LITERATURE REVIEW

The following literature review is divided into three main topics, and it provides some background on related work and concepts in this field of study. The first topic covers basic theory of optimization model and differential evolution approach and its application to a variety of optimization problems in field of water resources engineering. The second topic review pertaining to artificial neural network and intelligent control, and their application to various water resources management problems. Finally, the third topic describes and examines the technical features of the River Operation Model (ROM). The ROM was used as unsteady hydrodynamic and water quality simulation modeling and then was linked by optimization mathematical modeling, i.e. differential evolution for determination of the optimal costal gate operation strategies.

1. Differential Evolution Algorithm

1.1 General concepts of optimization problems

In computer science, optimization problem concerns the problem of finding the best solution from all feasible solutions. More formally, it finds a solution in the feasible region which has the minimum (or maximum) value of the objective function (Black, 1999). The aim of engineering design is to obtain the maximum benefit and at the same time requires minimization of any effects resulting from the design as well. Hence, most engineering problems are optimization problems (Price and Storn, 1997a). Typically, optimization method can be classified into two types: deterministic search algorithm and stochastic search algorithm. The deterministic search algorithm is conventional method, which is based on gradient method by means of calculus. The examples of this method are linear programming, integer programming, nonlinear programming, dynamic programming, etc. From many researches in the past, it was found that the disadvantage of the application of deterministic search algorithm in solving engineering problems, especially relatively

complex problems, is to obtain local optimum solution instead of global optimum solution. For this reasons, in recent years the stochastic search algorithm become more and more popular approach for optimization problems because of its capability in solving complicated problem such as nonlinear function, non-differentiate function, and multi-model optimization (Hendershot, 2004) and in obtaining global optimum solution. Genetic Algorithm (GA) (Goldberg, 1989), Simulated Annealing (SA) (Kirkpatrick *et. al.*, 1983), and Differential Evolution (DE) (Price and Stron, 1997b) are the examples of stochastic search algorithm (Prempree 2005; Charoenongart 2005). In addition, the main difference between stochastic search algorithm and deterministic search algorithm is that the first method only require information regarding objective function in solving the problems (Babu and Angira, 2003; Babu *et.at.,* 2003).

Currently, optimization techniques have been successfully applied in many water resources engineering problems. Unver and Mays (1990) employed a nonlinear programming model for real-time optimal flood operation of river-reservoir systems. Kuo (1995) used genetic algorithms for optimizing the benefits of water and crop management in an irrigation project. Ko *et al.* (1997) applied dynamic programming to the problems of multiobjective analysis of service-water-transmission systems. Rauch and Harremoes (1999) utilized genetic algorithms in real time control of urban wastewater systems. Basri (2001) deployed nonlinear programming and genetic algorithms for optimal design of subsurface barrier to control seawater intrusion. Fayad (2001) applied genetic algorithms for solving conjunctive water use problems. Laksanapanyakul (2001) used dynamic programming for management of water quality in a river. Tooychai (2001) applied stochastic dynamic programming for reservoir operation planning. Sethi *et al*. (2002) developed linear programming for determination of optimal crop planning and conjunctive use water resources in a coastal river basin. Muleta and Nicklow (2004) developed decision support model for watershed management using evolutionary algorithms. Shie-Yui *et al*. (2004) applied Non-dominated Sorting Genetic Algorithm-II (NSGA-II), for reservoir operation problem. Bhattacharjya and Datta (2005) applied genetic algorithms for optimal

management of coastal aquifers. Kulthai (2005) used linear and integer programming for estimating dry season cropping area.

1.2 Theory of differential evolution algorithm

Differential Evolution (DE) was initially introduced and developed by Price and Storn in 1996 in an attempt to solve the Chebychev Polynomial fitting Problem. DE is an evolutionary optimization technique or population-based optimization algorithm based on stochastic approach. It is an improved version of Genetic Algorithms (GA) for simple structure, ease of use, speed and robustness (Babu and Angira, 2003). The principal difference between GA and DE is that GA relies on crossover, which is a mechanism of probabilistic and useful exchange of information among solutions to locate better solutions whereas DE use mutation as the primary search mechanism. As with all evolutionary optimization algorithms, DE operates on a population, P_G, of candidate solutions, not just a single solution. DE uses a non uniform crossover that can take child vector parameters from one parent more often than it does from others. By using components of existing population members, it can construct trial vectors, recombine efficiently shuffles information about successful combinations, and enable the search for an optimum to focus on the most promising area of solution space (Vasan and Raju, 2005).

At present, there are numerous variants of DE. Price and Storn (1997a) gave the working principle of DE with single strategy. Later on, they suggested ten different strategies; namely:- DE/rand/1/bin, DE/best/1/bin, DE/best/2/bin, DE/rand/2/bin, DE/randtobest/1/bin, DE/rand/1/exp, DE/best/1/exp, DE/best/2/exp, DE/rand/2/exp, DE/randtobest/1/exp (Price and Storn, 2006). DE/x/y/z indicates DE for Differential Evolution, x is a string which denotes the vector to be perturbed, y denotes the number of difference vectors taken for perturbation of x, and z is the crossover method. The special variant used throughout this investigation is the DE/rand/1/bin scheme. This scheme will be discussed here and more detailed descriptions are provided. Since the DE algorithm was originally designed to work

with continuous variables, the optimization of continuous problems is discussed first. Handling discrete variables is explained later.

In general, the function to be optimizes, *f*, is the form:

$$f(x): R^D \rightarrow R \tag{1}$$

The optimization target is to maximize or minimize the value of this objective function *f(x)*,

$$\max \text{ or } \min \left(f(x)\right) \tag{2}$$

by optimizing the values of its parameters:

$$X = (x_1, x_2, x_3, ..., x_D), \quad X \in R^D \tag{3}$$

where X denotes a vector composed of D objective function parameters. Typically, the parameters of the objective function are also subject to lower and upper boundary constraints, $x^{(L)}$ and $x^{(U)}$, respectively:

$$x_j^{(L)} \leq x_j \leq x_j^{(U)} \quad j = 1, 2, 3, ...D \tag{4}$$

DE's algorithm consists of four main processes: initialization, mutation, crossover, and selection (Price and Storn, 1997a; Price and Storn, 1997b; Srinivas and Rangaiah, 2007; Onwubolu and Dravendra, 2006). The process starts with specifying DE's parameters, i.e. maximum number of generations, population size (NP), weighting factor (F), and crossover constant (CR). Then the four main steps are carried out as follows.

1) Initialization: As with all evolutionary optimization algorithms, DE works with a population of solutions (or candidate solutions), not just a single solution for optimization problem. The population, *P,* of generation, *G*, contains constant population size, *NP*, solution vectors calls individuals of the population (or candidate solutions) and each vector represents potential solution for the optimization problem.

$$P_G = X_{i,G} \quad i = 1,2,3,...,NP, \; G = 1,2,3,..G_{\max} \tag{5}$$

where *i* is index of the population and *G* is the generation to which the population belongs. Additionally, each vector contains *D* real parameters (chromosome of individuals):

$$X_{i,G} = x_{j,i.G} \quad i = 1,2,3,...,NP, \; j = 1,2,3,...,D \tag{6}$$

In order to determine a starting point for optimal seeking, the population must be initialized. Due to no more knowledge available about the location of a global optimum, the initial population is randomly generated from specified boundary constraints using the uniformly distribution random numbers to cover the entire solution space.

$$P_0 = x_{j,i,0} = x_j^{(L)} + rand_j[0,1] \cdot \left(x_j^{(U)} - x_j^{(L)} \right) \quad i = 1,2,3,...,NP, \;\; j = 1,2,3,...,D \tag{7}$$

where $rand_j[0,1]$ represents a uniformly distributed random value within range: [0.0,1.0] that is chosen a new for each *j*. After generation of initial population, the objective function values of all the individuals are calculated and the best solution is determined.

2) Mutation: DE uses a self-referential population recombination scheme, which is different from the other evolutionary algorithms, for creating new generation. From the first generation onward, vectors in the current population, P_G, are randomly sampled and combined to create candidate vectors for the subsequent

generation, P_{G+1}. The population of candidate, or mutant vectors, $P'_{G+1} = V_{i,G+1}$, is generated as follows:

$$V_{i,G+1} = X_{r1,G} + F\left(X_{r2,G} - X_{r3,G}\right),\ \text{i=1,2,3,...,NP} \tag{8}$$

where r_1, r_2 and r_3 belongs to set {1,2,3,…, NP} and $X_{r1,G}$, $X_{r2,G}$ and $X_{r3,G}$ represents the three random individuals chosen in the current generation, G, to reproduce the mutant vector for the next generation $V_{i,G+1}$. The random numbers r_1, r_2 and r_3 should be different from each other and also different from the running index, i. Hence, NP should be at least 4 to allow mutation. F is weighting factor, which is the real number between 0 and 2. This value is used to control the amplification of the differential variation between the two random vectors.

3) Crossover: This process is performed to increase the diversity of the perturbed parameter vectors. In this step, the trial vector, $U_{i,G+1}$ is produced by duplicating some elements of the mutant vector, $V_{i,G+1}$ or some elements of the target vector, $X_{i,,G}$ with proability equal to CR. For the first generation, target vector is the best vector from all individuals in initial population and it is the best vector from all individuals obtained from selection process (as will be described in the next topic) for subsequent generation. As shown in Figure 1, a random number (ran) is generated for each element of the target vector. If $randb(j) \leq CR$ or $j = rnbr(i)$, the element of mutant vector is copied, otherwise the target vector element is copied. The trial can be expressed as:

$$U_{i,G+1} = \left(U_{1i,G+1}, U_{2i,G+1}, ..., U_{Di,G+1}\right) \tag{9}$$

and the crossover process can be presented in mathematical form as:

$$U_{i,G+1} = \begin{cases} V_{i,G+1} & if\ \left(randb(j) \leq CR\right)\ or\ j\ =\ rnbr(i) \\ X_{i,G} & if\ \left(randb(j) > CR\right)\ or\ j\ \neq\ rnbr(i) \end{cases} \tag{10}$$

where *randb(j)* is the j[th] evaluation of a uniform random number generator with outcome ∈ [0,1]; *CR* is the crossover constant ∈ [0,1] which has to be determined by the users; *rnbr(i)* is a randomly chosen index ∈ 1,2,3,..,D which ensures that $U_{j,i,G+1}$ gets at least one parameters from $V_{j,I,G+1}$. Usually, suitable values for F, CR and NP can be found by experimentation after a few tests using different values. Practical advice on how to select control parameters NP, F and CR can be found in Storn and Price (1997a). A reasonable first guess is: F = 0.9, CR = 0.9 and NP = 10D.

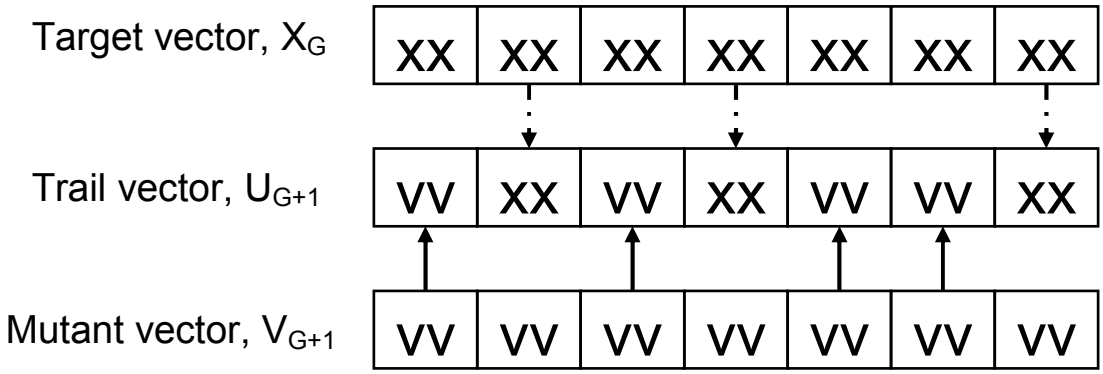


Figure 1 Illustration of the crossover process for $D = 7$ parameters.
Source: Modified from Stron and Price (1997); Srivinas and Rangaiah (2006).

4) Selection: The selection scheme of DE also differs from other evolutionary algorithms. On the basis of the target vector of current population, $X_{j,i,G}$ and the trail vector of next population $V_{j,i,G+1}$, the child population, $X_{j,i,G+1}$ is created as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} = u_{j,i,G+1} \text{ if } f\left(U_{i,G+1}\right) \text{ better than } f\left(X_{i,G}\right) \\ X_{i,G} \text{ otherwise} \end{cases} \tag{11}$$

In equation 11, the trial vector $U_{i,G+1}$ is compared to the target vector $X_{i,G}$ using the greedy criterion. If vector $U_{i,G+1}$ yields a better cost function value than $X_{i,G}$, then $X_{i,G+1}$ is replaced by the trial vector $U_{i,G+1}$; otherwise, the old value of the target vector $X_{i,G}$ is retained. The process of mutation, crossover, and selection is repeated until a

termination criterion such as maximum number of generation is satisfied. The algorithm then terminates proving the best point that has been explored over all the generations. The overall process for DE is presented in Figure 2. Figure 3 also shows the example of using DE for solving a simple objective function: $f(x) = x_1 + x_2 + x_3 + x_4 + x_5$.

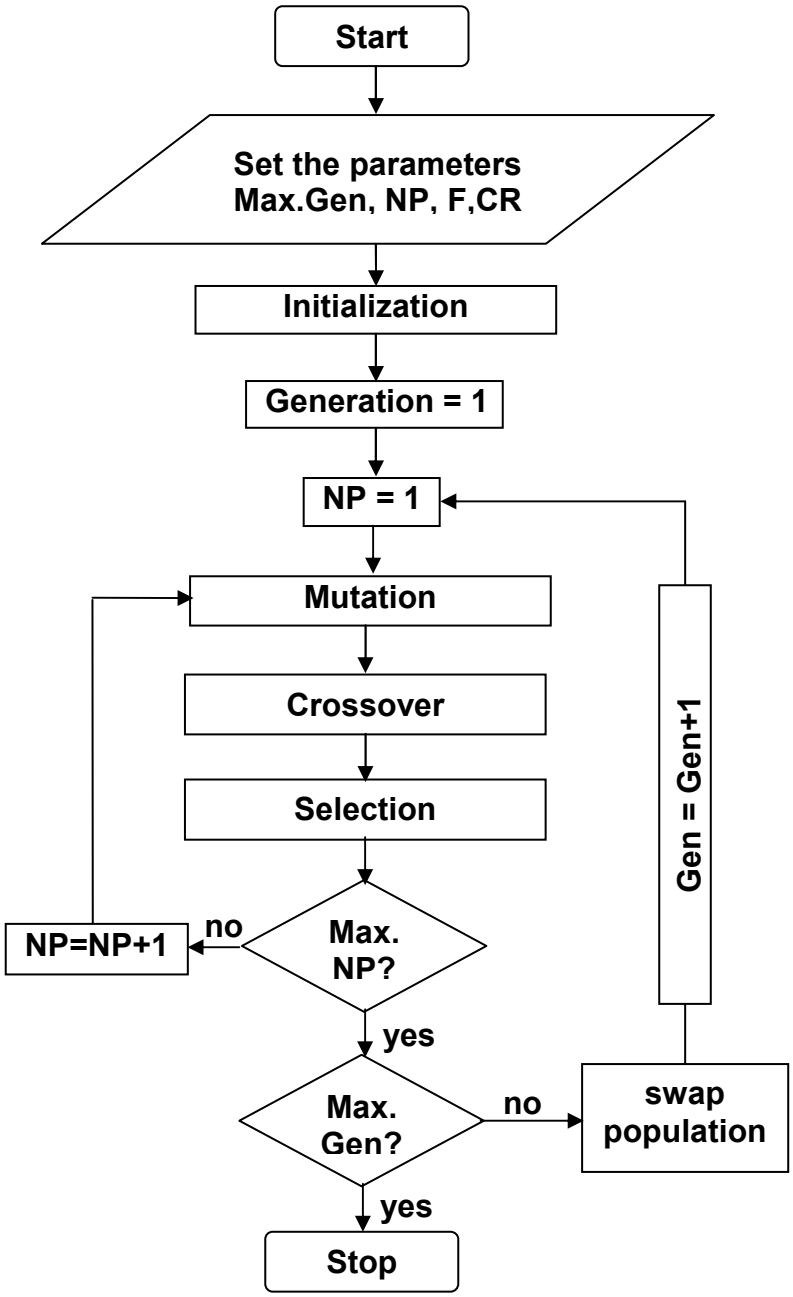


Figure 2 The overall processes of DE.

Source: Modified from Srivinas and Rangaiah (2006).

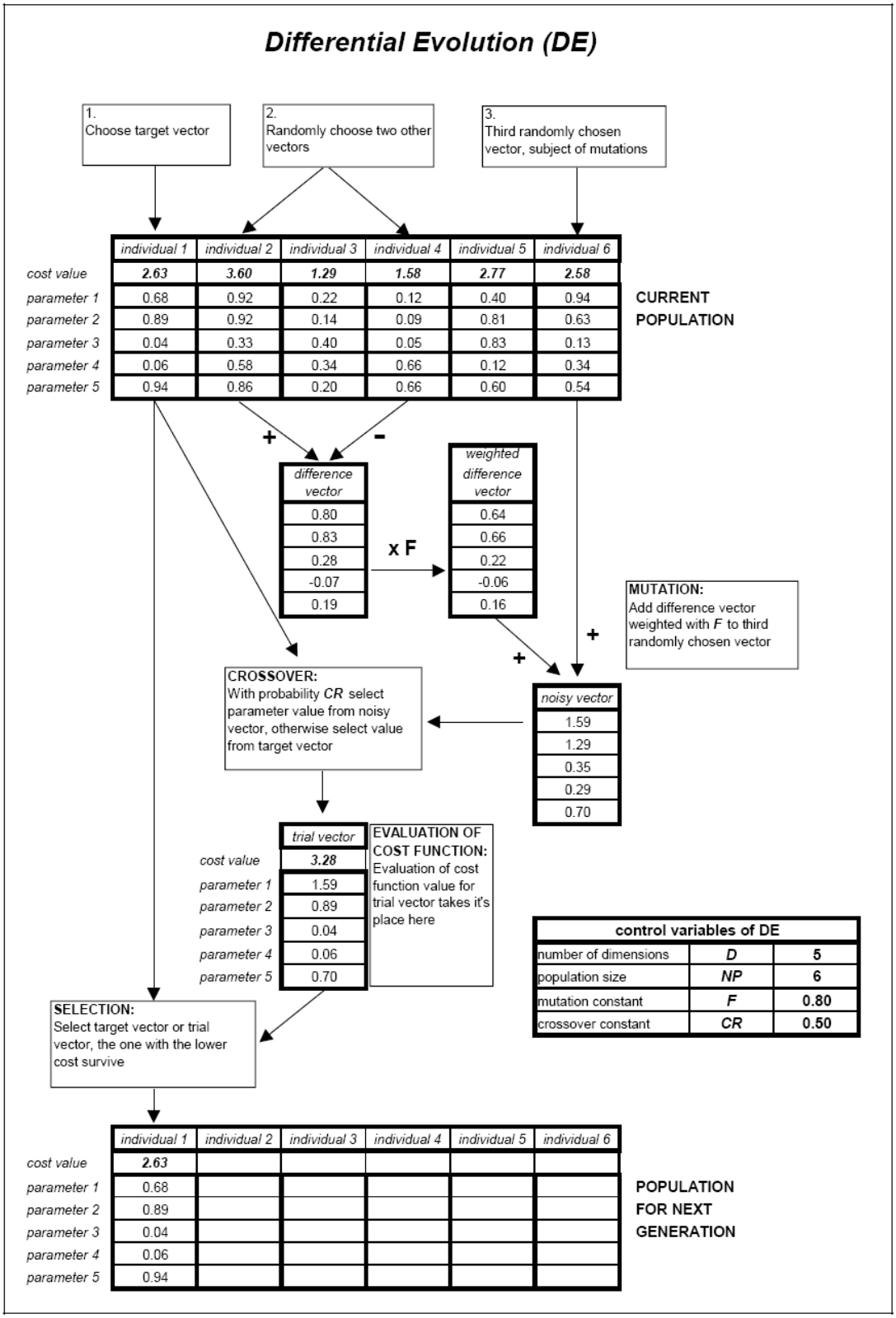


Figure 3 The example of using DE for solving a simple objective function. Source: Lampinen (2006).

1.2.1 Handling simple boundary constraints

It is important to notice that as results of the recombination operation of DE, it is possible to extend the search outside of the initialized range of the search space. It is also worthwhile to notice that sometimes this is a beneficial property in problems with no boundary constraints because it is possible to find the optimum that is located outside of the initialized range. However, in boundary-constrained problems, it is essential to ensure that parameter values lie inside their allowed ranges after recombination. Several methods have been proposed to cope with this problem (Lampinen, 2006; Onwubolu and Dravendra, 2006). A simple way to guarantee this is to replace parameter values that violate boundary constraints with random values generated within the feasible range:

$$U_{i,G+1} = \begin{cases} x_j^{(L)} + rand_j[0,1] \times \left(x_j^{(U)} - x_j^{(L)} \right), & \text{if } U_{i,G+1} < x_j^{(L)} \text{ or } U_{i,G+1} > x_j^{(U)} \\ U_{i,G+1} \text{ otherwise,} \end{cases} \tag{12}$$

where i ∈ [1,NP]; j ∈ [1,D].

Another simple but less efficient method is to reproduce the boundary constraint violating values according to equation 10 as many times as is necessary to satisfy the boundary constraints. The simple method that allows bounds to be approached asymptotically while minimizing the amount of disruption that results from resetting out of bound values (Price, 1999) is

$$U_{i,G+1} = \begin{cases} \left(x_j^{(G)} + x_j^{(L)} \right) \Big/ 2, & \text{if } U_{i,G+1} < x_j^{(L)}, \\ \left(x_j^{(G)} + x_j^{(U)} \right) \Big/ 2, & \text{if } U_{i,G+1} > x_j^{(U)}, \\ U_{i,G+1} \text{ otherwise.} \end{cases} \tag{13}$$

Another method presented and then applied in this study is to replace parameter values that violate boundary constraints with boundary value.

$$U_{i,G+1} = \begin{cases} x_j^{(L)}, & \text{if } U_{i,G+1} < x_j^{(L)}, \\ x_j^{(U)}, & \text{if } U_{i,G+1} > x_j^{(U)}. \end{cases} \tag{14}$$

1.2.2 Handling discrete variables

As described above, the DE algorithm was originally designed to work with continuous variables but the determination of optimal coastal gate operation involves solving the discrete variable problem. Therefore, it is necessary to investigate approaches to deal with discrete variable optimization. Round off method, which round off the variable to the nearest available value before evaluating each trial vector, is a simple, popular and effective way. To keep the population robust, successful trial vectors must enter the population with all of the precision with which they were generated (Price and Storn, 1997b). In its canonical form, the differential evolution algorithm is only capable of handling continuous variables. Extending it for optimization of integer variables, however, is rather easy. This is the method that was used for this work. In addition, the initialization of population and handling boundary constraint method should be modified as presented in equations 15 and 16 instead of equations 7 and 12, respectively (Lampinen, 2006; Onwubolu and Dravendra, 2006).

$$P_0 = x_{j,i,0} = x_j^{(L)} + rand_j[0,1] \cdot \left(x_j^{(U)} - x_j^{(L)} + 1 \right) \quad i = 1,2,3,...,NP, \quad j = 1,2,3,...,D \tag{15}$$

$$U_{i,G+1} = \begin{cases} x_j^{(L)} + rand_j[0,1] \times \left(x_j^{(U)} - x_j^{(L)} + 1 \right), & \text{if INT}(U_{i,G+1}) < x_j^{(L)} \text{ or INT}(U_{i,G+1}) > x_j^{(U)} \\ U_{i,G+1} \text{ otherwise,} \end{cases}$$

(16)

1.3 Multiple objective optimization problem in DE

The coastal gate operation problem concerns in an attempt to compromise several water quantity and quality parameters simultaneously. Hence, in this topic, the basic theory of multiple-objective optimization is reviewed. In principle, multiobjective optimization is very different from single-objective optimization. In single-objective optimization, one attempts to obtain the best design or decision, which is usually the global minimum or the global maximum, depending on whether the optimization problem is one of minimization or maximization. On the other hand, the multiple objectives may not be one solution that is the best (global minimum or maximum) with respect to all objectives. Solutions to a multiobjective optimization problem are mathematically expressed in terms of nondominated or superior point. In fact, none of the solutions in the nondominated set is absolutely better than any others, any one of them is an acceptable solution. Thus, one solution chosen by a designer may not be acceptable to another designer or in a changed environment (Srinivas and Deb, 1994).

In general, multiobjective optimization problem consists of a number of objectives and is associated with a number of inequality and equality constraints. The problem can be mathematically written as follows (Rao, 1991).

$$Minimize / Maximize\ f_i(x) \quad i = 1, 2, ..., N \tag{17}$$

Subject to

$$g_i(x) \le 0 \quad j = 1, 2, ..., J$$

$$b_k(x) = 0 \quad k = 1, 2, ..., K$$

The parameters x is a p dimensional vector having p design or decision variables.

The classical method for solving multiobjective optimization is to scalarize the objective vector into one objective. For example, the method of

objective weighting, the simplest of all classical techniques, combines multiple objective functions into one overall objective function, Z, as follows:

$$Z = \sum_{i=1}^{N} w_i f_i(x) \tag{18}$$

where $x \in X$, X represents the feasible region; the weight w_i are fractional numbers ($0 \le w_i \le 1$), and all weight are summed up to 1, or $\sum_{i=1}^{N} w_i = 1$. From equation 18, it shows that the optimal solution is controlled by the weight vector w. In addition, the preference of an objective can be changed by modifying the corresponding weight. Mathematically, a solution obtained with equal weights to all objectives may offer least objective conflict; however, as a real-world situation demands a satisfying solution, priority must be induced in the formulation. The advantage of using this method is that the emphasis of one objective over the other can be controlled, and the obtained solution is usually a Pareto-optimum solution. To apply the classical method for solving multiobjective optimization, the decision makers require knowledge of the individual optimum prior to vector optimization. For different situations, different weight vectors need to be used and the same problem need to be solved a number of times to obtain various alternative Pareto-optimal solutions instead of a Pareto-optimal solution.

Although several new techniques of differential evolution approach have been developed for serving the multiobjective optimization solutions, for example Differential Evolution for Multiobjective Optimization (DEMO), Pareto Differential Evolution (PDE) algorithm, and Vector Evaluated Differential Evolution (VEDE) (Madavan, 2002; Robic and Filipic, 2005; Parsopoulos *et al*., 2006), the weighting method is still chosen in this research. This is because it is quite comfortable for developing model and also writing computer program. Furthermore, one of the most important reason to select weighting method in this study is that such method requires computation time lesser than the new ones due to only solving a single point of solution. Hence, it is more appropriate to utilize this method for the problems concerning sequential optimal control, which involves calling the mathematical

simulation model several thousands of times for each time step of control horizon. Consequently, it leads to the requirement of a significant amount of the computation time for determining optimal control variables throughout control horizon. The weighting method has ever been used in the development of AQUARIUS model, which is an aiding tool for operational management of regional water systems by several water boards of the Netherlands (Lobbrecht and Solomatine, 1999; Lobbrecht et al., 2005).

1.4 Application of DE in water resources engineering problems

Differential Evolution was successfully applied in various optimization problems (Lakari *et al*., 2003; Hrstka and Kucerova, 2004; Onwubolu and Davendra, 2006). It is interesting to review some papers in which differential evolution has been applied to water resource engineering problems as follows.

Babu and Angira (2003) used differential evolution approach to solve a classical optimization problem of water pumping system. The objective function used in this study is to minimize increased pressure due to pump. Comparison is made with Branch & Reduce algorithm in terms of the number of objective function evaluations. The results indicated that the performance of DE is better than the Branch & Reduce algorithm.

Charoenongart (2005) applied differential evolution to the problem of an optimal planning process for pipe replacement of the pipe distribution network. The EPANET model, which is mathematical model for study of behaviors of water flow and water quality in piping system, was linked with computer code of differential evolution. The main objective of developed model is to find the most necessary replaced pipelines under the determined constraints of the pipe network. To demonstrate and evaluate the developed model, two case studies, including the pipe distribution networks of Nakhon Sawan province and of Wangthong House village, Bueng Kum District of MWA area, were modeled and simulated for determining pipelines to be replaced under the consideration of limited budget, pipe lifetime, and

hydraulic constraints. The study results illustrated that the developed could select the most appropriate pipelines to be replaces with reasonable replacement cost closed to the limited budget. As a result, the developed model is valuable for planners to effectively plan for replacement schedule and with the most beneficial use of available budget.

Prempree (2005) developed methodology for optimal design of pipe size and pump type in water distribution system using differential evolution algorithm. In this research, the computer simulation model was developed by integrating differential evolution approach and existing hydraulic simulation model, EPANET, together. The developed model was verified by using three case studies of water distribution network of Nakhon Sawan province. The first case study only concerned finding least cost of pipe network. The second case considered both pipe and operation cost of pipe network. Finally, the last case considered the optimal cost of the system including pipe cost, operation cost, and the most cost of appropriate pump. The study results showed that differential evolution could search the better optimal solution when compared to using original method and simulated annealing method under the same conditions. Thus, it confirmed that differential evolution algorithm is an efficient optimization method, which was succefully applied for the discrete optimization problem in the design of water distribution system.

Thasaduak and Chittaladakorn (2005) applied differential evolution for searching the optimal reservoir operation rule for efficient use of water corresponding to water supply and water demand. They proposed the decision making process for reservoir operation through the development of objective function. The developed model was composed of water balance model, and optimization model. To test the efficient of differential evolution for solving the problem of reservoir operation, the Mae Ngad Somboon Chol dam was used as a case study. The results showed that the Differential Evolution was the efficient method for searching the optimal operation rule curve with relatively quick converging to the solution.

Vasan and Raju (2005) used differential evolution to determine the suitable cropping pattern, which yields maximum net benefits, of Bisalpur project, Rajasthan, India. In this work, the performance of ten different strategies of DE for finding optimal cropping pattern was investigated with various needed DE's parameters, namely population size, crossover constant and weighting factors. In addition, the results of DE were also compared with the solution of Linear Programming (LP). It was concluded that DE can be appropriately applied for irrigation planning problem.

Based on available literature, it was found that no prior work applying DE or other optimization techniques has been done in which a model for planning coastal gate operations is developed.

2. Artificial Neural Networks

Artificial Neural Networks (ANNs) was first developed in 1940s. It is an information processing system that roughly emulates the behavior of a human brain by replicating the operations and connectivity of biological neurons (Tsoukalas and Uhrig, 1997). In general, neural network do not need detailed description or formulation of the underlying physical processes that it attempts to model. Furthermore, neural network is refers to as connectionist systems or distributed parallel processing systems. It is also comprised of a set of highly interconnected but simple processing units, called nodes or neurons, each responsible for carrying out a few rudimentary computations (Khalil, 2002). The nodes are usually organized into groups called layers. Each layer is referred to as an input layer, a hidden layer, and an output layer.

Figure 4 shows a typical neural network structure. The nodes in one layer are connected to those in the next layer or to those in the same layer through functional linkages, which are weighted to represent their connection strengths (Fayad, 2001; Khalil, 2002). Thus, the output of a node in any layer is determined by applying a

nonlinear transforming, sometimes called activation function, to the sum of the weights input which receive from the nodes of previous layers.

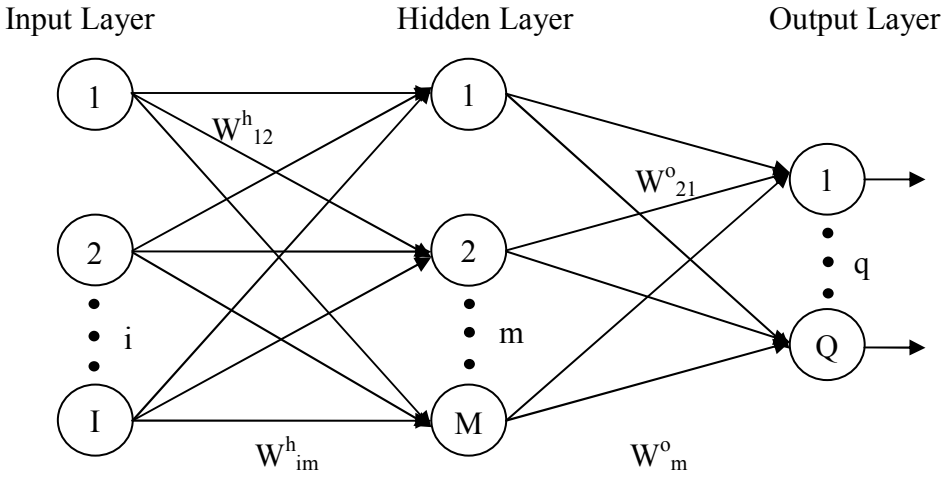


Figure 4 A typical neural network structure.

Neural network performs two major functions: learning and recall (Tsoukalas and Uhrig, 1997). Learning is referred to the process that adjusts the connection weights in network in order that the calculated output vector is close to desired output vector as much as possible. Conversely, recall is the process that receives an input data and then produces an output respond in accordance with the trained network. Typically, there are three types of learning process: supervised learning, unsupervised learning (i.e., self-organization), and reinforcement learning (Rogers and Dowla, 1994; Tsoukalas and Uhrig, 1997).

In supervised learning, series of connecting weights are adjusted in order to fit the series of inputs to another series of known outputs. That is, knowledge is acquired by the network through a learning process, and synaptic weights are used to store the knowledge. On the other hand, in unsupervised learning, there is no specific respond sought, but rather the respond is based on the networks ability to organize itself. Only the input stimuli are applied to the input buffers of the networks. The network then organizes itself internally so that each hidden neuron responds strongly to a different set of input stimuli. These sets of input stimuli represent clusters in the input space.

Reinforcement learning relies on a “tutor” grading the ANN’s output responses to the training patterns. A high grade results in synaptic weight reinforcement; a low grade results in an adjustment of the weights to determine whether the grade can be improved.

The vast majority of learning in engineering application, especially in water resources engineering problems, involves supervised learning (Lobbrecht and Solomatine, 1999; Lobbrecht and Solomatine, 2002; Lobbrecht *et al.*, 2005; Darsono and Labadie, 2006). As a result, this learning process is deployed in this dissertation research. In the following topic, two types of supervised learning algorithms, namely:- feed forward back propagation algorithm and generalized regression neural networks, are discussed in more detailed information.

2.1 Feed forward back propagation algorithm

Multilayer feed forward network with BP learning algorithm is one of the most popular neural network architectures, which has been deeply studied and widely used in many fields. The feed-forward architecture allows connections only in one direction, i.e. there is no back-coupling between neurons, and the neurons are arranged in layers, starting from a first input layer and ending at the final output layer with one or more hidden layers. The information passes from the input to the output side. Figure 4 also presents a typical three-layered feed-forward architecture (Rumelhart *et al.*, 1986). The three layers consist of an input layer, a hidden layer and an output layer. Each layer is made up of several neurons, and the layers are interconnected by sets of weights. The neurons in the input layer receive input directly from the input variables. The neurons in the hidden and output layers receive input from the interconnections. Neurons operate on the input and transform it to produce an analogue output. The transformation is performed in two stages. First, input from each neuron is multiplied with weights and a weighted sum is performed. Next, an activation function such as the sigmoid function converts such a weighted sum to be the output of each neuron, which becomes the input to neurons of the succeeding layer.

Learning is normally accomplished through an adaptive procedure or algorithm that incrementally adjusts weights of the connections in order to improve a predefined performance measure, such as average absolute error, R-squared, mean squared error. Due to supervised learning algorithm, the input data and expected (or output) data are specified. Initially, since the weights to the interconnections are randomly generated, the difference between the predicted and desired output values can be large. Learning therefore involves iteratively adjusting the connection weights to minimize these differences. And the other two parameters: learning rate, α, and a momentum rate, β, are selected. The learning rate, α, controls the incremental change in the interconnection weights during iterative training as a percentage of the difference between the desired or target output and the NN computed output. Thus, a high learning rate generally results in a larger weight change and faster learning. The momentum rate is a means to increase the rate of learning and at the same time, avoiding the possibility of getting trapped in local optima. The momentum rate is a multiplication factor to the change in the previous interconnection weights (Sivapragasam and Muttil, 2005).

.

Training of back propagation neural network involves two stages (Kumar *et al.*, 2002). In the first stage (forward pass), the input signals propagate from the network input to the output. The calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the following layer. In the second stage (backward pass), the calculated error signals propagate backward through the network, where they are used to adjust the weights. The weights of the output neuron layer are adjusted first because the target value of each output neuron is available to guide the adjustment of the associated weights. The learning process of back propagation neural network can be illustrated in Figure 5.

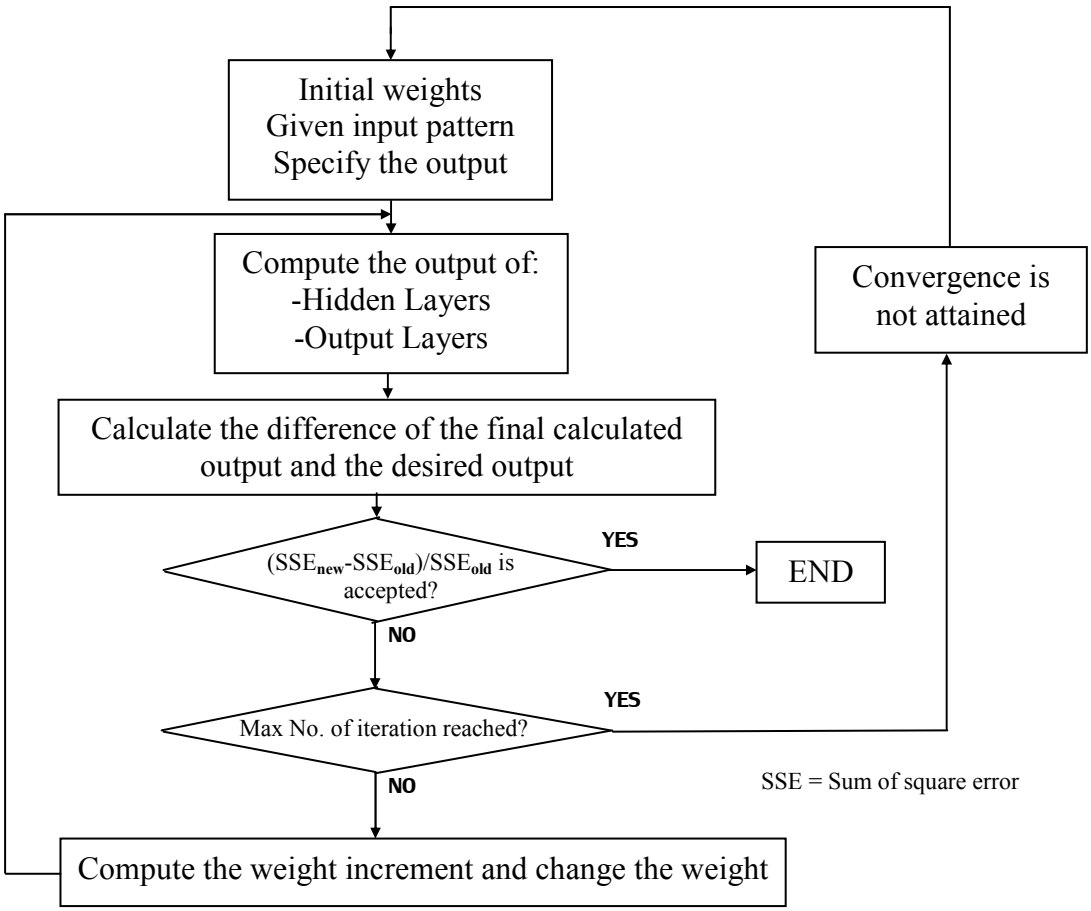


Figure 5 Flowchart of the Back Propagation Algorithm.
Source: Manusthiparom (2000).

2.2 Generalized regression neural networks

The generalized regression neural network (GRNN) was originally developed by Nadaraya (1964) and Watson (1964) and rediscovered by Specht (1991) to perform general (linear or nonlinear) regressions. The GRNN is a special extension of the radial basis function network (RBFN) (Tsoukalas and Uhrig, 1997). This type of neural network was utilized to solve a variety of problems such as prediction, control, plant process modeling or general mapping problems (Patterson, 1995). The GRNN (Tsoukalas and Uhrig, 1997; Niwa, 2003) is a memory-based feed forward network, consisting of 4 layers: input, hidden or pattern, summation, and output layers as illustrated in Figure 6. Whereas the neurons in the first three layers are fully

connected, each output neuron is connected only to some processing units in the summation layer. In the individually pattern units, they compute their activation using a radial basis function (RBF) instead of the sigmoid activation function often used in neural networks. The number of inputs is equal to the number of independent features. In general, the Gaussian kernel function as depicted in Figure 7 is used where σ is the width of the radial function or smoothing parameter. Typically, the larger values of the smoothing parameter of the pattern units lead to the smoother interpolation of the output vectors values. On the other hand, the smaller values of the smoothing parameter of the pattern units result in the hazard that wild points may have too great an effect on the estimate (Tsoukalas and Uhrig, 1997; Cigizoglu and Alp, 2006). The summation layer has two different types of processing units: the summation units and a single division unit. The number of the summation units is always the same as the number of the GRNN output units. The division unit only sums the weighted activations of the pattern units without using any activation function. Each of the GRNN output units is connected only to its corresponding summation unit and to the division unit; there are no weights in these connections. The function of the output units is to combine the signal received from the summation unit and that received from the division unit.

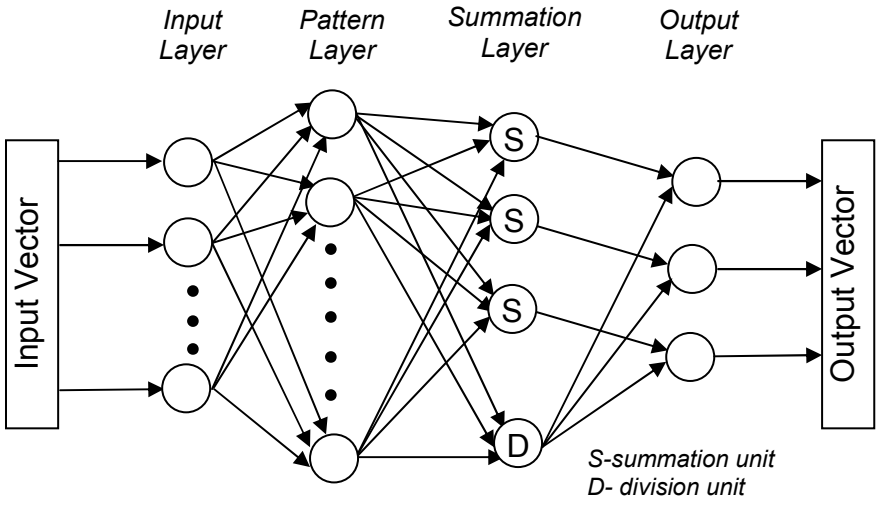


Figure 6 The architecture of the generalized regression neural network. Source: Tsoukalas and Uhrig (1997).

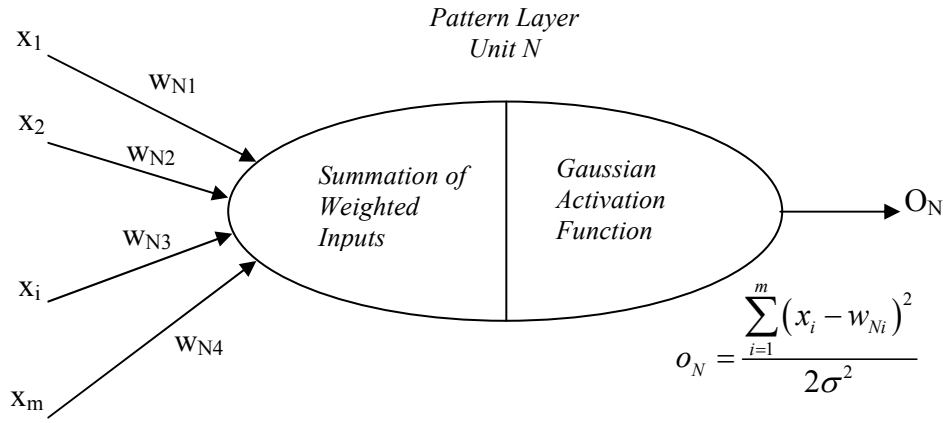


Figure 7 Pattern unit of the radial basis function network.
Source: Tsoukalas and Uhrig (1997).

Unlike back propagation algorithm which requires an iterative learning procedure, the GRNN is a one pass learning algorithm. It can be used for estimation of discrete and/or continuous variables and converges to the underlying regression surface. Consequently, it can quickly learn and fast converge to an optimal regression surface as the number of samples becomes large (Ben-Nakhi and Mahmoud, 2004; Cigizoglu and Alp, 2006). In addition, when training data set becomes large size, the estimation error approaches zero. The GRNN is also reported to respond better than back propagation in many types of problems (Noghondari and Rashidi, 2004). As one pass learning algorithm, the centers of the radial basis function of the pattern units and the connecting weights of the pattern units and the processing units in summation layer are assigned simultaneously. The unsupervised learning algorithm (a special clustering algorithm) is used for training of the pattern unit by specifying the radius of the clusters. The first input vector in the training set is determined as the center of the radial basis function of the first pattern unit. The next input vector is then compared with the center of the first pattern unit. If such different value is less than the prespecified radius, this input vector (the next input vector) is assigned to the same pattern unit (cluster). Otherwise, it becomes the center of the radial basis function of the next pattern unit. This process is repeated until all the other input vectors are

compared one-by-one with all the pattern units already set. As this process is running, the connecting weights between the input units and the corresponding pattern unit are adjusted. Furthermore, the values of the weights, which connect between the neurons in the pattern layer and the summation layer, are set using the supervised learning algorithm (Tsoukalas and Uhrig, 1997). The GRNN used in this study is genetic adaptive, i.e. it uses a genetic algorithm to find the input smoothing factor adjustment. This is used to adapt the overall smoothing factor to provide a new value for each input.

2.3 Neural network as intelligent controller

Neural network can be adapted for several engineering control tasks. By definition, control is action taken to achieve a desired result or goal (Tsoukalas and Uhrig, 1997). For instance, the use of a thermostat for controlling the temperature in a room, the thermostat turns the furnace on when the temperature is lower than a desired temperature; otherwise, the thermostat turns the furnace off. The intelligent controller refers to the application of artificial intelligence (AI) techniques such as neural network or fuzzy logic to produce control actions (Lobbrecht and Solomatine, 1999l; Lobbrecht and Solomatine, 2002). The model-based controller optimization is commonly used in current neural control system. It involves a model of the controlled system or process and consists of three main components; namely:- a reference model, a trainable intelligent controller, and the process or system under control. In supervised control, the neural network is used to mimic the behavior of reference model (see Figure 8). The coupling simulation and optimization model as will be described in the following chapter is used as reference model to conduct the off-line adaptive learning procedure of the intelligent controller. The neural receives the same input and (desired) output as the reference model, and training proceeds in the same way as described in the previous topics. When training is completed over the appropriate range of variables, the trained neural network can replace the reference model to control considered system (see Figure 9). It should be noted that adaptivity is a property of training and does not characterize the use of the trained system. In fact, the trained neural network will not automatically adapt well to the

changing properties of the controlled system. If the controlled system is changed, they must be re-trained on the basis of the new data (Lobbrecht and Solomatine, 2002). Tsoukalas and Uhrig (1997) pointed that the performance of the neural network control system can be no better than the control by the reference model.

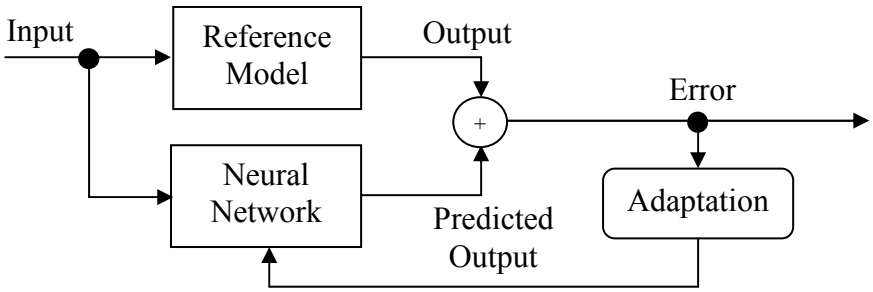


Figure 8 Training process of supervised control.

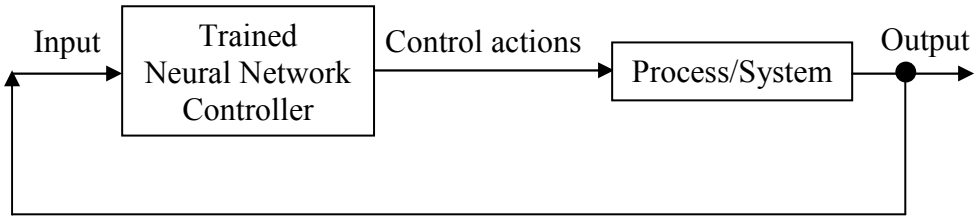


Figure 9 The use of trained neural network for control.

2.4 Combining artificial neural networks with genetic algorithms

The advantage of using the combination of artificial neural networks and genetic algorithms can be concluded in three different ways (Fayad, 2001). Firstly, genetic algorithms can be utilized as a learning algorithm for a neural network instead of using the conventional back propagation method (Rumelhart *et al*., 1968). Secondly, artificial neural network can be employed to establish the relation between input and output data of mathematical simulation model. And then this trained network then can be used to replace original mathematical simulation one for the purpose of decreasing significant amount of computational time of using combined

simulation-optimization models in optimization process (Rogers and Dowla, 1994; Rao et al., 2004; Muleta and Nicklow, 2005). Finally, genetic algorithm can be used to find the suitable neural network architecture and the optimal connectivity among input, hidden, and output layers (Tsoukalas and Uhrig, 1997). The genetic algorithm uses a fitness measure to determine which of the individuals in the population survive and reproduce. The measure of fitness for the neural network is the mean squared error of the outputs for the entire data set (Ben-Nakhi and Mahmoud, 2004). The genetic adaptive algorithm seeks to minimize this mean squared error. It is clear that this fashion is effective way and do not require considerable effort to obtain such architecture like using trial & error method.

2.5 NeuroGenetic Optimizer

The neural network development system (NeuroGenetic Optimizer or NGO, Version 2.6) was deployed to develop neural network controller for operating coastal gate corresponding multiple desired criteria simultaneously. NGO is software package developed by BioComp System, Inc. for use on Microsoft Windows based computers. The NGO is an automated neural network design and development system by using Genetic Algorithms to search the optimal neural network structures and select which input variables are the keys to your success. The NGO also provides seven different network types, including: Back Propagation, Continuous Adaptive Time Neural Network, Time Delay Neural Network, Probabilistic Neural Network, Generalized Regression Neural Network, Self Organizing Map, and Temporal Self Organizing Map for solving the problems. And it also configures to support four different application types such as Function Approximation, Diagnosis, Classification and Time Series Prediction.

Function Approximation is another word for “modeling”. In this type of application, the NGO is attempting to develop neural networks that model the relationships between the input variables and the output variables. Diagnosis applications involve developing systems that will predict whether a condition exists or not. The neural network is trying to learn to distinguish between two or more states.

Output values are usually binary, with the first column of output data set at “1” for the condition existing and “0” when not. Classification applications involve developing systems that will predict which category an item falls into. The neural network is trying to learn to distinguish between two or more categories you provide based on the features of the input data. Clustering applications involve developing neural networks that automatically place records with similar characteristics into like groups. The distinction between Classification and Clustering is that in Classification, you tell the NGO what groups each record falls into and the NGO builds networks that approximate the relationship between input variables and the group. In Clustering, the NGO builds SOM networks that automatically place records into desired groups without any specification of which group on your part. Time Series applications involve developing systems that will predict one or more variables over a period of time.

The NGO is being used in a wide variety of applications, including financial markets predictions, forecasting demand in banking and manufacturing, medical diagnosis, market classification, modeling manufacturing processes and resulting product quality, classification of biological organisms, job cost estimating, fraud detection and many others.

2.6 Application of ANNs in water resources engineering problems

In the field of water resources engineering, the Artificial Neural Networks (ANNs) have been successfully applied to various problems such as hydrological forecasting (EL-Din and Smith, 2002; Huang and Foo, 2002; Agarwal and Singh, 2004; Kumar *et al*., 2004), function approximation problems (Lorrai and Sechi, 1995; Sivapragasam and Muttil, 2005). In this section, it is interesting to review some papers concerning two types of the ANN’s application in water resources management such as deriving general operating rules and control problems.

2.6.1 Application of ANNs for deriving general operating rules

Since ANNs is more effective fashion than other techniques, it has become popular method for deriving general rule at present. The reasons why ANNs should be selected to serve such purpose are the following: a) ANN does not need programming, but it can be directly learned from the data; b) ANN is massively parallel, so it has high speed performance in decision making; c) ANN has the ability to generalize, i.e. to extend their decision making to novel data not seen by the network during the training; and d) ANN can be successfully applied in a complex decision problem such as in classification or pattern recognition (Cancelliere *et al.*, 2002).

Raman and Chandramouli (1996) deployed neural network for deriving a general operating policy of Aliyar reservoir in Tamil Nadu, India. The input and output information for training neural network were generated by using dynamic programming for 20 years of fortnightly historic data. The objective function for this case study was to minimize the squared deficit of release from the irrigation demand. In training process of a feed-forward neural network with one hidden layer, the value of initial storage, inflow, and irrigation demand were selected as input data, and the value of optimal release was chosen as output data. The performance of neural network procedure based on dynamic programming (DRN model) was compared with linear regression procedure based on dynamic programming, a stochastic dynamic programming, and a standard operating policy. It was found that DPN model provided better performance than the other models.

Cancelliere *et al.* (2002) applied neural network to derive monthly operating rules for an irrigation reservoir of the Pozzillo reservoir on Salso River. To obtain reservoir operation rules, two step process were proposed. In the first step, a dynamic programming technique, which determines the optimal releases subject to various constraints by minimizing the sum of squared deficits, was applied on a long period, including severe drought events. The Dynamic Programming (DP) and Constrained Dynamic Programming (CDP) were used herein. The CDP was utilized

to increase more realistic solution by introducing a penalty term into DP. In the second step, a neural network model based on the results from dynamic programming was developed to determine optimal release as a function of available information at the beginning of month, namely:- storage volume, and release at previous month. In this study, there were six different trained networks (one for each irrigation month) for determining the operating rules. After training of the networks, the obtained operating rules have been validated by using coupling reservoir simulation model and trained neural networks to simulate the behavior of the reservoir over shorter period, not include in the period used for training the networks, and through simulation of the soil water balance to evaluate a crop yield index. Results show that the use of neural networks should improve the reservoir performance during drought conditions.

Chandramouli and Raman (2002) developed a dynamic programming-based neural network model for optimal multireservoir operation. The supervised learning approach with the back-propagation algorithm was selected in this case study. The historic data of three reservoir system in Parambikulan Aliyar project were used to generate various operating information through dynamic programming and then these data were used as input and output data for neural network model. The value of initial storage, inflow, and irrigation demand of each reservoir were selected as input data, and the value of optimal release of each reservoir were specified as output data. The performance of the developed model was compared with (1) the regression-based approach used for deriving the multiple reservoir operating rules from optimization results; and (2) the single-reservoir dynamic programming-neural network model approach. The study results indicated that neural network based on dynamic programming results had the better performance in operating multiple reservoirs than the other models.

Chandramouli and Deka (2005) developed neural network based decision support model (DSM) for optimal operation of two reservoirs e.g. Aliyar and Thirumurthi reservoirs in the south of India. In this work, they applied a combination of a rule based expert system and ANN models for developing DSM. ANN was trained using the results from deterministic single reservoir dynamic programming

(DP). In addition to ANN models, the multiple linear regression equations were used to determine the relationship of single reservoir dynamic programming results. The function of rule based expert system is to fire the appropriate neural network based on specified criteria such as time period, storage level; the selected ANN model makes an estimation of the optimal release reservoir. Three different DSM models e.g. DSM1, DSM2, and DSM3 were developed. The DSM1 was developed by segregating the optimization results by considering different time periods; an ANN was trained for each time period. Further, the rule bases of DSM2 and DSM3 are made to fire the ANN by considering a particular time period and the storage available at the beginning of time period. As DSM1 and DSM2 used the information of storage, inflow, and demand at current time period as input data of ANN, DSM3 used the information of inflow and demand at previous time period as input data of ANN. The results of six different models (DP based on neural network, DP based on regression, DSM1 based on neural network, DSM1 based on regression, DSM2 based on neural network, and DSM2 based on regression) were compared in terms of reservoir performance in this study.

2.6.2 Application of ANNs for control

Lobbrecht and Solomatine (1999) applied Artificial Neural Network (ANNs) and Fuzzy Adaptive Systems (FAS) for the problem of water level management. Since Aquarius model requires high CPU time, especially for complex water system, for solving coupling simulation and optimization model in order to derive the optimal control actions, this model may not be appropriate real-time control tasks. Thus, the ANN and FAS were used to replicate control actions prepared by Aquarius model in both local and centralized dynamic control modes for controlling the polder water levels. The control variable was pumping rate of drainage station. The influencing parameters were water level, pump status, and moving average precipitation. For model simulation, the 30 years of real hydrological data were available. In addition to the extreme hydrological events, the random generated data were used to train ANN and FAS model. The obtained results illustrated that ANNs and FAS were able to replicate the behavior of the Aquarius control component with

accuracy in the range 90-97%. This also showed that it was possible to replace the slow computational components by the fast-running trained intelligent controllers.

Lobbrecht and Solomatine (2002) investigated the possibility of using machine-learning methods of Artificial Neural Networks (ANNs) and Fuzzy Adaptive Systems (FAS) to replicate behavior of an on-line deterministic model in preventing flooding of Delfland in the western part of the Netherlands. The local control, which involves a single regulating structure in a water system and is executed on the basis of data gathered in the vicinity of that structure, of Duifpolder and Woudse Droogmakerij. The centralized dynamic control, which typically requires data from various locations in the water system, of Woudse Droogmakerij areas was also considered in this study. AQUARIUS model was calibrated using large data set covering many years of precipitation and then employed to generated data sets for training the intelligent controllers. In addition to the existing actual time series data set, a data set representing excessive precipitation was constructed for the purpose of increasing a large variation in training data and the number of excessive events. For local control mode, water level and pump operation in previous time step t-1, and difference between water level in current and previous time step were used as input data and current pump operation was used as output data. In addition to input data used in local control, moving average of predicted precipitation and previous pump operation were used as input data for centralized control. They recommended that intelligent controller can be only quasi-optimum. It also requires retraining of intelligent controllers when the properties of a water system changed. The intelligent controllers appeared to be robust and capable of solving real-time control (RTC) problems on the basis of information measures only locally.

Lobbrecht *et al.* (2005) applied neural networks and fuzzy system for controlling water level of the overwaard polder, a drainage basin located in South-Holland, The Netherlands. The objective of this study is to investigate the possibility of using artificial intelligence techniques such as artificial neural networks and fuzzy systems for water system control. The existing AQUARUIS model, which is combined simulation and mathematical optimization model, was used as a reference

model for training intelligent controller. For training network networks and fuzzy system controllers, the data generated with the AQUARIUS model in dynamic control mode were used. The water levels in the upper and lower basins in the previous time step t-1, and 8 h moving average values of precipitation (t-1,…t-32) were used as input data and the number of pumps to be switched on at time was used output data. The study results showed that such two trainable intelligent controllers can save computational time when compared to using AQUARIUS. Moreover, it found that neural networks controllers can approximate the water levels in the upper and lower basin better than fuzzy system controllers.

Darsono and Labadie (2006) deployed a recurrent Jordan neural network architecture for real-time regulation of in-line storage in combined sewer systems. Due to computation time and complexity the primary limitation in employing the accurate models necessary for efficient real-time control, application of dynamic or recurrent artificial neural networks (ANNs) may provide the analysis speed, generalization ability and high fault tolerance needed for effective implementation. In this work, they utilized a highly accurate, but computationally time consuming, optimal control model, coupling simulation-optimization model, to provide the training data set for a recurrent ANN under a wide range of sewer inflow conditions. For training of the dynamic neural control module, which is a supervised learning process, rain gauge measurements for various historical storm events were used as input data set and the optimal gate controls, which were calculated offline by the optimal control module, were used as output data set. The data set, not included in the ANN training, was used to compare the performance of ANN controller and of optimal control module. The neural-optimal control algorithm is demonstrated in a simulated real-time control experiment for the King County combined sewer system, Seattle, Washington, USA. The results indicated that dynamic neural control module is effectively capable of an adaptive learning from optimal control model while satisfying the time constraints of real-time implementation.

Based on available literature, there is no prior work that applies ANNs controller or other machine learning techniques for real-time control of coastal

gate, which simultaneously concerns various environmental, ecological and hydraulic conditions.

3. River Operation Model

This section prepares a review of detailed description of River Operation Model (ROM). The ROM, developed by Royal Irrigation Department (RID) in 2004, is mathematical simulation modeling used in this work. This model includes five main sub-modules, namely:- Hydrodynamic Model (HD), Water Quality Model (WQ), Water Demand Model (WD), Rainfall-Runoff Model, and Forecasting Model. The more detailed information for each module can be discussed as follows.

3.1 Hydrodynamic Model

The HD is mathematical model which uses for analysis of the behaviors of water level, flow velocity, and discharge in the river network and flood plain for both steady and unsteady conditions. This model also comprises five sub-models, namely:- Node-Branch Model (NB), Flood Plain Model (FP), Structural Model (STR), Gate Operation Model (GOP), and Rating Curve Model (RC). The HD can thus analyze the flow through hydraulic control structure such as gate, weir, bridge, and culvert, and also can model the operation of gates to investigate the effects of various gate control strategies to water level, flow velocity, and discharge in the river network. The HD implicitly solves an integrated form of the Saint-Venant equations of continuity and momentum equations for one-dimensional unsteady open-channel flow. Such equations can be mathematically expressed as:

Continuity equation:

$$\frac{\partial Q}{\partial x}+\frac{\partial A}{\partial t}-q=0 \tag{19}$$

Momentum equation:

$$\frac{\partial Q}{\partial t}+\frac{\partial}{\partial x}\left(\frac{\beta Q^2}{A}\right)+gA\frac{\partial H}{\partial x}-g\frac{n^2 Q|Q|}{AR^{4/3}}=0 \qquad (20)$$

where Q is discharge (m^3/s); A is cross section of river (m^2); q is lateral discharge ($m^3/m/s$); H is water level (m); g is gravitational acceleration (m/s^2); n = Manning's roughness coefficient; β is correction factor; R is hydraulic radius (m); x is distance (m); and t is time (s).

Before applying this model for river system analysis purposes, Manning's coefficient of river reach and flood plain, and discharge's coefficient of hydraulic structures must be calibrated. That is, the observed water levels (or discharge) are compared to the simulated water level (or discharge). The input data for each sub-model of HD model are presented in Table 1. In addition to such input data, the HD requires upper boundary condition (upstream discharge), lower boundary condition (downstream water level), and initial condition, which is the water level and discharge data at the initial time of program execution. The output data for this model are the information of discharge, velocity, water level with respect to space and time.

Table 1 The input and output data requirement for hydrodynamic model.

Sub-model	Input data
NB model	cross section, length of river reach, roughness N
FB model	area elevation, flood cell link
STR model	structure dimension
GOP mode	gate opening and criteria for operation
RC model	discharge-elevation relationship

Source: RID (2004)

3.2 The Water Quality Model

The WQ is another sub-module of ROM. It uses for the study of mass transportation in watercourse. The water quality parameters, which can be analyzed by WQ model, are salinity, dissolved oxygen, biological oxygen demand, and pH. Typically, the dissolved oxygen and biological oxygen demand are analyzed simultaneously because the organic decomposition process requires oxygen from reaeration process. This model thus comprises three sub-models, namely:- BOD/DO Model, pH Model, and Salinity Model. The WQ can analyze each sub-model or all sub-models at one time. Before executing this model, the HD has to be run first and then the water level and discharge data obtained from HD are used for analysis of mass transportation. The WQ utilizes finite difference technique for solving the advection-diffusion equation as expressed in equation 21.

$$\frac{\partial(cA)}{\partial t}+\frac{\partial(ucA)}{\partial x}+\frac{\partial}{\partial x}\left(DA\frac{\partial(c)}{\partial x}\right)-S=0 \qquad (21)$$

where c is mass concentration (kg/m^3); A is cross section (m^2); u is flow velocity (m/s); D is diffusion coefficient (m^2/s); x is distance (m); t is time (s); and S is source/sing term (kg/m/s). The first term in equation 21 represents the rate of mass; the second term shows mass advection process. The third term represents mass diffusion process. And the last term is sink or source term of precipitation or decomposition. The diffusion coefficient (D_X) of each parameter, decay rate (K1), reaeration rate (K2), settling removal rate (K3) must be calibrated before applying this model for the analysis of water quality in river; i.e., the observed concentration of considered water quality parameters are compared to that from model simulation.

The input data required to run the WQ contain the details of types of nodes, types of boundary, initial condition of desired water quality parameters at nodes in river network, diffusion coefficient, the parameters of BOD and DO, namely:- decaying rate (K_1), settling rate (K_3), sediment oxygen demand (SOD), reaeration rate (K_2), temperature, and dissolved oxygen saturation (DOS), and

pollution loads. The outputs for this model are the information of concentration of considered water quality parameters with respect to space and time.

3.3 Water Demand Model

This model uses for the determination of water demand, especially irrigation water demand, of study area. In addition to irrigation water demand, other demands can be specified in this model. The water demand quantities of total activities are then used as input data for HD. The irrigation demand model (Kirdphitak, 1995) calculates the weekly irrigation water demand and return flow, and these values then summed up to become monthly irrigation water demand and return flow. Indeed, the irrigation areas in a considered watershed are divided into several irrigation blocks. The irrigation efficiency requires to be calibrated when applying this model. To perform model simulation, the information of irrigation area, cropping activity, effective rainfall, potential evapotranspiration, crop coefficient, other demands, and rainfall are required as input data for model. The output data is the quantity of water demand and return flow for each irrigation block.

3.4 Tank Model

The tank model, which is usually classified as a lumped conceptual model, is adopted herein as rainfall–runoff relationship model to determine upstream boundary data used in HD. This model was first introduced by Sugawara (1961) for analyzing the occurrence of runoff in several Japanese Rivers.

Figure 10 (Setiawan, *et al*., 2006) shows the Standard Tank Model and hypothetical water regimes in a watershed. This model is constructed of four vertical reservoirs, of which from top to bottom parts represents the Surface Reservoir (*A*), Intermediate Reservoir (*B*), Sub-base Reservoir (*C*), and Base Reservoir (*D*). In this concept, water can fill the underneath reservoir, and can go reversibly if evapotranspiration is so predominant. The horizontal outlet reflects the outflow, consisting of Surface Flow (*Ya2*), Subsurface Flow (*Ya1*), Intermediate Flow (*Yb1*),

Sub-base Flow (*Yc1*), and Base Flow (*Yd1*). Each outflow only occurs when the water level at each reservoir (*Ha, Hb, Hc* and *Hd*) is higher than its outlet (*Ha1, Ha2, Hb1* and *Hc1*). The outflow at each outlet is also influenced by the characteristics of the outlet, i.e., *A0, A1, A2, B0, B1, C0, C1,* and *D1*. The twelve parameters, namely:- *Ha, Hb, Hc* and *Hd*, *A0, A1, A2, B0, B1, C0, C1,* and *D1*, of the Tank Model need to be determined before using the model. In calibration process, the observed discharge data obtained from water measurement station are compared to the discharge data obtained from model calculation.

The input requirements for the Tank model contain physical characteristics of a watershed, hourly rainfall data, evapotranspiration data, and initial condition of soil water content. The outputs are the information of hourly runoff.

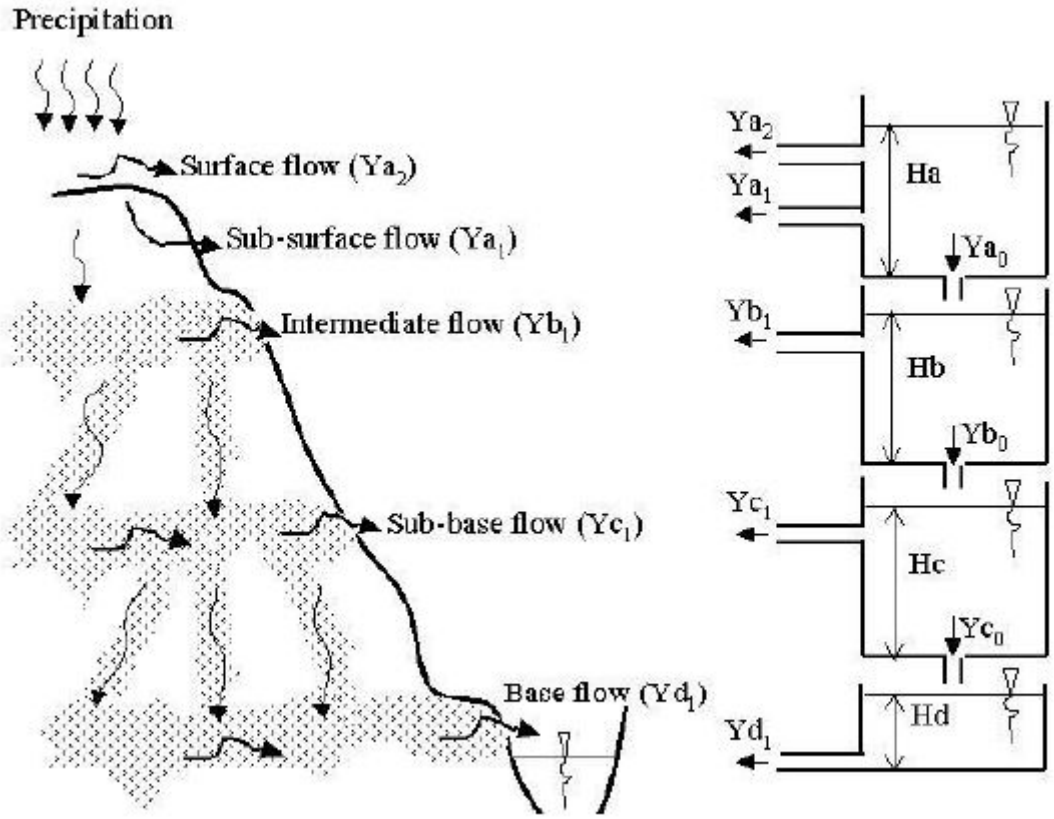


Figure 10 Schematic standard Tank Model.
Source: Setiawan, *et al*. (2006)

3.5 Forecasting Model

In the case of the requirements of water management in advance, Auto Regressive & Updating Procedure model (AR model) and harmonic analysis model as two Forecasting models are applied to synthesize upper boundary and lower boundary data, respectively. The detailed information of each model can be described as follows.

3.5.1 Harmonic Analysis Model (HA)

The fluctuation of sea water level results from the influence of the relative motion of the earth, moon and sun as well as the effect of oceanography. In general, the sea water level information is analyzed by means of harmonic analysis, which is the analysis of a certain periodic wave. Tidal characteristics can be explored using a harmonic analysis model as follows (RID, 2004):

$$\eta(t) = a_0 + \sum_{i=1}^{N} a_i \sin\left(\frac{2\pi t}{T_i} + \delta_i\right) \tag{22}$$

where $\eta(t)$ is the resultant tidal variation at a particular locality and is composed of N constitutents; a_0 is displacement from the reference datum to the mean level; N is the number of harmonic constituent; a_i is the amplitude of the i^{th} constitutents; T_i is period of the i^{th} constitutents; and δ_i is phase of the i^{th} constitutents.

The equation 22 is used in conjunction with the measured sea water level to determine the values of amplitude and phase when the period specified by using the following equations.

$$\lim_{\Delta t \to 0} \Delta t \frac{1}{\Delta t} \int_0^{\Delta t} \eta \sin \frac{2\pi T}{T_i} dt = \frac{a_j}{2} \cos \delta_j = X \tag{23}$$

$$\lim_{\Delta t \to 0} \Delta t \frac{1}{\Delta t} \int_0^{\Delta t} \eta \cos \frac{2\pi T}{T_i} dt = \frac{a_j}{2} \sin \delta_j = Y \tag{24}$$

$$a_i = 2\sqrt{X^2 + Y^2} \tag{25}$$

$$\delta_i = \tan^{-1}\left(\frac{Y}{X}\right) \tag{26}$$

$$a_o = \lim_{\Delta t \to 0} \int_0^{\Delta t} \eta dt \tag{27}$$

The application of harmonic analysis for sea water level forecasting in the duration of 1 to 2 days in advance consists of two stages. In the first stage, the measured sea water level at previous day (t-1) obtained from telemeterimg station is used to determine the value of mean amplitude and phase of specified constituent. For Pak Phanang River Basin Development Project, the two constituent of M2 (T= 12.4206 hrs.) and K1 (T= 23.9346 hrs.) are used (RID, 2004). In the second stage, such two parameters and the values of time (t) are substituted in harmonic equation, and then the forecasted sea water level is calculated at any times.

3.5.2 Auto Regressive Model (AR)

The forecast of discharge in river comprise the four main components, namely:- the measuring and sending real-time data, the hydrological and hydraulic model, the rainfall prediction, and updating procedure or data assimilation procedure. The accuracy of runoff prediction depends upon selected model quality, the precision of rainfall forecasting, and the efficiency of chosen updating procedure. The updating procedure or data assimilation procedure is feedback procedure by using the measured real-time data to improve initial state of system before forecasting. In addition, the updating procedure is deployed to correct the results obtained forecasting model during running processes. The updating procedure (RID, 2004) has several

types depending on selected variables for analysis such as input variables, model states, model parameter, and output variables. For Pak Phanang River Basin Development Project, the updating output variables procedure is selected since it is widespread method for forecasting discharge and is appropriate for conceptual rainfall-runoff model like Tank Model. Furthermore, this method has been used in several marketing software package such as MIKE II-FF, and ISIS.

Besides the accuracy of rainfall prediction, time of concentration of a interesting point is another factor resulting in the result of discharge forecasting. That is, if time of concentration of interesting point is more than forecast lead time, the result of model prediction is promising, but if time of concentration of interesting point is less than forecast lead time, the result of model prediction depend on the accuracy of rainfall prediction. For the discharge forecasting during dry season (no rainfall), the discharge in river depends on base flow, hence the discharge forecasting in this period of time is acceptable due to no need of rainfall prediction. The updating output variable method or error correction method is the correction process of discharge simulated by Tank model. Such method can be mathematically expressed in the following equation.

$$\hat{Q}_{t+1} = Q_{sim,t+1} + \hat{Z}_{t+1} \tag{28}$$

where $\hat{Q}_{t+1}$ is measured discharge at time t+1; and $Q_{sim,t+1}$ is discharge simulated by Tank model at time t+1 by means of rainfall forecasting; $\hat{Z}_{t+1}$ is model error at time t+1. The model forecast error can be determined using the following equation.

$$\hat{Z}_{t+1} = f\left(Z_t, Z_{t-1,\ldots..}, Q_{sim,t+1}, Q_{sim,t\ldots\ldots}\right) \tag{29}$$

where $\{Z_t\}$ is sequence of observed model error. When considering the equation 29, there is no term of rainfall data at previous time (t-1). In fact, this term is already

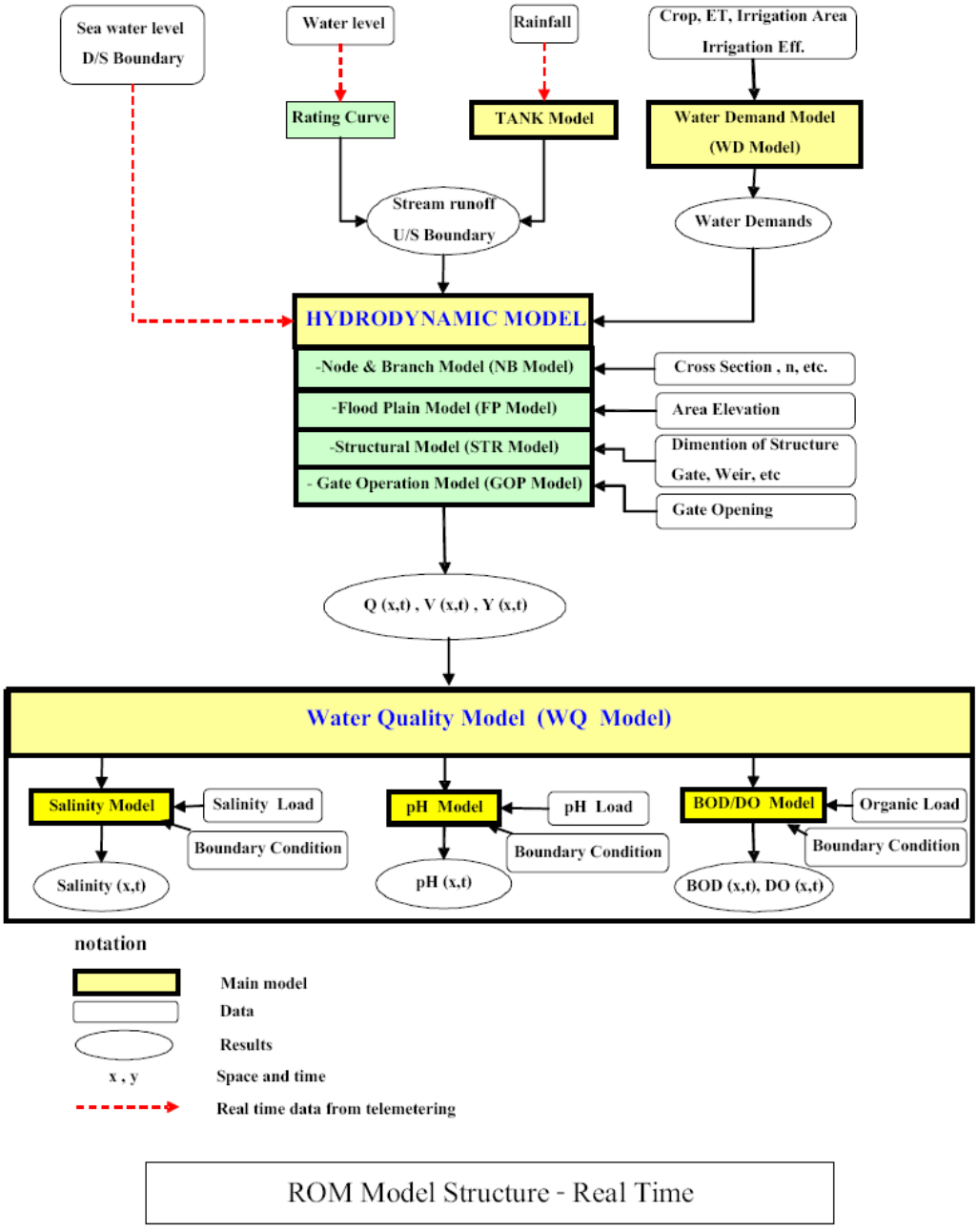
included in term of discharge. The relationship of equation 29 can be determined by the observed discharge data and the discharge data simulated by Tank model at the same time. There are various models which use for determining such relationship such as Autoregressive Model (AR), Autoregressive Moving Average Model (ARMA), Neural Networks, Genetic Programming, and Kalman Filter. For Pak Phanang River Basin Development Project, the Autoregressive Model (AR) was selected, which can show the equation as follow:

$$\hat{Z}_{t+1} = \sum_{i=1}^{P} \phi_i Z_{t+1-i} \tag{30}$$

where ϕ_i is model parameter; and P is order of model. The model parameter can be determined by using existing data, and the suitable order can be obtained from model calibration. The analysis results (RID, 2004) found that AR model order 3 is the most appropriate model for Pak Phanang River Basin Development Project, which can express the equation as follow:

$$\hat{Z}_{t+1} = 1.862Z_t - 0.912Z_{t-1} - 0.037Z_{t-2} \tag{31}$$

The ROM can be used to simulate situation both event mode, which uses for project planning and system analysis purposes, and real time mode, which cooperates with the data obtained from telemetering system. The schematic diagram of the model components in the case of real-time and forecast modes are presented in Figures 11 and 12, respectively.



ROM Model Structure - Real Time

Figure 11 Schematic diagram of ROM model: real time mode.
Source: RID (2004)

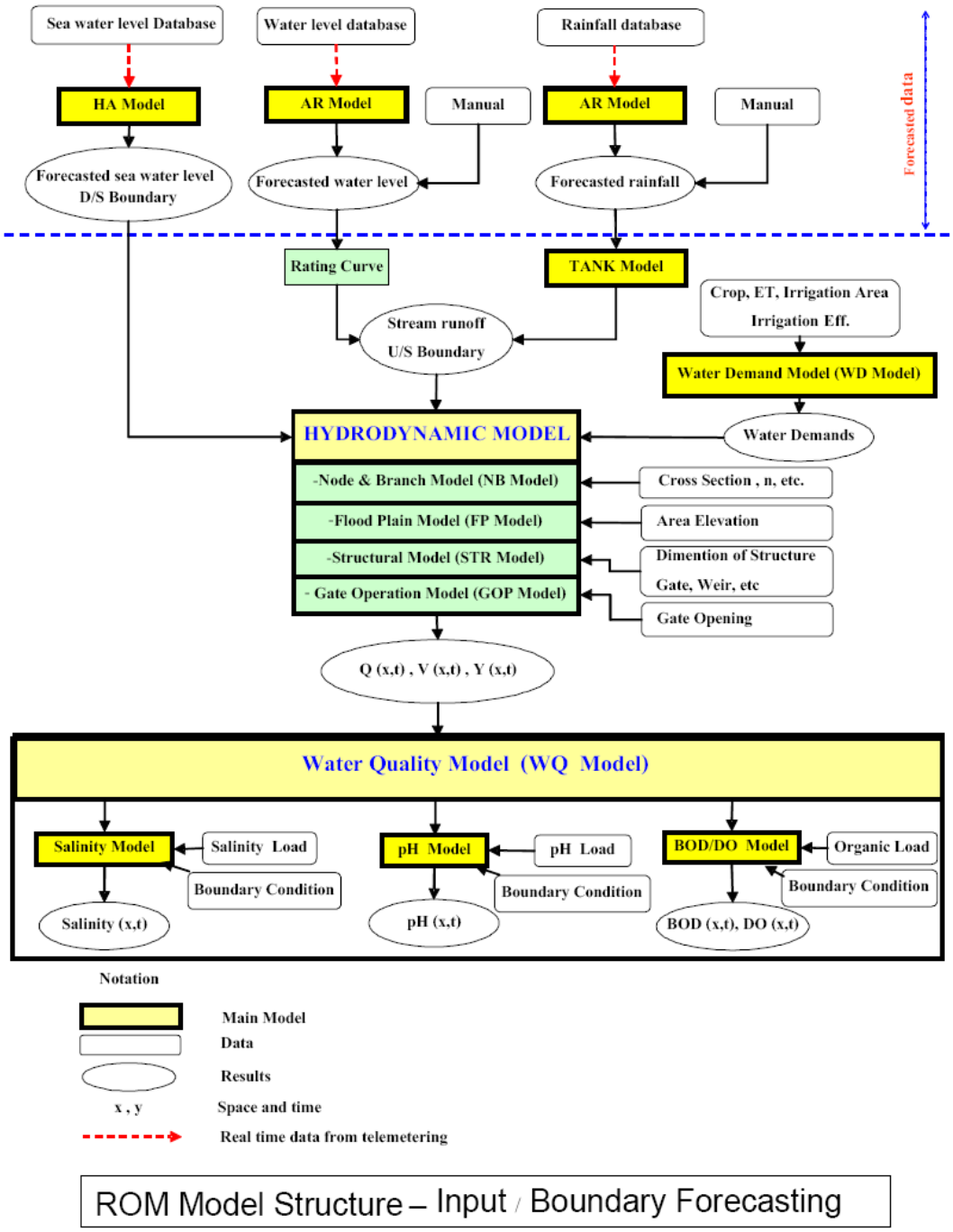


Figure 12 Schematic diagram of ROM model: forecasting mode.

Source: RID (2004)