manuscript: Wang, X, Li, L., Lockington, D., Pullar, D. and Jeng, D.-S. (2005): Self-organizing polynomial neural network for modelling complex hydrological processes. *Journal of Hydrology* (submitted, December 2005)

hand, conjugate gradient algorithm (CGA) has been used in ANNs to speed up the training process (Moller and Scaled, 1993; Thirumalaiah and Deo, 1998, 2000), but the global error problem is still unsolved in many case studies. Global optimization and data uncertainty are two important factors in ANNs' applications, especially for hydrological modelling (Maier and Dandy, 2000).

Recently, the group method of data handling (GMDH) algorithm has been successfully used to deal with uncertainty and nonlinearity of systems in a wide range of disciplines such as economy, ecology, medical diagnostics, signal processing and control systems (Tamura and Kondo, 1980; Ivakhnenko, 1995; Voss and Feng, 2002). Some simplified approximations, such as the two-direction regressive GMDH (Liu and Wang, 1989; Tong et al., 1996) and the revised GMDH algorithms (Chang and Hwang, 1999), have been introduced to model dynamic systems in flood forecast and petroleum resource prediction with some success. However, owing to its limited generic structure, GMDH tends to generate an overly complex polynomial when it is applied to estimate highly nonlinear systems (Tong et al., 1996). This may be the reason why there were only few applications of GMDH in modelling environmental and hydrologic systems.

More recently, a new class of polynomial neural networks (PNNs) combining GMDH with ANNs were explored for modelling highly nonlinear dynamic systems (Wang and Hu, 1999; Oh and Pedrycz, 2002; Aksyonova et al., 2003). Although the PNNs exhibited their capability in capturing nonlinear information, the method was lack of a robust network structure and its implementation was too complicated to deal with the uncertainty of hydrological data (Ivakhnenko, 1995; Oh and Pedrycz, 2002, 2003). In this paper, we improve a self-organizing learning feature in the partial polynomial nodes of PNNs to build a new, self-organizing polynomial neural network (SOPNN) for hydrological modelling.

The rest of the paper is organised as follows: the representation of complex hydrological systems is briefly discussed with nonlinear approaches in section 2. Based on the introduction of the basic structure and algorithm of SOPNN, a robust modelling process is described in section 3. Finally, a case study, using the SOPNN algorithm for the rainfall-runoff modelling in a small catchment, is presented to

compare the performance of SOPNN with those of traditional GMDH and BPN algorithms.

2. Complex Hydrological Systems and Nonlinear Modelling

Complex hydrological phenomena are responsive to both the spatial and temporal variability of meteorological, ecological, geological and environmental conditions in catchments. In representing these phenomena, we need to consider multiple variables taking into account the nonlinearity of involved physical processes at different spatial and temporal scales, and uncertainties in parameter estimates (ASCE, 2000). Consequently, the functional relationship between input units $X^n = \{x_1, x_2, \ldots, x_n\}$ and output units $Y^m = (y_1, y_2, \ldots y_m\}$ of a hydrological system has to be found with specific approaches:

$$Y^m = f(X^n). \tag{1}$$

For example, in the rainfall-runoff dynamic prediction, convoluted functions with "lag differences" are often used in the modelling, i.e.,

$$y_t = f(x_t, x_{t-1}, x_{t-2}, x_{t-3}, \cdots, y_{t-1}, y_{t-2}, y_{t-3}, \cdots), \tag{2}$$

where $x_t, x_{t-1}, x_{t-2}, x_{t-3}, \cdots$ are the input vectors (precipitation record) at time step *t*, *t-1, t-2, t-3,…* while y_t is the output vector (runoff) at time step *t*. The previously measured/predicted output vectors (runoff) $y_{t-1}, y_{t-2}, y_{t-3}, \cdots$ can also be part of the input to improve the precision of the current prediction. The function *f* is generally a nonlinear function.

2.1 Polynomial theory and GMDH algorithm

Any response function (1) in dynamic systems can be found in an unfolding formula of Volterra Series, known as the Kolmogorov-Gabor polynomial (Ivakhnenko, 1971):

$$y_n = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ijk} x_i x_j x_k + \cdots, \tag{3}$$

where $\{x_i, x_j, x_k, \cdots\}$ is the vector of input variables and $\{a_0, a_i, a_{ij}, a_{ijk}, \cdots\}$ is the vector of summand coefficients. This formula exhausts all combinations among input vectors (units); so it is considered to be the complete polynomial description of a system model. However, to determine the coefficients of the polynomial (3) for a general nonlinear system is rather difficult because they depend on not only the number of polynomial terms used but also the number of variables and data. Earlier researches have combined statistics, pattern recognition and least square methods to directly search for the best estimation of the polynomial; most of them, however, ran into problems with the ill conditional data structure and/or limited experimental data (Nalimov and Chernova, 1965; Ivakhnenko, 1971). The GMDH algorithm is one of the most successful fitting algorithms for obtaining an approximate description of formula (3) by combining partial polynomial units of two variables in multiple layers and sifting these units with certain sorting criteria (Ivakhnenko, 1971; Farlow, 1984). The structure of the GMDH algorithm is illustrated in Figure 1. The computation process comprises three basic steps (Chang and Hwang, 1999):

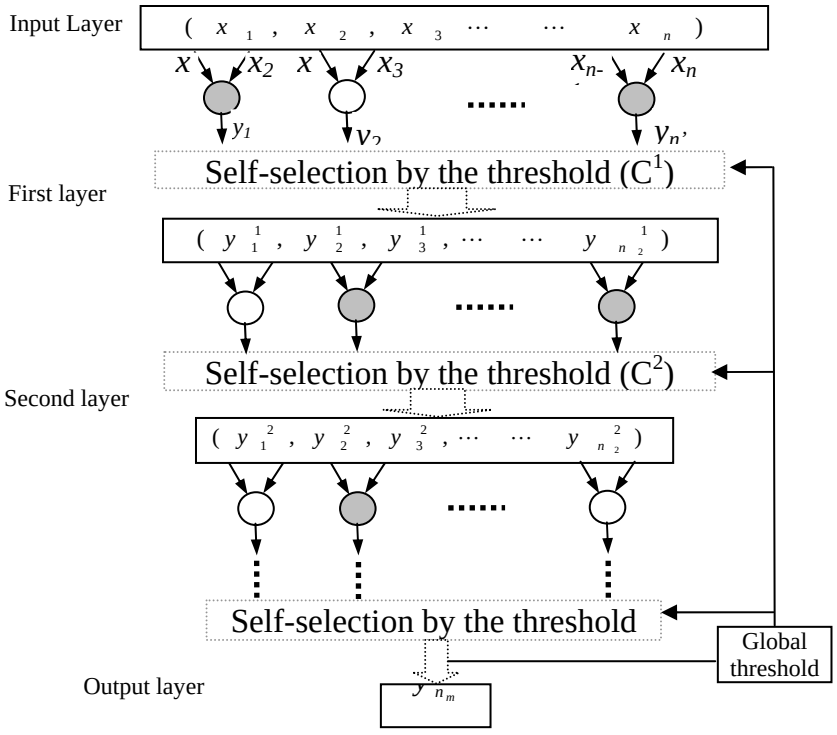


Figure 1. Structure of the GMDH algorithm. The shadowed nodes represent those that have significant contribution to the output and are selected to be input in the next layer.

Step 1 – Select input variables and divide them to pairs as a training set at each layer. As shown in Figure 1, a pair of input variables (x_i, x_j) composes a circle node which is a partial polynomial unit in the form of a quadratic regression polynomial:

$$Y_{ij} = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2, \quad (4)$$

where $\{a_0, a_1, a_2, \ldots, a_5\}$ are the partial polynomial coefficients which have to be estimated for each unit.

Step 2 – Select new variables as input of the next middle layer and truncate the subsequent computation. With the identification of the optimal output of partial polynomials at each layer, the selection of new variables enables the network to quickly converge to the target solution. Once the partial polynomial equations at each unit are selected, the residual error in each layer is further checked to determine whether the set of equations of the model should be further improved within the subsequent computation.

Step 3 – Build the final model and compute the predicted value. The final prediction model can be obtained with selected variables in each layer and the coefficients of partial polynomials between the connected layers.

The GMDH algorithm secures an optimal structure of the model from successive generations of partial polynomials after filtering out those intermediate variables that are insignificant for predicting the correct output. Most improvement of GMDH has focused on the generation of the partial polynomial, the determination of its structure and the selection of intermediate variables. However, every modified GMDH is still a model-driven approximation, which means that the structure of the model has to be determined with the aid of empirical (regression) approaches (Maier and Dandy, 2000). Thus the algorithms could not be said to truly reflect the self-organizing feature that is able to match the relationship between variables completely based on the prior knowledge.

2.2 ANN and Polynomial Neural Network (PNN) algorithms

ANNs are data-driven and learning systems which were originally developed to mimic the structure and function of the brain to simulate nonlinear functions of dynamic systems (Hopfield, 1982). The flexible feature of the ANNs makes them

suitable for modelling complex nonlinear hydrological processes. However, the weakness of the ANNs approach is that the selections of the network structure, the activation functions and criteria are always sensitive to the quality and quantity of input data. Most of the new methods have been made to improve these selections. Polynomial neural network (PNN) based on the GMDH structure and the feedforward training algorithm is one of such methods (Wang and Hu, 1999; Oh and Pedrycz, 2002; Aksyonova, et al., 2003).

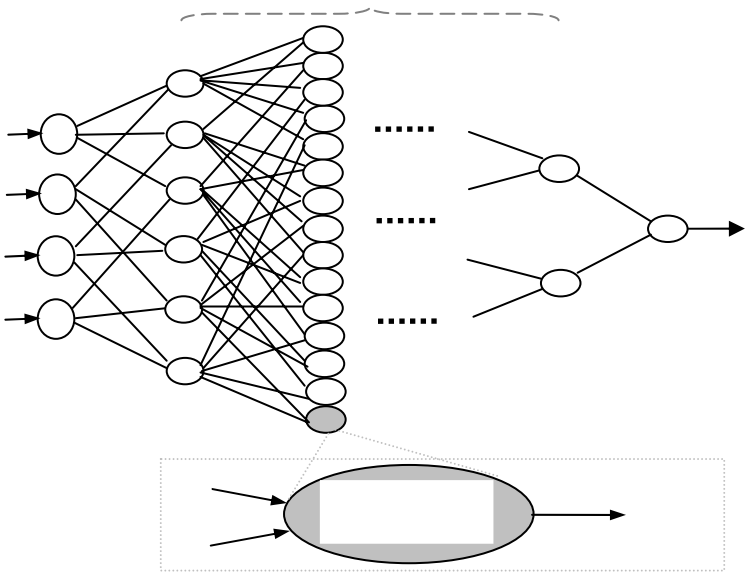


Figure 2: Basic structures of PNN and partial polynomial node.

PNN has a similar architecture to that of feedforward ANN, and consists of a set of middle hidden layers which are composed of a number of partial polynomial nodes. The basic structure of PNN is shown in Figure 2. Between the input layers $X^n = \{x_1, x_2, ..., x_n\}$ and the output layers $Y^m = (y_1, y_2, ... y_m\}$, the connect function and the objective function (typically mean square error) are determined by a training process consisting of three components: *connective weights* between nodes, analogous to neurons, which define the relative contribution of the input; *training laws* (criteria) that determine the adjustment of the weights during the training; and *a transfer function* that can be determined by a number of nodes and the connected weights.

The non-linear mapping feature of the PNN is carried out with both the transfer function and partial polynomial nodes in the training process.

The input–output relationship and training procedure of the PNN algorithm can be briefly described in the following steps:

1) Select input variables $X = \{x_1, x_2, ..., x_n\}$ and divide the available data into training and testing data sets;

2) Choose a pair of variables and determine the structure of partial polynomial (pair of input variables and the order). This is in contrast with traditional ANNs that use single-variable nodes;

3) Calculate the connective weights between the nodes of partial polynomial in the training process;

4) Identify the contributing nodes at each hidden layer and select the new input variables for the next hidden layer;

5) Check the residual error and stopping criteria, and build up the network relationship for prediction.

It is clear what distinguishes PNN from traditional ANNs is its partial polynomial structure in the nodes and the selection of nodes in the training process. By choosing the most significant input variables and corresponding polynomial order, the optimal extracted partial (polynomial) descriptions can be obtained. According to both the selection of nodes at each layer and the generation of hidden layers, the procedure leads to an optimal network structure of PNN.

However, difficulties exist with the PNN's selections of training criteria and network structure of polynomial. They need to be chosen carefully in a controllable region of the judging threshold in order to ensure that most nodes would be sifted and the network covers the expected output. Some improved PNNs have been developed to incorporate special control in the selection of valuable nodes at each layer. For example, Oh and Pedrycz (2002) introduced a battery of structures (types and order) of partial polynomial. Aksyonova et al. (2003) used a twice-hierarchical neural net structure in the iteration procedure without an increase of the power of polynomials and the number of terms. However, these developments still fall short of self-organizing selection of input and intermediate variables; and so the developed

computation processes are difficult to implement. This restricts PNNs' applications in hydrological modelling. In the following section, a new PNN algorithm is developed to improve the training process and network structure.

3. Self-Organizing Polynomial Neural Network (SOPNN) Algorithm

Self-organizing theory has been applied to the ANNs for a decade (Kohonen, 1990). Its self-organizing feature mapping (SOFM) is mostly used in ANNs for data classification (Moradkhani, et al, 2004). In this study, the self-organizing feature is incorporated into the training process for shifting and selecting the terms of partial polynomial in PNN, whereas the method used in SOFM is to group data within competitive patterns. Similar to other PNNs, the structure, criteria and training process are three most important components in the SOPNN algorithm.

3.1 SOPNN structure

The SOPNN uses a coverage network structure as shown in Figure 3. Different from the architecture of PNN, the SOPNN adds a sub-multi-neuron which involves an additional hidden layer that consists of six nodes of the terms of partial polynomial (see Figure 4). Within such a coverage network structure, the SOPNN polynomial combinations of input variables are searched in different levels of hidden layers while all the combinations of input variables are processed at every layer in general PNN. In the SOPNN, the network uses a selection criterion to self-organise neural nodes step by step through a forward training process (from input X^n to the output Y) and an error-back-propagation process (add residues d^n to the input space (X^n, d^n) in the iteration. The optimal SOPNN structure is obtained through systematic recursion and error-back-propagation layer by layer, in conjunction with the self-organising sifting step by step.

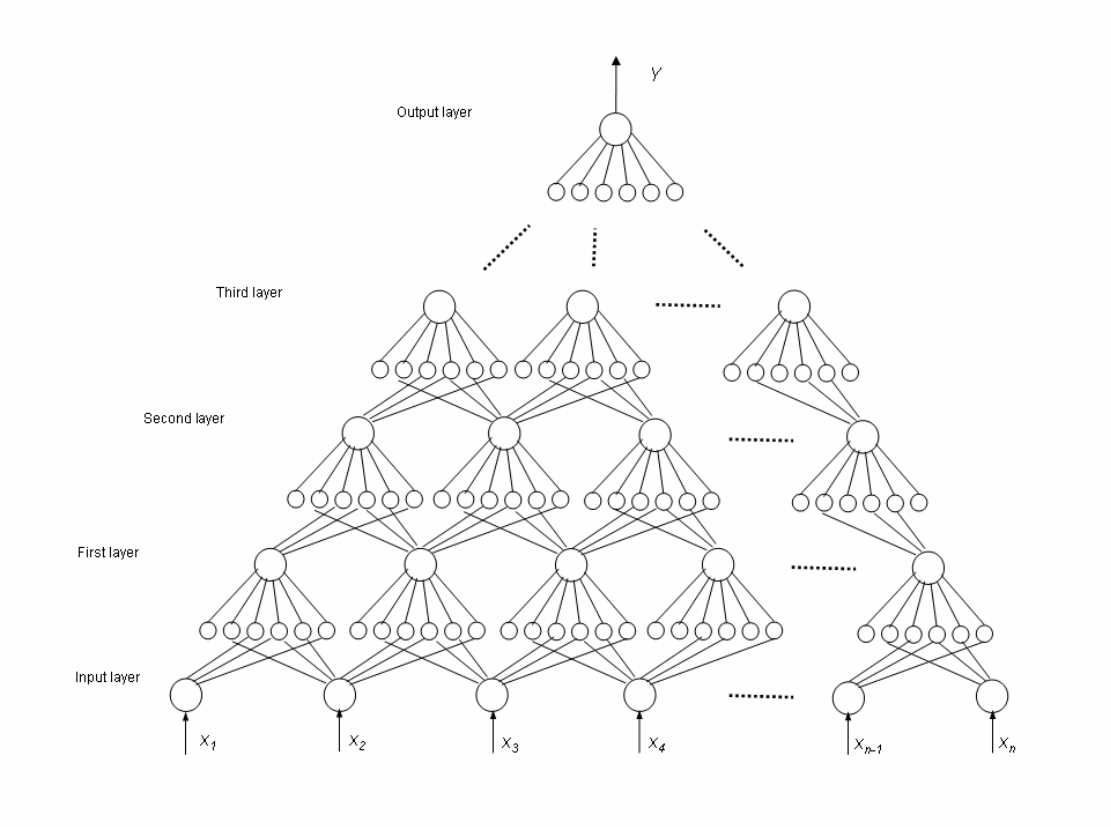


Figure 3: Structure of SOPNN algorithm

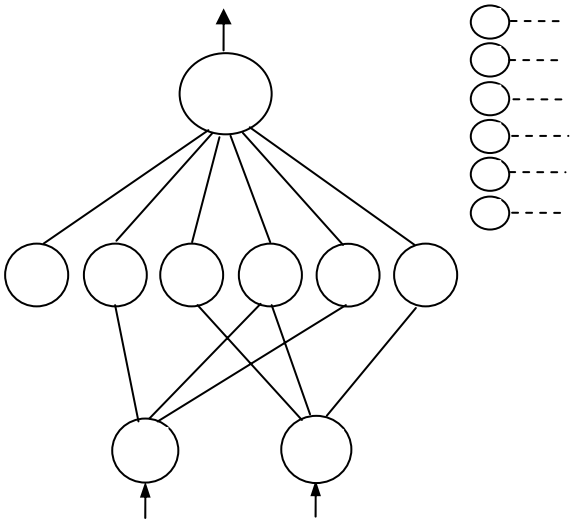


Figure 4: Schematic diagram of sub-hidden layer of a partial polynomial node.

The characteristics of the SOPNN structure are clear: (i) the effect of input variables is fully considered because all the combinations (the terms of polynomial) of

input variables are searched in the different hidden layers; (ii) the network is robust and converged to a final output node as the number of middle nodes are reduced by at least one each layer; and (iii) the computation is accelerated dramatically through self-organising sifting nodes and reducing duplicated computation of partial polynomial terms, e.g. the computation of the term $\{x_i^2\}$ in the case of N input variables is reduced from N-1 times in PNN to 2 times in SOPNN regularly.

3.2 SOPNN selection criteria

Several criteria for determining the structure of the partial polynomial and for selecting the units in the hidden layers have been used in different ANN algorithms. Traditionally, ANNs and PNNs used the *predicting error square sum (*PESS*)* as an objective function in each layer. The PESS is defined as follows:

$$PESS = \frac{1}{m}\sum_{t=1}^{m}[Y(t) - \hat{Y}(t)]^2 , \qquad (5)$$

where *Y(t)* is the expected output, and $\hat{Y}(t)$ is the estimated value of the object at time *t*. Iteratively, to consider the information at each node, SOPNN uses the following modified *PESS* (Okuno, 1976) as the critical function for sub-hidden layer

$$PESS = \frac{1}{m}\sum_{t=1}^{m}[\frac{Y(t) - \hat{Y}(t)}{1 - \bar{x}_t^T \bar{x}_t /(X^T X)}]^2 , \qquad (6)$$

where $\bar{x}_t^T = [a_0, x_i, x_j, x_i x_j, x_i^2, x_j^2]$ $i \neq j$; $t=1,2,\ldots,m$ and $X^T = [\bar{x}_1, \bar{x}_2 \ldots, \bar{x}_m]$.

On the other hand, instead of the *Bayesian Information Criterion* (BIC, Hsu, et al, 1995) used for global error adjustment as in most dynamic modelling, the SOPNN uses *Multiplied Weight Square Sum (MWSS)* as the critical function of global prediction error

$$MWSS = \sum_{t=1}^{m}\sum_{\lambda=1}^{l} w(\lambda) e_t^2(\lambda) \qquad (7)$$

where $w(\lambda)$ is a weighted function of previous λ steps ($\lambda \geq 1$, integer), $e_t^2(\lambda)$ is the predicting error of previous λ steps at time *t*. *MWSS(l, k)* is a calculated value of *MWSS* for partial polynomial *k* at layer *l.*

$$\overline{MWSS}\,(l)=\min MWSS(l,k). \tag{8}$$

When the following condition is met with a certain value $\hat{l}$

$$\frac{\overline{MWSS}(\hat{l}) - \overline{MWSS}(\hat{l}-1)}{\overline{MWSS}(\hat{l})} \le \varepsilon \quad , \tag{9}$$

where the threshold ε is given by one's experience, the sifting procedure will be stopped at layer *l*. This criterion, though equation (8) in search for the local minimal error at each layer, can determine whether minimization of the global error with a preset threshold has been achieved in the whole prediction domain (Tong, et al., 1996).

3.3 Training process

It can be seen from the SOPNN structure that whatever criteria are adopted, the main calculation of the algorithm will focus on estimating the weighted coefficients of partial polynomials, and calculating and propagating the prediction errors of all probable inputs at each layer. Here it is feasible to obtain the estimates of weighted coefficients and predicting error of the partial polynomial simultaneously by using the gradient descent method in the training process. The SOPNN procedure modified from the PNN training process is carried out in the following steps:

Step 1 – Select input variables and data sets, assuming that there are *m* samples and *n* input variable data (x_{il}) ($i=1,\ldots,n;\ l=1,\ldots,m$), which have a sequence like a time series related to output variable in the modelled dynamic system.

Step 2 – Choose a pair of input variables, namely $x_i = \{ x_{i1}, x_{i2}, \ldots, x_{im} \}$, $x_j = \{ x_{j1}, x_{j2}, \ldots, x_{jm} \} (i \neq j)$, *to build a quadratic polynomial node* $y_{ij}^k = (z_0^k, z_1^k, \ldots, z_5^k) = \{x_0, x_i, x_j, x_i x_j, x_i^2, x_j^2\}$ *in the hidden layer k*, where x_0 is a constant node. The structure of a partial polynomial node with a sub-hidden layer is shown in Figure 4.

Step 3 – The weights between input and output units are computed using the gradient decreasing method within the feed-forward training process. The initial

weights (w_q) are given random values between (-1, 1). With the input data and weights, the value $\hat{y}^k$ of output nodes can be calculated using the transfer equation which defines the basic operation in the hidden layer *k*:

$$\hat{y}^k = f_s(z_q^k \bullet w_q - b_q) \quad q = 0, \ldots, 5. \tag{10}$$

where b_q is an adjustable threshold and function $f_s(\bullet)$ is called the activation function which commonly uses the sigmoid function (ASCE, 2000).

The general error of the output layer nodes is computed by

$$d_k = (y^k - \hat{y}^k) f_s'(Y^k). \tag{12}$$

where $f_s'(\bullet)$ is the derivative of $f_s(\bullet)$ and Y^k is the sum of weighted $\hat{y}^k$ in the hidden layer *k*.

Step 4 – *Use error-back-propagation to update the weights and biases.* The weighting coefficient is updated using the following equation:

$$w_q(t+1) = w_q(t) + \alpha \cdot d_k \cdot z_q^k \qquad q=0,\ldots,5 \tag{13}$$

where *a* is an input training factor between (0, 1) for accelerating the convergence rate of network, *t* is the iterative count.

Step 5 – Check the stopping condition and criterion for adjusting unit import: Use equation (7) to calculate *PESS*, if $(PESS_1\text{-}PESS_2)/PESS_1 > \varepsilon_1$, it means that z_l^k improves *PESS* obviously and z_l^k is retained; otherwise z_l^k is rejected. Here, $PESS_1$ and $PESS_2$ are two values of *PESS* calculated before and after importing z_l^k. Note that the acceptance-or-rejection process is not performed in the first training step.

Step 6 – Check the condition for terminating the training. General ANN and PNN use a preset value as a threshold or a maximal time-step to stop the training process. Here, we take equation (8) and (9) as the determination of the stopping condition; if the condition is not satisfied, return to step 1, and replace inputs x_i, x_j with the combinations of Y^k_i, Y^k_j in the subsequent hidden layer *k+1*. Repeat all steps, replace and train each layer until the final layer of the network. Take its structure and matrix of the weighted coefficient as the final predictive model.

From the above the network structure and training procedure, the SOPNN algorithm compared with the GMDH method has at least two advantages: (1) The accuracy of partial polynomial fitting can be improved by adjusting the coefficient continuously after importing the gradient descent algorithm into the GMDH regressive procedure. The *data-driven* training in the SOPNN can find more accurate polynomials than those with coefficients determined only once by *the least mean square* method in the GMDH. (2) The polynomial structure, weights and estimated error of SOPNN can be determined simultaneously, as well as the acceptance or rejection of neural nodes. This is superior to the GMDH which has much higher global errors through multi-step regressions.

The SOPNN holds a converged network structure, unlike most ANNs and PNNs which determine the network structures by *trail and error*. That enhances the stability and robust of the SOPNN implementation. Moreover, the SOPNN algorithm incorporates a self-organizing sifting of the term nodes in the sub-hidden layer whereas the PNN only performs the sifting of partial polynomial nodes in the hidden layers. SOPNN also adopts a double checking of the error criteria to improve the accuracy and efficiency of the training process. Finally, SOPNN's superiority in the determination of variables and the self-organizing construction of network accelerates the calculation especially in multi-variable and highly nonlinear case studies.

4. Applications and comparison with other methods

In order to validate the SOPNN algorithm for hydrological modelling, it is applied to runoff prediction with a data set from a small catchment (Fentie, 2001). The results are compared with those obtained via two other nonlinear modelling approaches, i.e., general GMDH and BPN (Rumelhart, et al, 1986).

The rainfall-runoff data have been examined in the previous studies (Fentie, 2001; Pullar, 2003). The data comprise 202 minutes of observed values of rainfall and runoff at the outlet of a 9.6 ha catchment in the central Queensland, Australia (Figure 5). The rainfall-runoff process in the catchment is highly non-linear, and the variation of runoff during the measurement period is complex due to combined influences of soil infiltration, land cover and landscape environment. Both statistical and

physically-based models (LPIM) have been applied to simulate but failed to replicate the runoff data (Fentie, 2001). In this section, we will attempt to use the SOPNN together with the GMDH and BPN methods to model and predict the runoff in responses to the rainfall.

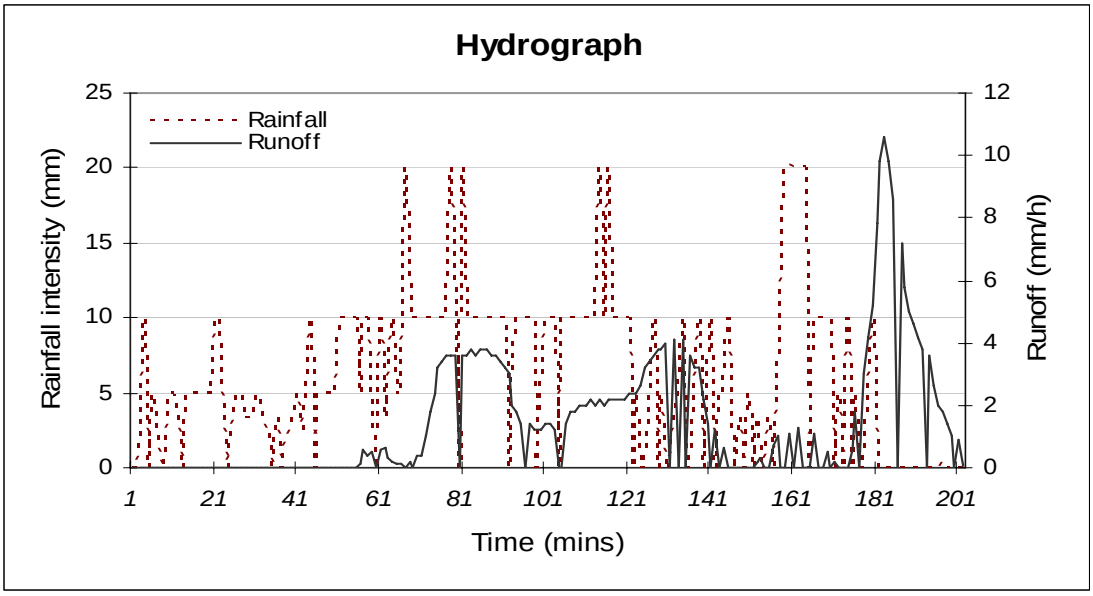


Figure 5: Hydrograph showing the measured rainfall and runoff

4.1 Data preparation

Two time series of 202 minutes (28/02/1988), rainfall record *r(t)* and runoff measurement *o(t)*, are used as the system input data set. Although it was found that a delay of 20 mins between rainfall and runoff existed due to infiltration and other landscape influence at the beginning, the rainfall records and runoff measurements are considered to be suitable for testing the SOPNN method. Since the catchment is very small (9.6 ha), the precipitation induced runoff immediately after the initial infiltration. The flow from upstream areas reached the gauge point in just a few minutes. In general ANN modelling, the selection of input variables is one of the hardest and yet important tasks. Although the SOPNN algorithm can, in theory, match any nonlinear relationship between rainfall and runoff, reasonable input with a suitable time lag can significantly improve the convergence speed as well as the modelling accuracy. Based on the postpositive reality of rainfall-runoff process and a

number of trial training tests, we selected the observation data of rainfall and runoff from previous 4 time steps as input variables, i.e., X = {*r(t-1), r(t-2), r(t-3), r(t-4), o(t-1), o(t-2), o(t-3), o(t-4)*}; and the runoff at time t as expected output, i.e., Y = {$o(t)$}. Also, we chose 188 mins of data as the training sample (i.e., data of the first 192 minutes including the beginning 4 minutes input data) for building the models; and data of the last 10 mins (from 193 minutes to 202 minutes) formed the testing sample (for model verification). The same data sets were used for testing all three different modelling approaches.

4.2 Modelling process

Firstly, with the GMDH approach, the two-direction regressive algorithm (Liu and Wang, 1989) was used for its simplistic computation process. Local error threshold 0.001 and global error threshold 0.01 were used to obtain a stable modelling solution. Secondly, a traditional ANN, Back-Propagation Network (BPN), is used for comparison with the SOPNN approach. In the BPN, a three-layer network structure was selected, and 12 nodes in the hidden layer and the training factor a = 0.55 were employed in order to obtain an optimal solution using the *trial-and error* approach. Finally, in the SOPNN implementation, similar thresholds to those in the GMDH and the same training factor (a=0.55) as that used in the BPN were utilised in the training process to achieve comparable modelling results. The normalisation of original data was performed in all above modelling processes.

4.3 Comparisons with the existing ANN models

With implementation of the above three nonlinear algorithms, the modelling and prediction results were obtained as shown in Figure 6. All three approaches gave close approximations of the actual observations, suggesting that these approaches are applicable for modelling complex hydrological behaviours such as the rainfall-runoff relationship modelled here. However, the SOPNN algorithm yielded slightly better results (Figure 6(c)) than those generated by the other two algorithms (Figure 6(a) & (b)).

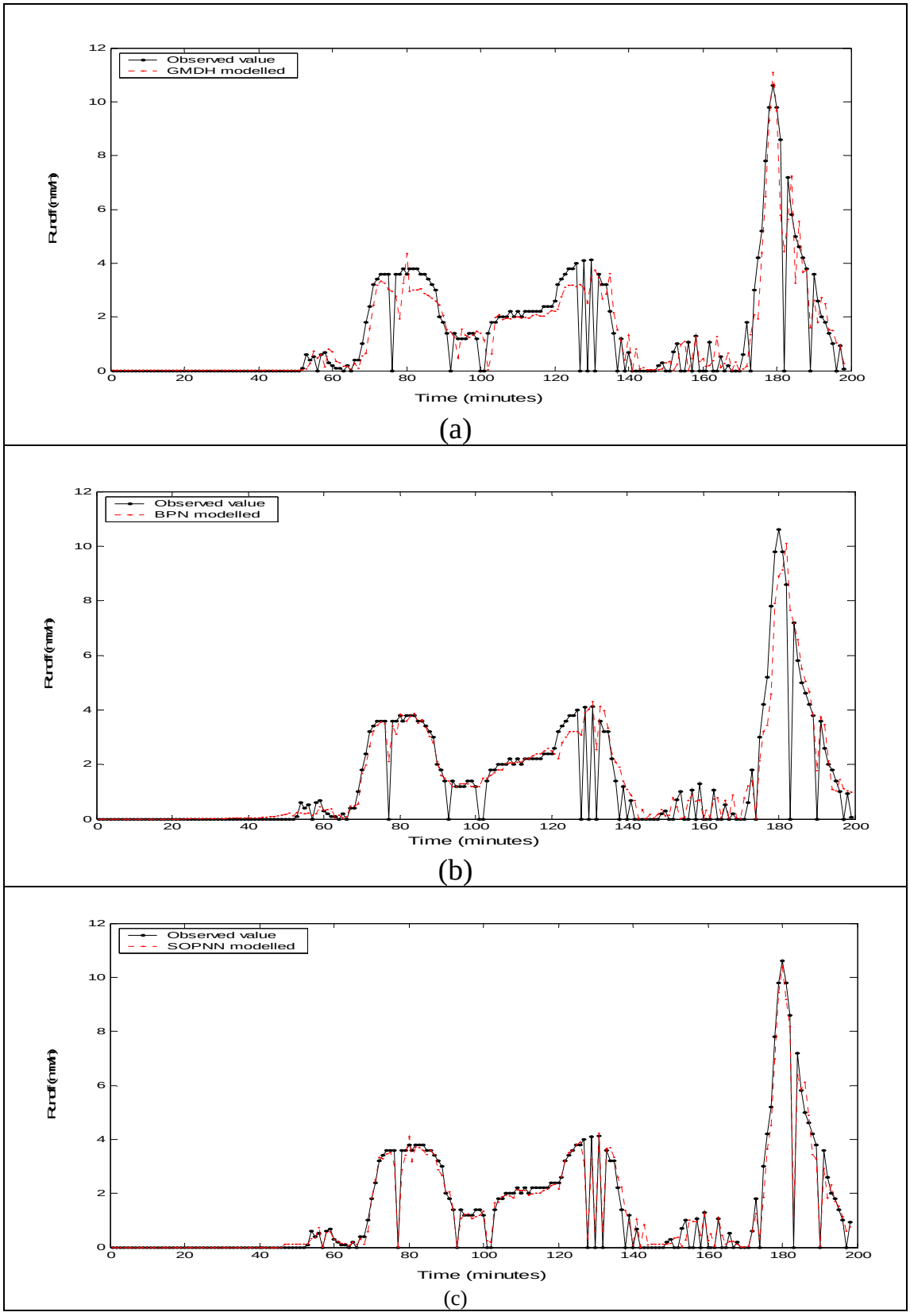


Figure 6: Measured discharge and modelled/predicted runoffs.

Table 1. Comparison of modelling/prediction precision among the three algorithms.

Algorithms		GMDH	BPN	SOPNN
Modelling	RMSE	0.8295	0.8783	0.3364
	R^2	0.9631	0.9588	0.9932
Prediction	RMSE	0.1796	0.1742	0.0959
	R^2	0.9031	0.9177	0.9808
Training times		1	35000	2000

Table 1 shows the comparison of modelling/prediction precision among the three different approaches based on two statistical measures. The RMSE, namely the *root mean squared error*, represents the deviations of modelling results from the measurement and has already been considered in the training process of each algorithm. The *correlation coefficient* R^2 between the observed values and modelled/predicted values is given by:

$$R^2 = \frac{n\sum_t Y(t)\hat{Y}(t) - \sum_t Y(t)\sum_t \hat{Y}(t)}{\sqrt{n\sum_t Y(t)^2 - \left(\sum_t Y(t)\right)^2}\sqrt{n\sum_t \hat{Y}(t)^2 - \left(\sum_t \hat{Y}(t)\right)^2}}, \qquad (14)$$

where n is the number of samples, and $Y(t)$ is the expected output, while $\hat{Y}(t)$ is the modelled/predicted output at times t.

In Table 1, the highest R^2 and lowest RMSE are found with the SOPNN modelling/testing results, demonstrating that the SOPNN provides a better approach for modelling the rainfall-runoff process. It should also be noted that the sequence of input variables affects the accuracy of the two-direction regressive GMDH algorithm tested, but does not affect the BPN and SOPNN algorithms because of their data-driven approach and free training process.

Figure 7 shows the comparison of scatter plots between modelled/tested results by the three algorithms and measured runoffs. From the figures, it is evident that the

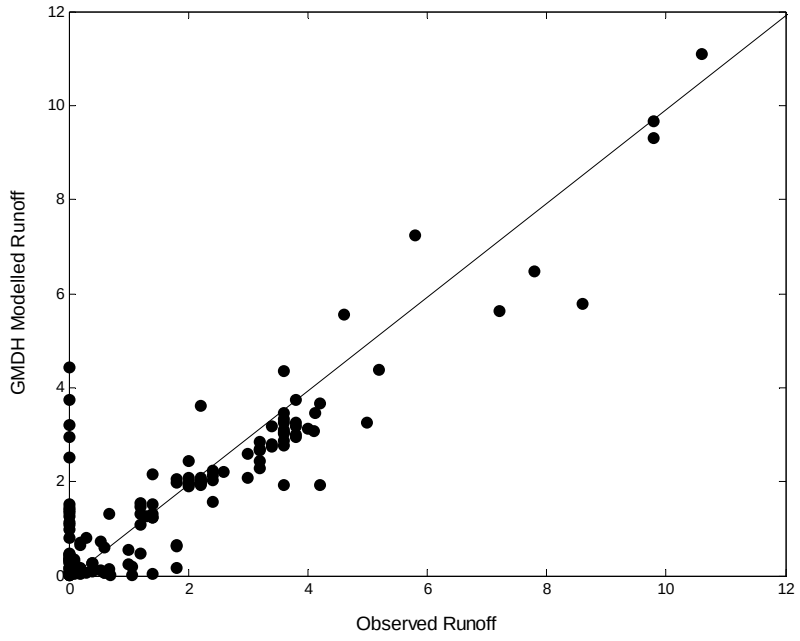
SOPNN performed better than the BPN and GMDH algorithms in simulating high and low values of runoff (peak and null). Although the GMDH algorithm can sift useless variables and the BPN hold strong capability of evacuating the noise through the error-backward in training process, they are short of fully considering the remove of both data noise and unnecessary variables in the modelling process. However, SOPNN does not have eh aforementioned shortage of GMDH and BPN models. It can be concluded from Figure 7 that the SOPNN has a better capability of removing data noise and useless variables in the training processes.

Figure 8 illustrates the comparison of errors between generated and observed values in the testing data domain. Similar to the performance in the modelling process, the modelled error line generated from SOPNN is much closer to the zero-value-line and flatter than the error lines from GMDH and BPN. The same results are also shown in Figure 7(c) (with the results close to the 1:1 line) and Table 1.

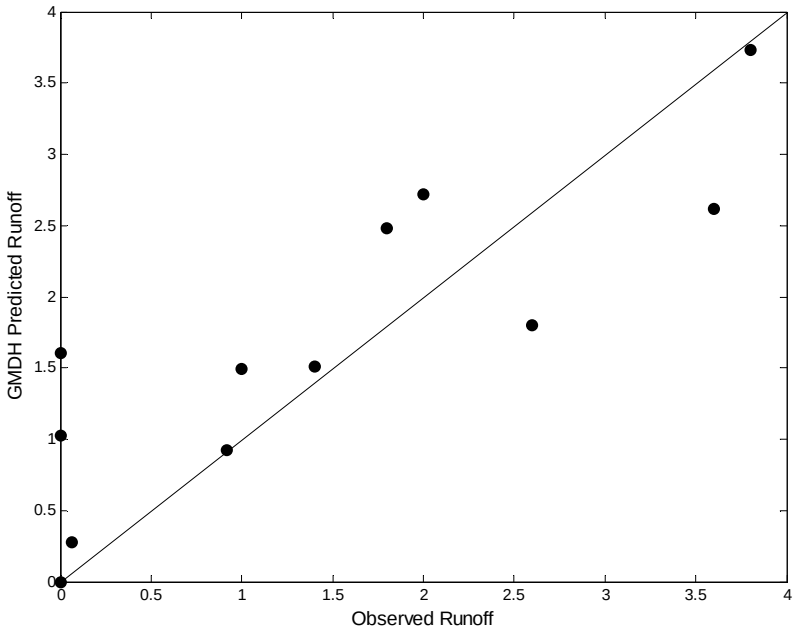
5. Concluding remarks

The self-organizing polynomial neural network (SOPNN) algorithm developed here combines the self-organizing polynomial (SOP) theory to overcome the drawback of the artificial selection of network structure, with the ANN mapping feature to seek optimal solutions for determining the coefficients of partial polynomial at each layer. The combination leads to SOPNN's capability of nonlinear modelling of complex hydrological systems. The algorithm is particularly suitable for systems involving uncertain parameter predictions because it can remove noise and sifts irrelevant information in the training process.

The case study on rainfall-runoff modelling and testing demonstrated that the fitting accuracy of SOPNN is superior to those of the GMDH and BPN. Generally, the SOPNN algorithm is robust in the modelling of complex nonlinear structures, but its superiority may be subtle in simple structure models. If combined with a parallel algorithm, the SOPNN would demonstrate the merits in functional fitting and real time predictions of complex hydrological behaviours.

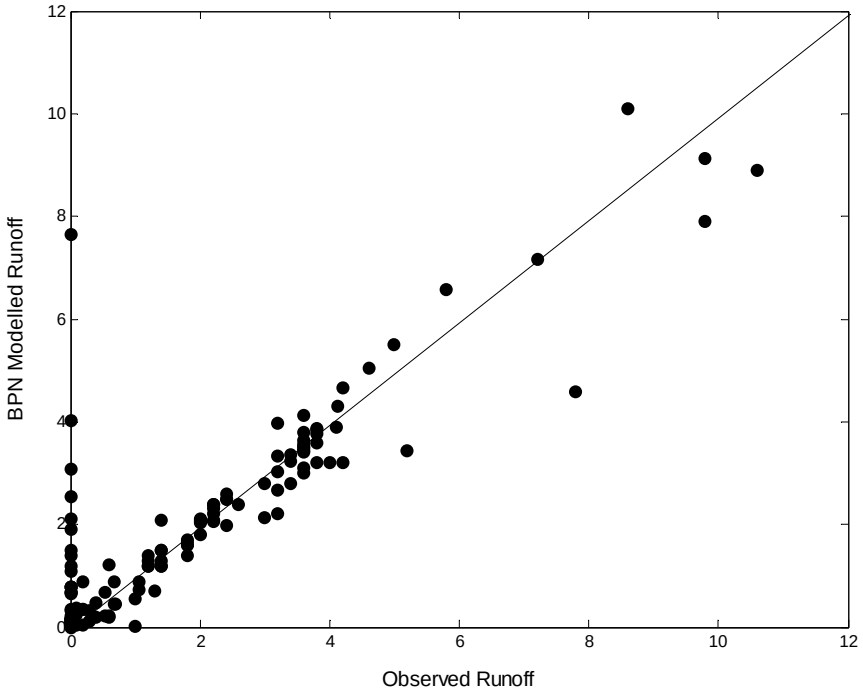


(a1)

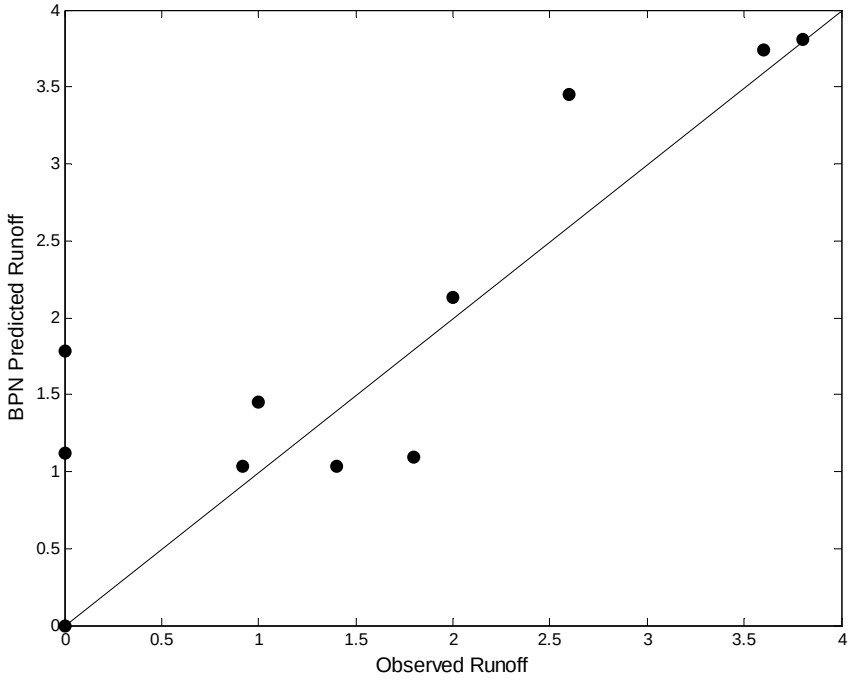


(a2)

Figure 7: Observed discharge versus the estimated values. The scatter plots (a1), (b1) and (c1) show the modelled runoff against the observed runoff, whereas the scatter plots (a2), (b2) and (c2) compare the tested runoff against the observed runoff. (to be continued)

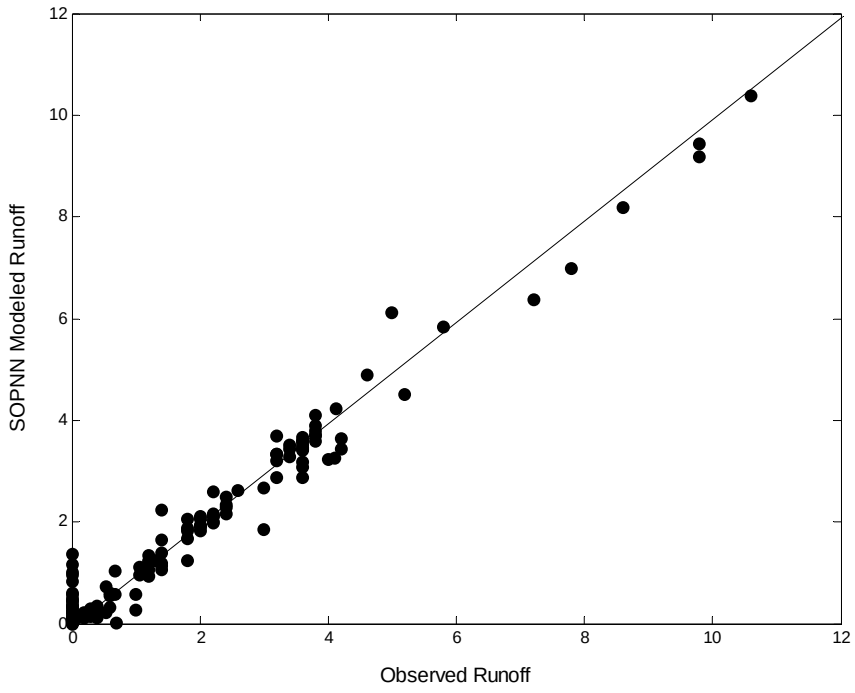


(b1)

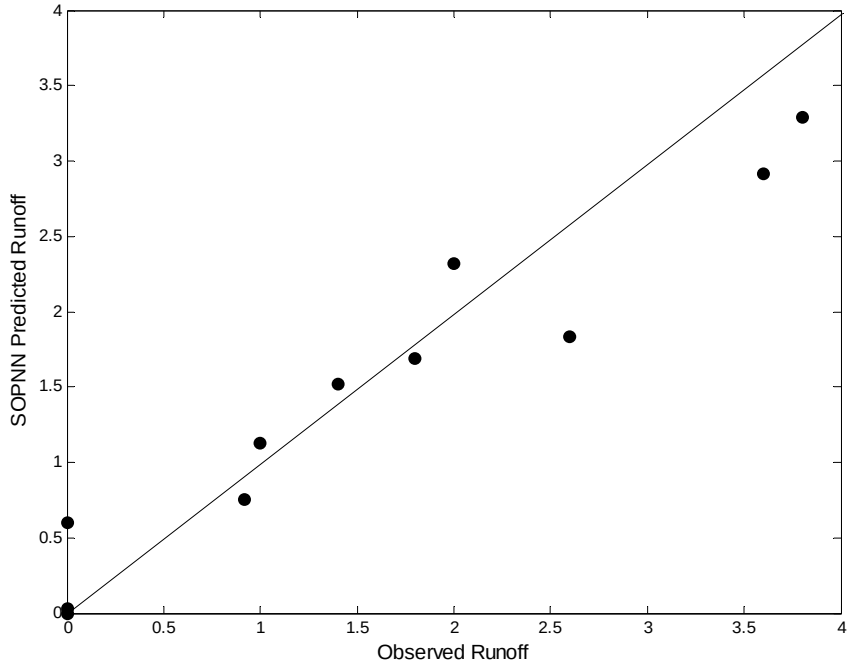


(b2)

Figure 7: Observed discharge versus the estimated values. The scatter plots (a1), (b1) and (c1) show the modelled runoff against the observed runoff, whereas the scatter plots (a2), (b2) and (c2) compare the tested runoff against the observed runoff. (to be continued)



(c1)



(c2)

Figure 7: Observed discharge versus the estimated values. The scatter plots (a1), (b1) and (c1) show the modelled runoff against the observed runoff, whereas the scatter plots (a2), (b2) and (c2) compare the tested runoff against the observed runoff.

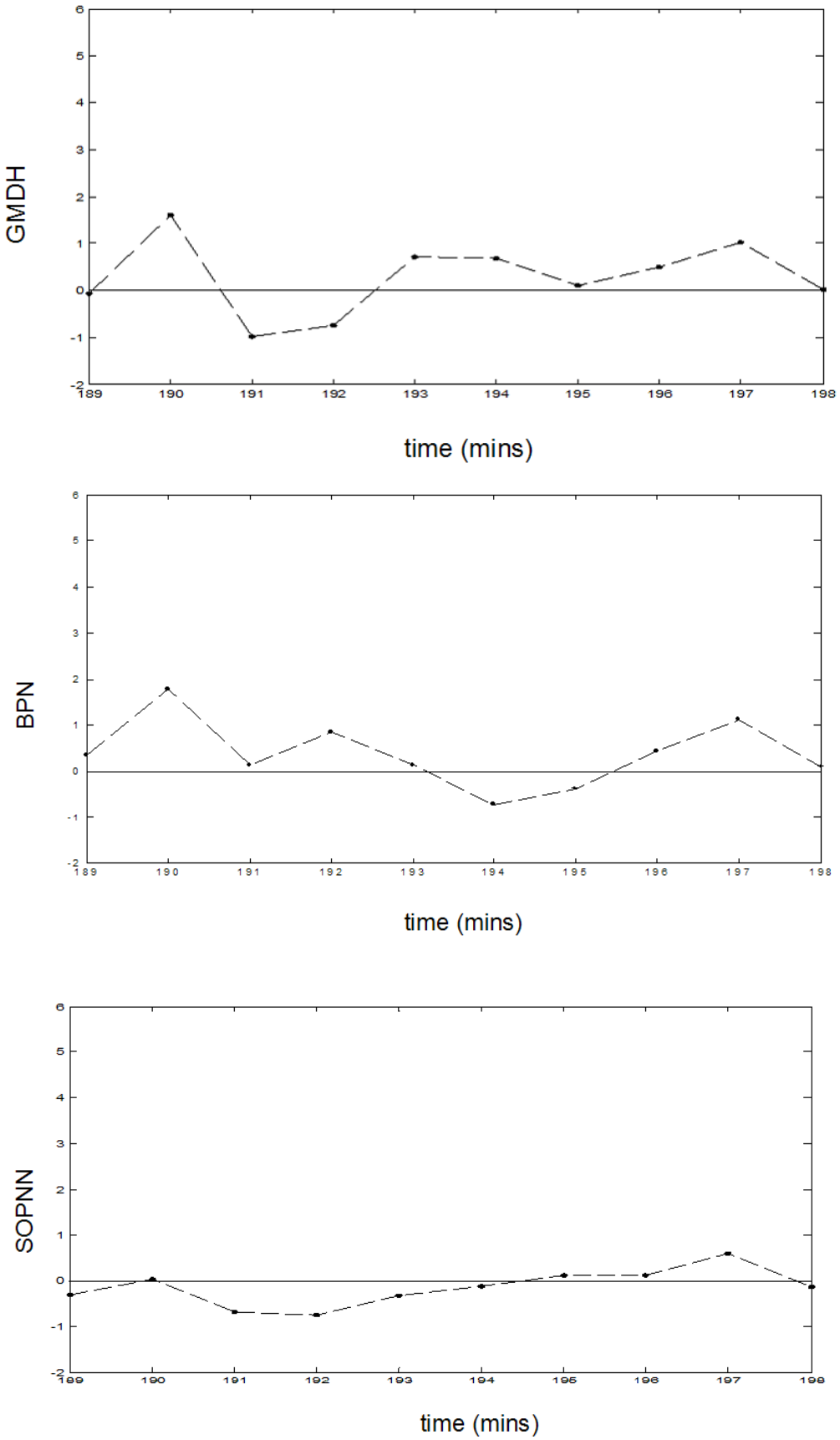


Figure 8: Comparison of the errors of predictions (differences from the observed and tested values).

References

Aksyonova, T., Volkovich, V.V. and Teiko, I.V., (2003), Robust polynomial neural network in quantitative-structure activity relationship studies, *Systems Analysis Modelling Simulation*, 43(10), 1331-1339.

ASCE Task Committee on the application of ANNs in hydrology, (2000a), Artificial Neural Networks in Hydrology: Preliminary Concepts, *Journal of Hydrologic Engineering, ASCE*, 5(2), 115-123.

ASCE Task Committee on the application of ANNs in hydrology, (2000b), Artificial Neural Networks in Hydrology: Hydrologic Applications, *Journal of Hydrologic Engineering, ASCE*, 5(2), 124-137.

Chang, F. J. and Hwang, Y.Y. (1999), A self-organization algorithm for real-time flood forecast, *Hydrological processes*, 13, 123-138.

Coulibaly, P., Anctil, F., Aravena, R. and Bobee, B., (2001), Artificial Neural Networks modelling of water table depth fluctuations, *Water Resources Research*, 37(4), 885-896.

Farlow, S.J. (1984), *Self-organizing Methods in Modeling: GMDH Type Algorithms*, Marcel Dekker, New York.

Fentie, B., (2001), *Catchment-scale Water Erosion and Deposition Modelling: A Physically-based Approach*, PhD thesis, University of Queensland.

Hsu, K.L., Gupta, H.V. and Sorooshian, S., (1995), Artificial Neural Networks modelling of rainfall-runoff process, *Water Resources Research*, 31(10), 2517-2530.

Hsu, K.L., Gupta, H.V. Geo, X., Sorooshian, S. and Imam, B., (2002), SOLO-An Artificial Neural Networks suitable for hydrologic modelling and analysis, *Water Resources Research*, 38(12), 1-38.

Hopfield, J.J. (1982), Neural Networks and physical systems with emergent collective computational abilities, *Proc. Nat. Academy of Scientists*, 79, 2554-2558.

Imrie, C.E., Durucan, S. and Korre, A., (2000), River flow using Artificial Neural Networks: generalization beyond the calibration range, *Journal of Hydrology*, 233, 138-153.

Islam, S. and Kothart, R, (2000), Artificial Neural Networks in Remote Sensing of Hydrologic Processes, *Journal of Hydrologic Engineering*, 5(2), 138-144.

Ivakheneko A G,. (1971), Polynomial Theory of Complex System, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-1(4): 364-378.

Ivakheneko A G. and Ivakheneko, G.A. (1995), The review of problems solvable by algorithms of the GMDH, *Pattern Recognition and Image Analysis*, 5(4): 527-535.

Kohonen, T., (1990), The self-organizing map, Proc. IEEE, 78, 1464-1480.

Krotov, G. I., and Kozubovskiy, S. F. (1987), Verification of dendroscale forecasting by a multiplicative GMDH algorithm, *Sov. J. Autom Inf. Sci*., 20, 1-7.

Kumar, A.R., Sudheer, K.P., Jain, S.K. and Agarwal, P.K., (2005), Rainfall-runoff modelling using articial neural networks: comparison of network types, *Hydrological processes*, 19, 1277-1291.

Liu, D. and Wang X.F., (1989), Two-direction iterative regression algorithm of GMDH, *Information and Control*, 5, 47-51.

Mahmoud, M. R. and Mahound, A. H. M. (2004), Modelling artificial neural networks for forecasting monthly Nile River natural flow. *Journal of Engineering and Applied Science*, 51(6), 1119-1134

Mamedov, M. I., and Ivankhnenko, N. A. (1987), Forecasting model of air pollution of an industrial city, *Sov. J. Autom. Inf. Sci.*, 20, 90-92.

Maier, H.R. and Dandy, G.C., (2000), Neural Networks for the production and forecasting of water resource variables: a review and modelling issues and application, *Environmental modelling and software*, 15, 101-124.

Moller, A F, A Scaled, (1993),.Conjugate Gradient Algorithm for Fast Supervised, Learning, *Neural Networks*, 6, 525-533.

Moradkhani, H, Hsu, K.L. Gupta, H.V. and Sorooshian, S, (2004), Improved streamflow forecasting using self-organizing radial basis function Artificial Neural Networks, *Journal of Hydrology*, 295, 246-262.

Nalimov V.C. and Chernova N. A., (1965), *Statistical methods of planning the extremum experiments*, Moscow.

Oh S.K. and Pedrycz, W., (2002), The design of self-organizing Polynomial Neural Networks, *Information Sciences*, 141, 237–258.

Pullar, D, (2003), Simulation modelling applied to runoff modelling using MapScript, *Transactions in GIS*, 7, 267-283.

Rumelhart, D.E., Hinton, G.E., and Williams, R. J., (1986), Learning internal representations by error propagation, Parallel distributed processing, 1, MIT Press, Cambridge, Mass., 318-362

Tamura, H., and Kondo, T. (1980), Heuristics free group method of data handling algorithm of generating optimal partial polynomials with application to air pollution prediction, *Int. J. Syst. Sci*., 11, 1095-1111.

Thirumalaiah, K. and Deo, M.C., (2000), Hydrological Forecasting Using Neural Networks, *Journal of Hydrologic Engineering*, 5(2), 181-189.

Thirumalaiah, K. and Deo, M.C., (1998a), River Stage Forecasting using Artificial Neural Networks, *Journal of Hydrologic Engineering*, 3(1), 26-32.

Thirumalaiah, K. and Deo M.C. (1998b), Real-Time Flood Forecasting Using Neural Networks, *Computer–Aided Civil and Infrastructure Engineering*, 13(2), 101-111

Tokar, A.S. and Markus, M., (2000), Precipitation-Runoff Modelling Using Artificial Neural Networks and Conceptual Models, *Journal of Hydrologic Engineering*, 5(2), 156-161.

Tong, X.H., Kuang, J.C., Wang X.Y. and Qi, T.X., (1996), Setting up prediction model of gas well prediction rate by various methods. *Natural Gas Industry*, 16 (6), 49-53.

Voss M.S. and Feng, X., (2002), Emergent system identification using particle swarm optimization, in *Complex Adaptive Structures* (ed: Hutchinson,I., Florida).

Wang X.Y. and Hu, W.Y., (1999), The self-organizing polynomial network algorithm based on the self-organizing theory. *System Engineering --- Theory & Practice*, 19(4), 51-56.

Wu Jr. S., Han J, Annambhotla S. and Bryant, S., (2005), Artificial Neural Networks for Forecasting Watershed Runoff and Stream Flows, *Journal of Hydrologic Engineering*, 5(2), 216-222.

Zealand, C.M., Burn, D.H. and Simonovic, S.P., (1999), Short term stream flow forecasting using Artificial Neural Networks, *Journal of Hydrology*, 214, 32-48.