Application of Recurrent Neural Networks to Rainfall-runoff Processes

Tsung-yi Pan, Ru-yih Wang, Jihn-sung Lai and Hwa-lung Yu
National Taiwan University
Taiwan

1. Introduction

Knowledge of the hydrological process is essential to the watershed and flood management. Due to the complexity of the interactions among the hydrological process, hydormeterological and geomorphological processes, a rigorous dynamic system model is required for the modelling purpose. Among them, the rainfall-runoff modelling is always considered as one of the most challenging part of hydrological process modelling. It has been shown in a variety of research fields that the application of recurrent neural network (RNN) can perform superior in dynamic system modelling (Pan and Wang, 2005). However, Maier and Dandy (2000) reviewed 43 hydrology journal articles with modelling of artificial neural networks (ANNs) published before 1998, where only Chow and Cho (1997) applied RNNs to forecast rainfall.

The application of RNNs to hydrological modelling is rapidly growing these years. Published between 2000 and 2008 spring, 14 papers in which RNNs have been used for simulation or forecasting of water resources variables are reviewed in terms of the modelling process. Due to the rapid increase in journals, it is unlikely that complete coverage has been achieved. Following the form of Maier and Dandy (2000), the major features of the models investigated are summarised in Tables 1 and 2, including background information (variable modelled, location of application, model time step, and forecast length), information about the data used (data type, normalization range, number of training samples, and number of testing samples), information about network architecture (connection type, method used to obtain optimal network geometry, and number of nodes per layer), information about the optimization algorithm used (optimization method, internal network parameters (hidden layer transfer function, learning rate, momentum value, epoch size, and initial weight distribution range)) and the stopping criterion adopted.

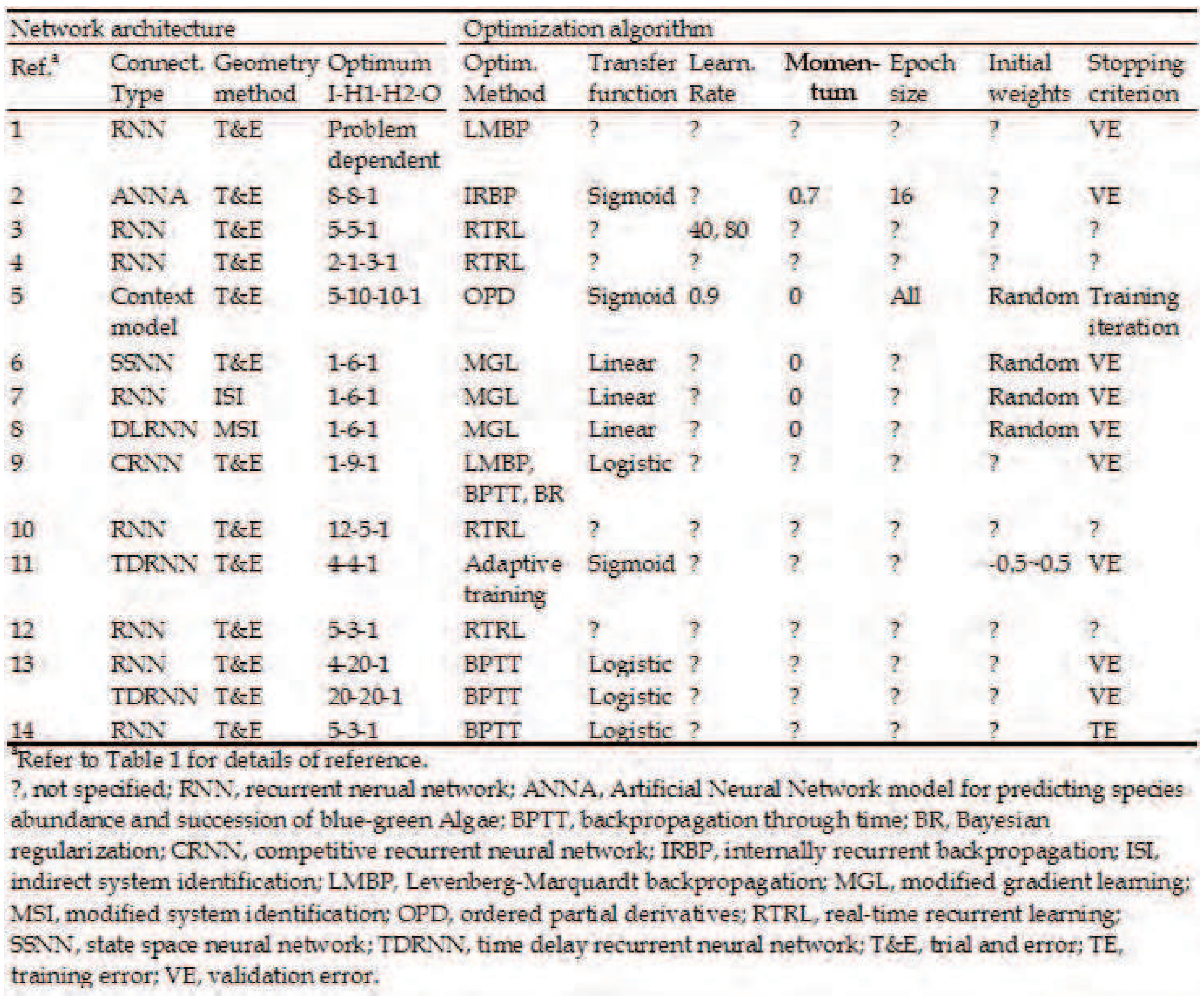
While hydrologists have not made an effort to construe the knowledge embedded in the trained RNN models, the recent studies strive to interpret physical significance from the internal architecture of RNN hydrological models, like Pan et al. (2004, 2005, and 2007). Therefore, this chapter will introduce the deterministic linearized recurrent neural network (denoted as DLRNN) and its application to rainfall-runoff processes.

Source: Recurrent Neural Networks, Book edited by: Xiaolin Hu and P. Balasubramaniam, ISBN 978-953-7619-08-4, pp. 400, September 2008, I-Tech, Vienna, Austria

Background information					Data				
Ref.	Author(s) and year	Input variable	Output variable	Location(s)	Time step	Forecast length	Data type	Normali-zation range	No
1	Coulibaly et al., 2000	Monthly mean flow, climatic indices	Annual flow	North-eastern Canada	Year	+1~11	Real	?	36 y
2	Walter et al., 2001	Phosphorus, ammonia, nitrogen, water temperature, solar radiation, volume, area, water depth	Chlorophyll-a	Burrinjuck Reservoir (Australia)	Day	+7	Real	0~1	15 y
3	Chang et at., 2004	Precipitation, flow	Flow	Da-chia River (Taiwan)	Hour	+1~2	Real	?	1 y
4	Chiang et al., 2007b	Precipitation	Flow	Keelung River (Taiwan)	Hour	+1	Real	?	13
5	Nagsh Kumar et al., 2004	Flow	Flow	India	Month	+1~12	Real/Synthetic	0~1	50
6	Pan and Wang, 2004	Precipitation	Flow	Keelung River (Taiwan)	Hour	+1~3	Real	0~0.9	10
7	Pan and Wang, 2005	Precipitation	Flow	Keelung River (Taiwan)	Hour	0	Real	0~0.9	10
8	Pan et al., 2007	Precipitation	Flow	Keelung River (Taiwan)	Hour	0	Real	0~0.9	10
9	Coulibaly & Baldwin, 2005	Flow, lake volume	Flow, lake volume	Nile River (Egypt), Saint-lawrence River (British), Great Salt Lake (U.S.A)	Month	+12	Real	?	199 GS Nil
10	Chiang et al., 2007a	Precipitation	Precipitation	Keelung River (Taiwan)	Hour	+1	Real	?	4 ty
11	Karamouz et al.,	Climate signals	Precipitation	Iran	Season	+2	Real	-1~1	22
12	Chiang et al., 2004	Flow	Precipitation, flow	Lan-yang River (Taiwan)	Hour	+1	Real	?	15
13	Coulibaly and Evora, 2007	Precipitation, max and min temperature	Precipitation, max and min temperature	Gatineau river basin (Canada)	Day	0	Real	?	366 pre 428 ten 268 ten
14	Coulibaly et al., 2001	Water table depth, precipitation, mouboun water level, max temperature, mean	Water table depth	Northwestern Burkina Faso (West Africa)	Month	+1	Real	?	7 y

?, no specified ; SLR, Saint-lawrence River; GSL, Great Salt Lake; NeuralSolutions v4 (NeuroDimension Inc., Gainesville, FL); MATLAB (The Mathworks

Table 1. Details of papers reviewed (background information and data).



Network architecture				Optimization algorithm						
Ref.[a]	Connect. Type	Geometry method	Optimum I-H1-H2-O	Optim. Method	Transfer function	Learn. Rate	Momen-tum	Epoch size	Initial weights	Stopping criterion
1	RNN	T&E	Problem dependent	LMBP	?	?	?	?	?	VE
2	ANNA	T&E	8-8-1	IRBP	Sigmoid	?	0.7	16	?	VE
3	RNN	T&E	5-5-1	RTRL	?	40, 80	?	?	?	?
4	RNN	T&E	2-1-3-1	RTRL	?	?	?	?	?	?
5	Context model	T&E	5-10-10-1	OPD	Sigmoid	0.9	0	All	Random	Training iteration
6	SSNN	T&E	1-6-1	MGL	Linear	?	0	?	Random	VE
7	RNN	ISI	1-6-1	MGL	Linear	?	0	?	Random	VE
8	DLRNN	MSI	1-6-1	MGL	Linear	?	0	?	Random	VE
9	CRNN	T&E	1-9-1	LMBP, BPTT, BR	Logistic	?	?	?	?	VE
10	RNN	T&E	12-5-1	RTRL	?	?	?	?	?	?
11	TDRNN	T&E	4-4-1	Adaptive training	Sigmoid	?	?	?	-0.5~0.5	VE
12	RNN	T&E	5-3-1	RTRL	?	?	?	?	?	?
13	RNN	T&E	4-20-1	BPTT	Logistic	?	?	?	?	VE
	TDRNN	T&E	20-20-1	BPTT	Logistic	?	?	?	?	VE
14	RNN	T&E	5-3-1	BPTT	Logistic	?	?	?	?	TE

[a]Refer to Table 1 for details of reference.
?, not specified; RNN, recurrent nerual network; ANNA, Artificial Neural Network model for predicting species abundance and succession of blue-green Algae; BPTT, backpropagation through time; BR, Bayesian regularization; CRNN, competitive recurrent neural network; IRBP, internally recurrent backpropagation; ISI, indirect system identification; LMBP, Levenberg-Marquardt backpropagation; MGL, modified gradient learning; MSI, modified system identification; OPD, ordered partial derivatives; RTRL, real-time recurrent learning; SSNN, state space neural network; TDRNN, time delay recurrent neural network; T&E, trial and error; TE, training error; VE, validation error.

Table 2. Details of papers reviewed (RNN architecture and optimization).

2. Deterministic linearized recurrent neural network

The RNN introduced in this chapter is to integrate a state space form into the neural network framework. The integration can provide not only the flexibility to represent any nonlinear functions but also the parallel inputs/outputs (causes/effects) relationships established between the neural model and the physical system (Pan & Wang, 2004). The presented RNN has five layers: input layer, hidden layer S, state layer, hidden layer O, and output layer. The input layer takes the input signals and delivers these inputs to every neuron in the next layer, hidden layer S, which represents any function that specifies the behaviour of states. State layer receives the signals from hidden layer S, and each neuron in this layer represents one state whose output value is the value of the state. After hidden layer O, which represents the features that relates the outputs of the neural network to the states, gets the signals from state layer, output layer takes the hidden layer O signals adds them to each output neuron. These outputs are, finally, the outputs of the RNN embedded in a state space form as Fig. 1.
The mathematical representation of a deterministic non-linear system in state space form is:

$$x_{k+1} = F(x_k, u_k) \tag{1}$$

$$y_k = G(x_k) \tag{2}$$

where u_k, y_k, and x_k with m, l, and n ranks denote, respectively, the input, output, and state vectors at time k. $F: n \times m \to n$ and $G: n \to l$ are two static linear/nonlinear mappings.

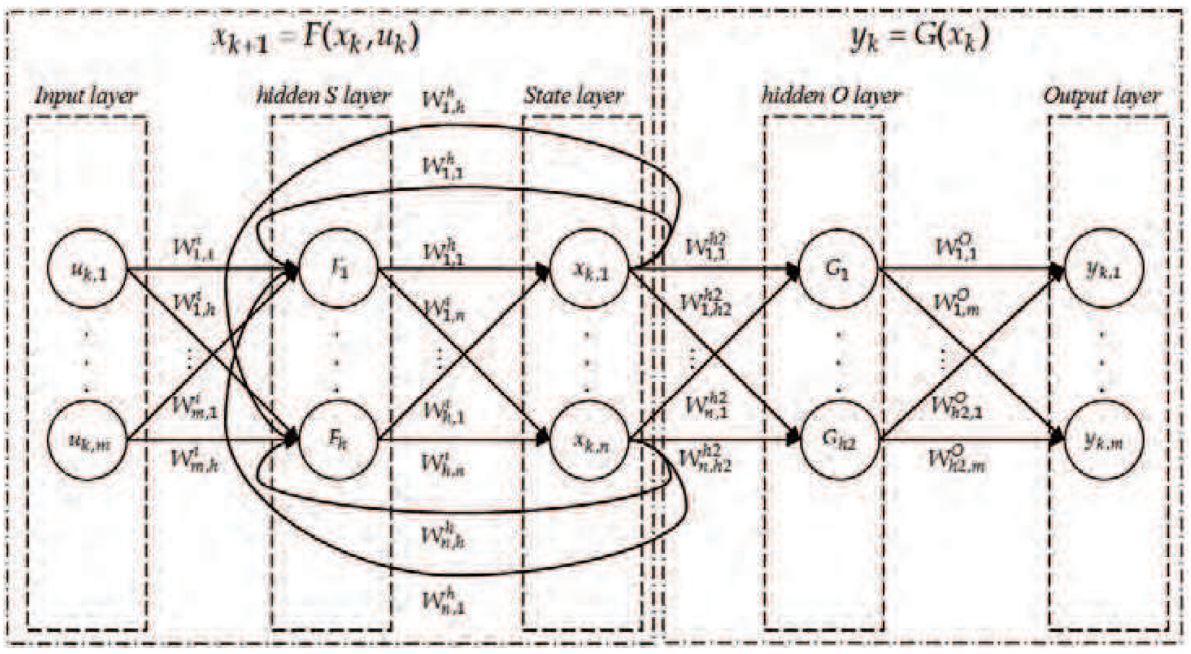


Fig. 1. The RNN embedded in a state space form.

A neural network containing a single hidden layer with bounded transfer functions in its neurons can be used for the representation of a variety of linear/nonlinear functions (Zarmarreño et al., 2000). Therefore, to apply the neural network for the linear/nonlinear mappings in Eqs. (1) and (2), the mathematical form of this special RNN can be written as:

$$\hat{x}_{k+1} = W^h \cdot f_1\left(W^r \cdot x_k + W^i \cdot u_k + B^h\right) \tag{3}$$

$$\hat{y}_k = W^o \cdot f_2\left(W^{h2} \cdot x_k + B^{h2}\right) \tag{4}$$

where W^h, W^r, W^i, W^o, and W^{h2} are matrices with dimensions $n \times h$, $h \times n$, $h \times m$, $m \times h2$, and $h2 \times n$ as the weights of the RNN, respectively. B^h and B^{h2} are two vectors with h and $h2$ elements as biases. f_1 and f_2 are linear/nonlinear functions depending on the behaviour of the system.

Previous works have established that linearized neural networks suffice to capture nonlinear systems. Botto and Costa (1998) designed a linear predictive control using a linearized neural network model. Henriques and Kuanyi (1998) stated that control design for linear systems has been well developed, and it is natural to make use of it in nonlinear plants. Hence, they applied as a linearized neural model. Furthermore, Rahman and Kuanyi (2000) studied a neural network method to linearizing control of nonlinear process plants, and used neural networks to model the process plant and to linearize the neural network model in a novel way. Additionally, the difference between a RNN and a linearized one is the linearity of the active function of each neuron in the hidden layer. In fact, however, it is not strictly necessary that a neural interpretation of the neuron contains a non-linear

function because the reduction of the diversity of activation functions, such as the sigmoid function, is beneficial (Ptitchkin, 2001). Although neural networks are known to be universal function approximators, except for unchanged the active functions, the weights and structure of the neural network are updated or modified during the entire approximating process. Moreover, a high-dimensional space nonlinearity problem can be suitably approximated by modifying the weights in the linear combinations of state variables with time. Consequently, the linear transfer function of the RNN applied herein is capable of simulating nonlinear rainfall-runoff process.
Considering the transfer functions of the RNN applied herein are set as linear functions and the biases are set at zero. Consequently, the Eqs. (3) and (4) are rewritten as:

$$\begin{aligned}\hat{x}_{k+1} &= W^h \cdot \left(W^r \cdot x_k + W^i \cdot u_k\right) \\ &= \left(W^h \cdot W^r\right)\cdot x_k + \left(W^h \cdot W^i\right)\cdot u_k = W_1 \cdot x_k + W_2 \cdot u_k \end{aligned} \tag{5}$$

$$\hat{y}_k = W^o \cdot \left(W^{h2} \cdot x_k\right) = \left(W^o \cdot W^{h2}\right)\cdot x_k = W_3 \cdot x_k \tag{6}$$

In the recursive equation (5), W_1, W_2, and W_3 are unknown weights to be identified by observed input/output sequences $\{u_0, u_1, \cdots, u_{N-1}\}$ and $\{y_0, y_1, \cdots, y_{N-1}\}$. By replacing the x_k term in the observed equation (6) with the solved recursive equation (5), the output response of the system is given as:

$$y_k = W_3 W_1{}^k x_1 + \sum_{p=2}^{k} W_3 W_1{}^{p-1} W_2 u_{k-p} \tag{7}$$

For a system initially at rest, i.e., $x_1 = 0$, Equation (7) is rewritten as:

$$y_k = \sum_{p=1}^{k} h_p u_{k-p} \tag{8}$$

where the unit hydrograph (UH) of the rainfall-runoff processes can be summarized as:

$$\begin{aligned} h_p &= W_3 W_1{}^{p-1} W_2 \quad \text{if } p \ge 2 \\ h_p &= 0 \qquad\qquad \text{if } p = 1 \end{aligned} \tag{9}$$

The impulse response terms $W_3 W_1{}^{p-1} W_2$ for $p \ge 2$ are known as the Markov parameters.

3. Calibration algorithm for DLRNN

3.1 Indirect system identification

The concept of indirect system identification algorithms is to obtain the UH ordinates first, called the constrained deconvolution step. The linear programming is selected herein to carry out the UH from the rainfall and runoff data. Then, the system matrices $[W_1, W_2, W_3]$ are identified from the UH ordinates via singular value decomposition (SVD), entitled the realization step.
In the realization step, the state space model can be represented as follows if $\overline{x}_k = T x_k$ for some nonsingular transformation matrix T (Romos et al., 1995):

$$Tx_{k+1} = [TW_1T^{-1}]Tx_k + [TW_2]u_k \tag{10}$$

$$y_k = [W_3T^{-1}]Tx_k \tag{11}$$

By considering $[TW_1T^{-1}]$ as $\overline{W}_1$, $[TW_2]$ as $\overline{W}_2$, and $[W_3T^{-1}]$ as $\overline{W}_3$, the system matrices of the transformed system are now $\left[\overline{W}_1, \overline{W}_2, \overline{W}_3\right]$, and these parameter matrices $\left[\overline{W}_1, \overline{W}_2, \overline{W}_3\right]$ are identified based on the deconvoluted impulse response sequence $\{\hat{h}_p\}$. Specifically, SVD is performed on the following Hankel matrix:

$$H_{t,t} = \begin{bmatrix} \hat{h}_1 & \hat{h}_2 & \hat{h}_3 & \cdots & \hat{h}_k \\ \hat{h}_2 & \hat{h}_3 & \hat{h}_4 & \cdots & \hat{h}_{k+1} \\ \hat{h}_3 & \hat{h}_4 & \hat{h}_5 & \cdots & \hat{h}_{k+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{h}_k & \hat{h}_{k+1} & \hat{h}_{k+2} & \cdots & \hat{h}_{2k-1} \end{bmatrix} = U \cdot S \cdot V^T = \left(U \cdot S^{1/2}\right) \cdot \left(S^{1/2} \cdot V^T\right) = \overline{\mathbf{o}}_k \cdot \overline{\mathbf{c}}_k \tag{12}$$

where $2k-1 \le M$. M is the memory of system. The transformed parameter matrices are identified from:

$$\overline{W}_1 = TW_1T^{-1} = \overline{\mathbf{c}}_{k,2}\overline{\mathbf{c}}^*_{k,1}; \; \overline{W}_2 = TW_2 = \overline{\mathbf{c}}_k \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}; \; \overline{W}_3 = W_3T^{-1} = [1,0,\cdots,0]\overline{\mathbf{o}}_k$$

where $\overline{\mathbf{c}}_{k,1}$ and $\overline{\mathbf{c}}_{k,2}$ denote the first and last $(k-1)$ columns of $\overline{\mathbf{c}}$, and the star denotes a pseudoinverse.

3.2 Subspace algorithm

Above indirect system identification algorithm computes the weights of a RNN from a Hankel matrix constructed using Markov parameters. However, using the Markov parameters as a starting point would be rather difficult to measure in some fields (Abdelghani & Verhaegen, 1998). The subspace algorithms are the automatic structure identification, and derive the model directly from the input-output data without estimating the Markov parameters as an intermediate step (Gustafsson, 2001; Ramos et al., 1995).
Before description of subspace algorithm, the past and future highly rectangular input/output Hankel matrices, H_1 and H_2 respectively, are defined by input-output data:

$$H_1 = \left[\frac{U_1}{Y_1}\right] = \left[\begin{array}{cccc} u_1 & u_2 & \cdots & u_j \\ u_2 & u_3 & \cdots & u_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_i & u_{i+1} & \cdots & u_{i+j-1} \\ \hline y_1 & y_2 & \cdots & y_j \\ y_2 & y_3 & \cdots & y_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_i & y_{i+1} & \cdots & y_{i+j-1} \end{array}\right] \text{ for j>>i>n} \tag{13}$$

$$H_2 = \left[\frac{U_2}{Y_2}\right] = \left[\begin{array}{cccc} u_{i+1} & u_{i+2} & \cdots & u_{i+j} \\ u_{i+2} & u_{i+3} & \cdots & u_{i+j+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{2i} & u_{2i+1} & \cdots & u_{2i+j-1} \\ \hline y_{i+1} & y_{i+2} & \cdots & y_{i+j} \\ y_{i+2} & y_{i+3} & \cdots & y_{i+j+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{2i} & y_{2i+1} & \cdots & y_{2i+j-1} \end{array}\right] \text{ for j>>i>n} \tag{14}$$

Two state vector sequences X_1 and X_2 are defined as $X_1 = [x_1 \mid x_2 \mid \cdots \mid x_j]$ and $X_2 = [x_{i+1} \mid x_{i+2} \mid \cdots \mid x_{i+j}]$. The subspace algorithm is presented as follows:

a) Compute the SVD of the concatenation of H_1 and H_2:

$$\left[\frac{H_1}{H_2}\right] = U_H \Sigma_H V_H^T = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \cdot \begin{bmatrix} \Sigma_{11} & 0_{(2mi+n)\times j} \\ 0_{(2li-n)\times(2mi+n)} & 0_{(2li-n)\times j} \end{bmatrix} \cdot V_H^T \tag{15}$$

where u_{11}, u_{12}, u_{21}, u_{22}, Σ_{11}, and V_H are the matrices with dimensions $(mi+li)\times(2mi+n)$, $(mi+li)\times(2li-n)$, $(mi+li)\times(2mi+n)$, $(mi+li)\times(2li-n)$, $(2mi+n)\times(2mi+n)$, and $j\times j$ respectively.

b) Compute the SVD of $u_{12}^T u_{11} \Sigma_{11}$ in order to determine the system order, n:

$$u_{12}^T u_{11} \Sigma_{11} = \left[U_q \mid U_q^{\perp}\right] \cdot \begin{bmatrix} \Sigma_q & 0_{n\times(2mi)} \\ 0_{2(li-n)\times n} & 0_{2(li-n)\times(2mi)} \end{bmatrix} \cdot V_q^T \tag{16}$$

where U_q, $U_q^{\perp}$, Σ_q, and V_q are the matrices with dimensions $(2li-n)\times n$, $(2li-n)\times 2(li-n)$, $n\times n$, and $(2mi+n)\times(2mi+n)$ respectively.

c) Compute the transformed state vector sequence:

$$\overline{X}_2 = U_q^T u_{12}^T H_1 = [\overline{x}_{i+1} \mid \overline{x}_{i+2} \mid \cdots \mid \overline{x}_{i+j}] \tag{17}$$

where $\overline{X}_2$ is the matrix with dimensions $n\times j$.

d) Compute the weights of the RNN by solving the overdetermined system of equations:

$$\begin{bmatrix} \overline{x}_{i+2} & \overline{x}_{i+3} & \cdots & \overline{x}_{i+j} \\ y_{i+1} & y_{i+2} & \cdots & y_{i+j-1} \end{bmatrix} = \begin{bmatrix} \overline{W}_1 & \overline{W}_2 \\ \overline{W}_3 & 0 \end{bmatrix} \cdot \begin{bmatrix} \overline{x}_{i+1} & \overline{x}_{i+2} & \cdots & \overline{x}_{i+j-1} \\ u_{i+1} & u_{i+2} & \cdots & u_{i+j-1} \end{bmatrix} \tag{18}$$

In the past few years, much attention has been paid recently to subspace algorithms when various time domain methods for identifying dynamic models of systems from modal experimental data appeared. However, this algorithm was seldom applied in the scope of hydrology. Except Ramos et al. (1995), they used one event of 29 data points (each 30 minutes long) and 365 daily data to evaluate the algorithm. To compare with daily data, hourly data used herein have more uncertainty and noisy. The suitability of subspace algorithm with hourly rainfall-runoff data, therefore, is re-evaluated based on a real typhoon event of the Keelung River in Taiwan as follows:

Firstly, a sequence of 100 data is generated from a state space model that was identified from rainfall-runoff data observed on Sep. 27, 1996. Indirect system identification algorithm was used to check if the subspace algorithm could identify the original system. The state space model is a 3-order system as following equations:

$$X_{k+1} = \begin{bmatrix} 0.9600 & -0.1087 & -0.0383 \\ 0.1087 & 0.8333 & -0.3317 \\ -0.0383 & 0.3316 & 0.6772 \end{bmatrix} \cdot X_k + \begin{bmatrix} -0.2274 \\ 0.2144 \\ -0.1485 \end{bmatrix} \cdot U_k \tag{19}$$

$$Y_k = \begin{bmatrix} -0.2274 & -0.2144 & -0.1485 \end{bmatrix} \cdot X_k \tag{20}$$

The generated sequence was identified as Equations (21) and (22). The results show that the system order determination in the step 2 of subspace algorithm is correct so that the impulse response can be simulated accurately.

$$X_{k+1} = \begin{bmatrix} 0.9559 & -0.1857 & -0.0944 \\ 0.0693 & 0.8517 & 0.5704 \\ 0.0237 & -0.1880 & 0.6628 \end{bmatrix} \cdot X_k + \begin{bmatrix} -0.1190 \\ 0.0716 \\ 0.0386 \end{bmatrix} \cdot U_k \tag{21}$$

$$Y_k = \begin{bmatrix} -0.4403 & -0.6683 & 0.6034 \end{bmatrix} \cdot X_k \tag{22}$$

The second test used the original observed data to identify a rainfall-runoff system. However, according to the identified UHs shown in Fig. 2, the subspace algorithm performed poorly because it was very sensitive to the noise in observed data. Therefore, the modified system identification combined with indirect system identification and subspace algorithm is introduced.

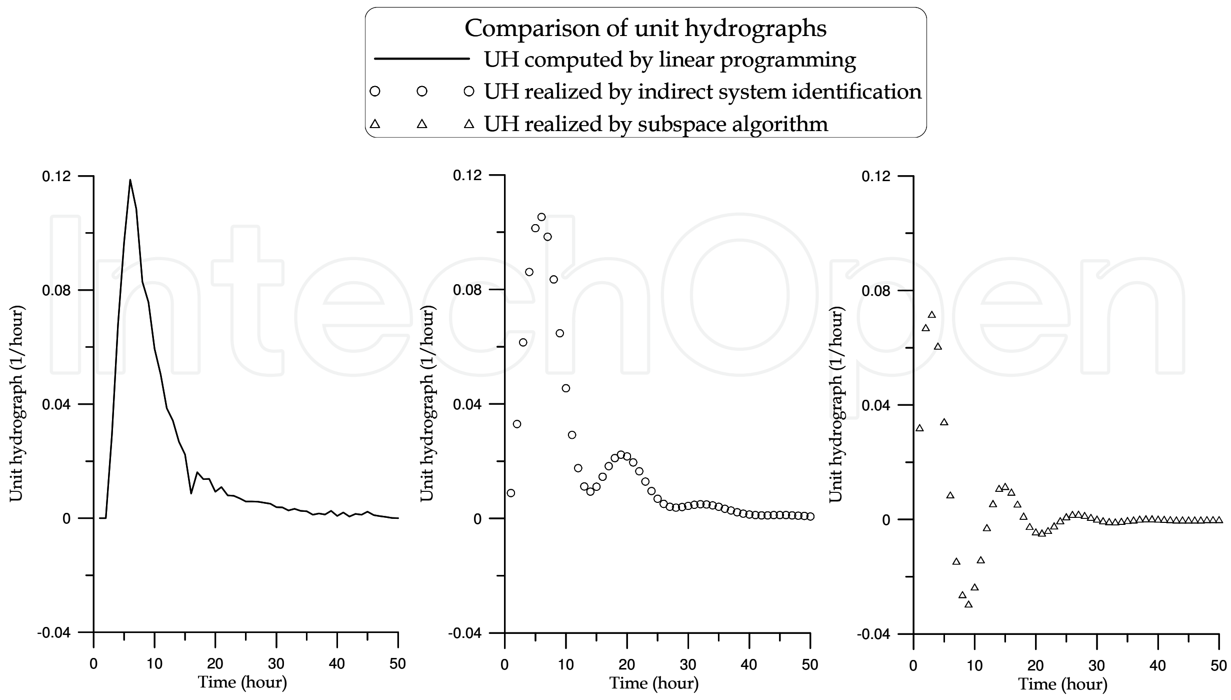


Fig. 2. UHs carried out via linear programming, indirect system identification, and subspace algorithm.

3.3 Modified system identification for hydrology

Figure 3 and the left part of Fig. 4 are the flowcharts of indirect system identification algorithm and subspace algorithm respectively. To compare with these two flowcharts, indirect system identification algorithm needs to subjectively decide the system order from a sequence of singular values in Equation (16). In practice, the singular values are not easily classified into significant and insignificant groups when the singular values descend slowly. Additionally, subspace algorithm can determine the system order objectively, but it is sensitive. Therefore, the constrained deconvolution step is considered, firstly, to compute a discrete UH from rainfall-runoff events for calibration. Secondly, a sequence of rainfall-runoff data generated form the discrete UH via convolution is synthesized. This synthesized data are without noise that helps subspace algorithm to get the system order. The right part of Fig. 4 surrounded by dotted line is the modified system identification for hydrology.

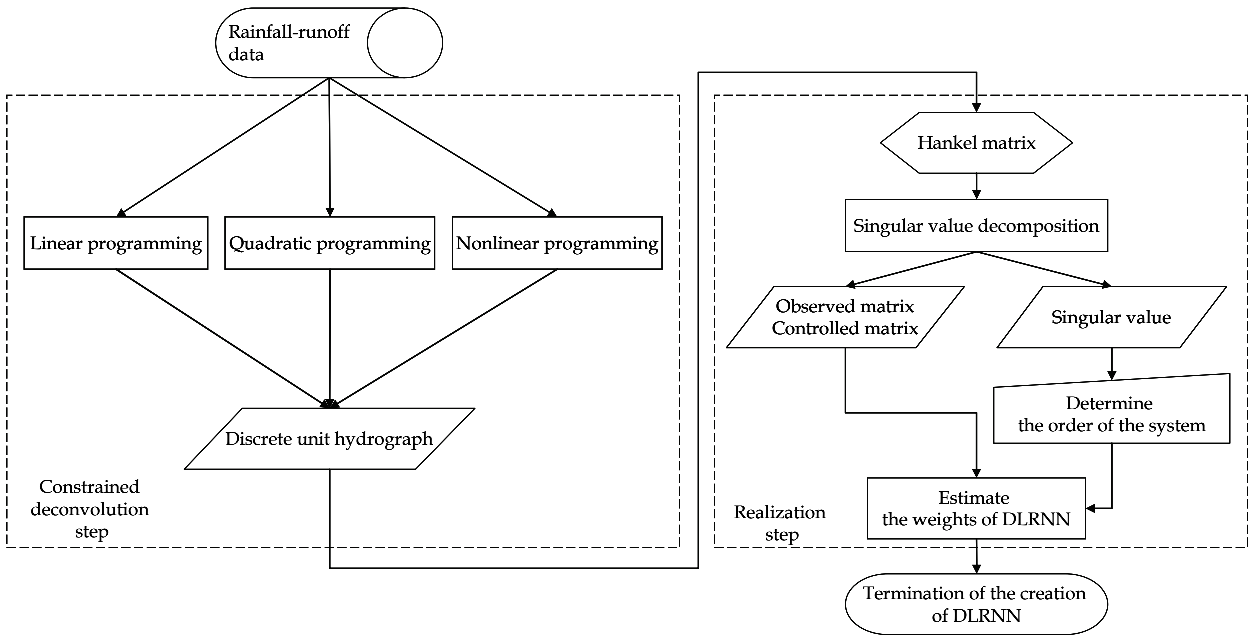

Rainfall-runoff data
Linear programming
Quadratic programming
Nonlinear programming
Discrete unit hydrograph
Constrained deconvolution step
Hankel matrix
Singular value decomposition
Observed matrix Controlled matrix
Singular value
Determine the order of the system
Realization step
Estimate the weights of DLRNN
Termination of the creation of DLRNN

Fig. 3. Flowchart of indirect system identification.

4. On-line learning algorithm for DLRNN

Dynamic RNN learning algorithms can be grouped into five major categories (Parlos et al., 2000), such as (1) the real time recurrent learning; (2) the backpropagation through time (BTT) method; (3) the fast forward propagation method; (4) the Green's function method; and (5) the block update method. All training algorithms above are gradient-based by which the learning trajectory is represented into the changes of weights of neurons.
The weights updated via gradient-based learning algorithms can be written as:

$$W_{new} = W_{old} - \eta \frac{dE}{dW} \tag{23}$$

where η denotes the learning rate, and E is the sum of square errors.

$$E = \frac{1}{2}\sum_{k=1}^{K}(y_k - d_k)^T(y_k - d_k) \tag{24}$$

where y_k is the output of the model, and d_k represents the desired output at time index k. The algorithm introduced herein is based on the gradient-based learning method developed by Atiya and Parlos (2000).

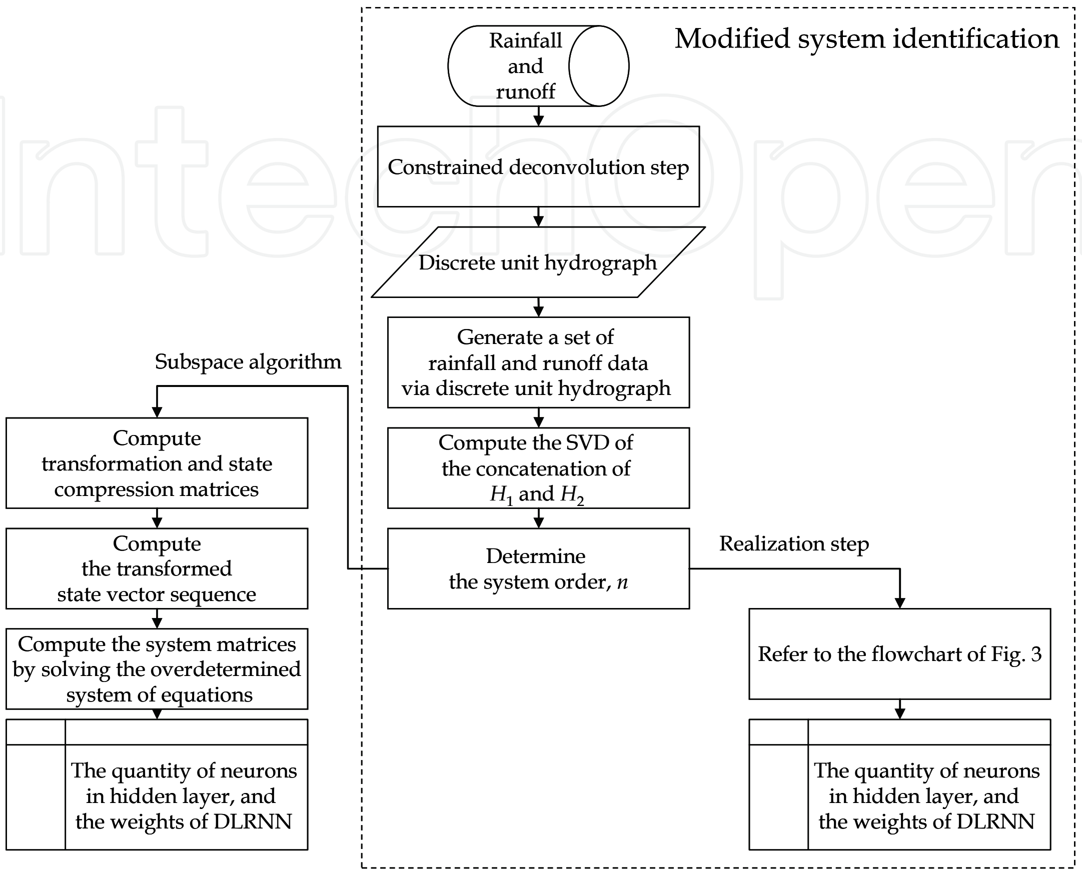

Modified system identification
Rainfall and runoff
Constrained deconvolution step
Discrete unit hydrograph
Generate a set of rainfall and runoff data via discrete unit hydrograph
Compute the SVD of the concatenation of H_1 and H_2
Determine the system order, n
Subspace algorithm
Compute transformation and state compression matrices
Compute the transformed state vector sequence
Compute the system matrices by solving the overdetermined system of equations
The quantity of neurons in hidden layer, and the weights of DLRNN
Realization step
Refer to the flowchart of Fig. 3
The quantity of neurons in hidden layer, and the weights of DLRNN

Fig. 4. Flowchart of modified system identification.

4.1 DLRNN learning algorithm

The idea of the algorithm adopted herein is to obtain an approximation for the gradient that can be efficiently computed via the interchange of the roles of the network states x_k and the weight matrix W. Let the states be considered as the control variables, and the change in the weights is determined upon the changes in x_k. The details of the algorithm are as follows: First, the network learning is formulated as constrained minimization problem, with the objective to minimize the sum of square error, E, given by Equation (24), and the constraints.

$$g_{k+1} \equiv W_1 \cdot x_k + W_2 \cdot u_k - \hat{x}_{k+1} = 0, \quad k = 0, \cdots, K-1 \tag{25}$$

According to the Equations (10) and (11), the error gradient can be written as follows:

$$\frac{dE}{dw_1} = \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w_1} \tag{26a}$$

$$\frac{dE}{dw_2} = \frac{\partial E}{\partial w_2} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w_2} \tag{26b}$$

$$\frac{dE}{dw_3} = \frac{\partial E}{\partial w_3} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_3} \tag{26c}$$

where $\frac{dE}{dw_n}$ for n = 1, 2, 3 equals 0 since E is the function of y. Consequently, the updated weights of W_2 and W_3 for time K can be derived ffom Equation (23), (24), (26b), and (26c) as follows:

$$W_{2,K+1} = W_{2,K} - \eta \cdot (y_K - d_K) \cdot W_{3,K} \cdot u_K \tag{27}$$

$$W_{3,K+1} = W_{3,K} - \eta \cdot (y_K - d_K) \cdot x_K \tag{28}$$

By taking the derivative of the Equation (25), one can get:

$$\frac{\partial g}{\partial w_1} + \frac{\partial g}{\partial x}\frac{\partial x}{\partial w_1} = 0 \Rightarrow \frac{\partial x}{\partial w_1} = -\left(\frac{\partial g}{\partial x}\right)^{-1}\frac{\partial g}{\partial w_1} \tag{29}$$

Solving Equations (26a) and (29), one can get:

$$\frac{dE}{dw_1} = -\frac{\partial E}{\partial y}\frac{\partial y}{\partial x}\left(\frac{\partial g}{\partial x}\right)^{-1}\frac{\partial g}{\partial w_1} \tag{30}$$

According to the convention that $(\partial u/\partial v)$ for two vectors u and v is the matrix whose (i, j)th element is $(\partial u_i/\partial v_j)$, the matrices in (28) can be evaluated from Equations (6) and (24) as follows:

$$\frac{\partial E}{\partial y}\frac{\partial y}{\partial x} = (e_1, e_2, \cdots, e_K)^T \cdot W_3 = W_3^T \cdot (e_1, e_2, \cdots, e_K) \tag{31}$$

where e_k is the error at time k: $e_k = y_k - d_k$,

$$\frac{\partial g}{\partial x} = \begin{bmatrix} -I & 0 & 0 & \cdots & 0 & 0 \\ W_1 & -I & 0 & \cdots & 0 & 0 \\ 0 & W_1 & -I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & W_1 & -I \end{bmatrix} \tag{32}$$

and

$$\frac{\partial g}{\partial w} = \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{K-1} \end{bmatrix} \tag{33}$$

where

$$X_k = \begin{bmatrix} x_k^T & 0 & 0 & \cdots & 0 \\ 0 & x_k^T & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & x_k^T \end{bmatrix} \tag{34}$$

I is the identity matrix, and 0 in Equations (32) and (34) is a matrix (or vector) of zeros. After calculating the gradient of *E* with respect to the states x_k, a small change at the states x_k in the negative direction of that gradient can be written as:

$$\Delta x = -\eta\left(\frac{\partial E}{\partial x}\right) \tag{35}$$

Replace $\frac{\partial E}{\partial x}$ by Equation (29), and Equation (33) can be rewritten as:

$$\Delta x = -\eta e = -\eta W_3^T\left(e_1, \cdots, e_K\right) \tag{36}$$

Since g, given by Equation (25), equals zero, one can get:

$$\frac{\partial g}{\partial w_1}\Delta w_1 + \frac{\partial g}{\partial x}\Delta x = 0 \tag{37}$$

After applying the transposition and the pseudoinverse in Equation (37), the change in weights can be determined as:

$$\Delta w_1 = -\left[\left(\frac{\partial g}{\partial w_1}\right)^T\left(\frac{\partial g}{\partial w_1}\right)\right]^{-1}\left(\frac{\partial g}{\partial w_1}\right)^T\frac{\partial g}{\partial x}\Delta x_1 \tag{38}$$

where

$$\left[\left(\frac{\partial g}{\partial w_1}\right)^T\left(\frac{\partial g}{\partial w_1}\right)\right]^{-1} = \begin{bmatrix} \sum_{k=0}^{K-1} x_k x_k^T & 0 & \cdots & 0 \\ 0 & \sum_{k=0}^{K-1} x_k x_k^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{k=0}^{K-1} x_k x_k^T \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} \left[\sum_{k=0}^{K-1} x_k x_k^T\right]^{-1} & 0 & \cdots & 0 \\ 0 & \left[\sum_{k=0}^{K-1} x_k x_k^T\right]^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \left[\sum_{k=0}^{K-1} x_k x_k^T\right]^{-1} \end{bmatrix} \tag{39}$$

From Equation (36), let

$$\gamma = \frac{\partial g}{\partial x} e^T = \frac{-1}{\eta} \frac{\partial g}{\partial x} \Delta x \tag{40}$$

and partition the vector γ into the K vectors as follows:

$$\gamma = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_K \end{pmatrix} \tag{41}$$

Using Eqs. (32) and (40), γ can be evaluated by following recursions:

$$\gamma_1 = -e_1^T \tag{42a}$$

$$\gamma_2 = -e_2^T + W_1 e_1^T \tag{42b}$$

$$\vdots$$

$$\gamma_K = -e_K^T + W_1 e_{K-1}^T \tag{42c}$$

Let

$$V_K' = \sum_{k=0}^{K-1} x_k x_k^T \tag{43}$$

Substituting Equations (33), (39), (40), and (43) into (38), one can get after some manipulation.

$$\Delta W_1 = \eta \left[\sum_{k=1}^{K} \gamma_k x_{k-1}^T \right] \cdot V_K'^{-1} \tag{44}$$

In order to alleviate the effect of most likelihood ill-conditioning problems caused by the matrix inversion in Equation (44), a small matrix εI is added to the outer product matrix V_K' as follows:

$$V_K = \varepsilon I + \sum_{k=0}^{K-1} x_k x_k^T \ , \tag{45}$$

where ε is a small positive constant. Then Equation (44) is rewritten as follows:

$$\Delta W_1 = \eta \left[\sum_{k=1}^{K} \gamma_k x_{k-1}^T \right] \cdot V_K^{-1} \tag{46}$$

Since the passed inputs, state variables, and observed outputs (u_1, x_1, d_1, …, u_{K-1}, x_{K-1}, d_{K-1}) are already available to get $\Delta W_{1,K-1}$, the on-line updated change in weights $\Delta W_{1,K}$ based on a new data point (u_K, x_K, d_K) can be written as follows:

$$\Delta W_{1,K} = \eta \left[\sum_{k=1}^{K-1} \gamma_k x_{k-1}^T + \gamma_K x_{K-1}^T \right] \cdot \left[V_{K-1} + x_{K-1} x_{K-1}^T \right]^{-1} \tag{47}$$

Furthermore, using the small rank adjustment matrix inversion lemma, the inverse of V_K can be obtained recursively in terms of the inverse of V_{K-1} as follows:

$$V_K^{-1} = \left(V_{K-1} + x_{K-1}x_{K-1}^T\right)^{-1} = V_{K-1}^{-1} - \frac{\left(V_{K-1}^{-1}x_{K-1}\right)\left(V_{K-1}^{-1}x_{K-1}\right)^T}{1 + x_{K-1}^T V_{K-1}^{-1} x_{K-1}} \tag{48}$$

and let

$$B_K = \sum_{k=1}^{K} \gamma_k x_{k-1}^T \tag{49}$$

Substituting Eq. (48) into Eq. (47), after simplification one can get the final on-line updated formula of W_1 as follows:

$$\Delta W_{1,K} = \Delta W_{1,K-1} + \eta \frac{\gamma_K x_{K-1}^T V_{K-1}^{-1} - B_{K-1} V_{K-1}^{-1} x_{K-1} \left[V_{K-1}^{-1} x_{K-1}\right]^T}{1 + x_{K-1}^T V_{K-1}^{-1} x_{K-1}} \tag{50}$$

5. Application

5.1 Study area and data pre-processing

With a length of 86 km and an area of 501 km^2, the Keelung River has a U-turn in the northeast Taipei county, and runs through Taipei city, where it joins the Dansuie River and flows out to sea, as shown in Fig. 5. The watershed upstream of Wu-tu with about 204 km^2 surrounding the city of Taipei in northern Taiwan was chosen for evaluating the simulation ability of the DLRNN for recognizing the transition of rainfall-runoff processes. Due to the northeast monsoon in winter and the typhoons in summer, the mean annual precipitation, runoff depth, and runoff coefficient are 2865 mm, 2177 mm, and 0.76, respectively. Owing to the rugged topography of the watershed, large floods caused by the short and steep runoff path-line arrive rapidly in the middle-to-downstream reaches of the watershed, and cause serious damage.

According to the records of three rain gauges (Wu-tu, Jui-fang, and Huo-shao-liao) and on discharge site (Wu-tu) in Wu-tu watershed, as shown in Fig. 5, 38 rainfall-runoff events from 1966 to 1997 were selected as study cases including 13 multi-peak and 25 single-peak events (Table 3). With 766 rainfall-runoff observations, the earliest 10 events, from 1966 to 1972, were used for calibration while the remainder events were used for validation. Through the Kriging method to calculate the average effective rainfall based on effective rainfall measurements from three rain gauges, current average effective rainfall (mm) and direct hourly runoff (m^3/s) are the input and output with no lead-time considered after be normalized between 0 and 0.9.

5.2 Criteria

The performances of rainfall-runoff simulations were evaluated by four criteria as follows:

(1) Coefficient of efficiency, *CE*, is defined as follows:

$$CE = 1 - \frac{\sum_{k=1}^{K}\left[Q_{obs,k} - Q_{est,k}\right]^2}{\sum_{k=1}^{K}\left[Q_{obs,k} - \overline{Q}_{obs}\right]^2} \tag{51}$$

where $Q_{est,k}$ denotes the discharge of the simulated hydrograph for time index k (m^3/s), $Q_{obs,k}$ is the discharge of the observed hydrograph for time index k (m^3/s), and $\overline{Q}_{obs}$ is the mean of the discharge of the observed hydrograph during whole event period K. The better the fit, the closer CE is to 1.

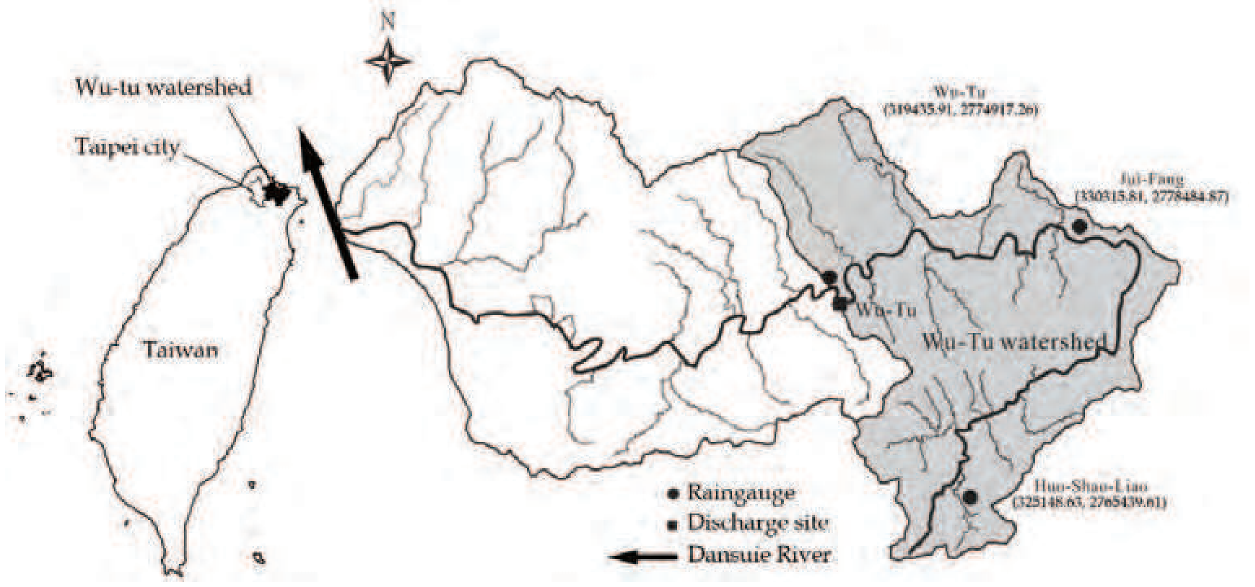


Fig. 5. The maps of Wu-tu watershed showing the study area near Taipei, Taiwan (the coordinates are TWD67 2-degree wide Transverse Mercator projection).

Typhoon name	Time (y/m/d)	Rainfall duration (h)	Rainfall depth (mm)	Max rainfall intensity (mm/h)	Max discharge (m^3/s)	Typhoon name	Time (y/m/d)	Rainfall duration (h)	Rainfall depth (mm)	Max rainfall intensity (mm/h)	Max discharge (m^3/s)
Cora	1966/09/06	48	247.9	20.0	770.7	*Gerald	1984/08/14	127	513.5	23.4	586.4
*Carla	1967/10/17	72	1088.0	52.9	921.2	Nelson	1985/08/22	46	341.4	25.0	1177.0
*Gilda	1967/11/16	59	339.5	29.1	706.9	Brenda	1985/10/03	38	248.4	15.1	626.7
Nadine	1968/07/26	61	252.1	15.1	219.7	*Abby	1986/09/17	91	521.3	28.8	579.0
Elaine	1968/09/29	72	686.6	44.5	1037.7	Alex	1987/07/27	30	187.0	40.7	519.8
*Storm	1969/09/09	89	678.5	24.1	848.4	*Gerald	1987/09/09	33	321.2	47.2	553.9
Elsie	1969/09/26	38	288.5	38.0	662.5	*Storm	1988/09/29	101	627.3	22.7	670.2
Agnes	1971/09/18	69	411.3	31.5	466.3	*Sarah	1989/09/10	61	322.5	27.7	401.2
Bess	1971/09/22	54	353.3	32.0	994.1	*Offlia	1990/06/22	49	251.0	20.6	500.0
Betty	1972/08/16	40	177.2	15.2	677.9	Yancy	1990/08/19	44	259.5	46.3	824.5
Storm	1973/09/20	22	292.5	37.3	862.3	Abe	1990/08/30	35	239.1	15.7	764.4
Wendy	1974/09/28	57	321.2	16.7	822.0	Storm	1990/09/02	26	192.8	32.2	842.5
Vera	1977/07/31	46	264.7	16.9	735.7	*Polly	1992/08/29	98	500.6	17.8	278.9
Storm	1977/11/15	72	292.2	15.2	538.4	Gladys	1994/09/01	18	184.1	31.3	434.2
Irving	1979/08/14	56	340.3	24.4	974.1	*Seth	1994/10/09	48	300.7	12.2	451.3
Storm	1980/11/19	42	266.9	21.9	687.1	Herb	1996/07/31	44	313.6	31.8	1082.9
*Cecil	1982/08/09	34	235.7	23.9	626.4	*Zane	1996/09/27	84	440.6	29.9	666.0
Storm	1984/06/02	18	212.7	46.1	1403.5	Winnie	1997/08/17	47	343.5	24.1	1034.8
Freda	1984/08/06	30	242.1	30.7	501.5	Amber	1997/08/29	42	329.8	30.2	953.5

* Multi-peak event

Table 3. Information about the 38 events selected from Wu-tu watershed.

(2) The error of peak discharge, EQ_p (%), is defined as follows:

$$EQ_p(\%) = \frac{Q_{p,est} - Q_{p,obs}}{Q_{p,obs}} \times 100\% \qquad (52)$$

where $Q_{p,est}$ denotes the peak discharge of the simulated hydrograph (m^3/s) and $Q_{p,obs}$ is the peak discharge of the observed hydrograph (m^3/s).
(3) The error of the time for peak to arrive, ET_p, is defined as follows:

$$ET_p = T_{p,est} - T_{p,obs} \tag{53}$$

where $T_{p,est}$ denotes the time for the simulated hydrograph peak to arrive (*hours*) and $T_{p,obs}$ represents the time required for the observed hydrograph peak to arrive (*hours*).
(4) The error of total discharge volume, $VER(\%)$, is defined as follows:

$$VER(\%) = \frac{\left(\sum_{k=1}^{K} Q_{est,k} - \sum_{k=1}^{K} Q_{obs,k}\right)}{\sum_{k=1}^{K} Q_{obs,k}} \times 100\% \tag{54}$$

where $Q_{est,k}$ denotes the discharge of the simulated hydrograph for time index k (m^3/s) and $Q_{obs,k}$ is the discharge of the observed hydrograph for time index k (m^3/s). The better the fit, the closer EQ_p, ET_p and VER are to 0.

6. Result and discussion

A developed DLRNN is applied to perform rainfall-runoff simulation and recognize the transition of rainfall-runoff processes using UHs realized from the DLRNN weights. First, the DLRNN is compared with a forward neural network to demonstrate the advantage of RNNs. DLRNNs identified using indirect system identification and modified system identification then are compared. Furthermore, control system theory is employed to consider a DLRNN in canonical form and compare it with that identified using modified system identification. Finally, rainfall-runoff processes recognition using DLRNN is described.

6.1 Comparison between DLRNN and FNN (Pan et al., 2007)

Through the modified system identification based on the earliest 10 events, a DLRNN with 4 neurons in the hidden layer is calibrated, as shown in Fig. 6. Due to the full connection between neurons in hidden layer, the DLRNN totally has 24 weights for storing information. Therefore, it is fair to have the same control on the quantity of weights for comparing the DLRNN with the feed-forward neural networks (FNNs) although the structures of FNNs with inputting information as a time delay pattern that constitutes the tapped delay line information are classified as local or global RNNs according to the definition by Tsoi and Back (1997). Based on the rule of Equations (55) and (56), observed runoff and rainfall data are used in sequence to constitute the tapped delay line inputs as the input layer illustrated in Fig. 7. In hidden layer of Fig. 7, a bias neuron always delivers a negative impulse as a threshold to each hidden neuron. All FNNs compared with the DLRNN herein are trained using the same calibrated data via the back-propagation learning algorithm, the most common learning algorithm for FNNs.

$$\text{input neuron} = \text{rainfall}\big(k - (\text{int}((n+1)/2)+1)\big), \text{ if } n \text{ is odd;} \tag{55}$$

$$\text{input neuron} = \text{runoff}\left(k - \left(\text{int}\left(n/2\right)\right)\right),\ \text{if } n \text{ is even} \tag{56}$$

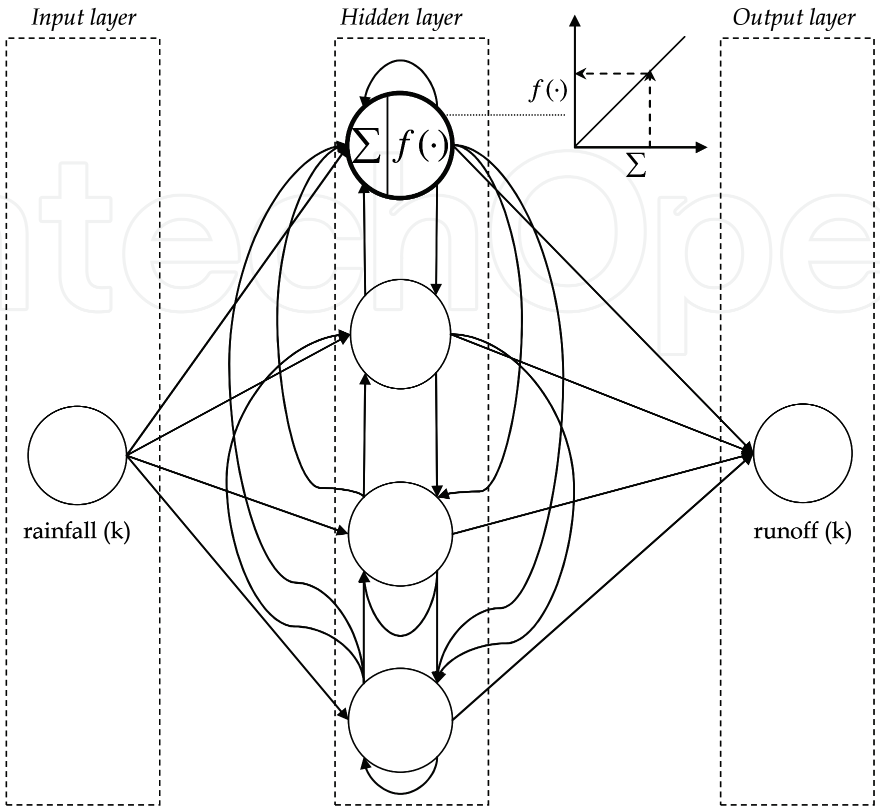


Fig. 6. Architecture of DLRNN identified via modified system identification.

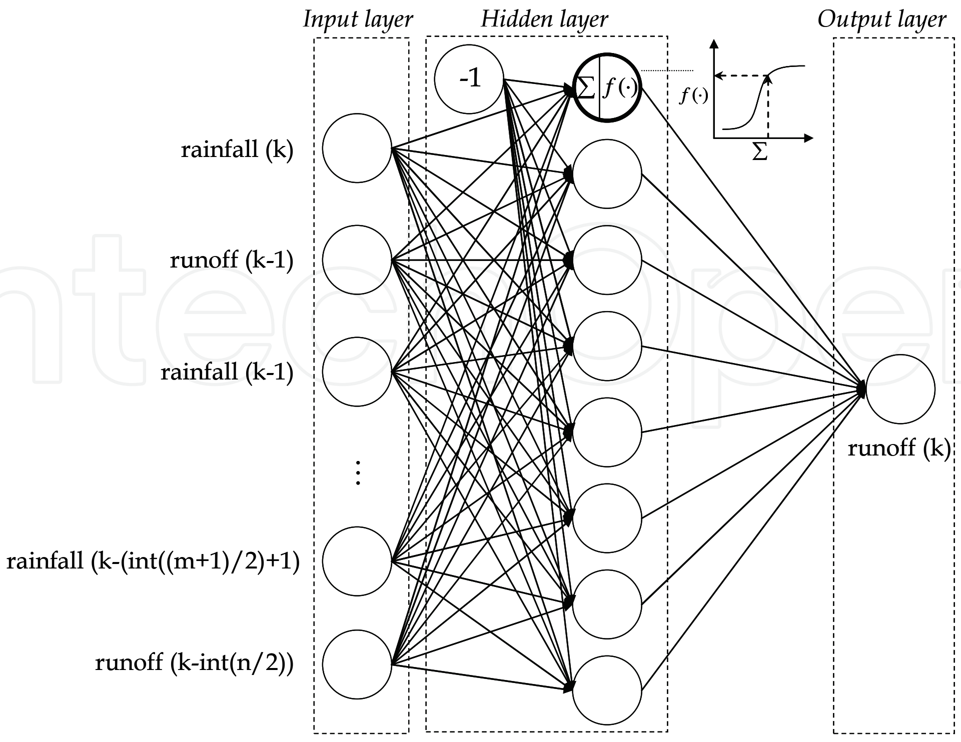


Fig. 7. The structure of FNNs with the tapped delay line inputs.

Model Form	Feed-forward neural network								DLRNN
Number of neurons in hidden layer	1	2	3	4	5	6	7	8	4
Number of neuraons in input layer	22	10	6	4	3	2	2	1	1
Number of neural network's weights	24	24	24	24	25	24	28	24	24
CE	0.943	0.982	0.975	0.974	0.957	0.952	0.950	-0.074	0.926
EQp (%)	10.363	4.377	4.432	4.445	4.545	4.687	5.322	48.936	12.438
ETp (hour)	2.079	1.184	1.184	1.526	1.895	1.921	1.921	7.474	1.036
VER (%)	5.504	2.088	2.242	2.383	2.513	3.101	3.295	24.509	4.769

Table 4. The averages of absolute criteria of the DLRNN and FNNs to simulate the rest 28 events (Pan et al., 2007).

Table 4 shows the averages of the absolute criteria of the DLRNN and the FNNs in which FNN(1-8-1) is the only neural network without any feedback connection. According to the average absolute criteria, the FNN(1-8-1) performs poorly because it is merely a static system without memory and only executes mapping from rainfall to runoff. However, the FNNs with tapped delay line inputs, such as FNN(2-7-1) to FNN(22-1-1), perform superiorly. The result shows the importance of a feedback connection and using tapped delay line inputs to the FNN. Chiang et al. (2004) also noticed that the feature of feedback connections is especially important and useful for grasping the extraordinary time-varying characteristics of the rainfall-runoff processes. The neural network with only one rainfall input can not achieve a satisfactory mapping to the current runoff because the rainfall-runoff processes are dynamic systems. One more tapped delay line input, like FNN(2-7-1), gives the feed-forward neural network the last-time-step status of the runoff, and raises the CE over 0.94. However, the DLRNN only needs the current rainfall as the input to get a satisfactory simulation because the feedback connections in hidden layer give the DLRNN the function to calculate the state of the rainfall-runoff process recurrently.

6.2 Comparison between DLRNNs based on two identification methods

Vos et al. (2005) commented that a disadvantage of artificial neural networks is that the optimal form or value of most network design parameters differ for each application and cannot be theoretically defined, which is why they are commonly found using trial-and-error approaches. However, the identification methods mentioned herein provide a deterministic solution. This chapter considers the indirect and modified system identification for identifying DLRNNs. In the realization step of the indirect system identification, a series of singular values is carried out through the singular value decomposition, and it can be illustrated in Fig. 8. If the singular values can be separated distinctly into two groups, namely the significant and the neglected groups, the number of neurons in the hidden layers of a RNN equals to the size of the significant group. From Fig. 8, the first two singular values are relatively significant and the number of neurons in the hidden layers are at least 2. However, the other singular values do not decrease noticeably, making it difficult to optimize the number of neurons of the hidden layers. Furthermore, the relation between the coefficient of efficiency and the number of neurons in the hidden layers of the DLRNN determined using trial-and-error method is illustrated as the open dots in Fig. 9. The CE increases from 0.70 to over 0.86 while the number of neurons in hidden layers exceeds 2 in Fig. 9. Six neurons in the hidden layer are selected as the optimum DLRNN (denoted as DLRNN(1)), denoted as the solid dot at the right side of Fig. 9) using the best coefficient of efficiency (*CE*=0.87043).

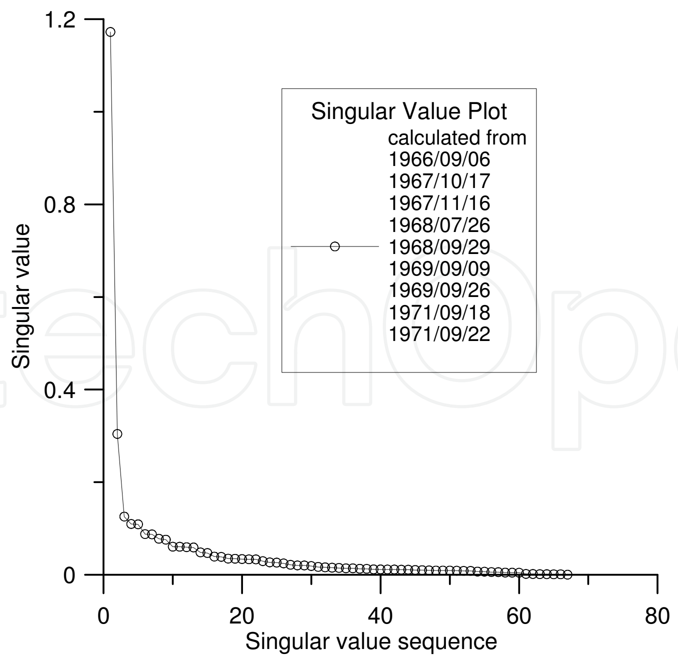


Fig. 8. The singular value plot from the realization step of indirect system identification.

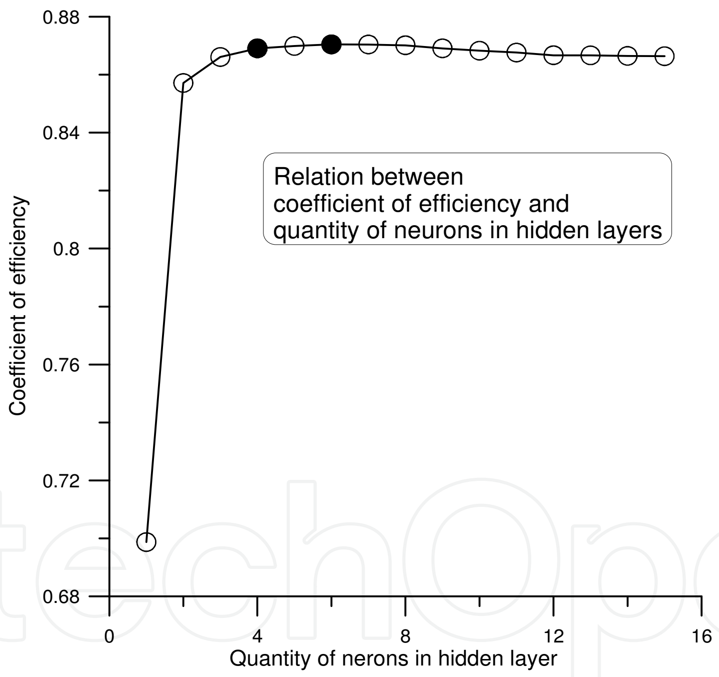


Fig. 9. The relation between coefficient of efficiency and number of neurons in hidden layers of the DLRNN.

Another DLRNN (denoted as DLRNN(2)) has four neurons in the hidden layer, as determined using modified system identification (solid dot at the left side of Fig. 9). Owing to part of the subspace algorithm being included in modified system identification, the four neurons in the hidden layer are chosen without any referable plot, such as singular value plot. From Fig. 9, the *CE* of DLRNN(2) is just 0.00028 less than that of DLRNN(1). However, DLRNN(2) reduces 48 weights of DLRNN(1), to 24 weights. The 50% reduction in weights from DLRNN(2) demonstrates that the combination of modified system identification and the advantages of indirect system identification and subspace algorithm provide an efficient algorithm for applying DLRNN in hydrology.

6.3 Comparison between DLRNNs in different forms

The DLRNN adopted herein is a fully RNN and has full connections between neurons in different layers. However, using a state space model is well known to over parameterize the estimation problem, while using canonical forms, as illustrated in Fig. 10, is far more economical for estimating the linear model. Figures 6 and 10 show that the DLRNNs have the feed-back connections in the hidden layers that belong to the local recurrent structures. The DLRNN in a canonical form has the same number of neurons as the original DLRNN, but the DLRNN in a canonical form has the minimum connections and weights to achieve the same performance. Hence, the comparison between the two DLRNNs in canonical form is of interest in this investigation. Some experiments are designed to clarify this issue. First, in the flowchart illustrated in Fig. 4, the original DLRNN(1) is transformed into a DLRNN in the canonical form after identifying the quantity of neurons in hidden layer and the weights of the DLRNN. Figure 10 shows that the DLRNN in the canonical form is clearly not a fully RNN. 28 validated events are fed to the model, and a new on-line learning method developed by Pan and Wang (2004), is applied to develop the DLRNN into a fully RNN via on-line learning. Table 5 lists the average absolute criteria. The table reveals that the canonical and non-canonical form DLRNNs do not differ significantly, and the on-line learning algorithm always derives a fully RNN from a DLRNN in the canonical form.

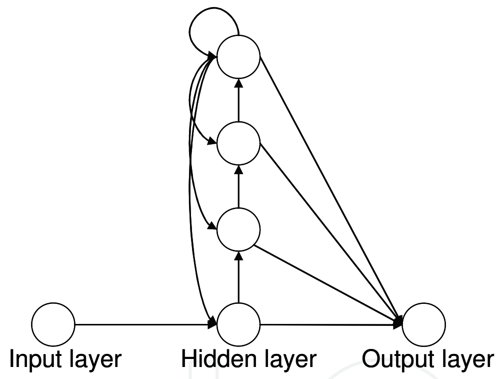


Fig. 10. The DLRNN in canonical form.

model type	*CE*	*EQp* (%)	*ETp* (hour)	*VER* (%)
original model	0.926	12.438	1.036	4.769
canonical form	0.925	12.704	1.071	4.845

original model: a DLRNN identified via modified system identification.

canonical form: a DLRNN in canonical form.

Table 5. The averages of absolute criteria of the DLRNNs in two forms.

6.4 Recognition of the transition of rainfall-runoff processes (Pan et al., 2007)

A streamflow or discharge hydrograph is a graph showing the flow rate as a function of time at a given location on the stream. In effect, the hydrograph is “an integral expression of the physiographic and climatic characteristics that govern the relations between rainfall and runoff of a particular drainage basin" (Chow, 1959). UH is a hypothetical unit response of the watershed to a unit input of rainfall that has been widely adopted by hydrologists to represent the mechanism of rainfall-runoff processes. Through the visualization of the transition of rainfall-runoff processes by UHs, the duration of a storm event, the time to peak flow, and the peak flow can be detected from the UHs. Therefore, the DLRNN learning algorithm is applied to modify the weights of the DLRNN on-line for detecting the transition of UHs based on the connection between the DLRNN and UH representation by treating the weights as Markov parameters. The structure of DLRNN can analogize the rainfall-runoff processes in a simple manner. The number of neurons in the hidden layer calibration by modified system identification describes the dimensions of the state space for the rainfall-runoff processes. Each neuron in the hidden layer represents a state variable that is controlled by rainfall and interacts with all state variables recurrently. Although the state variables can not be measured directly, UH can be represented based on their weights to describe the transition of rainfall-runoff processes.

Equations (8) and (9) reveal the relationship between the UH and the weights of DLRNN. Equation (9) also illustrates the relationship between the system responses to a unit impulse and the weights of DLRNN used herein. The time variance of the weights of a DLRNN can be used to recognize the transition of rainfall-runoff processes. Figure 11 illustrates the transition of UHs of the single-peak typhoon in Aug. 17, 1997, while Fig. 12 shows the simulation of this typhoon through DLRNN with on-line learning. At the beginning of the simulation, the weights of the DLRNN are identified from the earliest 10 events to form a generalized model. When comparing these two figures, the change of the UHs reveals the peak arrival is between the 15th and 30th hours. The time to peak of this typhoon is approximately 8 hours, shown in Fig. 11. The 8-hour duration is significantly increased after the time to peak of UH is calibrated as 3 hours. The rainfall process is fed to DLRNN to simulate runoff, as illustrated in Fig. 12, and the simulated runoff should follow the trends of the rainfall process. The rainfall-runoff simulations are evaluated as effective if the trends of rainfall and runoff are identical.

Another study case, Zane typhoon, is a multi-peak rainfall-runoff process out of the 38 selected events (Table 5). Figure 14 illustrates the variation between observed rainfall and runoff, and shows the excellent simulation performance from DLRNN. Figure 13 characterizes the transition of the rainfall-runoff process as the changes of UHs. During the first 20 hours of Zane typhoon, the simulated runoff is slightly higher than observed runoff (Figure 14), and this phenomenon demonstrates that the peak of the actual UH is lower than the UH realized from DLRNN. Through the on-line learning, the peak of the UH realized from DLRNN decays during first 20 hours. However, the largest peak of observed runoff is higher than the simulated runoff, and this shows that the actual UH of the rainfall-runoff process changes with time. Therefore, the peak of the UH realized from DLRNN increases after on-line learning. Furthermore, the difference between observed and simulated runoffs around the 60th hour demonstrates again the property of DLRNN that simulated runoff goes with the trends of the rainfall process. Additionally, a common conceptual model, called linear reservoir model, is introduced to compare with the DLRNN. It is an objective comparison in which both two models consider rainfalls as inputs. Results show that DLRNN performs better than the linear reservoir model.

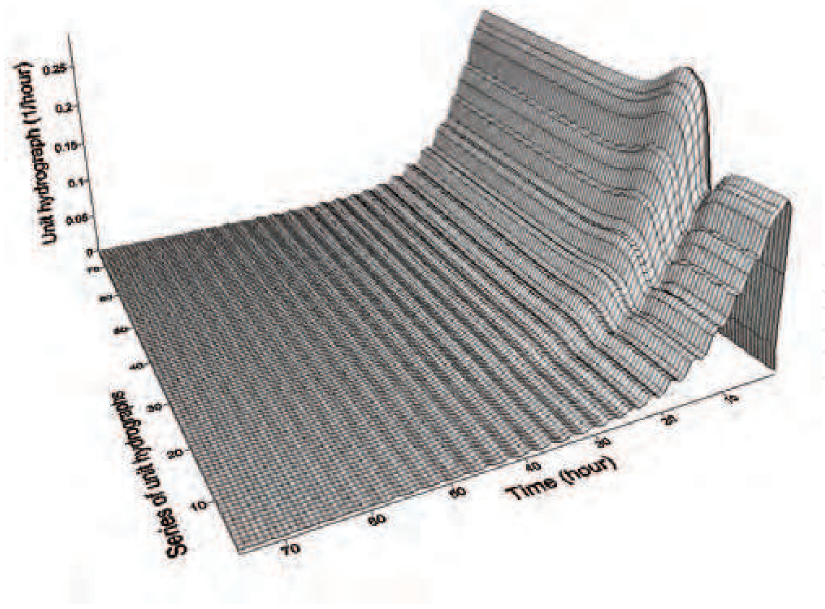


Fig. 11. The transition of UHs of the single-peak typhoon in Aug. 17, 1997.

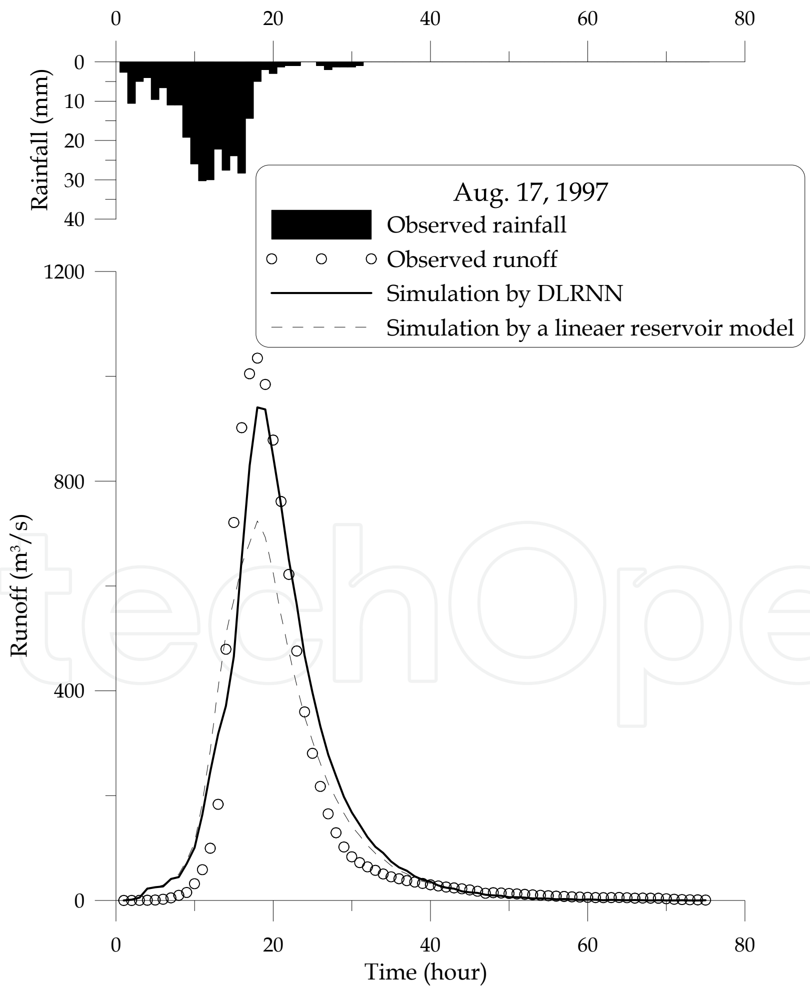


Fig. 12. Simulation of Winnie typhoon in Aug. 17, 1997 via DLRNN with on-line learning and a linear reservoir model.

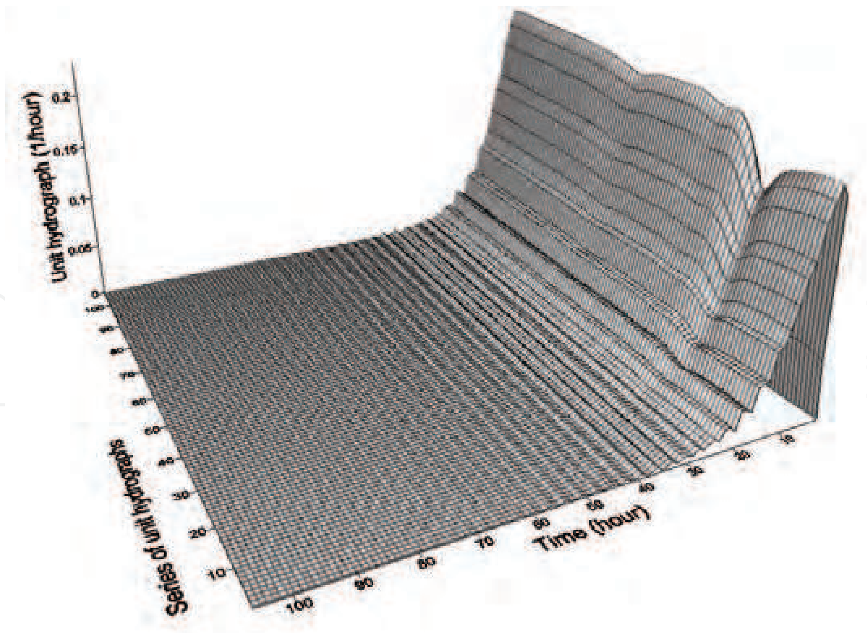


Fig. 13. The transition of UHs of the multi-peak typhoon in Sep. 27, 1996.

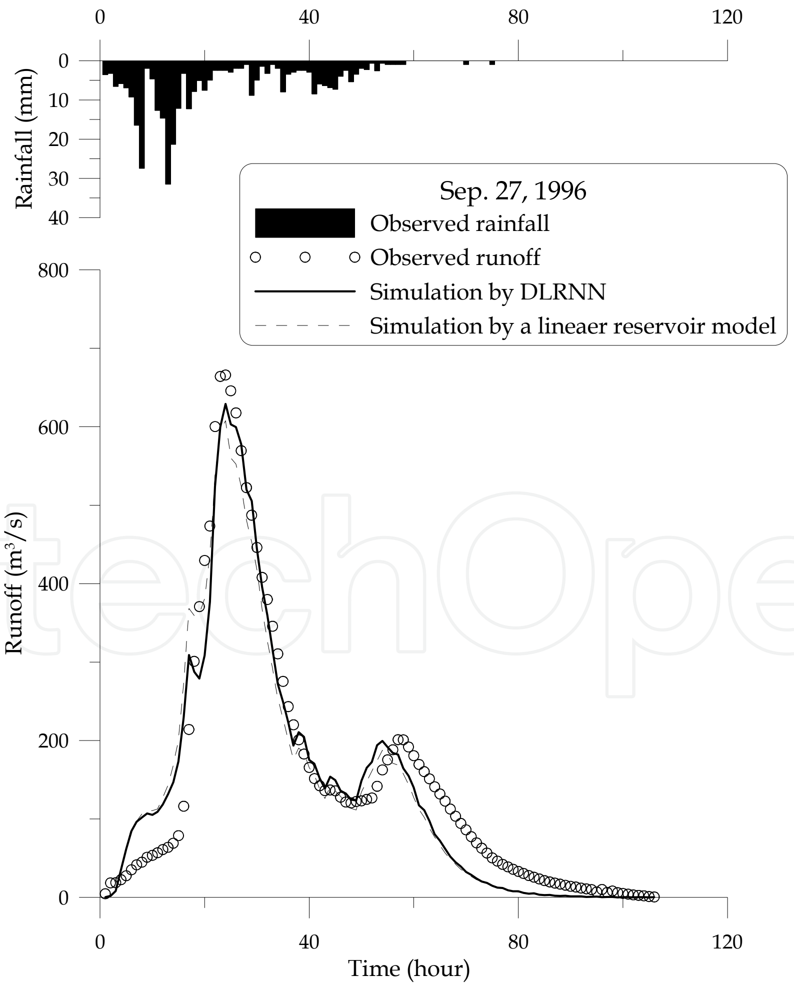


Fig. 14. Simulation of Zane typhoon in Sep. 27, 1996 via DLRNN with on-line learning and a linear reservoir model.

A generalized UH identified from multi-event rainfall-runoff records can represent the hydrological feature of the watershed. However, due to the complex interaction with other hydrometeorological and geomorphological processes within the hydrological cycle, the true UH of a rainfall-runoff process can not be predetermined before the event happens. DLRNN has the capability to shape the generalized UH to catch the transition of rainfall-runoff processes by real time modifying weights. The case study shows that the representation of UHs from DLRNN weights and the tracing ability of the DLRNN. The transition of the rainfall-runoff processes is visualized by the representation of UHs that furthers the interpretation of DLRNN weights.

7. Conclusion

In this chapter, the application of a DLRNN is demonstrated to simulate rainfall-runoff processes and recognize the transition of UHs in hydrology. Although most neural networks are black-box models that lack physical meanings of weights, the DLRNN developed in this chapter connects its weights with UHs that reveal the physical concepts from the network based on the special structure of RNNs. Without trial and error method, the structure and the weights of DLRNN can be quickly determined through a modified form of system identification that combines indirect system identification with the subspace algorithm. Then, the DLRNN learning algorithm based on the interchange of the roles of the network state variables and the weight matrix is derived for on-line training.

In this chapter, the DLRNN introduced can not only simulate rainfall-runoff processes, but also recognize the transition of UHs. Owing to the feedback connections, DLRNN performs rainfall-runoff simulations as dynamic systems, and the advantage of DLRNN's dynamic feature has been proven after the comparison between DLRNN and FNN. The investigation of the connections between weights and physical meanings is an extension of neural networks applied in hydrological field due to the linearization of the RNN. Based on the linearization, weights of DLRNN are treated as Markov parameters to realize the transition of UHs. Through on-line learning, DLRNN modifies the weights to capture the relation between rainfall and runoff every time step, and the transition of rainfall-runoff processes can be emerged based on the changes of UHs.

Furthermore, a modified system identification that combines indirect system identification with subspace algorithm is described to calibrate the DLRNN. This method determines the quantity of neurons in hidden layer and the weights of the network. It overcomes the drawback of costing time by traditional trial and error search for optimum structure of DLRNN. Additionally, the different forms of DLRNN have also been discussed herein. The results show that the performances of DLRNNs in different forms are close. Hence, the transformation of canonical form can be ignored in the flowchart of simulation via DLRNN. Finally, four criteria have been applied to evaluate the performance of rainfall-runoff simulation via DLRNN. The results show that the performance is satisfactory and DLRNN is competent to simulate dynamic systems, like rainfall-runoff processes.

8. Future research

Although feed-forward neural networks are commonly adopted to solve hydrological problems, applying RNNs to deal with the issues of hydrology is still a novel technique because the structure and the learning algorithm of RNN are more complex than those of FNN. This chapter has demonstrated an example to show how RNN applies to hydrological problems. However, further research is necessary. As Sudheer mentioned (2005), hydrologists have not endeavored to construe the knowledge embedded in the trained ANN models, other than the recent research attempts to assign physical significance to the internal architecture of ANN hydrological models. Therefore, how to abstract more physical interpretations from the weights or the architectures of RNN, like the connection between UHs and the weights of DLRNN, is one of the major issues. Furthermore, in order to clarify some opacity in RNN, the DLRNN mentioned herein is only a single-input-single-output (SISO) system with a nonlinearity-interpretation trade-off. With construing the knowledge embedded in, an ideal multi-input-multi-output RNN without any trade-off for rainfall-runoff simulation is needed.

9. References

Abdelghani M. & Verhaegen M. (1998). Comparison study of subspace identification methods applied to flexible structures. *Mechanical Systems and Signal Processing*, Vol. 12, No. 5, pp. 679-692, ISSN 0888-3270.

Anctil F.; Asce M. & Rat A. (2005). Evaluation of neural network streamflow forecasting on 47 watersheds. *Journal of Hydrologic Engineering*, Vol. 10, No. 1, pp. 85-88, ISSN 1084-0699.

Atiya A.F. & Parlos A.G. (2000). New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 697-709, ISSN 1045-9227.

Boto M.A. & Costa J.S. (1998). A comparison of nonlinear predictive control techniques using neural network models. *Journal of Systems Architecture*, Vol. 44, No. 8, pp. 597-616, ISSN 1383-7621.

Chang L.C.; Chang F.J. & Chiang Y.M. (2004). A two-step-ahead recurrent neural network for stream-flow forecasting. *Hydrological Processes*, Vol. 18, pp. 81-92, ISSN 0885-6087.

Chow T.W.S. & Cho S.Y. (1997). Development of a recurrent sigma-pi neural network rainfall forecasting system in Hong Kong. *Neural Computing and Applications*, Vol. 51, No. 5, pp. 921-927, ISSN 0941-0643.

Chiang Y.M.; Chang L.C. & Chang F.J. (2004). Comparison of static-feedforward and dynamic-feedback neural networks for rainfall-runoff modelling. *Journal of Hydrology*, Vol. 290, pp. 297-311, ISSN 0022-1694.

Chiang Y.M.; Chang F.J.; Jou B.J.D. & Lin P.F. (2007a). Dynamic ANN for precipitation estimation and forecasting from radar observations. *Journal of Hydrology*, Vol. 334, pp. 250-261, ISSN 0022-1694.

Chiang Y.M.; Hsu K.L.; Chang F.J.; Hong Y. & Sorooshian S. (2007b) Merging multiple precipitation sources for flash flood forecasting. *Journal of Hydrology*, Vol. 340, pp. 183-196, ISSN 0022-1694.

Chow V.T. (1959). *Open Channel Hydraulics*, McGraw Hill, ISBN 007085906X, New York.

Coulibaly P; Anctil F.; Rasmussen P. & Bobée B. (2000). A recurrent neural networks approach using indices of low-frequency climatic variability to forecast regional annual runoff. *Hydrological Processes*, Vol. 14, pp. 2755-2777, ISSN 0885-6087.

Coulibaly P.; Anctil F.; Aravena R. & Bobée B. (2001). Artificial neural network modeling of water table depth fluctuations. *Water Resources Research*, Vol. 37, No. 4, pp. 885-896, ISSN 0043-1397.

Coulibaly P. & Baldwin C.K. (2005). Nonstationary hydrological time series forecasting using nonlinear dynamic methods. *Journal of Hydrology*, Vol. 307, pp. 164-174, ISSN 0022-1694.

Coulibaly P. & Evora N.D. (2007). Comparison of neural network methods for infillin missing daily weather records. *Journal of Hydrology*, Vol. 341, pp. 27-41, ISSN 0022-1694.

Gustafsson T. (2001). Subspace identification using instrumental variable techniques. *Automatica*, Vol. 37, No. 12, pp. 2005-2010, ISSN 0005-1098.

Henriques J. & Dourado A. (1998). A multivariable adaptive control using a recurrent neural network. *Proceedings of international conference on engineering applications of neural networks, engineering applications of neural networks*, pp. 118-121, UK, June 1998, Gibraltar.

Karamouz M.; Razavi S. & Araghinejad S. (2008). Long-lead seasonal rainfall forecasting using time-delay recurrent neural networks: a case study. *Hydrological Processes*, Vol. 22, pp. 229-241, ISSN 1099-1085.

Maier H.R. & Dandy G.C. (2000). Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. *Environmental Modelling & Software,* Vol. 15, pp. 101-124, ISSN 1364-8152.

Nagesh Kumar D.; Srinivasa Raju K. & Sathish T. (2004). River flow forecasting using recurrent neural networks. *Water Resources Management*, Vol. 18, pp. 143-161, ISSN 0920-4741.

Pan T.Y. & Wang R.Y. (2004). State space neural networks for short term rainfall-runoff forecasting. *Journal of Hydrology*, Vol. 297, pp. 34-50, ISSN 0022-1694.

Pan T.Y. & Wang R.Y. (2005). Using recurrent neural networks to reconstruct rainfall-runoff processes. *Hydrological Processes*, Vol. 19, pp. 3603-3619, ISSN 0885-6087.

Pan T.Y.; Wang R.Y. & Lai J.S. (2007). A deterministic linearized recurrent neural network for recognizing the transition of rainfall-runoff processes. *Advances in Water Resources*, Vol. 30, pp. 1797-1814, ISSN 0309-1708.

Parlos A.G.; Rais O.T. & Atiya A.F. (2000). Multi-step-ahead prediction using dynamic recurrent neural networks. *Neural Networks*, Vol. 13, pp. 765-786, ISSN 0893-6080.

Ptitchkin V.A. (2001). Models of dynamic neural networks and automatic control systems, *Proceedings of the second international conference on neural networks and artificial intelligence*, Republic of Belarus, October 2001, Minsk.

Rahman M.H.R.F. & Kuanyi Z. (2000). Neural network approach for linearizing control of nonlinear process plants. *IEEE Transactions on Industrial Electronics*, Vol. 47, No. 2, pp. 470-477, ISSN 0278-0046.

Ramos J.; Mallants D. & Feyen J. (1995). State space identification of linear deterministic rainfall-runoff models. *Water Resources Research*, Vol. 31, No. 6, pp. 1519-1531, ISSN 0043-1397.

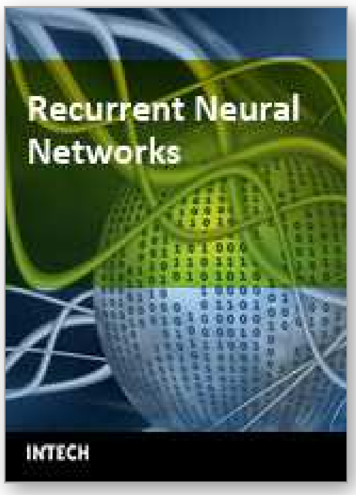
Sudheer K.P. (2005). Knowledge extraction from trained neural network river flow models. *Journal of Hydrologic Engineering*, Vol. 10, No. 4, pp. 264-269, ISSN 1084-0699.

Tsoi A.C. & Back A. (1997). Discrete time recurrent neural network architectures: a unifying review. *Neurocomputing*, Vol. 15, pp. 183-223, ISSN 0925-2312.

Vos N.J. & Rientjes T.H.M. (2005). Constraints of artificial neural networks for rainfall-runoff modeling: trade-offs in hydrological state representation and model evaluation. *Hydrology and Earth System Sciences Discussions*, Vol. 2, pp. 365-415, ISSN 1812-2108.

Walter M; Recknagel F.; Carpenter C. & Bormans M. (2001). Predicting eutrophication effects in the Burrinjuck Reservoir (Australia) by means of the deterministic model SALMO and the recurrent neural network model ANNA. *Ecological Modelling*, Vol. 146, pp. 97-113, ISSN 0304-3800.

Zarmarreño J.M.; Vega P.; García L.D. & Francisco M. (2000). State-space neural network for modelling, prediction and control. *Control Engineering Practice*, Vol. 8, pp. 1063-1075, ISSN 0967-0661.



Recurrent Neural Networks

Edited by Xiaolin Hu and P. Balasubramaniam

ISBN 978-953-7619-08-4
Hard cover, 400 pages
Publisher InTech
Published online 01, September, 2008
Published in print edition September, 2008

The concept of neural network originated from neuroscience, and one of its primitive aims is to help us understand the principle of the central nerve system and related behaviors through mathematical modeling. The first part of the book is a collection of three contributions dedicated to this aim. The second part of the book consists of seven chapters, all of which are about system identification and control. The third part of the book is composed of Chapter 11 and Chapter 12, where two interesting RNNs are discussed, respectively.The fourth part of the book comprises four chapters focusing on optimization problems. Doing optimization in a way like the central nerve systems of advanced animals including humans is promising from some viewpoints.