Department of Civil Engineering,

Imo State University, P.M.B. 2000,

Owerri, Nigeria

4/ 3/ 2019

Dear Prof. Michael Horsfall Jnr,

RE: COST- BENEFIT ANALYSIS OF URBAN WATER SUPPLY AND DISTRIBUTION SCHEME (JASEM-11-1817)

Details of Corrections Made

The following corrections were effected on my manuscript (JASEM-11-1817) as shown below and are hereby forwarded to you for further processing

PAGES	LINES	ERRORS	CORRECTIONS
2	8	Supply Schemes	Mention the water schemes
3	Figure 1	Faint	Was made bold and changed
6		Table 5	Table 4
6	4	Table 6	Table 5
7		Table 6	Table 5
7	1	Tables 7 and 8	Tables 6 and 4
8		Table 7	Table 6
8	1	Tables 8 and 9	Table 7 and 8
8	2	Double words	Remove one
9-11		Tables 8, 9, 10, 11 and 12	Tables 7, 8, 9, 10, 11 and 12
Abstract	1, 2		Lines 1, 2 of abstract were removed
Abstract	3	'The' before Awka city, 'is done'	'the' is deleted 'is done' replaced with 'was carried out'
References			All the references were formatted according to the recommended journal guidelines

Thank you.

Dr. A. O. Ibeje,

Corresponding Author

CHAPTER ONE
INTRODUCTION

1.1 Background of Study

The internet, a global network of computer networks, has become one of the world's largest and most promising information resources. The web, an internet service capable of accessing complex information in the form of graphics, styled-text, and even sound and video, has allowed people to increase their sharing and use of information. The use of Web-Based Systems (WBS) has grown dramatically from obscurity in the 1980s to become common place in businesses, universities and governments; where they are now used for many diverse applications. The web-based environment is emerging as a very important development and delivery platform for real time flood forecasting system and watershed management decision making (Miller et al., 2004).

However, hydrological modeling which includes reservoir inflow prediction tends to remain in the domain of the model developer and to be applied within a consulting framework. The models are inaccessible to decision makers who are not specialist modelers (Taylor et al., 1998) and appear to be designed for experts and professionals or use as in-house tools (Parson, 1999).

Environmental information is usually held in government, academic and commercial water institutions. While data are currently available for public use through the internet by some institutions and government agencies, many of

these institutions provide the data in varying formats and use complicated relational databases (Dia et al., 1997). Public participation and involvement are important components in the implementation of a water resources project and in flood management planning (Wood et al., 1985). Recently, researchers have begun to consider practical and societal impacts of using forecasting systems to support public participation (NCGIA,1996). Having realized these impacts, it has been concluded that there is a growing demand for public access to environmental information, as well as a need for publicly accessible environmental information systems (Haklay, 2001). Forecasting systems, therefore are often designed for decision support, but they lack the capacity for a collaborative decision making (Jankowski et al., 1997). A forecasting system must support collaborative environments (Karimi and Blais, 1997), designed for sharing, executing, and comparing model results, especially when the decision makers and/or stakeholders are from separate locations (Correia et al., 1997).

Lack of software flexibility impedes current forecasting system. Software flexibility describes the process by which modeling software is tailored and modified to satisfy corporate, departmental, and user requirements (Bundock and Raper, 1991). Each organization requires a system that is in many ways unique to its own structure with consideration to its specific problems, requirements, goals, and objectives. At present, most forecasting and simulation software customization processes are time-consuming and expensive; require technical expertise in many computer languages and very often produce poor

results (Raper and Livingstone, 1993). Most systems depend on a specific computer platform (WMS reference, 1999) while MMS operates with the UNIX environment (MMS manual, 1998). Arc/Info, the most popular GIS software, runs on a UNIX platform and the windows-based version (Windows NT), has been released (Binge, 2000). Thus, platform dependence is one of the issues that restrict users from data and software sharing. The importance of integrating simulation models with WBS technology is well recognized (Burrough, 1997; Van Deursen and Kwadijk, 1993; Goodchild, 1992; Raper and Livingstone, 1993). Web-based modeling has been a powerful method for integrating and analyzing data from different sources in comprehensive flood management (Correia et al., 1997).

Hydrological models are used mostly in the research stage of environmental projects, providing information for the development of new hydrological designs or the evaluation of existing designs (Collier, 1994; Todini and Di Bacco, 1995; Haggett, 1998). Models often supplement on-site monitoring because of the expensive and time-consuming nature of monitoring. Hydrological model users become skilled at overcoming the three big challenges of traditional hydrological modelling (Parson, 1999): compiling the proper input files; running the model to create output files and evaluating the output files. There is a need for watershed models to interface with economic, social, political, administrative, and legal models. Thus watershed models will become a component of the larger management modeling strategy. At present,

however, the effective use of forecasting models requires specialized computer hardware and software that are typically not available to communities. Consequently, communities currently lacking the ability to access forecasting systems do not significantly benefit from access to valuable data collected and maintained by government agencies, universities and private organizations around the world.

1.2 Problem Statement

Dadin-Kowa dam was commissioned in 1988 for irrigation, domestic water supply and flood control (Ibeje et al., 2012). Over the years, it has become very difficult to determine the area of cultivable land in each year because there is no prior information of available reservoir inflow. Since daily variation of reservoir inflow is closely linked to daily variation of rainfall amounts, it would be necessary to develop rainfall-inflow model for prediction of reservoir inflow. The prediction of daily reservoir inflow from rainfall-inflow model would assist farmers engaged in irrigation agriculture. This is because they would have fore knowledge of the reservoir inflow that determines the available volume of water that could be used for irrigation. By so doing, sustained and projected agricultural production can be made. Therefore, there is an urgent need for the development of rainfall-inflow model for the prediction of reservoir inflow in order to ensure adequate planning for sustained irrigation agriculture.

Secondly, the Dadin-kowa reservoir, just as most reservoirs in Nigeria is located in a semi-arid region. The precarious rainfall pattern in semi-arid region has

become a huge challenge to hydropower generation. In recent years, Dadin-kowa has lost large amount of water useful for hydropower generation. This is because excess rainfall during the rainy season often filled the reservoir and made it overflow at the end of rainy season. This has been as a result of lack of rainfall-inflow model necessary to predict the reservoir inflow. Hence, the reservoir managers are unable to plan for the excess inflow that could be used for hydropower generation. There is evidently a need for rainfall-inflow model for accurate projection of anticipated excess inflow that can be harnessed for hydropower generation and management.

Thirdly, reservoir inflow prediction is faced with a huge task of timely dissemination of forecasts to users. In moments of flood warning, predictions of reservoir inflow are not often received by the public in real time. This has very often led to loss of lives and property. In periods of droughts, farmers may not be informed of shortage of irrigation water which often resulted in wilting of crops. The most damaging aspect of lack of timely prediction of reservoir inflow is the loss of huge revenue by the industries that rely on hydropower energy. In order to tackle the problems, this research is set out to develop a web-based forecasting software which will enable all water users to assess the inflow prediction in the internet. This would go a long way to ameliorate problems resulting from late information of reservoir inflow predictions.

Fourthly, it is evident from literatures (Bundock and Raper, 1991; Jankowski et al., 1997; Van Deussen and Kwadijk, 1993) that coupling hydrological model

with the web is difficult. Most hydrological softwares often disconnect intermittently from the web (Goodchild, 1992; Raper and Livingstone, 1993). This has often caused abrupt cut in communication when the software wants to download data from a website (Collier, 1994 and Haggett, 1998). Thus, web-based hydrological modelling has become hindered. It is therefore very necessary in this study to develop a web-based reservoir inflow forecasting software that can couple very effectively with the web by using improved technology and efficient web-based software.

1.3 Objectives of Study

The primary objective of this study is the development of user-friendly Web-based software for daily reservoir inflow prediction. The specific objectives of the research are itemized as follows:

i. To acquire available rainfall and inflow data from Dadin-Kowa Reservoir;
ii. To calibrate and verify rainfall-inflow model using Artificial Neural Network (ANN);
iii. To develop an inflow forecasting software using the developed rainfall-inflow model.
iv. To effectively couple the software developed in (iii) with Global Weather Forecasting System for daily download of rainfall forecasts using Java and three-tiered web architecture.

1.4 Justification of Study

Advantages of the Web include openness, user-friendly interfaces, interactivity, flexibility, and fast communication. It is relatively cheap and therefore gives the general public access to a variety of both WBS and data of varying degree of sophistication. This direct access, as a means of allowing wider involvement and participation in environmental decision making is an important prerequisite of watershed management. The provision of accurate, timely and comprehensive real-time and long-term hydrologic predictions is a key component in the implementation of integrated water resources management programs as well as for environmental disaster mitigation efforts for events such as floods and droughts. The key requirement of inflow cum flood forecasting is near-to-real time data acquisition and analysis.

This research will provide a good model for individuals and small organizations that sought to provide services with limited resources. It is the author's wish to contribute to the WBS community by promoting discussion about this particular Web-based forecasting. A reader of this thesis can develop skills in user interface design and programming. Even though the prototype does not have the polish of a professional software product, it does provide many valuable contributions for the general public, concerned citizens, and hydrology students. For example, the prototype can be used as an educational tool. Running the model is simple, and the output is portrayed clearly and concisely.

Summarily, this research is justified because of the following reasons:

i. **Automatic Updating:** This project will produce an automatic hydrological model capable of updating itself through the web.

ii. **Daily Hydropower Forecasts:** Daily predictions of hydropower generation from the reservoir can be reasonably ascertained through daily prediction of the reservoir inflow.

iii. **Decision Support for Flood Warning:** This study will serve as a useful gateway for a reliable real-time flood warning system.

iv. **Rainfall-Inflow Model:** Dadin-Kowa Reservoir would have rainfall-inflow model captured in practical user-friendly software as an innovation from this research.

1.5 Scope of Study

i. This study will be limited to the formulation, calibration and verification of a black box model which will be implemented using the artificial neural network technology. Mechanistic models which include lumped and physically distributed models will not be considered in this study.

ii. It must be emphasized that the development of unit hydrograph for the catchment area under study is excluded from this work. Also channel and flood routing through the reservoir shall not be considered in the research.

iii. The research effort focuses primarily on the use of 1991 to 2001 observed as well as predicted inflow records of the River Gongola in

Nigeria to formulate a rainfall-inflow model which is capable of predicting inflow through the internet.

iv. It should be noted that the predictions made by the model will be limited to short term, i.e., real time. Here, "short term" shall be defined to mean that the model shall make only daily predictions of reservoir inflow up to the maximum of six days ahead.

v. There shall not be any development of a flood warning system for the reservoir. Instead, this research will culminate in the integration of the model into web-based software which will be capable of making the Dadin-Kowa reservoir inflow predictions from any part of the world.

CHAPTER TWO
LITERATURE REVIEW

2.1 Hydrological Models

During the past few decades, a great deal of research has been devoted to the formulation and development of approaches and models to improve the quality of hydrological prediction including mechanistic models (Jaquin and Shamseldin, 2006; Refsguard, 1997; Romanowicz, 2007; Yu-Chi et al.,; 2006) and black box models (Jain and Srinivasulu; 2004; Lin et al., 2006; Liong and Sivapragasan, 2002; Nayak, 2008; Yu et al., 2006). The mechanistic models, which have been studied for more than fifty years and widely used nowadays, laid emphasis on the description of physical laws and tried to make a comprehensive and incisive understanding of every step in rainfall-runoff (RR) process. However, the mechanistic models used to model such processes would require a large amount of high quality data associated with hydrological, meteorological, natural geographical characteristics as well as human activities, while the burden of data constrains the application of mechanistic models. For a large scale hydropower system, selecting one of the mechanistic models will be difficult because of lack of data. In contrast, the black-box models, that at first were designed to identify the connection between inputs and outputs, are widely applied to forecast streamflow because of their requirement of little data and

their simple formulation. The earlier methods include time series techniques and multiple linear regression method (Irvine and Eberhardt, 1992; James, 1991). As an alternative to the aforementioned mathematical models, ANNs, fuzzy logic, genetic programming and support vector machines map input to output without need to identify the physics a priori have been widely applied to hydrological field. The models used in the streamflow forecasting are empirical, conceptual or a combination of both. Empirical models use mathematical equations having no relation to the system's physics. Conceptual models use the hydrological concepts in order to simulate the basin behaviour. Conceptual models usually have two main components: a rainfall-runoff model, which transforms rainfall in runoff through the water balance in the hydrological components such as interception, upper soil zone, groundwater and overland flow; and a routing model, which simulates the flow in rivers and reservoirs. The conceptual rainfall-runoff model like IHACRES has been used in many streamflow studies.

The use of physically-based models allows an understanding of the hydrological process being modelled to be incorporated in the equation that describes it. Provided that the model can be suitably described and calibrated, a reasonably accurate model output can be produced. The forecasting ability of these models can be improved by coupling them with statistical models that use flood stage or discharge observations to account for inaccuracies in the models resulting from errors introduced by boundary conditions, model parameters, and input data.

The time required for model development and calibration, however, is considered a drawback associated with the use of physically-based models. These models have also been criticized for often ignoring the spatially distributed, time-varying, and stochastic properties of the rainfall-runoff process (Zealand et al., 1999), and for difficulties associated with the availability of data for real-time forecasting. Rainfall-runoff models can be lumped or distributed. Lumped models, as opposed to distributed ones continue to constitute a viable solution for the operational needs of estimating flows in watersheds. They are inexpensive, are relatively easy to operate, have low computing requirements, and can provide quick and reasonably accurate estimations at the watershed outlet. Such models are expected to be widely used well into the future. The shortfall of lumped hydrological models is that heterogeneous precipitation over a watershed cannot be considered. Indeed, only the mean areal precipitation is usually considered as an input to lumped models, unless a specific subdivision of the watershed can be made and accommodated by the model. In their review of rainfall-runoff models, Singh and Woolhiser (2002) stressed the effect of spatial variability of precipitation on the production of stream flow in a watershed, and this effect has been a longstanding issue in hydrology as demonstrated in the work of Naden (1992), and Faures et al. (1998). Dawdy and Bergman (1969), and Wilson et al. (1979) indicated that errors in the estimation of rainfall intensity are very likely to limit the accuracy of rainfall-runoff models, and this would be particularly prevalent for lumped models. For small

basins, this type of model is very useful since it has a simple structure and can be easily updated in parameter or state variables. Distributed hydrological models have been used in recent years for a range of different water quantity and quality simulations, but have not yet found widespread use in the field of flood forecasting. The distributed nature of such models provides the potentials for simulations of superior accuracy to purely data-driven models, and allows simulations results to be provided for multiple locations within a watershed. The distributed models can be distributed by sub-basin or by grids. The advantage of distributed models is that they take into account the spatial variation of physical characteristics of the basin and rainfall conditions; it is much more difficult to update the parameters or state variables of these types of models.

Hydrological models are of a major importance for the analysis of climatic change repercussions and water resources balance (Singh et al., 1995). So, they permit the evaluation of water resources and facilitate their management while valuing different choice consequences. Hydrological models of varying degrees of complexity and scale are now available ranging from basin scale models to macro-scale models like ECOMAG (Rajat et al., 2000; Motovilov et al., 1999) approaching that of GCM scale (105 Km^2) and can accept atmospheric model data as their input. Some of the distributed hydrological models which are in common use are MIKE-SHE model (Abbott et al., 1986), TOPMODEL (Beven and Kirby, 1979), WATBAL (Kundsen et al., 1986). A coupled atmospheric runoff and groundwater model was developed and used to study the impact of

changes in regional climate on water resource in Rio Grande Basin (Keeley et al., 2000). Hydrological models provide basis for simulating changes in water table configuration (Yu and Schwartz, 1996; Yu and Schwartz, 1998). Environmental Modelling and Prediction by Gongbing et al. (2003) provided comprehensive coverage on hydrological modeling and forecasting (both short-term and long-term) with succinct illustration of few important models.

2.1.1 Existing Hydrological Models

The following list provides an overview of existing hydrological models. These models exhibit a great diversity ranging from purely empirical approaches created to predict long-term loads to surface waters (MONERIS: Behrendt and Optiz, 2000) to hybrid dynamic-three-dimensional (3D) models working on a GIS-platform, such as MIKE SHE (Boggild et al., 1999) and SWAT (Rosenthal et al., (1995). Hydrological models and their first description are listed in chronological order as follows:

CREAMS	Rudra et al. (1985)
AGNPS	Young et al. (1989)
HBV	Harlin (1991)
HSPF	Chew et al. (1991)
PRMS	Yan and Haan (1991)
ACRU	Kienzle and Schulze (1992)
CASC2D	Julien et al. (1995)
SWAT	Rosenthal et al. (1995)
WASMOD	Schimming et al. (1995)

DHSVM	Nijssen et al. (1997)
SWIM	Krysanova et al. (1998)
MIKE SHE	Boggild et al. (1999)
HMS	Yarnal et al. (2000)
MONERIS	Behrendt and Opitz (2000)
WASIM	Rode and Lindenschmidt (2001)
ARCEGMO	Klocking and Haberlandt (2002)
J2000	Krause (2002)
DRIPS	Ropke et al. (2004)
DWSM	Borah et al. (2004)
MARTHE	Thiery and Amraoui (2001)
TRACE	Herbst et al. (2005)
MIKE BASIN	Ireson et al. (2006)

MONERIS predicts diffuse emissions of nutrients for mid-sized to large catchments and includes a module that describes the emission path of runoff. The temporal resolution of MONERIS is low amounting to one year. The model fails to provide good estimates of nutrient emissions for catchments smaller than $50km^2$. Correlations between modeled and measured loads were significantly good, but total nitrogen loads are generally overestimated (Behrendt and Optiz, 2000).

CREAM (Rudra et al., 1985) is composed of three modules: hydrology, erosion and chemistry and was created to predict diffuse emissions via runoff. CREAMS is a precursor of the leaching model GLEAMS. Hence, runoff is similarly modelled in both models and is based on the Soil Conservation Society (SCS) curve number approach. Results of simulation on a field scale

generally matched the observed order of magnitude (Yoon et al., 1992). The GIS-based hydrological model SWAT (Rosenthal et al., 1995) has a modular structure and consists of hydrological, sedimentological, and chemical subroutines applicable to watershed scales. The hybrid model spatially based on hydrological response units includes both conceptual and physical approaches. A central part of SWAT is the general water balance equation. Surface runoff is determined by the SCS Curve Number approach. Frede et al. (2002) found that physical soil properties affect total runoff moderately, but highly influence surface runoff in SWAT. The model was found to be less efficient in predicting runoff in relation to land cover in a semi-arid watershed; therefore calibration was strongly recommended (Hernandez et al., 2000). Nonetheless, SWAT was found suitable for predicting annual flow volumes, and nutrient loads (Borah and Bera, 2004). Monthly predictions were generally good, except for months with extreme storm events and hydrologic conditions (Borah and Bera, 2004).

Similar to SWAT, MIKE SHE (Boggild et al., 1999) has a modular structure and calculates 3D surface, sub-surface, and stream flow involving distributed grid points. In a case study in an arctic environment, the model was found to overestimate measured runoff, because modeled surface retention of melting water was too low. However, there has been little information on how well MIKE SHE works simulating the transport of pesticides. MIKE BASIN (Ireson et al., 2006) is another product within the MIKE family and functions as an extension of ArcView. The water resource management tool is raster-based and

works on a basin scale. In a case study, the main flaw of MIKE BASIN was that it failed to simulate high water flow, but otherwise satisfactory results were achieved (Ireson et al., 2006). Further watershed-scale models, such as AGNPS (Young et al., 1989), CASC2D (Julien et al., 1995), and PRMS (Yan and Haan, 1991) were found to be eligible to simulate diffuse pollutant loads to surface water (Borah and Bera, 2004). Muleta et al. (2006) tested AGNPS simulating soil erosion and nutrient transport in an Ethiopian catchment and succeeded in identifying hot spots of sediment and nutrient release. The 2-D raster-based model MARTHE (Thiery and Amraoui, 2001) has successfully been tested to predict salinity in groundwater (Weinthal et al., 2005) and may be applied to simulate transport and fate of pesticides, as well. However, until present, there are scarce published results of such simulations using MARTHE as platform. In the model SWIM (Krysanova et al., 1998) presented a three-level scheme of spatial disaggregation from basins to sub-basins. The processes of transpiration and percolation within soils are implemented in SWIM. Retention of water and solutes is described by means of a dimensionless retention coefficient. SWIM was successfully validated for the Elbe catchment (Hattermann et al., 2005), but these authors recommend to accompany macroscale simulations of runoff with empirical investigations in small catchments, in order to identify the dominant hydrological processes. TRACE is a recent development in hydrological modeling documented by Herbst et al., (2005). The Richards equation-based numerical model calculates the three-dimensional saturated/unsaturated water

flow. For the modelling of regional scale pesticide transport, TRACE was combined with the plant module SUCROS and with 3-D LEWASTE, a hybrid Lagrangian/Eulerian approach to solve the convection/dispersion equation (Herbst et al., 2005). A first-step application of TRACE/3-D LEWASTE to a $20km^2$ test area for a ten-year period was used to identify hot spots of isoproturon in groundwater. In general, the model results were consistent and reasonable. Ropke et al., (2004) developed a simple model (DRIPS) on horizontal pesticide transport. In this model, surface runoff is described as a function of rainfall and water infiltration. In contrast to the majority of hydrological models, other parameters such as slope and surface roughness are disregarded in this model. Horizontal attenuation is considered by implementing partitioning between the soluble and solid phases (K_D) and degradation of pesticides. The authors found a good correlation between measured and modelled pesticide concentrations, but in an uncertainty analysis, the confidence interval spanned several orders of magnitude. Therefore, the results of DRIPS have to be evaluated cautiously, although this model seems to be an efficient alternative to more elaborate hydrological approaches.

All hydrological models explicitly describe water runoff, but they were seldom created to model exclusively transport of pesticides. Although hydrological models often use the same SCS curve number approach to relate land use to runoff as do leaching models including a runoff component, the former provide more realistic results because of their larger horizontal resolution. In addition,

hydrological models differentiate between surface and subsurface runoff and thereby their performance is improved again. However, surface and subsurface attenuation processes of pesticides are often insufficiently described and for peak flow, modelled water runoff did not always match measured results. Therefore, in order to calculate more realistic results, hydrological models need to be augmented in temporal resolution.

Most hydrological models can account for changes in land use. For example, Wang et al. (2005) reported a successful test of ANN AGNPS combined with a lake model, when several scenarios of sediment and nutrient loadings were calculated for different land use scenarios. In contrary, Klöcking and Haberlandt (2002) tested the model ArcEGMO for changes in land use and found that problems of impact studies in large river basins resulted mainly from a huge spatial heterogeneity of land use and a rough input database. These authors stated that simple approaches are needed to set-up possible land use changes on the basis of easily available spatial data. The role of crops for the fate of pesticides has been described in leaching models, but hydrological models only consider the effect of vegetation on surface roughness, rather than of pesticides export by harvesting. This deficit is easy to remove. In the contrary, it remains doubtful if a more detailed description of retention and detention by nonlinear sorption and desorption would improve the performance of hydrological models. At least, an elaboration of sorption processes would also increase the

number of input parameter required, which in turn would be barely available in high resolution at large scales.

2.1.2 Evaluation of Hydrological Models

Many methods of evaluation and selection of forecasting models were published by WHO (1986). In particular, several methods of evaluating the credibility of streamflow forecasting models were summarized by Tao and Lennox in Hipel et al. (1994). In principle, a good model should offer the least difference of predicted discharges to the observed discharges. Some of the statistical measures (Hipel et al., 1994) are:

2.1.2.1 Ratio of Standard Deviation of Predicted to Observed Discharges

The ratio of standard deviation of predicted and observed discharges would indicate better model as it approaches to 1.

$$CO = \sqrt{\frac{\sum(y_f - \bar{y}_f)^2}{\sum(y_o - \bar{y}_o)^2}} \qquad (2.1)$$

where;

CO = Coefficient of observation

y_o = observed value inflow

y_f = forecast value inflow

$\bar{y}_o$ = observed value of inflow

$\bar{y}_f$ = predicted value of inflow

2.1.2.2 Root-Mean-Square Error (RMSE): The RMSE would indicate a better model as the value approaches zero.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} SE}{N}} \qquad (2.2)$$

where;

N = number of data points

SE = standard error

2.1.2.3 Square of the Pearson Product Moment Correlation Coefficient: The square of the Pearson product moment correlation coefficient would indicate a better model as it approaches 1.

$$r^2 = \left[\frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i^N (x_i - \bar{x})^2 \sum_i^N (y_i - \bar{y})^2}}\right]^2 \qquad (2.3)$$

where;

r^2 = coefficient of determination

y_i = observed value of inflow

$\bar{y}$ = observed mean value of inflow

x_i = observed value of rainfall

$\bar{x}$ = observed mean value of rainfall

2.1.2.4 Mean of Percent Error (PE): The PE indicates a better model when its value approaches zero.

$$PE = \frac{\sum_N \left[\frac{y_f - y_0}{y_o} X \frac{100}{1}\right]}{N} \qquad (2.4)$$

where;

N = number of data points

y_o = observed value inflow

y_f = forecast value inflow

2.2 Reservoir Inflow Forecasting

Forecasts of river flow can be made in the short term over periods of a few hours or a few days of lead time; and in the long-term up to nine months (Krzyszofowicz, 2001). Usually short-term flow forecasting can be used for flood management but there are many other contexts in which short-term forecast are useful, such as navigation in unregulated rivers where the load transported is dependent on the flow depth; irrigation and water supply and integrated water uses, such as flood control and hydropower.

Long-term flow forecasting has been used to describe the method to forecast flow in seasonal systems (Villanueva et al., 2001 and Druce, 2001). With the use of climate; models (Tucci et al., 2002) or empirical and probabilistic variables and flow (Anderson et al., 2001), such forecasting has been improved. Long-term forecasting can decrease the uncertainty of the economical evaluation of some commodities related to water resources such as; planning energy prices in systems where hydropower has an important share of the production, of agricultural products for non-irrigated areas; and management of water conflicts.

2.2.1 Short-term Reservoir Inflow Forecasting

Short-term forecasting also called real-time forecasting can be done continuously or only after a warning condition. The former is usually done when it is required for operational purposes, such as hydropower and navigation. In hydropower systems, the planning is usually based on the flow statistics and adjusted on a monthly, weekly and daily basis. Flood forecasting is usually done during the flood season after a warning condition is reached in the basin, such as a specific river level, rainfall or climate condition. It can be classified according to the required lead or basin time of response to rainfall. These floods may be flash floods and medium and large basin floods. Flash-floods are mainly a combination of a meteorological event, usually a convective-storm with a particular hydrological situation such as a small basin, steep slope and low infiltration capacity. Forecasting is strongly dependent on the quantitative precipitation forecast (QPF), since the time between rainfall and peak flow is very small for warning and relief measures during the flood (Krzyszofowicz, 1995). Flash floods mostly concern rural basins but, in large cities with greater impervious areas, the basin time of concentration decreases and increases the peak flow. Hydrologic modeling of urban flood potential has witnessed an up surge in interest recently (e.g., Ogben et al., 2000; Lee and Heaney, 2003; Zheng and Smith, 2003) because the hydraulic properties of these areas such as large expanses of impervious areas, smoothed and compacted land surfaces and modification of natural flow paths, conditions suitable for reduced infiltration, storage, and friction losses, creating conditions

favourable to high-peak flow responses. Managing the urban drainage system of conduits and controlling the traffic during wet season days of heavy rain require a warning system, based on a quick evaluation and forecast. The main characteristic of this type of flood forecasting is the requirement of rainfall evaluation for actual and future time steps and of Quantitative Precipitation Forecasts (QPF). Medium-basin flood forecasting can be achieved through a combination of upstream observation of water level and rainfall evaluation in the intermediate basin, together with the upstream level or discharge in a flood-forecast model, there are two main modules rainfall-runoff sub-basin simulation; and river routing.

2.2.2 Long-term Reservoir Inflow Forecasting

Long-term flow forecasting can be done through statistics of flood and seasonal hydrological behavior or correlation among variables such as ocean temperature and rainfall or flow. In a basin where the seasonal conditions are well explained, the flow can be predicted after the rainy period by the recession curve of the hydrograph in the dry months (Villanueva et al., 1987). In systems with long memory where the storage is large and the flow velocity is small, the forecast of a few months of lead time is allowed. However, in a basin with low memory (groundwater storage), the capacity of a long-term flow forecast depends on the climate variable forecast or other variables which has correlation with the climate such as the ENSO index or others.

Long-term flow forecasting can be done through the following:

i. Prediction of local seasonal statistics (the same values are predicted every year if the random term is not used);
ii. Stochastic model taking into account the seasonal and temporal correlation;
iii. Empirical models which relate ocean or climate variables with flow and with some lead time;
iv. Deterministic climate and hydrological models (these models are highly dependent on the capacity of the climate model to forecast rainfall).

Tucci et al. (2002) used two of the above models to forecast the monthly flow in the Uruguay River Basin with 3-6 months lead time. This basin has a low memory since its soil depth and the groundwater storage are small. There is no seasonal behavior and the mean flow does not show any seasonal variation over the years, but the standard deviation among the years is great. The models used were a stochastic model based on flow and rainfall inputs; a deterministic forecast developed with a GCM (Global Climate Model) of CPTEC (Climate Centre in Brazil), which forecast rainfall and a hydrological distribution model (10x10km) for large basins (Collis, 2001; Choan and Tucci, 2001) which used rainfall to forecast the flow. The hydrological model was fitted based on five flow gauges by a multi-objective optimization. It was verified for another 12 flow gauges over a period of 10 years. The CPTEC model presented a bias on the rainfall forecast and was corrected by a statistical distribution for each

model grid and month. This was done for a period of four years (1995-1998). The forecasts of the deterministic models were based on five ensembles (out of 25 of the climate models). The standard error of the predicted monthly flow of the stochastic mode is 1839 m^3s^{-1}. The climate-hydrological model reduced the standard error to 1245 m^3s^{-1}.

2.2.3 Probabilistic Reservoir Inflow Forecasting

In the last years there has been an increasing interest towards the probabilistic forecast of hydrological variables. A probabilistic approach aims at quantifying the prediction reliability through a probability distribution function. (Krzyszofowicz, 2001) or a prediction interval (Chatfield, 1996) for the unknown future value. The valuation of the uncertainty associated with the forecast is seen as a fundamental information, not only to correctly assess the prediction, but also to compare forecasts from different methods and to evaluate actions and decision conditionally on the expected values (e.g., Krzyszofowicz, 2001; Todini, 2004).

Several probabilistic approaches have been proposed, including

1) Methods that use resampling techniques to assess parameter and model uncertainty, such as the metropolis algorithm (Kuczera and Parent, 1998) or the Generalized Likelihood Uncertainty Estimation (GLUE) methodology (Beven and Binley, 1992) (see also Freer et al. (1996) for an application to runoff prediction),

2) Method based on processing the forecasting errors of past date to produce the probability distribution of future values (e.g., Montanari and Brath, 2004), and

3) Methods that evaluate the uncertainty propagates from the rainfall forecasts to the river discharge prediction, as the Bayesian forecasting system (BFS) proposed by Krzyszofowicz (1995).

Combined rainfall prediction and rainfall-runoff simulation procedures for estimation of future flood stages conditions generally attempt only to offer a 'best' estimate of future river watershed discharge conditions without giving any information in regard to the confidence of the forecast being made. Information about the uncertainty in forecasts, however, can be beneficial in a number of ways, especially when this uncertainty is described in the form of a probabilistic forecast. Risk-based decision-making becomes possible when probabilistic rather than deterministic forecasts are provided. Risk-based flood warning is also made possible through probabilistic flood stage forecasting; where the probability of exceedance of the design-flood levels can be provided. This has the benefit of reminding the user that a given flood forecast is not certain and alerts the user to the range of flood stage heights that could potentially be experienced. This would help to remove the confusion during and after flood events that would otherwise likely occur if a flood stage prediction were exceeded, leading to damage or loss of life as a result of misguided faith in what was the best but by no means perfect estimate of

failure conditions. Uncertainty in water shed runoff predictions results as a consequence of an inability to perfectly observe and predict rainfall conditions, and the inadequacy of the mathematical model used to approximate a highly complex physical system. The uncertainty related to the estimations of future rainfall conditions can be referred to as precipitation uncertainty, and the uncertainty related to the model structure, estimated model parameters and observed hydrological data can be collectively referred to as hydrologic uncertainty (Krzyszofowicz, 2007).

Precipitation uncertainty is generally regarded as the most influential cause of uncertainty in a flood forecast (Moore, 2002). Ensemble or Monte Carlo simulation-based forecasts of future hydrological conditions may be used to estimate the uncertainty in a flood stage forecast due to uncertainty in rainfall forecast input. An ensemble forecast produced in this manner, however, cannot alone provide a complete probabilistic forecast as it is only capable of estimating an output distribution model flood stage, incorporating uncertainty in the precipitation input while ignoring the hydrologic uncertainty arising from all other sources of uncertainty (Krzyscofowicz, 2001). Attempt to date to produce probabilistic forecasts of flood stage have considered rainfall as an averaged or point process using a coarse temporal resolution of the order of one hour, and have used lumped physical model or black box model to model the rainfall-runoff process. Examples include the precipitation uncertainty processor developed by Kelly and Krzyszofowicz (2000), which uses a time

series of 6-Rivers watershed average precipitation amounts as inputs for a lumped hydrologic model and the real time flood forecasting system of Lardet and Obled (1994), which uses stochastically generated hourly time series of rainfall as a lumped input to a rainfall runoff model.

2.3 Real-Time Reservoir Inflow Forecasting Methods/Techniques

The techniques available for real time flood forecasting may be classified into four groups:

i) Deterministic modelling

ii) Statistical modelling and

iii) Data-driven/modelling computational:

a) Support vector machines

b) Genetic programs

c) Fuzzy logic

d) Artificial neural network

2.3.1 Statistical Methods for Reservoir Inflow Forecasting

Method based on statistical approach makes use of the statistical techniques to analyze the historical data with an objective to develop a method for the formulation of flood forecasts. The methods thus developed can be presented either in the form of graphical relations or mathematical equations. A large number of data, covering a wide range conditions are analyzed to derive the relationships which inter-alia include gauge to gauge relationship with or without additional parameter and rainfall peak stage relationships. These

methods are more commonly used in Nigeria where Central Water Commission is the central authority for the issues of real time flood forecast in Nigeria.

2.3.2 Support Vector Machines (SVM) for Reservoir Inflow Forecasting

The foundation of Support Vector Machines (SVM) was given by Vapnik, a Russian mathematician in the early 1960s (Vapnik, 1995). Based on the structural risk minimization principle from statistical learning theory. It has gained popularity due to its many attractive features and promising empirical performance. SVM has been proved to be effective in clarification by many researchers in many different fields such as electronic and electrical engineering, civil engineering, mechanical engineering, medical, financial and other (Vapnik, 1998). Recently, it has been extended to regression problem (Keeman, 2001). In the river flow modelling field, Liong and Sivapragasan (2002) compared SVM with Artificial Neural Networks (ANN) and concluded that SVM's inherent properties gave it an edge in overcoming some of the major problems in the application of ANN (Han et al., 2006). Nonlinear modeling of river, flows of the bird creek catchments in the USA with SVM was reported to have its limitations (Han et al., 2002). Dibike et al. (2001) presented some results showing that Radial Basis function (RBF) is the best Kernel function to be used in SVM models. However, Bray (2002) found linear kernel outperformed other popular kernel function (radial basis, polynomial, sigmoid). Bray (2002) illustrated the difficulties in SVM identification for flood forecasting problems. It is clear that, due to its short history there are still many

knowledge gaps in applying SVM in flood forecasting and some conflicting results from different researchers are a good indication that this technique is still in its infancy and more exploratory work is necessary to improve our understanding of this potentially powerful tool from the machine leaning community.

2.3.3 Genetic Programming (GP) for Reservoir Inflow Forecasting

The GP method that is a subset of genetic algorithm generally approaches a solution using evolutionary processes including crossover, mutation, duplication, and deletion (Koza, 2004). It involves regression models over a series of generations based on the Darwinian principles of natural selection (Koza, 2004). It starts with solving a problem by creating a massive amount of simple random function on a population pool. The simple parent function mate and reproduce massive amount of children offspring functions. Each offspring function is measured against the training data. Those offspring functions that closely match the training data may be kept and be allowed to reproduce while some of the poor-fitted offspring may be terminated. The selected offspring function, determined by their fitness, can reproduce another generation of grand children functions. Each grand children function may be tested against the training data for its fitness. The good-fitted grandchildren functions may be kept and used to reproduce the next generation. Some low-fitted grand children functions may be terminated. This population of functions is progressively evolved over a series of generations. The search for the best result in the

evolutionary, process involves applying the principle of survival of the fittest. The GP can reproduce and terminate millions of functions over thousands of generation to find the strongest function that fits the training input data the most. Regression models generated from the GP are free from any particular model structure (Chang and Chen, 2000). The glass box characteristic of GP reveals structures of the regression models, which is the significant advantage of the GP over black box approaches such as neural network. The GP model could be the best solver for searching highly non-linear spaces for global optima via adaptive strategies. Linear Genetic Programming (LGP) is an extension of the GP family. The LGP expresses in a line-by-line mode. Execution of the program is a mimic of calculating multiple calculations in a normal calculator through a series of simple line-by-line processing steps (Heywood and Zincir-Heywood, 2002; Song et al., 2003). In a study by Makkeasorn et al. (2005), in which a comparative study of genetic programming and Neural Network (NN) models was made, preselected input variables offered an improved performance on neural network models over the all-input-variable NN models. But they were generally worse than the GP derived models. The NN model with a 3-day forecasting scheme had $r^2 = 0.50$ based on a preselected variable, while the NN model with a 3-day forecasting scheme based on all input variable had only $r^2 = 0.26$.

2.3.4 Fuzzy Logic for Reservoir Inflow Forecasting

A fuzzy logic model (Zadeh, 1973) is a logical-mathematical procedure based on a "IF – THEN" rule system that allows for the reproduction of the human way of thinking in computational form. In general, fuzzy rule system has four components;

a) ***Fuzzification of the Input:*** Process that transforms the "Crisp" (traditional) input into a fuzzy input;
b) ***Fuzzy Rules:*** IF-THEN logic system that links the input to the output variables;
c) ***Fuzzy Inferences:*** Process that elaborates and combines the rule outputs
d) ***Defuzzificaiton of the Output:*** Process that transforms the fuzzy output to a crisp output.

The most widespread methodologies for developing fuzzy rule system are those proposed by Mambani (1974) and Takagi and Sugeno (1985). The Mambani method (FL-M) follows exactly the above mentioned scheme, whereas the Takagi-Sugeno method (FL-TS) uses a composite procedure for fuzzy inference and output defuzzification. With reference to the Mambani method being the Kth crisp input variable defined as a_k , A_i, j, k its corresponding ith fuzzy output number relevant to the ith rule, the generic Mambani rule $(R_i)_m$ is:

if a_i is $\hat{A}_{i,j1}$ and a_2 is $\hat{A}_{i,j,2}$ and ... and a_k is $\hat{A}_{i,j,k}$ $(R_i)_m$:
then B_i,j (2.5)

The degree of fulfillment v_i of the ith rule can be obtained with the product inference (Larson, 1980), then the weighted sum combination is used to define

the final output membership function μ_B generated by the fuzzy rule system for the crisp input vector (a_1, .., a_k) (Bardossy and Duckstein, 1995). Finally, the crisp output number b is obtained by applying the central defuzzification method to μ_B. Alvisi et al. (2006) showed that the two models based on the fuzzy logic approaches performed better when the physical phenomenon considered are synthetic by both a limited number of variables and IF-THEN logic statements, while the ANN approach increased its performance when more detailed information was used. As regards the reliability as parts, of it was shown that the models based on the fuzzy logic approaches may fail unexpectedly to forecast, in the sense that in the testing phase, some input combination are not recognized by the rule system and thus no forecasting is performed. This problem does not occur in the ANN approach.

2.3.5 Artificial Neural Networks for Reservoir Inflow Forecasting

An ANN is a mathematical model which has a highly connected structure similar the brain cells. They consist of a number of neurons arranged in different layers: an input layer an output layer and one or more hidden layers (Fig 2.1).The input neurons receive and process the input signals and send an output signal to other neurons in the network. Each neuron can be connected to the other neurons and has an activation function and a threshold function, which can be continuous, linear or non-linear functions. The signal passing through a neuron is transformed by weights which modify the functions and thus the output signal reaches the following neuron. Modifying the weights for all

neurons in the network, changes the output. Once the architecture of the network is defined, weights are calculated so as to represent the desired output through a learning process where the ANN is trained to obtain the expected results. Information available is used to define a learning or training data set and a validation data set (Rumelhert et al., 1986). Several different architectures and topologies have been proposed for ANN. They differ in terms of architecture, in the learning process and in the training strategy (Nussbaum et al., 1996). A linear model can be represented adequately by a single layer network, while a non-linear model is generally associated with a multiple layer network. (WC, 1999).The use of ANN techniques in water resources and stream flow prediction is relatively new and has been reported by French et al. (1992); Zurada (1992); Hall and Minns (1993); Zealand et al. (1999); Abrahart et al. (1998); Zhu and Fugita (1994); Hsu et al (1993); Abrahart and See (1998); Minns (1998) and Salazar et al. (1998), among others. Artificial neural Networks have a structure where nonlinear function are present and the parameter identification are based on techniques which search for global maximum in the space of feasible parameter values, and hence can represent the nonlinear effects present in the rainfall-runoff processes. An important advantage of ANN compared to classical stochastic models, is that they do not require variables to be stationary and normally distributed (Burke, 1991). Non-stationary effects present in global phenomena, in morphological changes in rivers can be captured by the inner structure of ANN (Dandy and Mainer, 1996).

Furthermore, ANNs are relatively stable with respect to noise in the data and have a full generalization potential to represent input-output relationships (Zealand et al., 1999).

Neuron Model: The elements that build up the neural network are called neurons. The simplest neuron contains a weight and specific function. The operation of a neuron can be described as a scalar input p that is transmitted through a correction to the neuron, where it is multiplied by the weight w for this special connection. The scalar product wp is formed. In the simplest neuron construction wp is the only argument n for the transfer function f. The transfer function f produces the scalar output a. This procedure is shown in Figure 2.3. A bias b can also be added to the neuron. The bias b is simply the value added to the scalar product wp before the transfer function f.

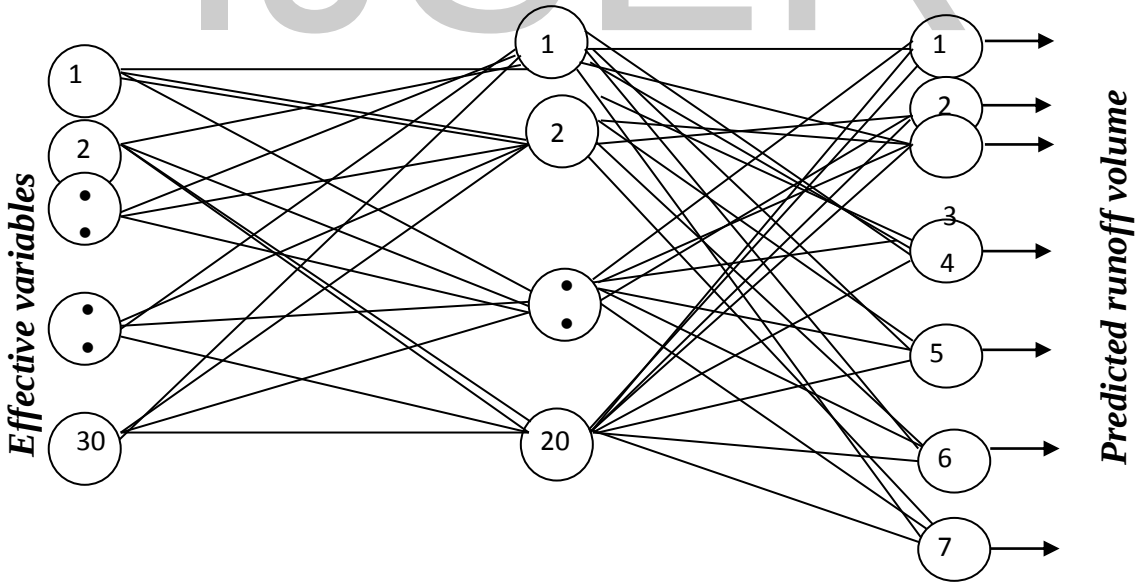


Fig 2.1: Artificial Neural Network [ANN (30, 20, 7)]

Source: Doling and Vares, (2003)

This sum becomes the argument n for the transfer function f. Both the weight w and the bias b are adjustable scalar parameters for the neuron. This is the central

idea of the neural network, that such parameters as w and b can be adjusted so that the network performs in the way that is desired. The most commonly used and basic functions are described as follows. The hard-limit transfer function limits the output from the neuron to either 0 or 1, the hard-limit transfer function's output a is 1. The hard-limit transfer function can be seen in Fig.2.5.

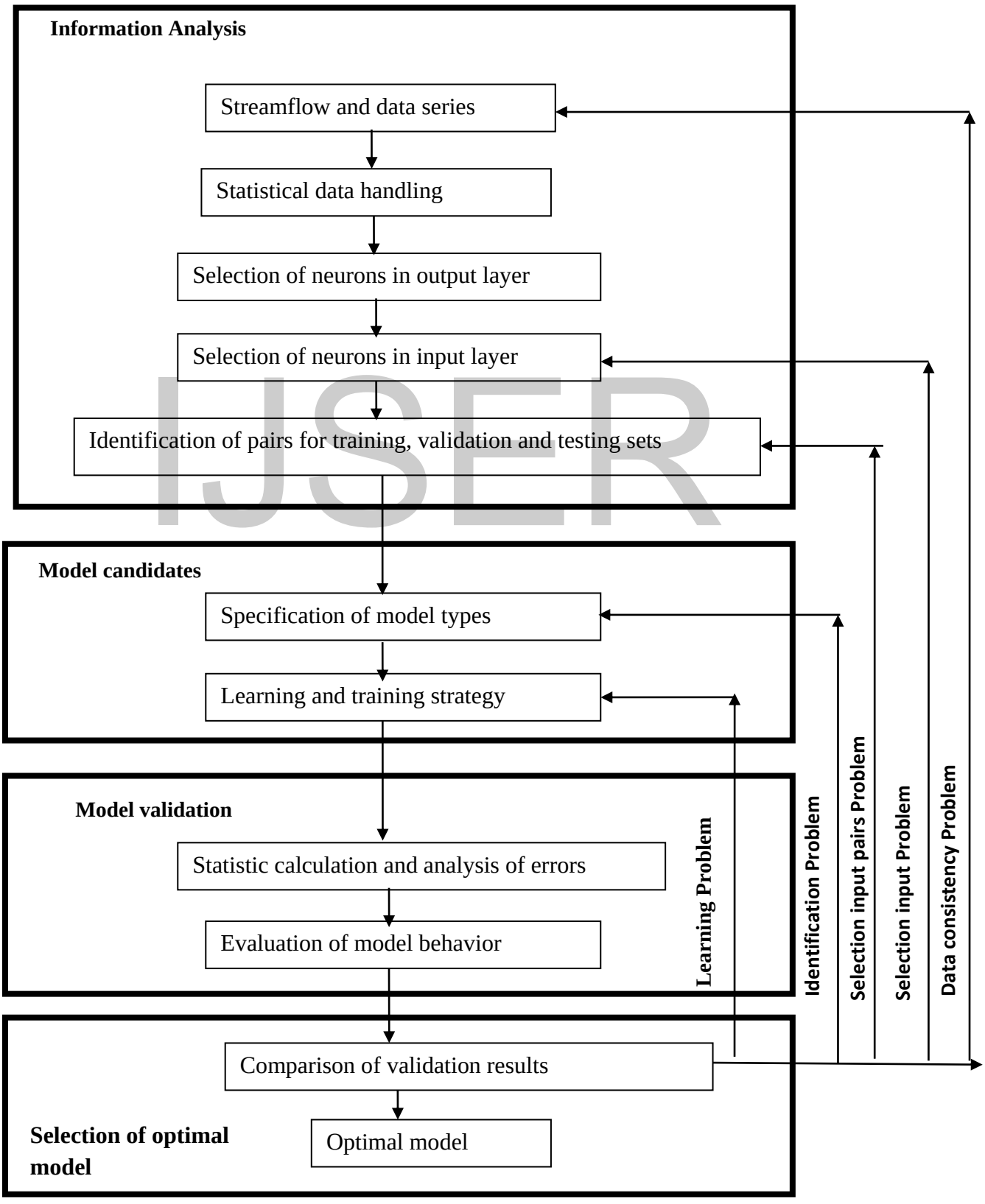


Fig. 2.2: Stages in ANN Model Formulation
Source: Dolling and Vares, (2003)

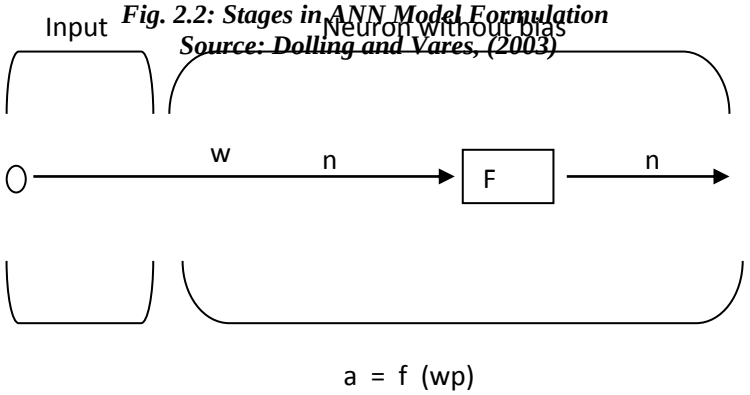


Fig 2.3: A Neuron without Bias
Source: Mathworks (2005)

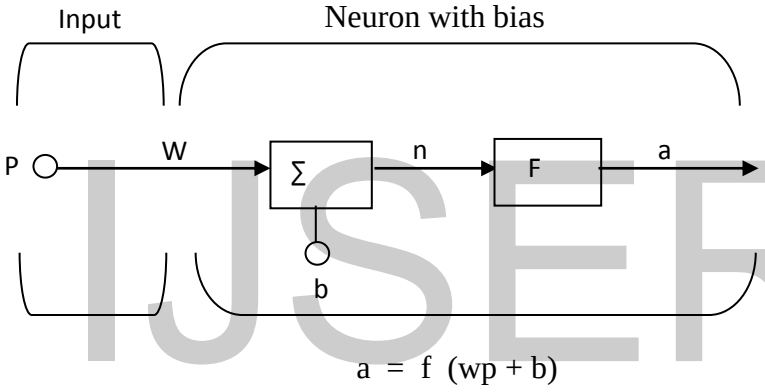


Fig 2.4: A Neuron with Bias
Source: Mathworks (2005)

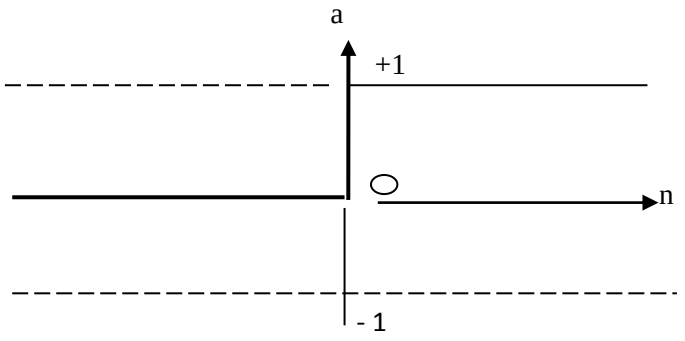
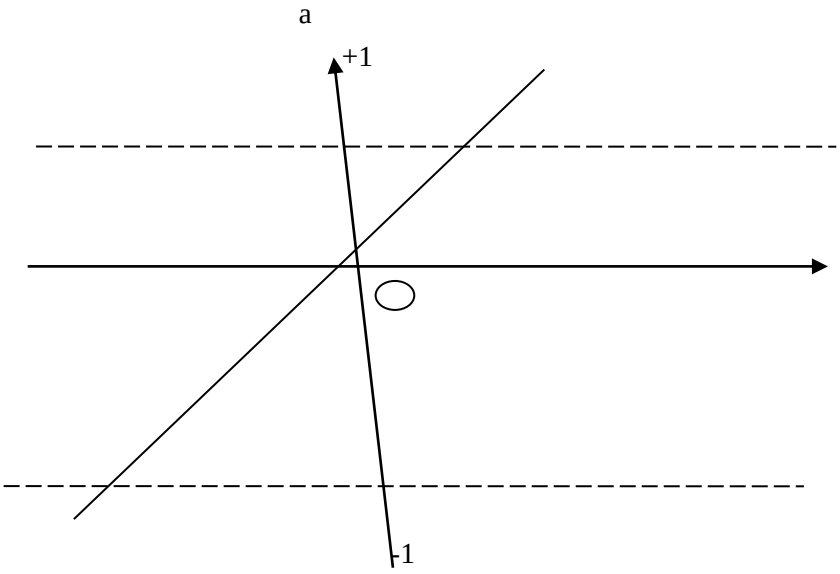


Fig 2.5: The Hard-limit Transfer Function
Source: Mathworks (2005)

The linear transfer function is used when the problem that shall be solved has linear characteristics. The linear transfer function can be seen in Fig 2.6



a = purelin (n)

Fig 2.6: The Linear Transfer Function
Source: Mathworks (2005)

The tan-sigmoid transfer function takes the argument n, which may consist of any values between plus and minus 1 and makes the output a, into the range of -1 to 1. The tan-sigmoid transfer functions can be seen in Fig 2.7.

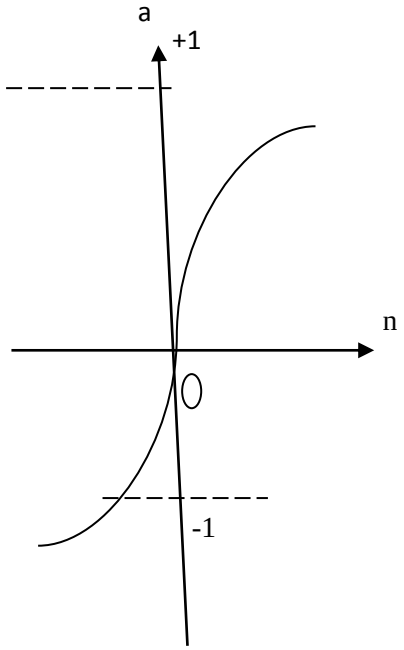


Fig 2.7: The Tan-sigmoid Transfer Function
Source: Mathworks (2005)

One neuron can handle many different inputs at the same time, this will be easy to illustrate when the input is a vector. Presenting the input to a neuron as a vector is the most common way when using Neural Network. The individual input elements: P_1, P_2, P_3, …… P_R are multiplied with the weight of its connection: $W_{1,1.}$, $W_{1,2,}$ $W_{1,3,}$ …… $W_{1,R}$

The sum of these products is called W_p

$$W_p = W_{1,1}.P_1 + W_{1,2}.P_2 + W_{1,3}.P_3 + \ldots + W_{1,R}.P_R \quad (2.6)$$

If the neuron has a bias b, it shall be added to W_p before entering the transfer function. This sum is the argument n for the transfer function.

$$n = W_p + b \quad (2.7)$$

This procedure can be seen in Fig 2.8.

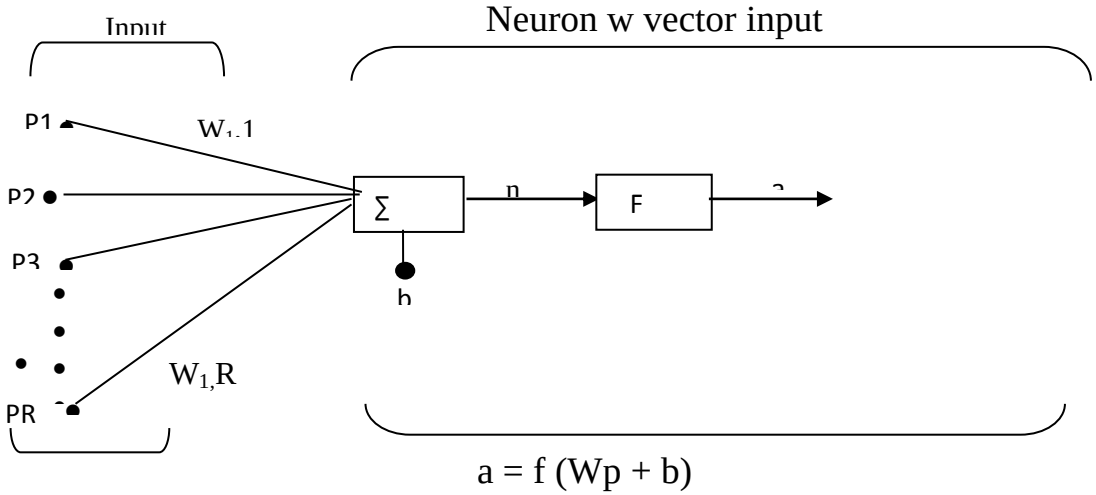
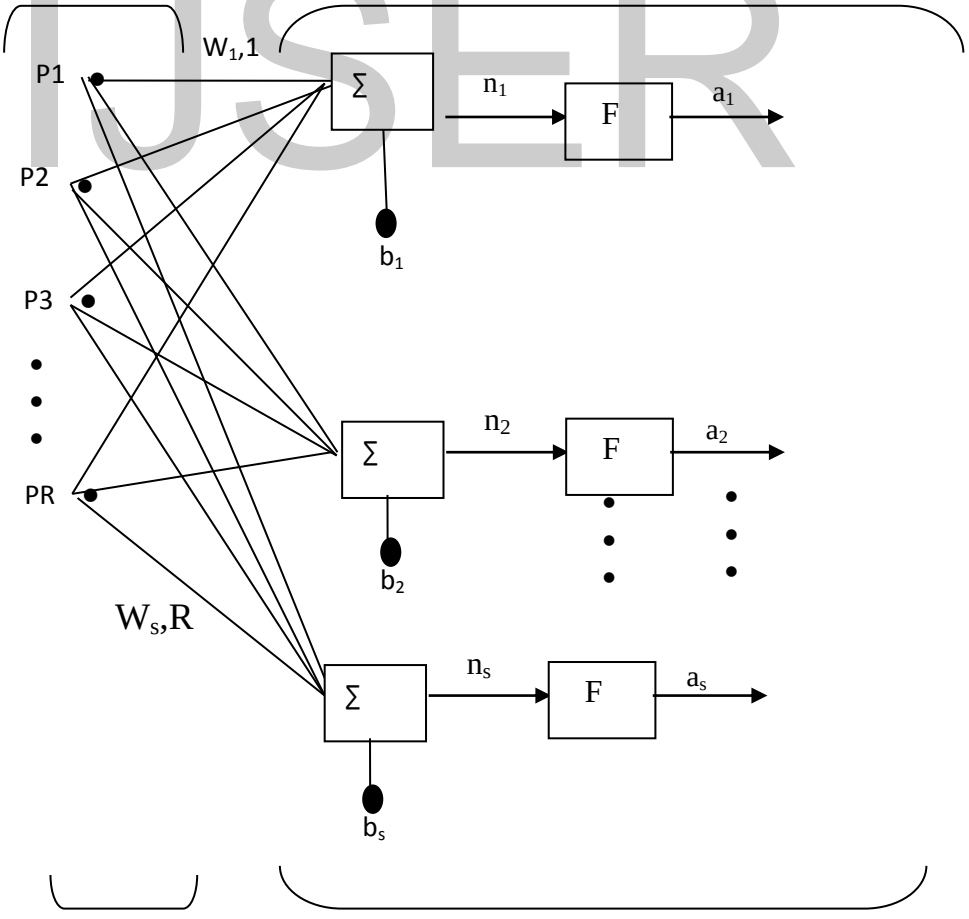


Fig 2.8: Neural Network Computations
Source: Mathworks (2005)

Network Structure: A layer in a network includes the combination of weights, the multiplication and addition operation, the bias and transfer function. However, the input to a network is not included in the layer. Two or more of the neurons can be combined in one layer and work together in parallel. In this network, each element of the input vector is connected to each neuron through the weighted inputs and bias b to form its own argument $n_{i.}$ The different arguments ni form an input vector n that is transformed by the transfer function f. At last, the neuron layer forms an output consisting of a column vector a.



$$a = f(W_p + b)$$

Fig 2.9: Layer Structure
Source: Mathworks (2005)

To create a layer of neurons having different transfer functions f, simply put two of the networks in parallel. Both networks will have the same input p. A network can contain one or more layers. If a network consists of more than one layer, the different layers are located after each other. This means that the output from one layer will be the input p for the following layer; this process is called feed-forward. Because of the feed-forward process, a distinction also has to be made between weight matrices that are connected to the input and weight matrices that are connected between the layers. Weight matrices connected to the input are often called input weights and weight matrices connecting layers are often called layer weights. If the layer consists of more than one neuron each layer has a weight matrix W, a bias vector b and an output vector a. A network structure, consisting of three layers with S neurons in each layer can be seen in Fig 2.9. In a network, it is common for the different layers to have different numbers of neurons. The layers in a multilayer network play different roles. The layer that produces the network output is called output layer, the other layer are called the hidden layers. Multiple-layered networks are powerful and can be used for many purposes in different fields.

Neural Network Training Algorithms:

(a) ***Back-propagation Algorithm:*** Back-propagation is by far, the most commonly used method for training multilayer feed forward networks. This algorithm was popularized by Rumelhert (1986). The break-through was perhaps not so much the application of the chain rule, but the demonstration that layered networks of differentiable nonlinearites could perform useful nontrivial calculations and that they offer (in some implementations) attractive features such as fast response, fault tolerance, the ability to "learn" from examples and some ability to generalize beyond the training data. The simplest implement of standard back-propagation learning updates the network weights and biases in the direction in which the performance function decreases most rapidly – the negative of the gradient. One iteration of the algorithm can be written;

$$x_{k+1} = x_k + \propto_k g_k \qquad (2.8)$$

where x_k = vector of current weights and biases; g_k= current gradient; and α_k = count learning rate.

(b) Conjugate Gradient Algorithm: The basic back-propagation algorithm adjusts the weights in the steepest descent direction (negative of the gradient). This is the direction in which the performance function decreases although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithm, a search is performed, along conjugate directions which produce generally faster convergence than steepest decent direction (Adeni and Hung,

1995). All of the conjugate gradient algorithms start out by searching the steepest decent direction (negative of the gradient) on the first iteration.

$$P_o = -g_o \tag{2.9}$$

where P_o = initial training pattern and g_o= gradient at initial iteration

A line search is then performed to determine the optimal distance to move along the current search direction

$$x_{i+1} = x_k + \propto_k g_k \tag{2.10}$$

where x_k = vector of current weights and biases; g_k = current gradient; α_k = count learning rate and $x_i + 1$ = input vector for i + th training pattern.

Then the next search direction is determined so that it is conjugate to the previous search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction.

$$P_k = -g_k + \beta_k P_{k-1} \tag{2.11}$$

where g_k= current gradient

The various versions of conjugate gradient are distinguished by the manner in which the constant β_k is computed. For the Fletcher-Reeves update, the procedure is;

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \tag{2.12}$$

where g_k^T = transpose of current gradient; g_{k-1} = previous gradient and g_{k-1}^T= transpose of previous gradient.

This is the ratio of the norm squared of the current gradient to the norm squared of the previous gradient.

(c) Cascade Correlation Algorithm: Unlike back-propagation or the conjugate gradient, here the network configuration is not fixed (Thirumaleh and Deo, 1998). Hidden nodes are added one by one starting from zero during the training until the training termination criterion is reached, this algorithm does not involve learning by descending down the error gradient but by maximizing the effect (or correlation) of the new hidden models output on the residual error. It also does not involve the transmission of error backward as in the back-propagation scheme.

(d) Levenberg-Marquardt Algorithm: The Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian Matrix (More, 1977). When the performance function has the form of a sum of squares (as is typical in training feed forward networks), then the Hessian Matrix can be approximated as;

$$H = J^T J \tag{2.13}$$

And the gradient can be computed as;

$$g = J^T e \tag{2.14}$$

where J = Jacobean Matrix, which contains first derivatives of the network errors with respect to the weight and biases; J^T = transpose of Jacobean Matrix and e = vector of network errors. The Jacobean matrix can be computed through

a standard back-propagation technique that is much less complex than computing the Hessian matrix. The Levenberg-Marquardt algorithm uses this approximation to the Hessian, matrix in the following Newton-like update:

$$x_{k+1} = x_k - [J^T J + \mu J]^{-1} J_e^{-1} \quad (2.15)$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian Matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift toward Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way the performance function will always be reduced at each iteration of the algorithm.

Ozgur Kisi (2007) investigated the potential of different ANN algorithms in daily streamflow forecast. The results indicated that the Levenberg-Marquardt algorithm takes a small fraction of the time taken by the other three algorithms for training of the network. The algorithm of back-propagation takes a usually high number of interactions and time for training. The ANN LM gave the best-flow forecasts compared to the other models in short term daily streamflow forecasting. The ANN CG and ANN CC models are also found to produce more satisfactory streamflow forecasts than the ANN_BP.

Descriptive Neural Networks: The structure of a Kohonen neural network is designed so as to identify patterns in data, as with multivariate statistical clustering techniques. This network is therefore a descriptive tool that is used increasingly in hydrology and water resources, in applications such as classification of water shed conditions (Liong et al.,2000); the determination of hydrological homogenous regions (Hall and Minns, 1993); the identification of river pollutant sources (Gotz et al., 1998); and the study of algae bloom (Bowden et al., 2002). The network is made of an input layer of neurons that receives the data and an output layers, often structured in a planar surface. The weight vector of each output neuron is the same scale as the input vector given to the network**.** The elements of all the weight vectors must be calibrated so as to cover the whole data domain. Following the feeding of an input vector I at a given iteration, the weight vector (W_j) of each of the output neuron is updated as follows (Kohonen, 1990):

$$W_j^{(t)} = W_j^{(t-1)} + h_j\left(1 - W_j^{(t-1)}\right) \quad (2.16)$$

This formulation simply gives the weight vector to be closer to the input vector, where h_j is expressed in this application as;

$$h_j = h_o \exp\left(1 - \left({}^{d_{j,a}}/_{5}\right)^2\right) \quad (2.17)$$

In this expression, $d_{j,a}$ is the distance between the most suitable output neuron a and another output neuron j as determined on the output map (layer). When j =

a, the exponential equals 1 and the values of h_j is at its maximum value of (h_o). The value of h_j decreased as the distance between activated neuron a and neuron j increases. Parameter h_o gives the magnitude of the updating, while parameter h_j is the scaling factor on the distance, and indicates the extent of the output map affected by the updating. Both parameters are set at a high value at the start of the calibration process to ensure rapid spreading of the outputs neurons over the data domain, and are reduced gradually so that only small adjustments are performed at the end of the calibration process. The calibration process ensures that all the patterns present in the data are defined in a meaningful coordinate system, and this is why the Kohonen neural network is often called a self-organized map. The Kohonen neural network reduces the dimension of a problem, from an n-dimension input vector to 2-dimension solution, so as to produce a clearer view of the patterns (Kohonen, 1990).

2.4 Current Web-based Modelling Technologies

The Internet began as a US. Department of Defense project in the late 1960s, responding to concerns that the communications infrastructure could be destroyed by a nuclear attack. During the 1980s, it was opened to government agencies and education and research institutions, and in the early 1990s to the public and commercial organizations. Today, with as many as 160 million Internet users worldwide (Morgan, 2000), it is expanding at a rate of 80-100% a year. Although no committee or federal agency manages the Internet, some organizations do direct aspects of its design, including the Internet Society

(ISOC), the Internet Engineering Task Force (IETF), and the Internet Architecture Board (TAB). The World Wide Web, created in 1991 by Tim Berners-Lee at the European Particle Physics Laboratory (CERN) in Geneva, is an application that operates on the Internet. It is based on a client-server model in which a client, using a Web browser, retrieves documents from a Web server.

2.4.1 Client-Server Web-based Modelling

The Web comprises four types of components, the browser, the server, the hypermedia document, and the uniform Resource Locator (URL). Its underlying network protocol is the Hypertext Transfer Protocol (HTTP). A hypermedia document is a file, often called a Web page, written in the Hypertext Markup Language (HTML). A browser displays such a file by interpreting its HTML commands. A URL, often called a hypertext link, is a reference within a hypermedia document to a location in the same or another hypermedia document. HTP, a simple protocol performing one task at a time, is simple but inefficient in that it establishes a TCP27 connection for every request. Moreover, HTTP does not permit a client to issue parallel multiple requests in the same work session. A URL has two basic parts, the address and the protocol name. The address can be a 1P28 address (e.g., 195.147.147.140) or a domain name (e.g., www.kciac.uk), an IF address that has been resolved by a Domain Name Server (DNS). The protocol name specifies one of the Internet protocols such as the File Transfer Protocol (FTP) or HTTP. For a local file, the protocol name is “file.” Lynx, the first Web browser, developed by Lou Montulli in 1992

at the University of Kansas, displayed text-based HTML only. Mosaic, developed in 1993 by the National Center for Super-Computing Applications (NCSA) at the University of Illinois, allowed platform-independent data presentation and was itself multi-platform. The current major browsers, Netscape Communicator and Microsoft Internet Explorer, display a limited number of document formats regulated by the WWW Consortium (W3C). Before the Web, client-server data access was not in general use and often was built using a limited set of component types: clients, networks, servers, and databases. The Web changed client-server data access by providing a universal infrastructure, millions of users, a global scope of access, and thousands of potential developers (Chang and Harkey, 1998).

Client-Server Network Architecture: Client-server systems are characterized by a clean separation of functions between the customer and the provider of service, concurrent access to shared resources, client initiation of dialog, server-location masking, platform independence, communication by message passing; implementation-independent server interface design, scaling of client or server numbers or computing power, and central control of server implementation (Orfali et al., 1996). The Internet's TCP/IP protocol is not a single protocol, but a suite of protocols. Its specifications are publicly available as Requests for Comments (RFCs) which document Internet standards and are maintained by the Internet Architecture Board (lAB, 2003). TCP/IP fits loosely within the Open Systems Interconnection (OSI) Reference Model. Following the OSI

model, a framework for protocol design, helps to identify the functionality of a protocol and encourages standardization. Microsoft's TCP/IP, for example, is compliant with RFC-published TCP/IP standards. However, the Microsoft implementation includes a number of features not found in other versions of TCP/IP.

2.4.2 Architecture of Client-Server Web-based Modelling

Network systems have evolved through monolithic, two-tier client-server, and three-tier client-server architectures. The first-generation monolithic architecture is based on a host-terminal model centralized an organization's applications on one mainframe. The monolithic architecture has been largely supplanted by the two-tier client-server architecture, consisting of fat clients and a data-repository server. A fat client stores all or a portion of the application logic along with the user interface. Since a client and the server are tightly coupled, modification of one requires modification of the other (Finger et al., 1996), leading to problems of scalability and maintenance (Brodie and Stonebraker, 1995). To overcome these problems, the three-tier client-server architecture evolved in which a thin client contains only the user interface, such as a Web browser. Three-tier systems can be easier to maintain and less expensive than their two-tier counterparts (Harnedy, 1996). The major difference between the two is that the application server provides an abstract interface for both the client and the database, creating a more loosely coupled

interface and allowing unrelated clients to utilize the same server (Finger et al., 1996).

(a) Web Servers: A Web server comprises three major software components: the HTTP interface, the HTML page generator, and the server interface. Additional functions include Common Gateway Interface script handling, database access for search engine requests, and administration features. The major Web servers currently available include: Apache, a Unix platform (Ricert, 1996); Microsoft Internet Information Server, a Windows platform; and Netscape Enterprise Server, a Unix and Windows platform. Other servers include FreeBSD (FreeBSD, 2003), Web Commander, Lotus Domino (Domino, 2003) and Servertec Internet Server (SIS. 2003).

(b) Markup Languages: HTML, now about 10 years old, is one of a class of languages that can be defined using the Standard Generalized Markup Language (SGML), developed in the mid-l980s and standardized by the International Standards Organization (ISO) in 1986. Table 2.1 shows the HTML chronology (Harnedy, 1996).

Table 2.1: HTML Chronology

Date	Event
1989	**A subset of SGML is used to distribute documents among various CERN labs,**
1991	**Tim Berners-Lee creates the original HTML document**
1993	**T. Berners-Lee and D. Connolly create the HTML Version 1.0 specification. An HTML + discussion document is created, which provides the basis for HTML version 2.0.**
1994	**An HTML 2.0 standardization effort is underway**
1995	**The HTML 3.0 working draft is abandoned**

1996	**The proposed ISO/IEC International Standard for HTML version 2.0 is completed.** **HTML 3.2 merges the Netscape and Microsoft extensions**

Source: Harnedy, 1996

HTML commands in a document, also called marks or tags, define the structure and the presentation of the data in the document. HTML documents can also incorporate active objects like ActiveX controls and Java applets. A displayed HTML page can contain interfaces to a wide variety of helper and plug-in applications. Using active objects, HTML tags specific text, graphics, interactive buttons, forms, and other objects on-screen which interact with a user, immediately, not waiting for another click that might be required for defining a rectangle line or polygon. While SGML separates content from presentation, HTML mixes the two, reducing its flexibility considerably. The Extensible Markup Language (XML) is a compromise. Like SGML, it is a language for specifying markup languages. However, it is not as complex as SGML but does extend HTML with some of SGML's functionality. XML allows a Web page author to create specialized tags allowing XML pages to provide functionality not possible with HTML

(c) Common Gateway Interface: The Common Gateway Interface (CGI) is a specification standard that allows a Web server to exchange data with other applications. A user at the client begins by opening the Web site and navigating among its pages. For example, the user might request a flood prediction map by completing a form that causes the server to invoke the flood-map script, which in turn invokes the mapping software. Once the mapping program has

completed, the script converts the results to HTML and returns them to the server for transmission to the user's browser. Although a CGI script can be written in almost any language, Perl has become the popular choice (Hunter, 1998). Most criticisms of CGI concern its performance. As shown in Fig 2.10, each client request creates a process at the server, requiring significant processing overhead. Another problem is that a CGI script cannot interact with a Web server (Orfali et al., 1996).

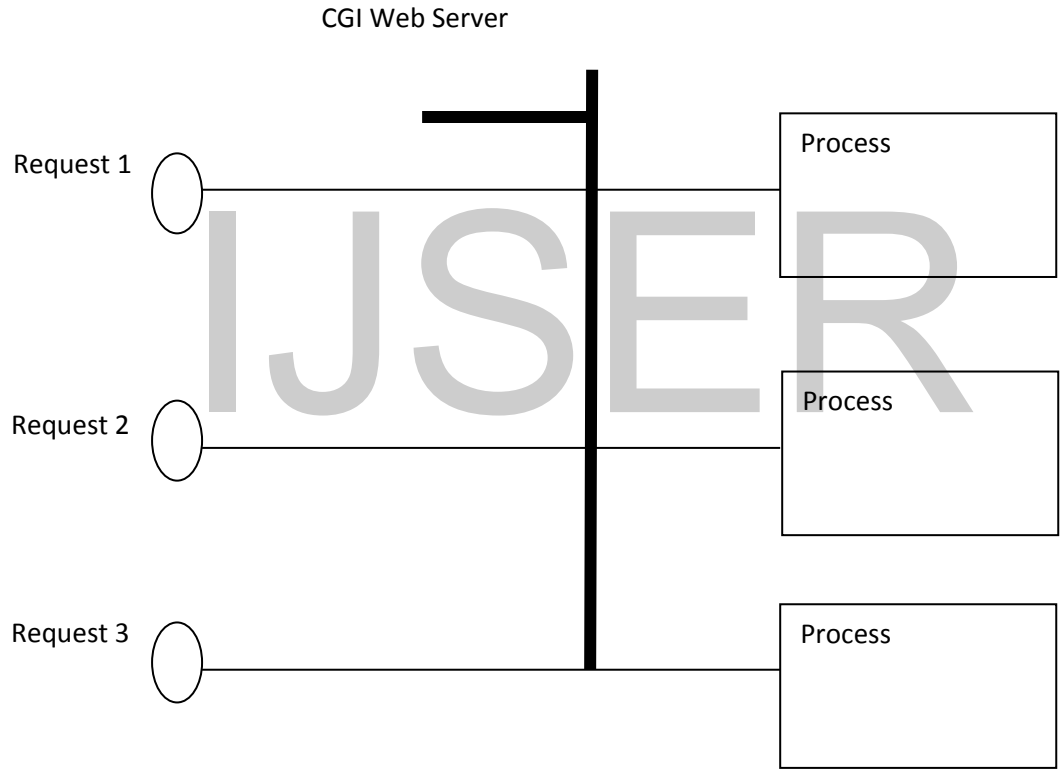


Figure 2.10: GGI life Cycle

Source: Chi (1999)

Perl (practical extraction and report language) is an easy-to-learn and rich language producing compact programs (Srinivasan, 1997). It is useful not only

for CGI programming but also for more general applications (Wall et al., 2000; Schwartz et al., 1997; Harnedy, 1996; Srinivasan, 1997)

(d) Java: Web programming languages, used to create active content in both servers and clients, include Java, C, and C++, although Java is probably the most common. It is an object-oriented (OO) language that is somewhat smaller and simpler than C or C ++; (Hardy, 1999; Bertino and Martino, 1993; Abnous and Khoshafian, 1990). But its main attributes are platform independence (Lemany and Perkins, 1997) and improved security (Eckel, 1998; Traister, 1993; Grand and Knudsen, 1997). Java's primary role is to create applets, mini-applications that execute within a Java-enabled browser but with the ability to communicate with a server. A Java platform has two basic parts (Kramer, 1996), a virtual machine and a Java application programming interface (Java API). The virtual machine provides the interface for system calls, while the Java API provides the interface for user programs written in Java (Gosling et al., 1996). Fig. 2.11 shows the basic components of a Java platform. Java objects that share the same set of attributes and methods: the data and the operations; are often grouped together into an object class. An object instance contains the object's graphical characteristics, its location, and its associated attribute values. A composite object is one that is composed of other objects and whose operations propagate to its constituent objects.

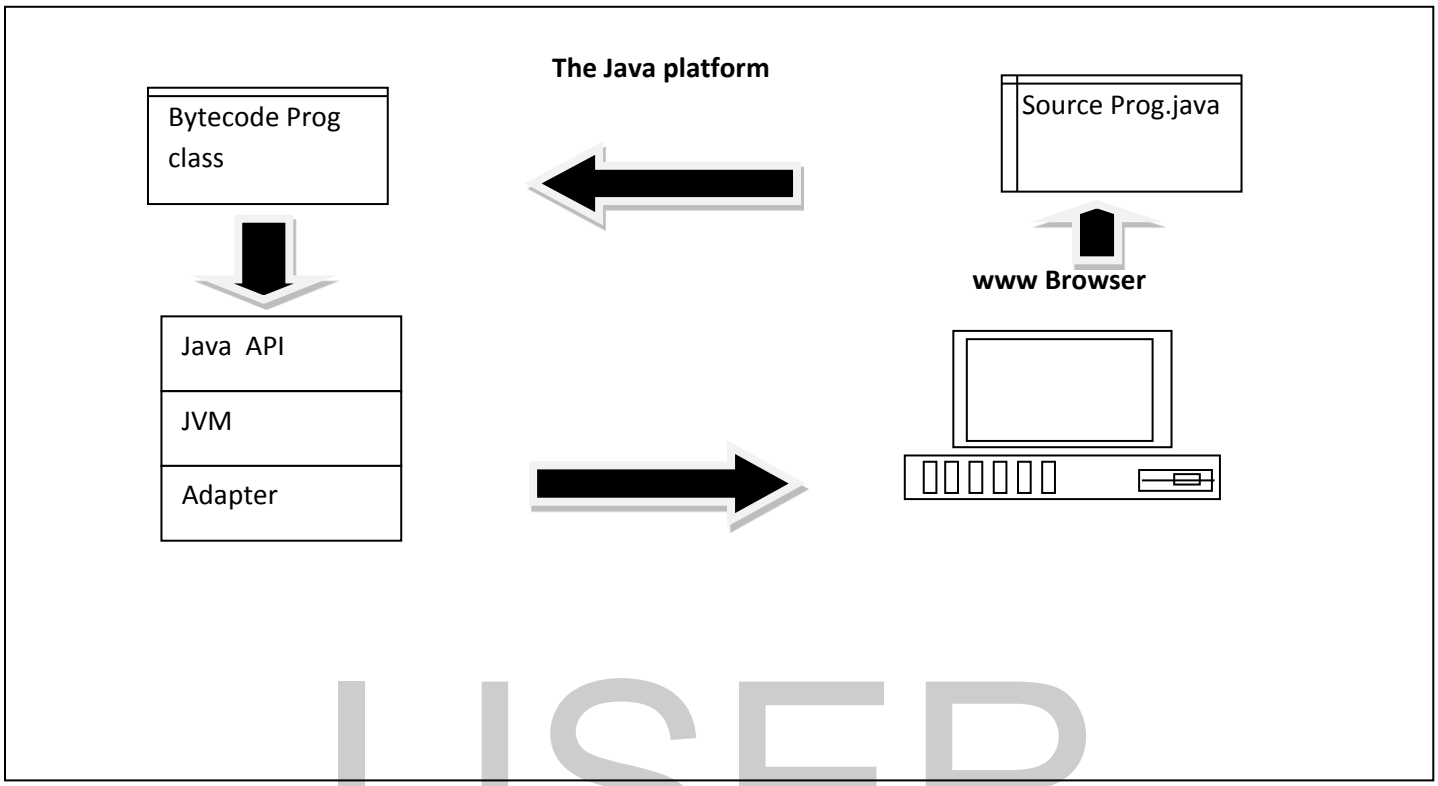


Fig 2.11: Java Platform

Source: Waleed, 2003

Very often, the terms "composite object" and "object class" are used interchangeably (Davis, 1994; Milne et al., 1993). A method is a rule that specifies an object's behaviour. The process of associating an operation with a method is known as binding (Milne et al., 1993). Each object class can have several methods and specific behaviours defined and can also inherit methods from its parent class. When a method operating on an object is invoked by sending a message to the object, the behaviour bound to it is executed (Hardy, 1999). Java allows functionality to be transmitted with data. Thus, a system can

receive data and the ability to manipulate the data at the same time. For example, on the server an application could create an object, ask the object to populate itself from a local database, and then send this object to a client Web browser. The client could then ask the object to create an HTML representation of itself for display, or open up an interface for editing itself. After editing, the object can be sent to another client across the network. It could then be connected to a local database and asked to save itself. Applications such as managing image and graphic databases, scientific databases, geographic information systems, and multimedia databases require relatively complex structures for objects, data types, and application-specific operations (Elmasri and Navathe, 1994). OO languages like Java are intended to address these requirements (Budd, 1991; Booch, 1994; Gamma et al., 1995). For information about object-oriented programming in WBS applications, see (Kjerne and Dueker, 1986; Herring, 1991; Yourdon, 1994; Worboys, 1993; Chance et al., 1999; Davis and Maidment, 2000). For object-oriented applications in hydrology, (see Morten and Vefsnmo, 1997; Alfredsen, 1999; Alfredsen, 2000; Alfredsen and Saether, 2000). Java platform-independence is achieved through the Java Virtual Machine (JVM) which read, and interprets Java programs and invokes any required platform commands (Gosling and McGilton, 1996). Since each platform has its own JVM. A Java class can inter-operate across machines with different architectures (Daconta, 1996; Venners, 1996; Zhou et al., 2001). In Fig. 2.12 unfortunately, the layered structure comes with a loss in speed

(Chappell, 1996) when compared to other compiled languages. Java facilitates the construction of image-processing applications (Baxes, 1994) including restoration, compression, segmentation, and representation (Lyon, 1995). Animation, extremely important when implementing time-series analyses over spatial models, is easily performed in Java.

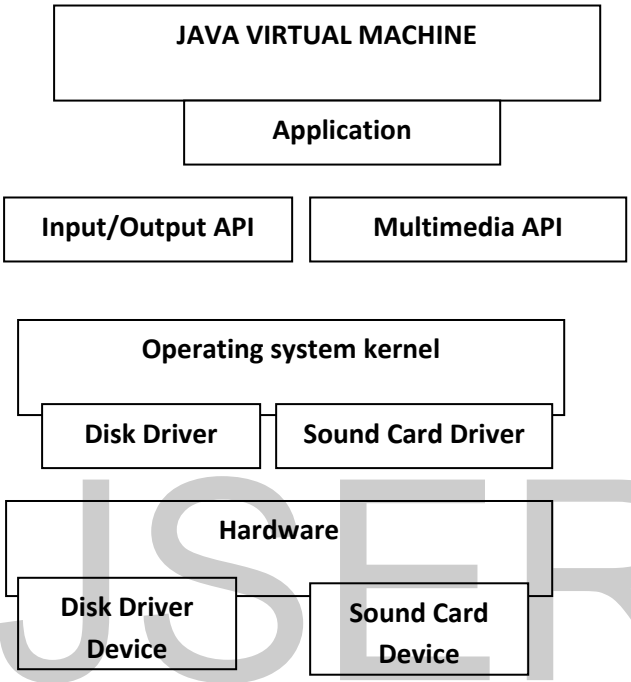


Fig 2.12: Computer System Layering with Java

Source: Waleed, 2003.

For example, Java can display a computer animation of a sequence of maps from the same area in different time periods. In such a case, the movement of water across the study area (runoff routing) could be made quite evident. Java has many built-in facilities for developing rich networking applications. It includes classes for socket programming for different types of protocols. For example, the Java URL class, an abstraction of a URL, allows a Java application to open a remote file as if it were local. It also supports network connections with the socket class, which provides a client-side socket interface similar to the

standard Unix socket, for the creation of distributed clients and servers (Gosling and McGilton, 1996). Finally, Java classes may contain some geo-processing services such as display, zoom in, zoom out, and map production. With a Java-based browser, a user can connect to an HTTP server, download an applet and retrieve the required spatial data.

(e) Applets and Servlets: A Java application program has no security restrictions within its name space. A Java applet, on the other hand, is a small Java program that typically provides active content, such as animation, of a part of an HTML page. Such a page is configured to give the applet only space on the screen to draw and links to the applet's code on its server. Specific client security restrictions on applets include the inability to access files - unless the applet comes from a digitally signed archive and the signature is verified as coming from a trusted source, make a network connection — except to the server from which it came, start any programs on the client, or read certain system properties such as the user home directory. A Java servlet is a special kind of Java application that runs on a Web server within a process with some security, error checking, and garbage collection (Hunter and Crawford, 1998). A servlet is usually implemented as a Java class that can be loaded dynamically to expand the functionality of a server (Hunter, 1998). Servlets provide a Java-based solution for server-side programming problems such as inextensible scripting capabilities, platform-specific APIs, and incompatible interfaces. Like applets, the major benefit of servlets is that the Web browser can provide a

universal client environment. A servlet can be used to generate dynamic HTML content. Dynamic HTML page is usually written by a servlet and returned to the caller on the client side. The servlet offers improvements over CGI, being faster and more secure (Chang and Harkey, 1998). It also reduces the development effort in server-side Web applications (Chi, 1999).

(f) ActiveX: Component reuse is a central goal in software design. An ActiveX control is a reusable software component intended for use in Web sites, desktop applications, and development tools. ActiveX is the commercial name for what originally was called Object Linking and Embedding (OLE) and OLE Control Extensions (OCX). A Windows application can access an ActiveX OCX control over a computer network by adhering to the Distributed Component Object Model (DCOM) specification. ActiveX is strongly tied to Windows at present because of its reliance on the Visual Basic environment (Lee, 2001). Like Java applets, ActiveX controls and executes within a constrained environment, like a Web browser. Unlike Java applets, however, ActiveX controls can be written in various languages and are delivered in machine code, normally for a Windows environment. While Java applets are supported primarily only by Web browsers, ActiveX controls are supported by many types of containers. Also, ActiveX controls, downloaded when a Web page is visited, are saved in the client, whereas Java requires downloading of Java applets each time a page is visited. Finally, since ActiveX controls have greater freedom in the client, they present a security risk. This is addressed by a practice whereby developers of

ActiveX controls sign their work digitally. Security is a major issue in comparing Java and ActiveX (Blundon, 1996; Coffee, 1996; Semineno, 1997; Stron, 1999). Java supports two methods: server trust and component constraints, commonly called sandboxing. However, one disadvantage of sandboxing is to prohibit certain useful tasks. ActiveX controls support only the server-trust method. Consequently, ActiveX security relies on the user making correct decisions in accepting programs. A major concern with ActiveX controls is the possibility of a virus in the machine code. Another problem is that they tend to be large compared to applets. Table 2.2 shows a comparison between Java and ActiveX.

Table 2.2: A Comparison between Java and ActiveX

	Java	ActiveX
Development language	**Java**	**Visual basic, C ++. Java**
Execution environment	**Virtual machine**	**Internet explorer, windows**
User interface	**Virtual machine**	**Windows**
Component API	**Java beans**	**ActiveX**
Computer platform	**Any**	**Wintel, Macintosh**
Database API	**JDBC**	**ODBC**
Security	**Sandbox, signed code**	**Signed code**
Distribution API	**IIOP (Internet Inter-ORB)**	**DCOM (Distributed COM)**

Source: Blundon (1996)

(g) Client-side Java versus Server-side CGI: Java is a client-side capability, while CGI is server-side. CGI provides no protection against invasion of the host computer's resources, while Java isolates the client from the server to protect against invasion of the server's resources. Java has limitations in

interface with other types of software since it is a single language facility, while CGI programs can be written in standard languages to easily interface with existing database and other software (Schwartz et al., 1997). CGI programs can be easily integrated with HTML, while Java tends to be standalone with only limited connectivity to HTML. For example, a Java image cannot act as an HTML anchor, while a CGI image can. While graphics operations in CGI are usually difficult for the user, Java has built-in classes that allow graphic manipulation such as zooming, display, and other operations. CGI is slow and resource intensive, requiring a software process for each invocation. On the other hand, a Java applet is kept in memory and run as a thread, a much less expensive operation. The biggest advantage of Java is the machine independence of applets. A Java program is translated into an intermediate language that can be interpreted by any system supporting applets. CGI programs, on the other hand, must be recompiled for each server. Although CGI programs produce HTML or standard images, they are client independent; however, they must be recompiled if moved to a different server.

CHAPTER THREE

METHODOLOGY

3.1 Development of Rainfall-Inflow Model

3.1.1 The Study Area

(a) Reservoir Operation and Management: The study area is the Dadin-Kowa reservoir. The Dadin Kowa Dam is in Yamaltu local government area of Gombe State in the north east of Nigeria (See Fig.3.1 and Fig.B.2). Dadin-kowa town is located between latitudes 10^{o} to 10^{o} 20^{o} N and longitudes $11^{o}01^{o}$E and $11^{o}19^{o}$E . The dam is located about 35 kilometers to the east of Gombe town, and provides drinking water for the town. The dam was completed by the federal government in 1984, with the goal of providing irrigation and electricity for the planned Gongola sugar plantation project. The water supply project was built at a cost of about ₦8.2billion by CGC Nigeria, a Chinese company. In 2010 it was providing about 30,000 cubic meters of water daily, treated at a plant three kilometers from the dam before being piped to storage reservoirs in Gombe (Ogbu, 2010). In August 2001 the federal government announced that it would spend $32 million to complete the Dadin Kowa Dam power generation facilities (Nwezeh, 2001). In March 2009 ₦7 billion was allocated to complete the hydro-electrical generation component of the dam, and another ₦500 million to complete the canal, which would irrigate 6,600 hectares of farmland (Timawus, 2010).

(b) Physical Characteristics: The climate of Dadin-kowa is characterized by a dry season of eight months, alternating with a four months rainy season. As in other part of Nigerian savanna, the precipitation distribution is mainly triggered by a seasonal shift of the Inter-Tropical Convergence Zone (ITCZ). For the years 1977 to 1995, the mean annual precipitation is 835mm and the mean annual temperature is about 26^{o}C, whereas relative humidity for the area has same pattern of 94% in August and dropping to less than 10% during the harmattan period (Yamaltu L.G.A., 1999). The relief of the town ranges between 650m in the western part to 370m in the eastern parts.

(b) Reservoir Structures: Dadin-Kowa Dam is a multipurpose dam which impounds a large reservoir of water from Gongola River. It has a storage capacity of 1.77 billion cubic meters for irrigation with 950km^2 area of farmland (Ibeje et al., 2012). Its flood spillway has a discharge capacity of 1.1110m^3/s (see Appendix A).

3.1.2 Materials for Development of Rainfall-Inflow Model

(a) Laptop: A portable Toshiba Centrino Dual-Cord Computer CPU of 1.73GHz, 1.2GB Random Access Memory (RAM). The operating System is Microsoft XP Professional.

(b) Matlab Software*:* Version 7.5.0.342 (R2007b) of Matlab R2007b Software; the language of technical computing.

Fig. 3.1: The Location of Dadin-Kowa Reservoir

Source: FMIA (2012)

(c) Quantitative Precipitation Forecasts (QPFs): The Global Forecast System (GFS) is a global Numerical Weather Prediction (NWP) computer model, which produces QPFs up to 16 days ahead at each data assimilation cycle (00, 06, 12 and 18 UTC), but with decreasing spatial and temporal resolution over time. The NWP model is run in two parts: the first part has a higher resolution and goes out to 180h in the future; the second part runs from 180-384h at a lower resolution. The resolution of the model varies in each part of the model: horizontally, it divides the surface of the Earth into 35 or 70km grid squares; vertically, it divides the atmosphere into 64 layers and temporally it produces a forecast for every third hour for the first 180h; after that, they are produced for every twelfth hour. The GFS is the only global NWP model for which all output including QPFs is available for free over the internet and as such is the basis for non-state weather companies, e.g. Wunderground.com, Weatheron-line.co.uk, Weather.com.au and t7online.com. The GFS-QPFs that were used in this study were the modeled precipitation forecasts over the entire Nigeria.

3.1.3 Data Collection

The hydrological data used to conduct this research consisted of daily rainfall amounts of Gongola River Basin and daily inflow for the Dadin-kowa reservoir. Records of daily rainfall amounts and daily inflow were spread over eleven-year period (1991 – 2001) (Ibeje et al., 2012).

The rainfall data was collected from Nigeria Meterological Agency (NMA) data bank for the meterological stations within the Gongola River Basin. The

existing inflow data was obtained from the Dadin-kowa dam archives. The inflow data measurements were taken at the Dadin-kowa inflow gaugimg station located at the dam (Ibeje et al., 2012).

3.1.4 Procedure for Data Analysis

3.1.4.1 Data Subsets for Model Calibration, Verification and Data Training

Preparation of inputs for calibrations and verification of models requires splitting the datasets into two sub-datasets in model development. The first sub-dataset contained about two-third of the original data, which was used for the calibration of the inflow forecasting models (Hall and Minns, 1993). The second sub-dataset, which is unused (unseen) during the model calibrations, was thus prepared solely for the verification of the inflow forecasting models (Hall and Minns, 1993). Within such a two-stage effort of model development, models have to be created based on the calibration of datasets, and then verified by the verification dataset. The ultimate predication accuracy may be confirmed in the verification.

3.1.5 Technique for Rainfall-Inflow Modelling

3.1.5.1 Multi Layer Perception (MLP)-Artificial Neural Networks (ANN)

The MLP-ANNs used in this study is a feed forward neural network which generally consists of three layers viz; input, hidden and output layers. This consisted of a mass of interconnected and parallel processed neurons in every layer, and each neuron can transfer an input of vector variable to an output of

scalar variable through a nonlinear function called “Sigmoid”. The input layer is connected with the predictors outside, synchronously the output layer connects to the predictands, and the hidden layer is the connective bridge of them. The signal from “normalized” predictors according to “feed forward” mode, is transmitted in order from the input layer through the hidden layer to the output layer, and is finally converted to the predictands. Output y at time T is calculated from the function expressed in equation (3.1):

$$y\,(T/\theta) = g\,(\delta\,(T), \theta) \qquad (3.1)$$

where δ (T) is input vector (regressor) and θ represent activation function which is sigmoid function for MLP.

3.1.5.2 Improvement Error Back-propagation Algorithm

The error back-propagation algorithm is widely used to adjust the weight matrix and the biases matrix of the networks through an interactive training procedure for the purpose of revealing the relationship between the predictors and the predictands. From an initial random weight matrix and bias matrix, this algorithm searches the “weight space” by using gradient decent method to minimize the overall error between the predicted outputs and the observed values. A method with self-adaptive learning rate and self-adaptive momentum coefficient is used here to accelerate the training process and to prevent the algorithm from converging at a local minimum. The learning rate α and the momentum coefficient β are automatically adjusted as follows (Sun et al., 2007)

a) when $e_f^{(k)} < 0$

$$\alpha (K+1) = \alpha (k) [1+ ч. e^{-er(k)}] \quad (3.2)$$

$$\beta (k + 1) = \beta (k) [1 + v. e^{-er(k)}] \quad (3.3)$$

b) when $e_r^{(k)} \geq 0$

$$\alpha(k+1) = \alpha (k) [1-ч. e^{-er(k)}] \quad (3.4)$$

$$\beta(k+1) = \beta(k) [1-v.e^{-er(k)}] \quad (3.5)$$

where k is the epoch index, ч, v Σ (0, 1) are two parameters (0.5 and 0.5 in this study, an approach adopted from Sun et al., 2007) and $e_r^{(k)}$ is an artificial coefficient. The overall error is defined as follows:

$$e_r^{(K)} = \frac{\Delta E(K)}{E(K)} = \frac{E(K)-E(K-1)}{E(K)} \quad (3.6)$$

By this dynamic adjustment, the learning rate increases when the overall error reduces and decreases over a certain rate. Additional momentum term is used to import the effect caused by the changes of the weights matrix in the previous epoch and this item can effectively smooth the descending path and prohibit converging at a local minimum. So the changes of weights matrix W_{ji} and bias matrix q_j are adjusted as follows:

$$\Delta W_{ji} (k+1) = \sum_{p=1}^{p} \propto_j d_{pj} + \beta_j \Delta w_{ji}(k) \quad (3.7)$$

$$\Delta q_{i\ (k+1)} = \sum_{p=1}^{p} \propto_j d_{pj} + \beta_j \Delta q_j(k) \quad (3.8)$$

in which p = 1, 2… p represents the p_{th} training pattern.

3.1.5.3 Levenbergy-Marquardt Back-propagation (LMBP) Training

In prediction context, MLP-ANN training consists of providing input-output examples to the network, and minimizing the objective function (i.e., error function) using either a first order or a second order optimization method. This so-called supervised training can be formulated by minimizing the weight function, the sum of the nonlinear least squares between the observed and the predicted outputs, defined by:

$$E = \frac{1}{2}\sum_{p=1}^{n}\sum_{k=1}^{m}\left(y_{pk} - \hat{y}_{pk}\right)^2 \quad (3.9)$$

where *n* is the number of patterns (observations) and *m* the total output units, y represents the observed response ("target output") and $\hat{y}$ the model response ("predicted output"). In the case of one output unit (*m* = 1) reduces to

$$E = \frac{1}{2}\sum_{p=1}^{n}\left(y_p - \hat{y}_p\right)^2 \quad (3.10)$$

which is the usual function minimized in least squares regression. In the BP training, minimization of *E* was attempted using the steepest descent method and computing the gradient of the error function by applying the chain rule on the hidden layers of the MLP-ANN (Rumelhart et al., 1986). Consider a typical multi layer MLP-ANN. whose hidden layer contains *M* neurons. The network is based on the following equations:

$$net_{pj} = \sum_{i=1}^{N} W_{ji}x_{pi} + W_{jo} \quad (3.11)$$

$$g\left(net_{pj}\right) = \frac{1}{1+e^{-net_{pj}}} \quad (3.12)$$

where net_{pj} is the weighted inputs into the jth hidden unit, N the total number of input nodes, W_{ji} the weight from input unit i to the hidden unit j, x_{pi} a value of the ith input for pattern p, W_{jo} the threshold (or bias) for neuron j, and $g(net_{pj})$ the jth neuron's activation function assuming that $g(net_{pj})$ is the sigmoid function. Note that the input units did not perform operation on the information but simply passed it onto the hidden node. The output unit received a net input of

$$net_{pj} = \sum_{j=1}^{M} W_{kj} g(net_{pj}) + W_{ko} \quad (3.13)$$

$$\hat{y}_{pk} = g(net_{pk}) \quad (3.14)$$

where M is the number of hidden units, W_{kj} represents the weight connecting the hidden node j to the output k, W_{ko} is the threshold value for neuron k, and $\hat{y}_{pk}$ the kth's predicted output. Recall that the ultimate goal of the network training is to find the set of weights W_{ji} connecting the input units i to the hidden units j and W_{kj} connecting the hidden units j to output k, that minimize the objective function (Eq. (3.13)). Since Eq. (3.13) is not an explicit function of the weight in the hidden layer, the first partial derivates of E were evaluated with respect to the weights using the chain rule, and the weights were moved in the steepest-descent direction. This can be represented mathematically as

$$\Delta W_{kj} = -\eta \frac{\partial E}{\partial W_{kj}} \quad (3.15)$$

where η is the learning rate which simply scales the step size. The usual approach in BP training consists in choosing η according to the relation $0 < \eta < 1$. From Eq. (3.15), it is straightforward that BP can suffer from the inherent

slowness and the local search nature of first order optimization method. However, BP remains the most widely used supervised training method for MLP-ANN because of the available remedies to its drawbacks. In all, second order nonlinear optimization techniques are usually faster and more reliable than any BP variant (Masters, 1995). Therefore, LMBP for MLP-ANN was used for data training. The LMBP uses the approximate Hessian matrix (second derivatives of *E*) in the weight update procedure as follows:

$$\Delta W_{kj} = -[H + \mu I]^{-1} J^T r \qquad (3.16)$$

where *r* is the residual error vector, μ a variable small scalar which controls the learning process, $J = \nabla E$ is the Jacobian matrix, and $H = J^T$ denotes the approximate Hessian matrix usually written as $\nabla^2 E = 2J^T J$. In practice, LMBP is faster and finds better optima for a variety of problems than do the other usually methods (Hagan and Menhaj, 1994). To improve the network training speed and efficiency, the LMBP was used with the early Stopped Training Approach (STA).

3.1.5.4 Early Stopped Training Approach (STA)

The main issue in training MLP-ANN for prediction is the generalization performance. MLP-ANN, like other flexible nonlinear estimation methods such as kernel regression, smoothing spines, can suffer from either underfitting or overfitting. While a too complex ANN (i.e., too much hidden nodes) may likely fit the noise leading to overfitting, an insufficiently complex network (i.e., insufficient hidden nodes) can fail to detect the regularities in the data set

leading to underfitting. Underfitting produces excessive bias in the model outputs whereas overfitting produces excessive variance. Here, in order to avoid underfitting and overfitting, STA was introduced with the LMBP training. STA allowed the use of complex network without overfitting since the training is stopped as soon as some non-zero criterion was met. Therefore, the efficiency of the method depended highly on the stopping criterion. There are a number of plausible stopping criteria that have been reported to be superior to regularization methods (Finnoff et al., 1993). Here the stopping criterion used involved a tradeoff between training time and generalization error and proceeded as follows.

The available data were split into two parts:

(1) A *training set*, used to determine the network weights;

(2) A *prediction* (or *test*) *set*, used to verify the effectiveness of the stopping criterion and to estimate the expected performance in the future.

Recall that E is the objective function (Eq. (3.9)), then $E_{tr}(t)$ represents the mean square-error per example over the training set after epoch t, and $E_{\text{ver}}(t)$ is the corresponding error on the prediction set. As long as $E_{\text{ver}}(t)$ decreased, training continued. When the prediction error started to increase, data training was stopped. The final result of the training in this case was the set of weights that exhibit the lowest prediction error that can be written.

$$E_{low}(t) = minE_{ver}(t');\ t' \leq t \tag{3.17}$$

where $E_{low}(t)$ is the lowest prediction error obtained in epochs up to t and $E_{ver}(t')$ is the error on the prediction set. The stopping criterion can be defined as

$$GR(t) = 100\left(\frac{E_{ver}(r)}{E_{low}(r)} - 1\right) \quad (3.18)$$

where *GR(t)* is the percent of generalization loss at epoch t. This corresponded to the relative increase of the prediction error over the actual lowest. In fact, Eq. (3.17) determined the optimum (or lowest error) on the prediction set, while Eq. (3.18) estimated the relative increase of the prediction error over the observed minimum (or optimum). Generalization loss was obviously a serious reason to stop training, and therefore to avoid the overtraining. As the generalization error was estimated by cross-prediction with a holdout set, this allowed for comparing solutions and stopping when the prediction error was effectively minimum. Using STA with second order optimization method (LMBP) should reduce the training time, therefore the model could be retrained online to adapt to changing future events. Here, the training process was not required to converge; rather, the training process was used to perform a direct search of a model with superior generalization performance.

3.1.4.5 Design of MLP-ANN Architecture

The number of predictors and predicands specified the number of neurons in the input and output layers respectively. An experiment with trial-and-error measure, recommended as the best strategy by Shamseldin (1997) was used to determine the number of neurons in the hidden layer. In general, the architecture of multi-layer MLP-ANN can have many layers where a layer represents a set

of parallel processing units (nodes). The three-layer FNN used in this study contained only one intermediate (hidden) layer. MLP-ANN can have more than one hidden layer; however theoretical works have shown that a single hidden layer is sufficient for ANNs to approximate any complex nonlinear function (Cybenko, 1989; Hornik et al., 1989). Indeed many experimental results seem to confirm that one hidden layer may be enough for most forecasting problems (Zhang et al., 1998; Coulibaly et al., 1999). Therefore, in the study, one hidden layer FNN was used. It is the hidden layer nodes that allow the network to detect and capture the relevant pattern(s) in the data, and to perform complex nonlinear mapping between the input and the output variables. The sole role of the input layer of nodes is to relay the external inputs to the neurons of the hidden layer. Hence, the number of input nodes corresponds to the number of input variables. The outputs of the hidden layer are passed to the last (or output) layer which provides the final output of the network. The network ability to learn from examples and to generalize depends on the number of hidden nodes. A too small network (i.e., with very few hidden nodes) will have difficulty learning the data, while a too complex network tends to overfit the training samples and thus has a poor generalization capability. Finding a parsimonious model for accurate prediction is particularly critical since there is no formal method for determining the appropriate number of hidden nodes prior to training. Therefore, in this research, the trial-and-error method commonly used for network design was used.

3.1.6 Performance Assessment of Rainfall-Inflow Model

A number of error measures (Dawson et al., 2007; Legatees and McCabe, 1999) have been developed to assess the goodness of fit performance of hydrological forecasting models but no standard has been specified since each measure can just assess one or two aspects of the runoff characteristic. Three commonly used error measures, therefore, were employed in this study to make the evaluation of the model predictions. They are the Mean Absolute Error (MAE), the Mean Squared Relative Error (MSRE) and the Coefficient of Determination (r^2), respectively defined as follows:

$$MAE = \frac{\sum_{i=1}^{n} /Q_i - \hat{Q}_i/}{n} \tag{3.19}$$

$$MSRE = \frac{\sum_{i=1}^{n} \frac{(Q_i - \hat{Q}_i)^2}{Q_i{}^2}}{n} \tag{3.20}$$

$$r^2 = \left[\frac{\sum_{i=1}^{n} (Q_i - \bar{Q})(\hat{Q}_i - \tilde{Q})}{\sqrt{\sum_{i=1}^{n} (Q_i - \bar{Q})^2 (\hat{Q}_i - \tilde{Q})^2}} \right]^2 \tag{3.21}$$

where Q_i is the observed discharge, $\hat{Q}_i$ is the simulated discharge, $\bar{Q}$ is the mean of the observed discharges, $\tilde{Q}$ is the mean of the simulated discharges and n is the length of the observed/simulated series.

The MAE (Nash and Sutcliffe., 1970), which ranged from 0 to + ∞, was used to measure how close predictions were to the eventual outcomes. Theoretically, a coefficient of zero (MAE = 0) meant the best model with a perfect performance. The MSRE, which ranged from 0 to + ∞, could provide a balanced evaluation of

the goodness of fit of the model as it was more sensitive to the larger relative errors caused by the low value and the best coefficient would be zero (MSRE = 0). The $r^{2,}$ which ranged from 0 to 1, was a statistical measure of how well the regression line close to the observed data and coefficient of one (r^2=1) indicated that the regression line perfectly fitted the observed data.

3.2 Design of Reservoir Inflow Predicting Software

The detailed software design started from the identifications of system functional requirements. It defined the requirements from the perspective of the users of the system based on the experience in the past industrial applications. The document mainly described the functional requirements of user account management, target configuration, method configuration, end user forecast, analysis and report, search function, data requirements and graphic user interface requirements.

(a) Software Architectural Design: WBS was designed as a component-based software system. The component of GUI Controller accepted all HTTP requests and translated the requests to events. It passed these events to other components for processing and re-directed to correct resulting pages. The component of User Account Management (UAM) handled user account. It implemented the WBS user management functionality. Administrator of WBS used this functionality to manage the user registration and user information. The core component of the system is the Forecasting Engine. It was designed as an

independent deployable component that can be assembled with other systems. It implemented comprehensive core functions of forecasting with other six components, which are:

- User Forecast (UF) that handles the forecasting process for end-users;

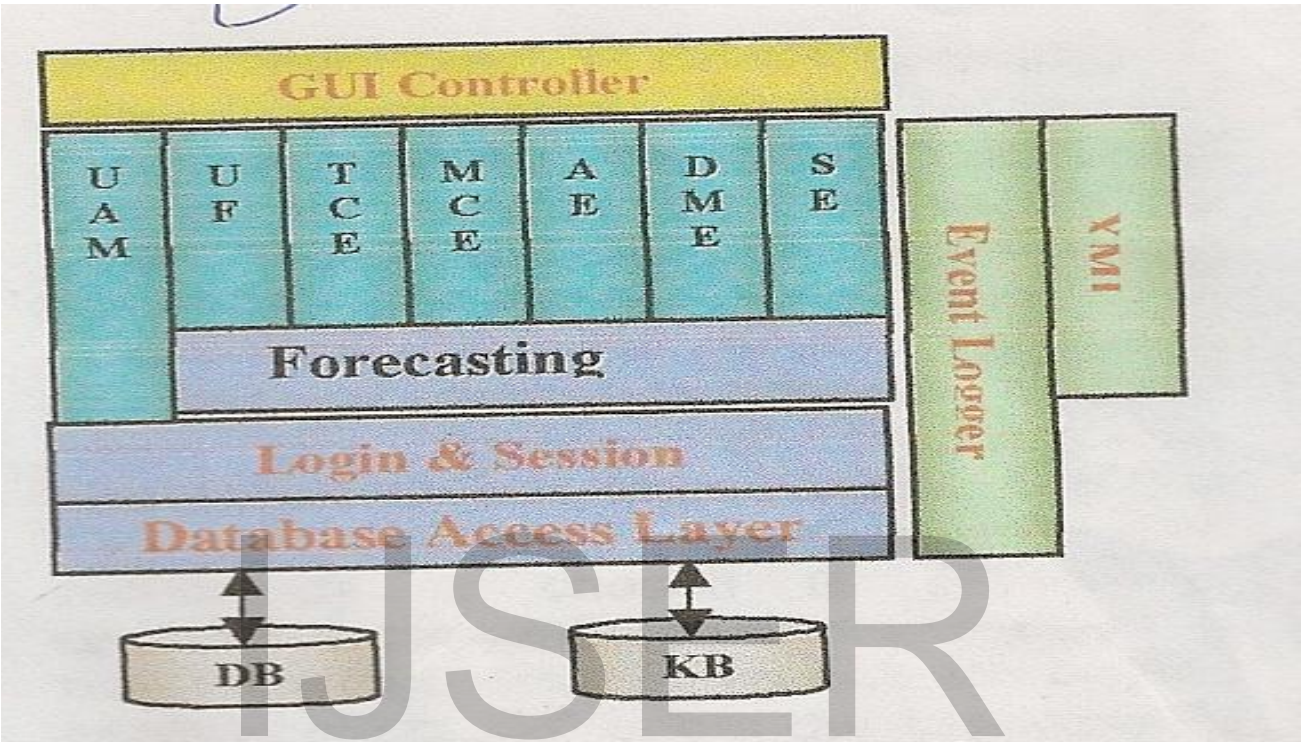


Fig. 3.2: WBS System Architectur

Source: Waleed (2003)

- Target Configuration Executor (TCE) that handles the target configuration process for super user to set up target and assign the target to end users;
- Method Configuration Executor (MCE) that handles method configuration process for super user to configure the forecasting method for targets;

- Analysis Executor (AE) that handles the forecasting method analysis processes to check and analyze forecasting accuracy;
- Data Management Executor (DME) that handles the data management for super user to load and edit the data;
- Search Executor (SE) that implements the search function for end users to search previous forecasting results as well as for administrator to search user account.

Login and Session Management provided the authentication for end users and stored session related data. Database Access Layer linked with the dataBase (OB) and Knowledge Base (KB) to store and retrieve the data. Event Logger was used by all other component to record significant events into log files. XML utilities were used to transfer and standardize the data from backend data sources to WBS database.

3.2.1 Graphical User Interface (GUI) Design

Within the scope of the project, a Graphical User Interface (GUI) was designed based on the system functional requirements and the workflow. The GUI of the system was designed with the Web enabled capability by using Microsoft FrontPage 2000. It covered in two different portions: access control and users menu.

3.2.2 Forecasting Engine Design

WBS forecasting engine was designed with four main modules and four supporting modules. The four main modules are user forecast, forecast configuration, analyzer and data extractor. The four supporting modules are data access layer, knowledge base, database and component interface API. The main modules received requirements from EJB controller through the component interface API, then extracted and used data and knowledge from the supporting modules. Their decisions or results were in turn used to update the database and the knowledge base through the data access layer.

3.2.3 Software Workflow Design

The workflow design defined and described how WBS system works from the user point of view. Workflow diagrams were used to represent system scenario and dynamic behavior.

3.2.4 Web Tier/EJB Controller Design

The design of Web tier and EJB controller defined the way to achieve the system functionality in the JSP (JavaServer Page) and EJB (Enterprise Java Bean). The Model-View-Controller (MVC) application architecture concept was adopted. MVC helped in the process of breeding an application up into logic components that can be structured more easily. JSP technology was chosen to provide a method of developing servlets. Along with all benefit, servlets offer, JSP offered the capability to rapidly develop servlets where

content and display logic are separated, and to reuse code through a component-based architecture. Separate EJB controller was implemented for each component. They could be plugged in and played in application server platform without affecting other components. This also facilitated the maintenance of WBS system.

3.2.5 Database Schema Design

Both logic schema and physical schema models for WBS core database were defined. The logical schema model was used as the key input of the physical schema model, which in turn, served as the main input to the physical implementation of the WBS core database.

3.2.6 Database Access Layer API Design

The Database Access Layer (DAL) was used by a number of modules that was designed to perform different functional tasks. These tasks included the execution of user forecasting, configuration of target, configuration of methods, analysis of forecasting results, presentation of data to the web-tier, management of user access control, and the import data utility.

3.3 Development of Web-based Forecasting System (WBFS)

3.3.1 Materials for Developing Web-based Forecasting System

Support tools and resources for building the (WBFS) comprised of data-collection hardware, computer hardware and software. Reliability and cost were

major factors in selecting both hardware and software. To reduce costs, free software packages available on the internet were used wherever possible.

(a) Data-collection Hardware: Data capture and input, generally the most time-consuming task in building any WBS, have two main requirements (Worboys et al., 1993). First, one provided both the physical devices for capturing data external to the system and a method for inputting the captured data into a database. The devices that were used in the WBFS were automatic loggers, wireless modems, and NT servers. The field data were collected automatically in real time and transmitted via wireless modem to an NT server for storage. It was downloaded in nearly real time by FTP and stored in a database, which is immediately accessible to the WBS model for further analysis. The field data consist of daily rainfall forecasts from online weather forecasting sites. A Java-based subsystem is used in the WBFS. The data-collection configuration is shown in Fig. 3.3.

(b) Software Component: A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. It could be independently deployed and run on a predetermined deployment infrastructure or framework.

The main advantages of software component are their scalability and reusability. The software component technology was applied to the WBS architecture system design, Web tier/EJB controller design, forecasting engine design and database tier design.

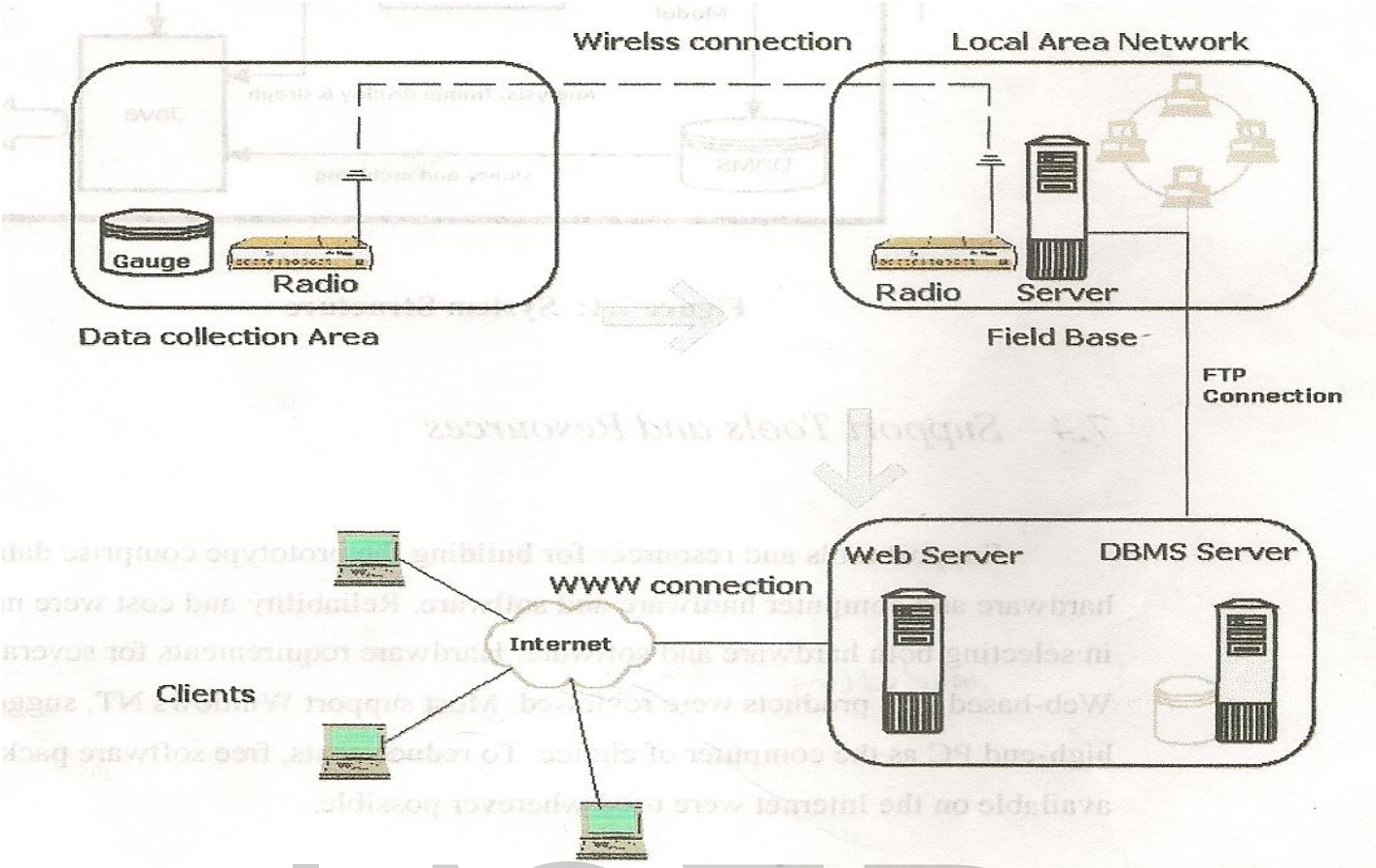


Fig. 3.3: Connection Model

Source: Waleed (2003)

(c) J2EE Technology: Java 2 Enterprise Edition (J2EE) was developed by Sun Microsystems, Inc in 1995. J2EE consists of multitier applications by replacing the client application with a Web browser and HTML pages powered by servlet/JavaServer Pages (JSP) technology. WBFS was designed in J2EE three-tier architecture. JSP was adopted to create dynamic pages for implementing presentation in Graphical User Interface (GUI). Each page was implemented with JSP page that talked to JavaBean class to retrieve data. JavaBean was also a part of JEE technology. It is a reusable, interoperable software component that

can be visually manipulated with builder tool. Session beans and entity beans were applied in the WBFS Web tier/EJB controller design.

(d) Weblogic Application Server: WebLogic application server allows a user to quickly develop and deploy reliable, secure, scalable and manageable applications. It manages system-level details so that the user can concentrate on business logic and presentation. A concept of WIFS was done for approving Web technologies by using Weblogic application server.

(e) Computer Hardware and Accessories

***1. Web-Server Computer*:** A dell PowerEdge computer, with a 1000MHz Intel Pentium M processor, 500-Mb memory, and a 500 Gb hard disk, was chosen for the Web server.

2. Database-server Computer: The database-server computer is similar to that of the Web server computer. A database server allows a client to pass Structured Query Language (SQL) request messages to the database, which resides on the same computer, and to return results.

3. Software: All software packages and data sets used were obtained free from the Internet or a government agency.

4. Operating System: Windows NT 4.0 was selected for the software, over Unix. Although not the best technical choice, because it limits the choice of Web servers, it is sufficient to run the system and is cheaper.

5. Web Server: A Web server supports the Internet protocols, including HTTP. For the software, Microsoft's Internet Information Server (IIS) was chosen. Although other Web servers were considered, IIS was selected mainly because it is part of the Windows NT option pack and can be downloaded from the Internet. For the Java-based application, Servertec Internet Server 1.10.3 was used. It is a small, fast, scalable, and easy-to-administer platform-independent application and Web server.

6. Database System: The database system is clearly a fundamental component. Some database systems are large and multi-featured, like Oracle and Sybase, but are expensive. Others are small and free, like mSQL and MySQL mSQL were selected, primarily because the storage requirement was not large. An operational system would require a more powerful database system.

7. Java-servlet Engine: Because the Web server chosen did not support the Java servlet API, a Java-servlet engine was needed. Although primarily a Web server, the choice was the Servertec Internet Server, which can also be used as a servlet engine for any application or Web-Server supporting Apache Modules, ISAPI, NSAPI, CGI or Java It runs on any platform supporting the Java Runtime Environment (IRE) version 1.1.x or later, and supports all industry-standard platforms. A servlet can be thought of as a Java applet that runs on the server side. Java servlets have the following advantages over CGI scripts:

i. Servlets are more efficient because they are loaded only once;

ii. Servlets are more secure due to Java's memory management support and

iii. Servlets are easier to develop and faster to deploy. Their main disadvantage is that some Web servers do not support the Java servlet API.

8. Development Tools: Development languages that were used include Java, Pen, and HTML, with various Web browsers. The Java Development Kit 1.1.8 Java compiler was used. Java has two important APIs, the applet API and the servlet API. Both were used for model calculation and drawing charts and servlets for data querying.

9. Structured Query Language: Structured Query Language (SQL) was used to retrieve, create, update, and delete tuples in a relational database. mSQL was used with mSQL-JDBC, a database-access API for the mSQL database engine that conforms to the Java Database Connectivity (JDBC) API. mSQL and the mSQL-JDBC are freely available for non-commercial use. The JDBC API supported both two-tier and three-tier models for database access.

10. Java-based Applications: Because users often experience difficulty transferring their existing knowledge to the realm of data handling (Davis and Medycki-Scott, 1994), careful attention must be paid to the user interface (Chopra, 1996; Peng, 1997; Wang, 1997). The software Java-based interface used the AWT7° library (Krutsch et al., 2001; Rodngues, 2001),

3.3.2 Procedure for Implementation of Web-based Forecasting System

(a) System Requirements: Functional requirements for web-based modeling software include the ability to invoke remote services, share information, and execute functions across heterogeneous computing environments. More specifically, the requirements include;

i. Real-time data acquisition and analysis
ii. User-side operation with a Web browser only
iii. Performance of under a few seconds per request
iv. Low-cost, particularly for the user

Technical support requirements include hardware, software, an Internet connection, and some development tools. The Internet connection had high-bandwidth and connected to the Internet through a local area network with a firewall. Firewall is a system that protects a local area network from unauthorized Internet access, a firewall server controls all communication passing between the Internet and the local network.

(b) System Architecture: The final system structure of the prototype is shown in Fig. 3.4, illustrating the relationships among the six major system components. The data collection sub-system, providing data from devices in the field, fed into the dynamic model, shown in the upper left of the diagram. The dynamic model communicated with both the DBMS server, shown in the lower left, and the Java user interface, shown in the lower right. The diagram showed that the

communication interface between the model and the Java user interface transmitted time series files. Finally, the user's web browser, shown on the right side, ran on the user machine. With limited hardware and software resources, the functionality of the software was moderate yet it provided frequently used hydrological techniques. It is a raster-based system permitting the integration of environmental modelling functions with classical WBS functions such as database maintenance and screen display. WBS functions and modelling functions were incorporated in a single WBS–modelling language for performing both WBS and modelling operations. The following section present some of the factors and reasoning that was considered as the software that was designed.

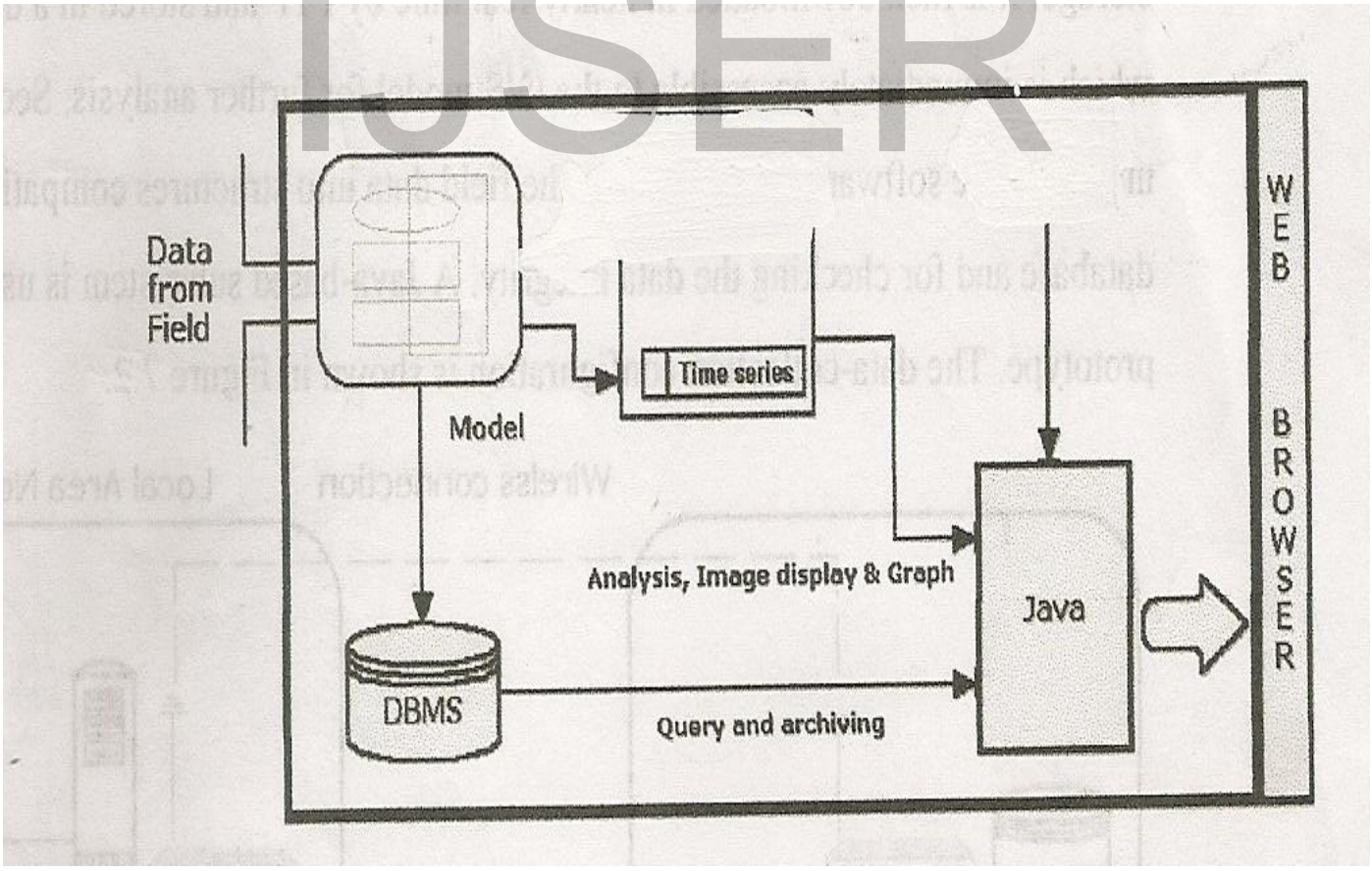


Fig. 3.4: System Structure

Source: Waleed (2003)

(c) Database Connection: The software used two data-retrieval channels. One downloaded data files for local rendering or analyzing. These files were pre-deployed onto the Web server; an applet simply downloaded them from the appropriate URL using standard HTTP. The second obtained statistical, availability, and other analyses about data. This latter channel was isolated inside the DataManager interface. The software was configured to support three database engines, mSQL, MySQL, and Access. The data managers available are:

i. mSQL direct connection — used in current version
ii. Access ODBC direct connection — ready for connection to Access
iii. MySOL direct connection — ready for connection to MySOL
iv. Servlet with mSQL backend — used in current version
v. Servlet with Access backend — ready for connection to Access
vi. Servlet with MySQL backend — ready for connection to MySQL

The first three data managers allowed an applet to manipulate the data directly using a JDBC connection. These represented direct-connection methods a 2-tier approach. The last three data managers, allowing an applet to manipulate data using a special proxy servlet using a three-tier approach, represented servlet connection methods.

(d) Direct Connection: A Java database connection was chosen over the combination of HTML forms, CGI, and scripts because of its superior handling

of complex systems. JDBC (Hamilton et al, 1998), a Java API for executing SQL statements, consisted of a set of classes and interfaces written in Java (Horstmann et al., 1984) and provides access to a wide range of relational databases. The JDBC driver category that was employed was implemented using a native-protocol pure Java driver. This type of driver converted a JDBC call directly into the network protocol used by the DBMS, allowing for a direct call from the client machine to the DBMS server. An applet was downloaded from a Web server to the client's machine where it used JDBC to access the server. The applet was restricted by the Web-browser security so it could not access the client's local files and could only make a connection to the originating host. The driver accessed the database in the server by communicating with the JDBC applet driver provided by the database vendor. Fig 3.5 illustrates the database connectivity. Applets were not allowed to connect to servers other than those from which they came. Of course, if one signed an applet, one can access any data and execute any program. Firewalls, and some routers, restrict access by restricting ports; therefore they must be carefully designed.

(e) Servlet Connection: Three-tiered applications consisted of a client supporting the user interface, a Java servlet incorporating the application logic, and the database. Processes can be managed and deployed separately from the GUI and the database. Three-tiered systems were claimed to be more scalable,

robust, and flexible (Orfali et al., 1996). They can integrate data from multiple sources and the following characteristics were significant:

- The user interface cannot access the database directly — only through a well-defined API in the middle tier.
- The connection between the middle tier and the database tier is fast.
- Applets contain only view-related classes. The middle tier handles database- specific functions.

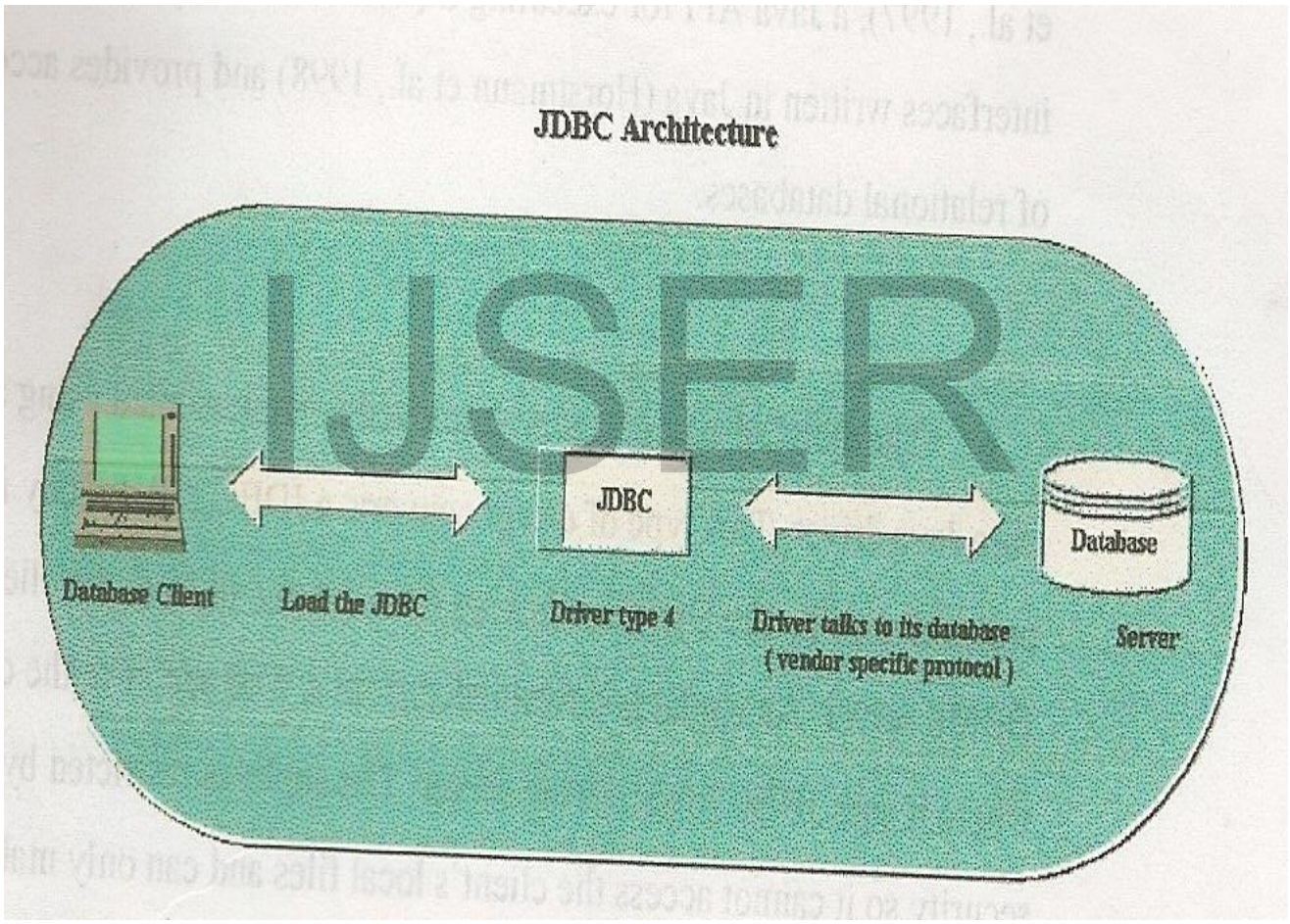


Fig. 3.5: Database Connectivity (JDBC)

Source: Waleed (2003)

(f) Database Manager Implementation: The software contained two implementations of the DataManager interface, BaseDBDataManager and

ServletDataManager. The former connected to a specified database using JDBC but, for the following reasons, was not very suitable for Internet applications:

i. Data was shared among all Internet users

ii. Database connections would not work behind firewalls

iii. Database drivers required some native code.

iv. An applet was required to download all database-related classes.

ServletDataManager, developed to take advantage of the three-tier architecture, is in fact just a very simple forwarding mechanism. The user interface encoded information and sent it to the servlet in the middle tier, which decoded it and passed it to the appropriate DataManager.

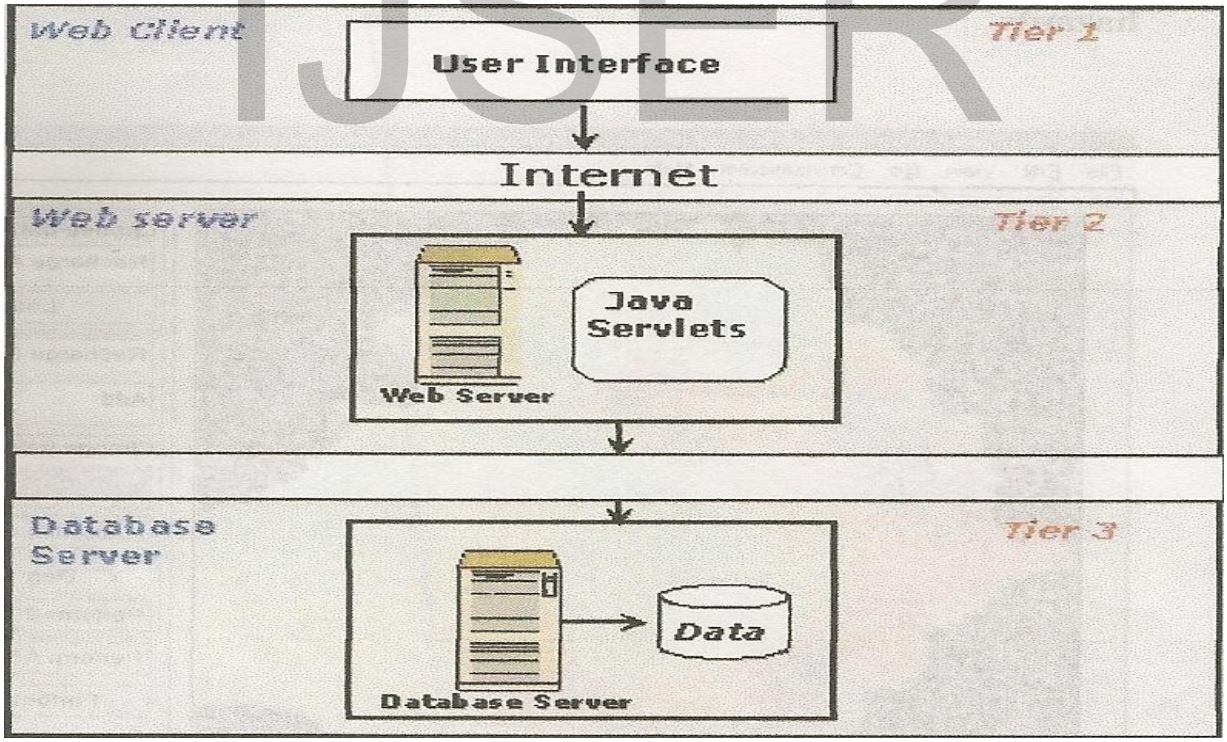


Fig. 3.6 Three-tier Approach

Source: Waleed (2003)

After a database connection had been established, through a JDBC or servlet data manager, the servlet data-manager page was activated.

***(g) Java User interface**:* The Java GUI comprised four pages. The analysis page provided analysis function including slope, aspect, accumulated flux, overlay, zoom, histogram, and metadata. The time-series (TSS) graph page provided time-series analysis. The data-manager page provided database querying via JDBC or the servlet. Fig. 3.8 illustrates the menu system.

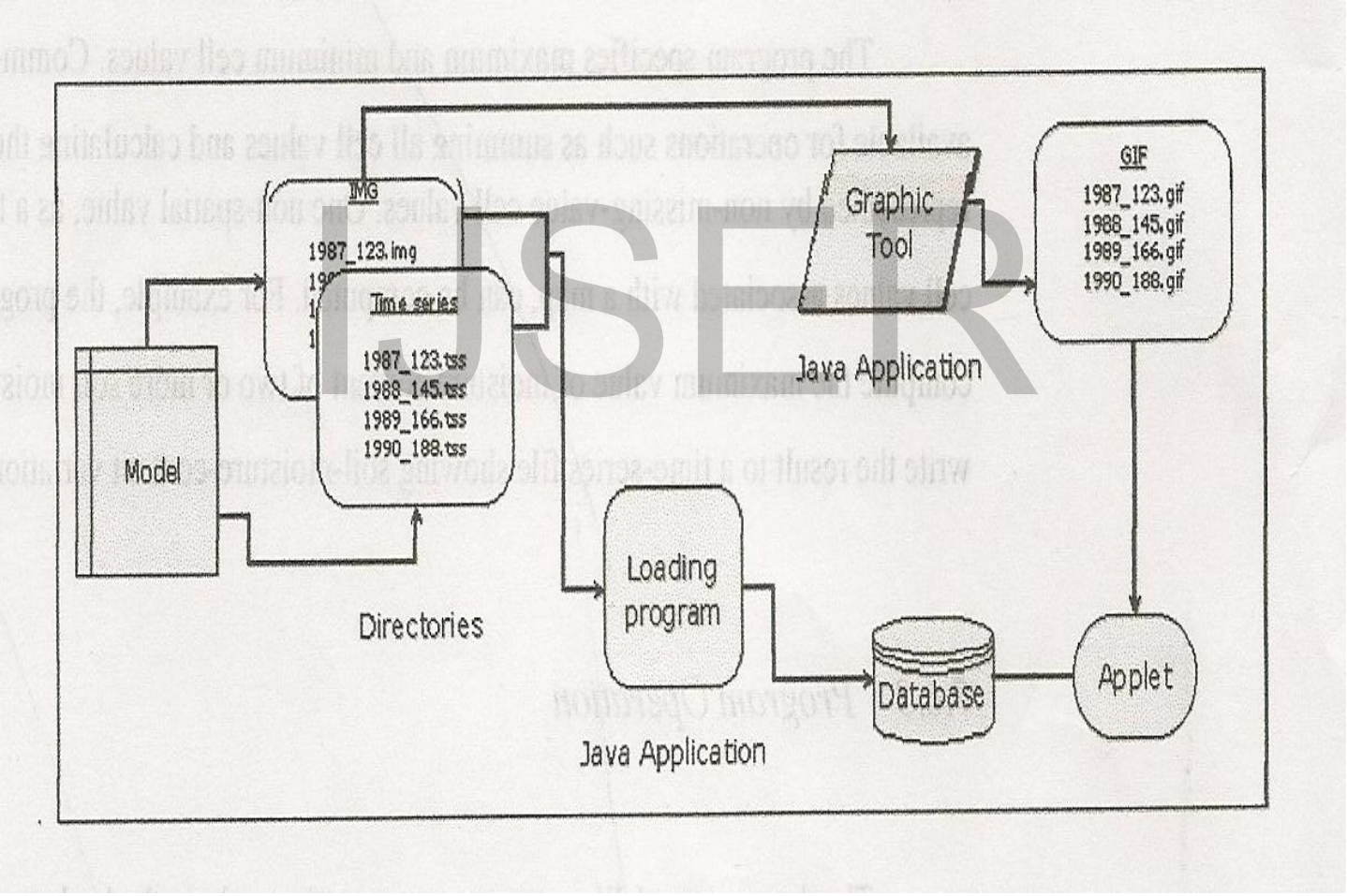


Fig. 3.7: Interactions within the Model

Source: Waleed (2003)

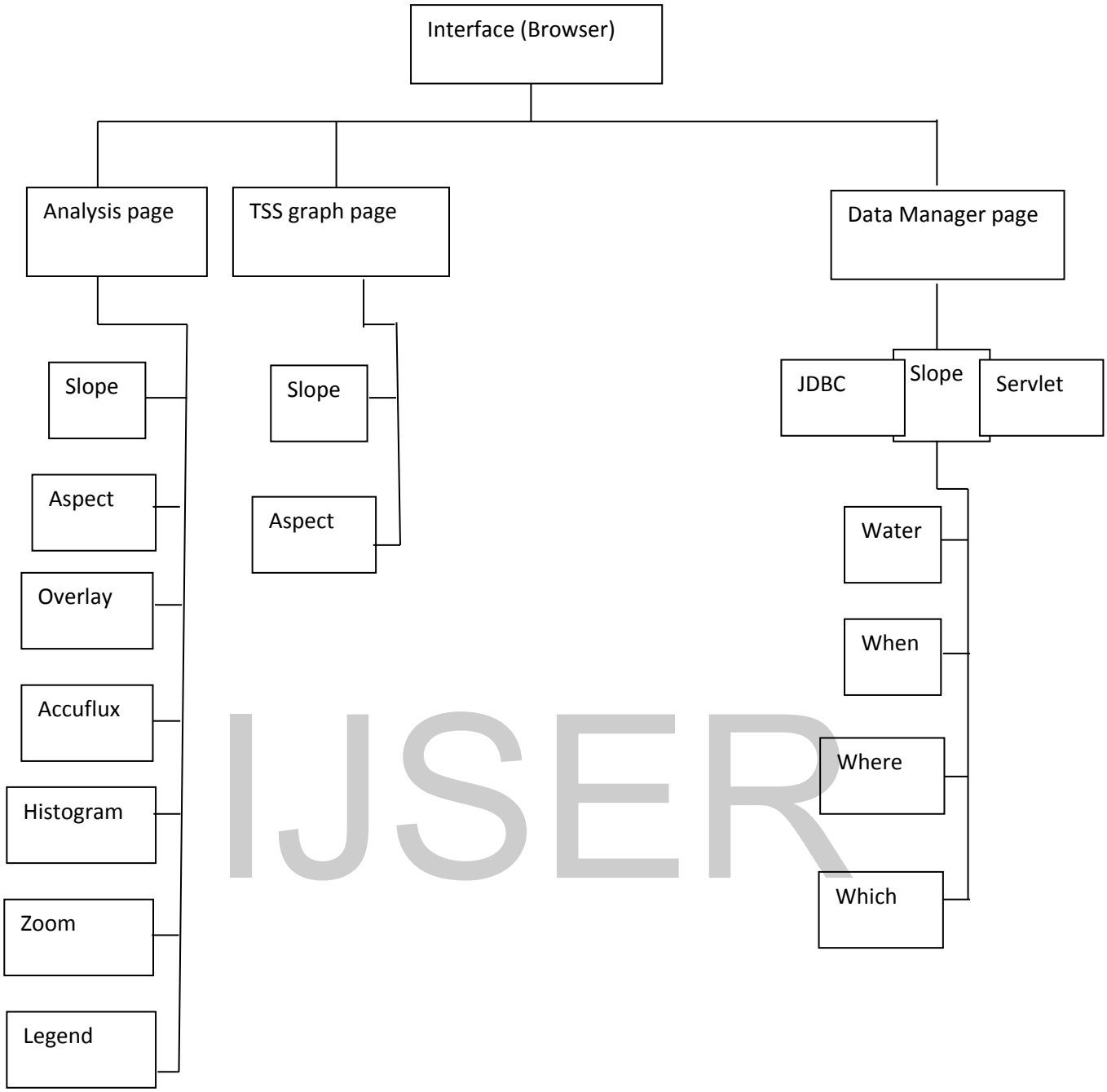


Fig. 3.8: GUI Menu System

Source: Waleed (2003)

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 Results of Rainfall-Inflow Modelling

Figures 4.1 and 4.2 show the variation in the inflow to Dadin-kowa Reservoir. The inflow increased gradually after the commissioning of the dam in 1988 until there was a decline in the inter-annual inflow in 1997. The effect of this is the inability to fill the reservoir due to siltation. The flow also exhibited a noisy pattern. The annual inflow showed that the reservoir inflow increased from the month of May to August, after which it declined in the month of September. The month of August is notably the wettest month in the area. Other months in the year never had any reasonable records of inflow over the years. This is a clear demonstration of the climatic characteristics of the reservoir catchment area. There were usually no rainfalls during those months of no inflows.

4.1.1 The Developed MLP-ANN Rainfall-Inflow Model

Tables 4.1, 4. 2 and 4.3 show the r^2 (coefficient of determination) tests for the analysis to determine the number of input nodes, hidden nodes and the training epochs for MLP network trained using back propagation algorithm (MLP-BP); From Equations 3.2 to 3.5, the learning rate, α and the momentum coefficient, β were automatically adjusted according to Sun et al. (2007) to yield: α= 0.001 and β = 0.85.

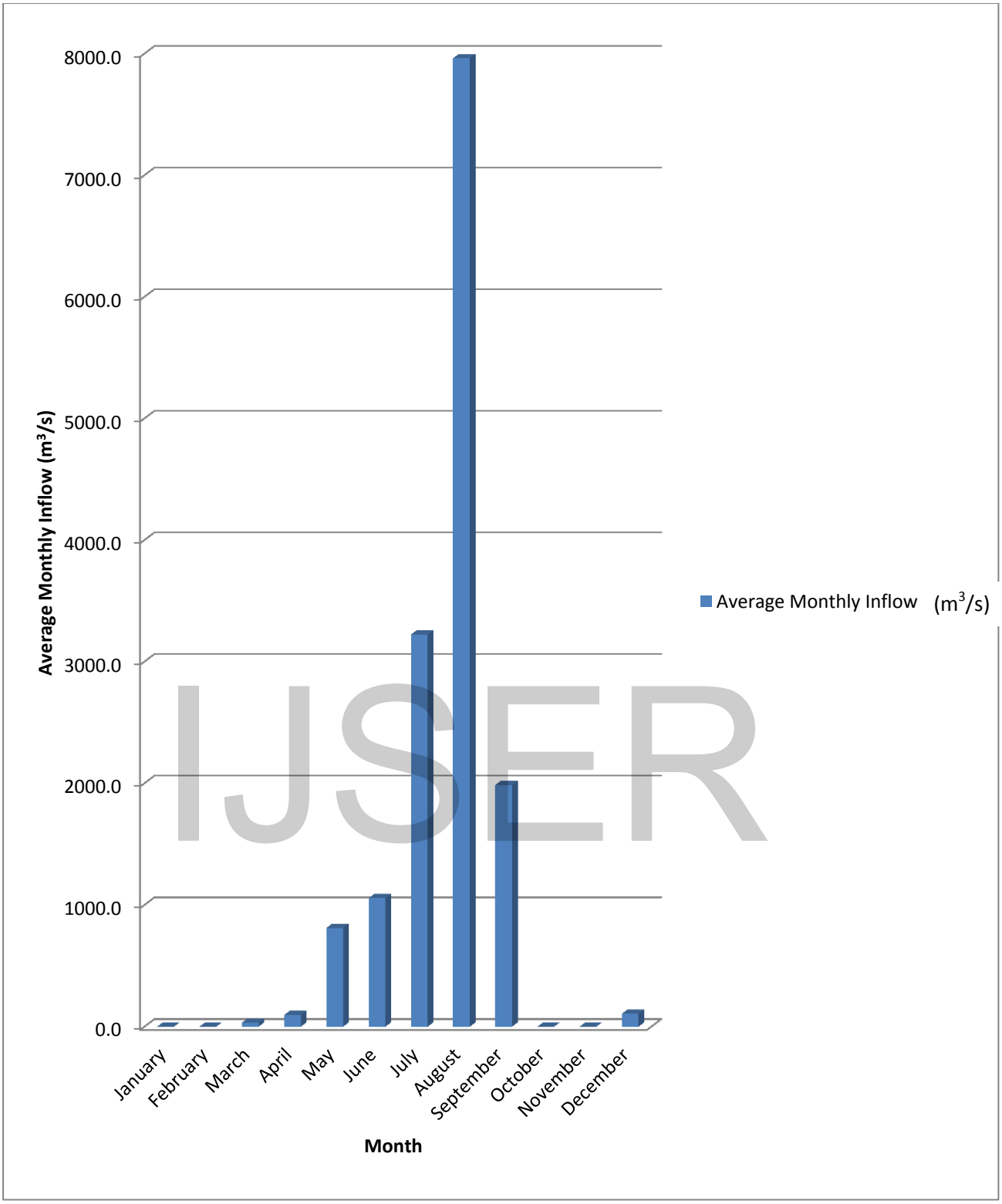


Fig. 4.1: Annual Inflow of Dadin-Kowa Reservoir

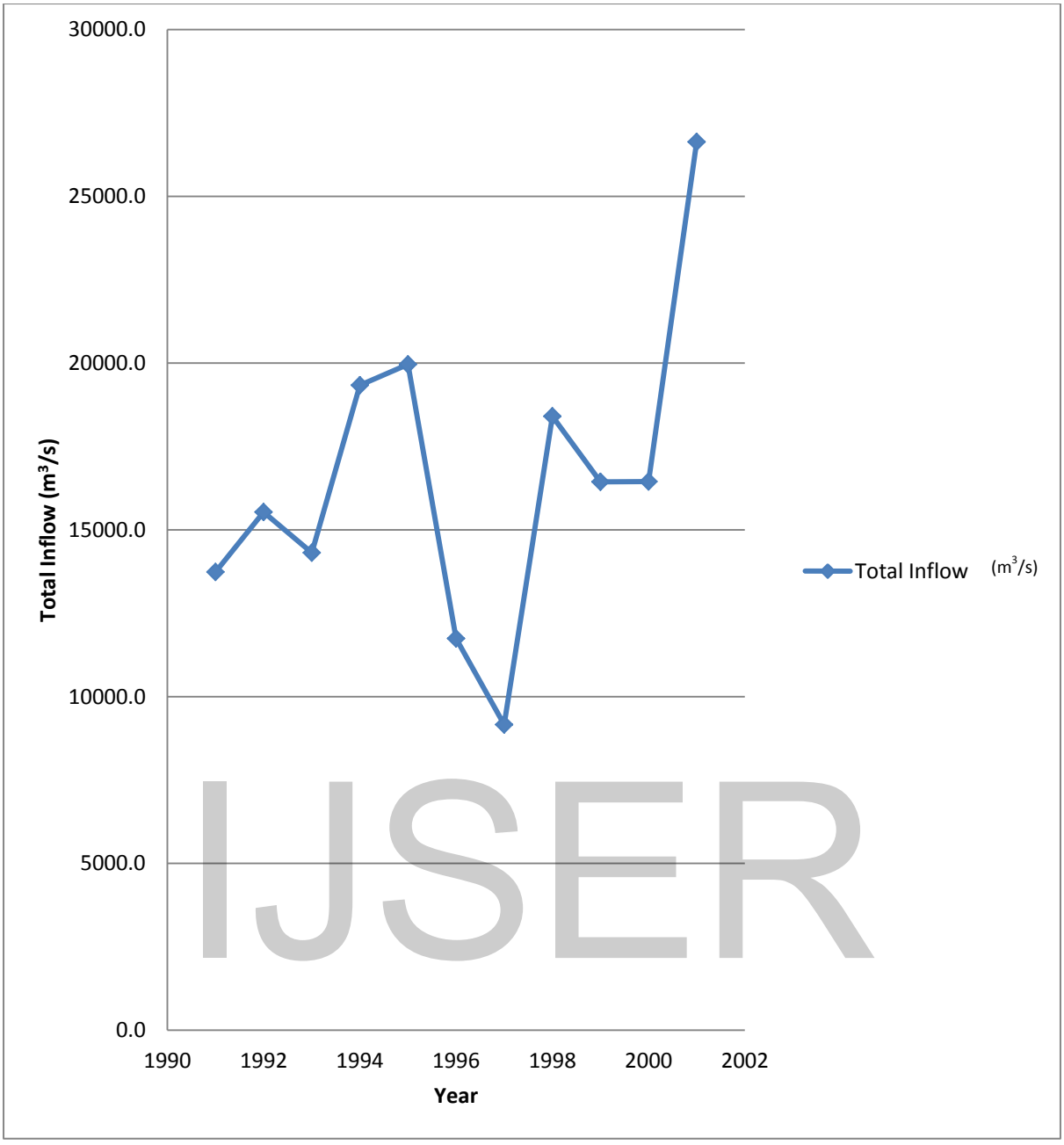


Fig. 4.2: Seasonal Inflow of Dadin-Kowa Reservoir

The results in Table 4.1 were produced by assigning hidden node for the neural networks model and the number of input nodes was varied to identify the best input node required by the neural network. It is clear that the best r^2 tests were produced with one input node, i.e., when only the last inflow lag is used. By assigning input node to one and changing the number of hidden nodes, the

results in Table 4.2 were obtained. The results indicated that the best number of hidden nodes for the network is 7.

Table 4.1: Variation of r^2 Tests with Different Input Nodes

No. of input nodes	MLP-BP
1	0.9290
2	0.3927
3	0.3635
4	0.4949
5	0.4949

The analysis to find the adequate training epochs was carried out and the results are shown in Table 4.3. The results suggested that the adequate training epoch is 1000 for MLP-BP.

Table 4.2: Variation of r^2 Tests with Different Hidden Nodes

No. of hidden nodes	MLP-BP
1	0.5959
2	0.8946
3	0.9290
4	0.9536
5	0.9521
6	0.9508
7	0.9570
8	0.9543
9	0.9507
10	0.9296
11	-
12	-
13	-
14	

In order to test the generalization properties of the neural networks models, r^2 tests for multi-step-ahead (MSA) predicting of the inflow were calculated. Numerous MLP-BP-ANN structures were tested in order to determine the optimum number of hidden layers and the number of nodes in each. The architecture (Fig. 4.3) of the best MLP-BP-ANN model for predicting reservoir inflow was composed of one input layer with one input node, one hidden layer with seven nodes and one output layer with one output variable.

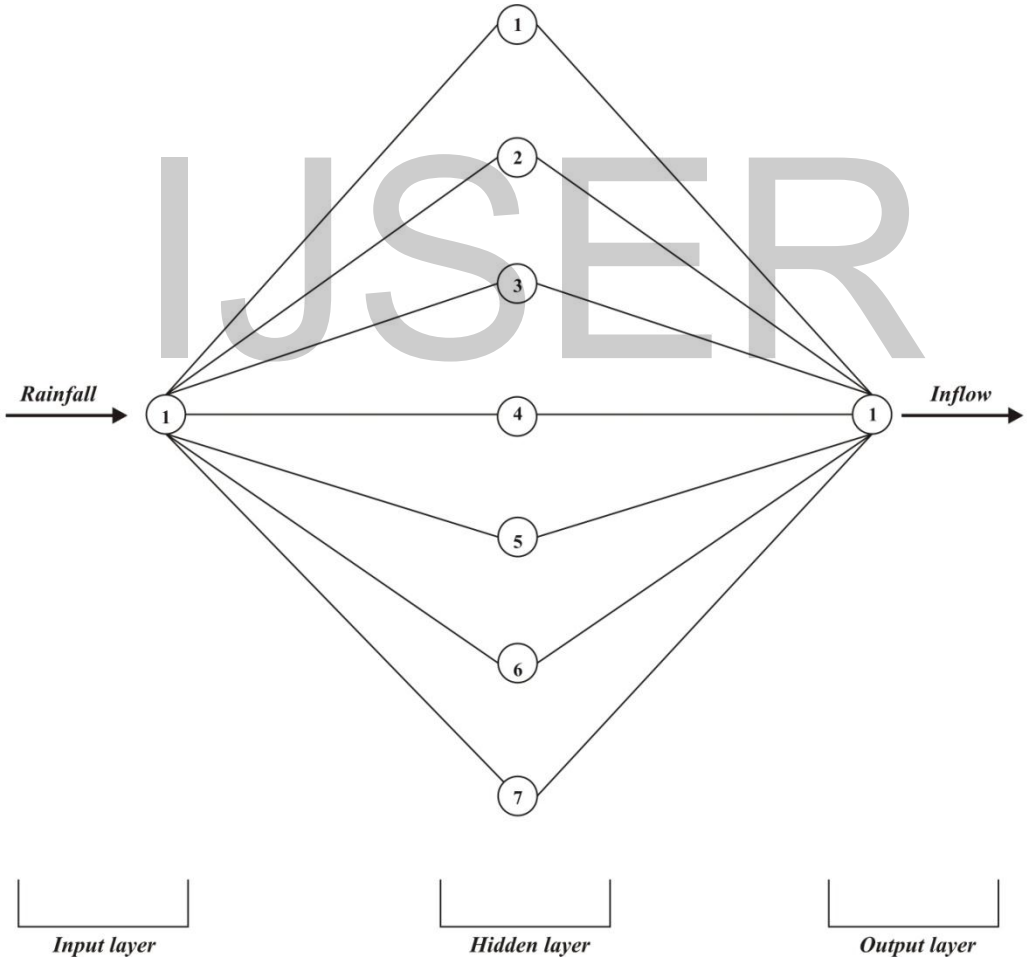


Fig. 4.3: The Architecture of the MLP-BP-ANN model

Thus, the MLP-BP architecture used: input node = 1, Hidden Nodes = 7, Training Epochs = 1000.

Table 4.3: Variation of r^2 Tests with the Number of Training Epochs

No. of epoch	MLP-BP
1	0.6300
2	0.6278
3	0.6233
4	0.6129
5	0.5847
6	0.5123
7	0.3538
8	0.1210
9	0.0854
10	0.2498
12	0.5184
14	0.7317
16	0.8567
18	0.8996
20	0.9242
100	0.9570
1000	0.9706
1500	-
2000	-
3000	-

r^2 tests for lead-time from 1-day up to 6-day of the inflow were calculated over both the training and independent data sets as shown in Table 4.4. The results for training data set indicate that MLP-BP gave good r^2 tests up to 4-day, 5-day and 6-day lead-time respectively, where their r^2 test values are about 0.8. The

results for independent data set in Table 4.4 showed that MLP-BP gave good r^2 tests up to 4-day, 5-day and 6-day ahead, respectively.

Another aspect that needs to be considered for on-line modelling is the complexity of the model. For on-line modelling and prediction, all the calculation for parameter estimation or adjustment must be carried out within the sampling time. Hence, the number of adjustable parameters should be as small as possible. For MLP network, the adjustable parameters are composed of connection weights between input nodes and hidden nodes, connection weights between hidden nodes and output node, and also the threshold in hidden nodes. Therefore, for one output network, the number of adjustable parameters for the network can be calculated using the following formula (Finnoff et al., 1993):

$$NAP_{MLP} = n_i \, x \, n_h + 2n_h = (n_i + 2)n_h \quad (4.1)$$

where *NAP*, n_i and n_h are the short form for number of adjustable parameters, number of input nodes and number of hidden nodes respectively. Based on the formula and the previously determined structure for the networks, the number of adjustable parameters for MLP networks was 21.

Table 4.4: r^2 Tests Calculated over both the Training and Independent Data Sets

Lead-time (days)	Training data set	Independent data set
	MLP-BP	MLP-BP
1	0.9888	0.9706
2	0.9771	0.9342
3	0.9491	0.8899
4	0.9098	0.7955
5	0.8672	0.7232
6	0.8138	0.6311

4.1.2 Results of Predicted/Observed Reservoir Inflow Hydrographs

Figures 4.5a-h show the relationship between the predicted and observed discharges for model calibration during the month of August (1991-1998). Figures 4.4a-h present the predicted and observed inflow hydrographs. Figures 4.6a-c show the predicted and observed inflow hydrographs for model verification.

Fig. 4.4a: Predicted and Observed Inflow Hydrographs of August 1991 in Model Calibration

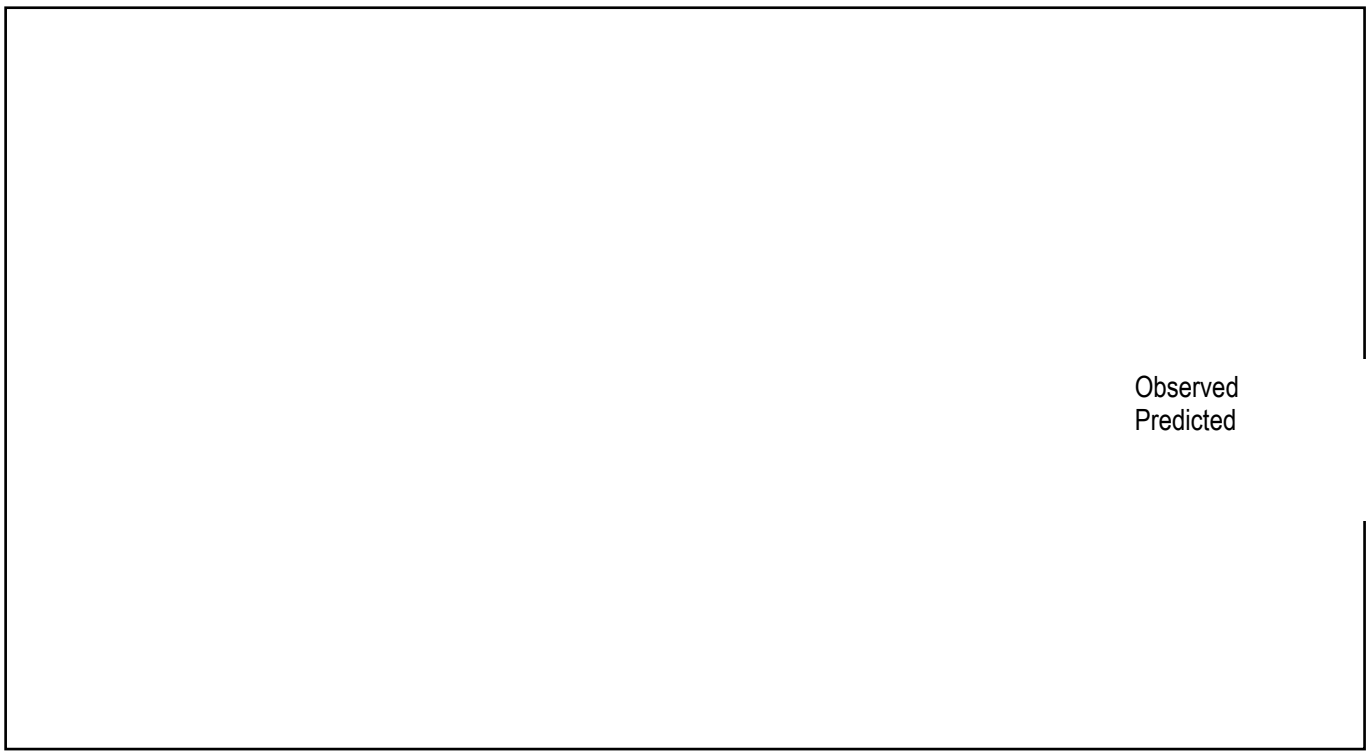


Fig. 4.4b: Predicted and Observed Inflow Hydrographs of August 1992 in Model Calibration

Fig. 4.4c: Predicted and Observed Inflow Hydrographs of August 1993 in Model Calibration

Fig. 4.4d: Predicted and Observed Inflow Hydrographs of August 1994 in Model Calibration

Fig. 4.4e: Predicted and Observed Inflow Hydrographs of August 1995 in Model Calibration

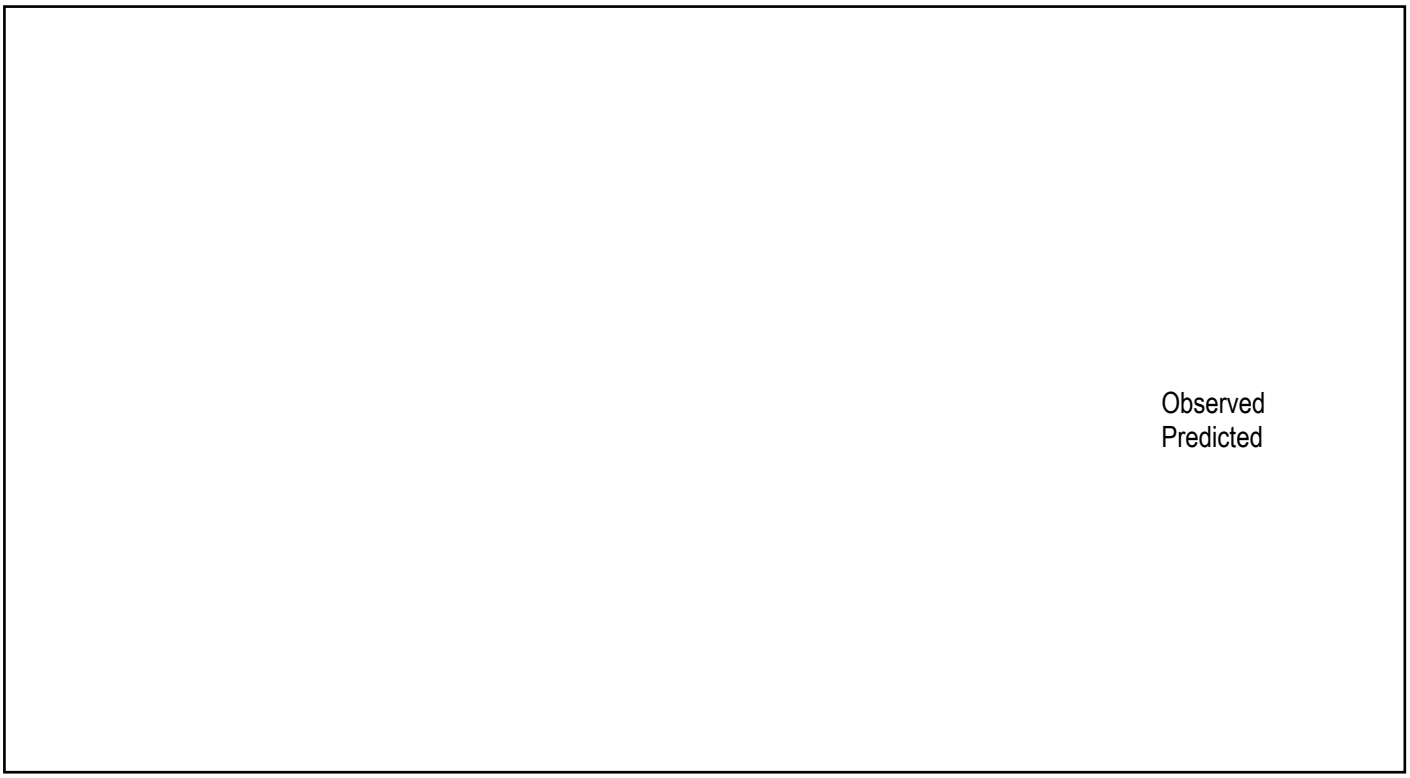


Fig. 4.4f: Predicted and Observed Inflow Hydrographs of August 1996 in Model Calibration

Fig. 4.4g: Predicted and Observed Inflow Hydrographs of August 1997 in Model Calibration

Fig. 4.4h: Predicted and Observed Inflow Hydrographs of August 1998 in Model Calibration

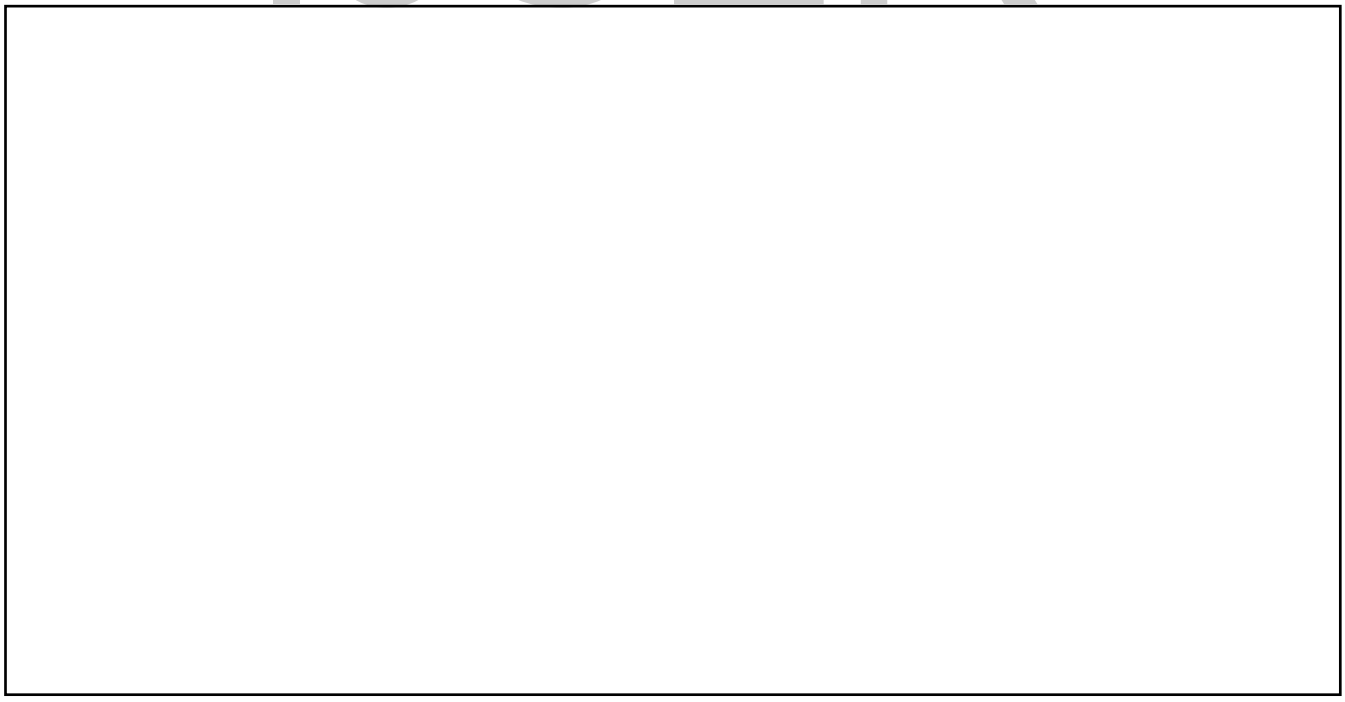


Fig. 4.5a: Comparison of Observed and Predicted Discharges of August 1991 in Model Calibration

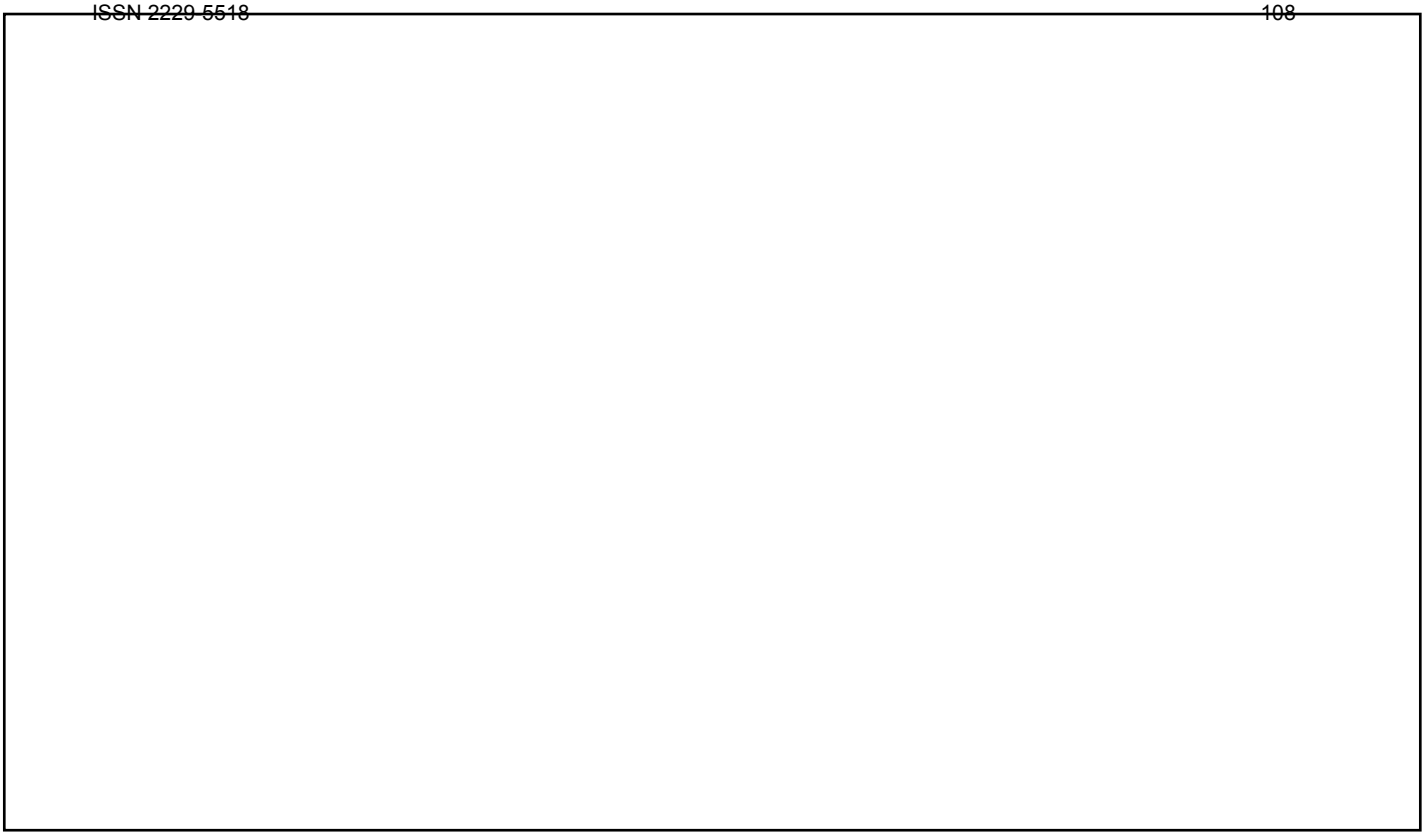


Fig. 4.5b: Comparison of Observed and Predicted Discharges of August 1992 in Model Calibration

Fig. 4.5c: Comparison of Observed and Predicted Discharges of August 1993 in Model Calibration

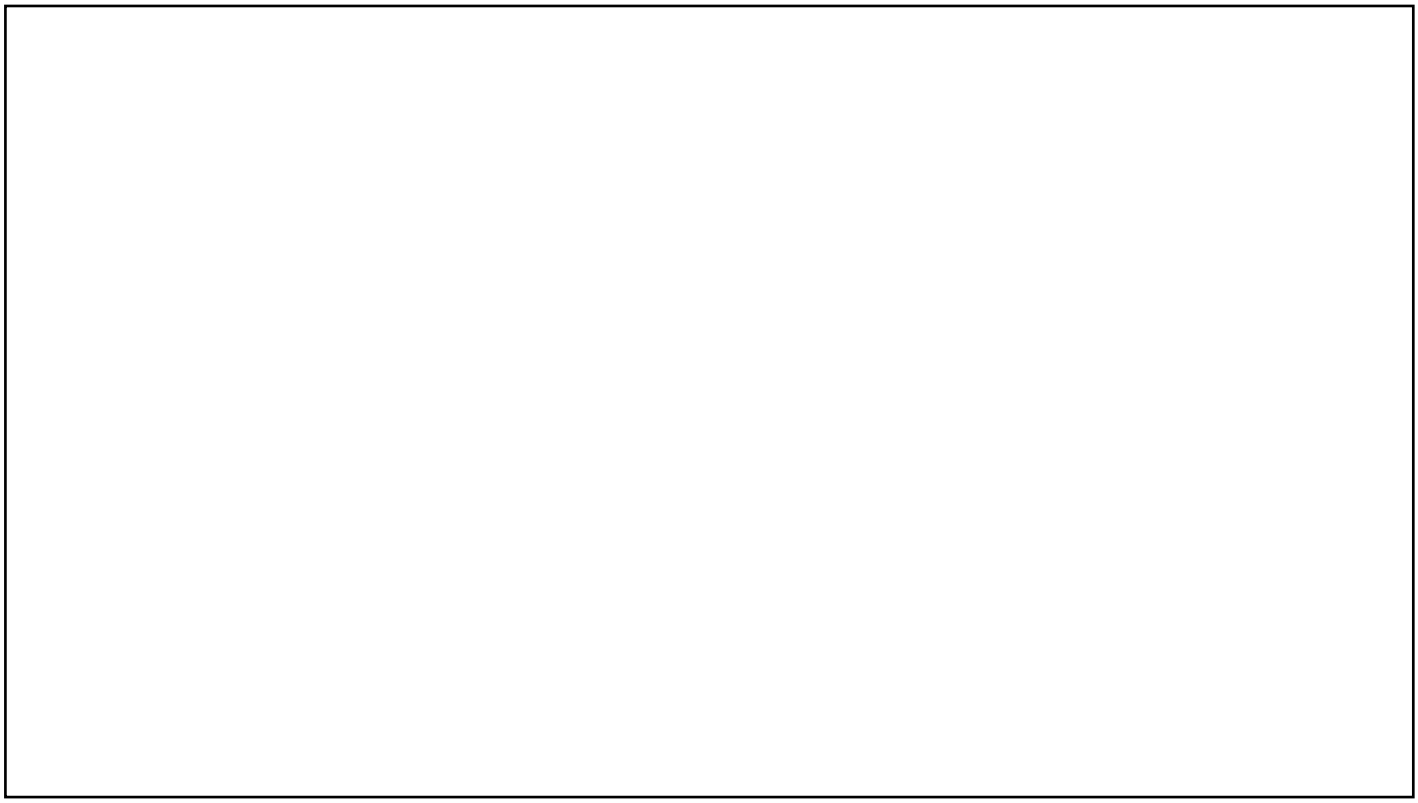


Fig. 4.5d: Comparison of Observed and Predicted Discharges of August 1994 in Model Calibration

Fig. 4.5e: Comparison of Observed and Predicted Discharges of August 1995 in Model Calibration

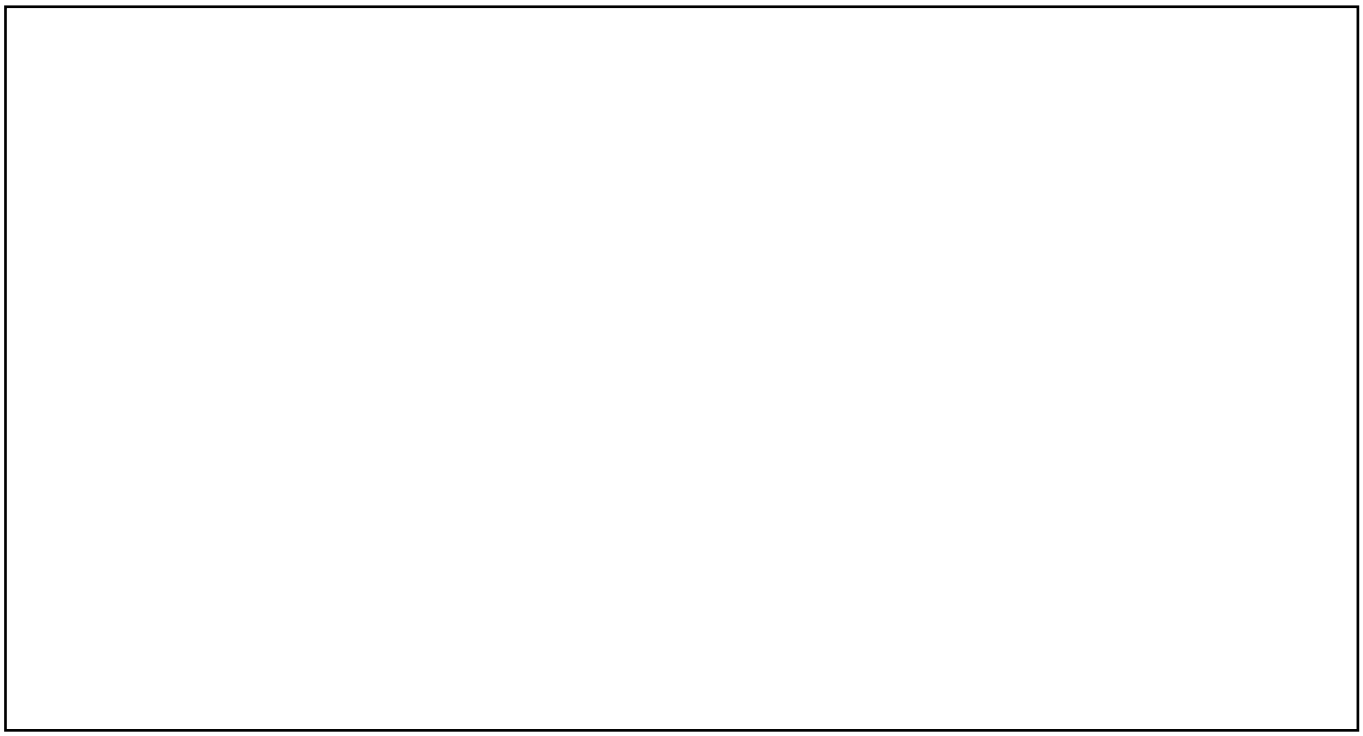


Fig. 4.5f: Comparison of Observed and Predicted Discharges of August 1996 in Model Calibration

Fig. 4.5g: Comparison of Observed and Predicted Discharges of August 1997 in Model Calibration

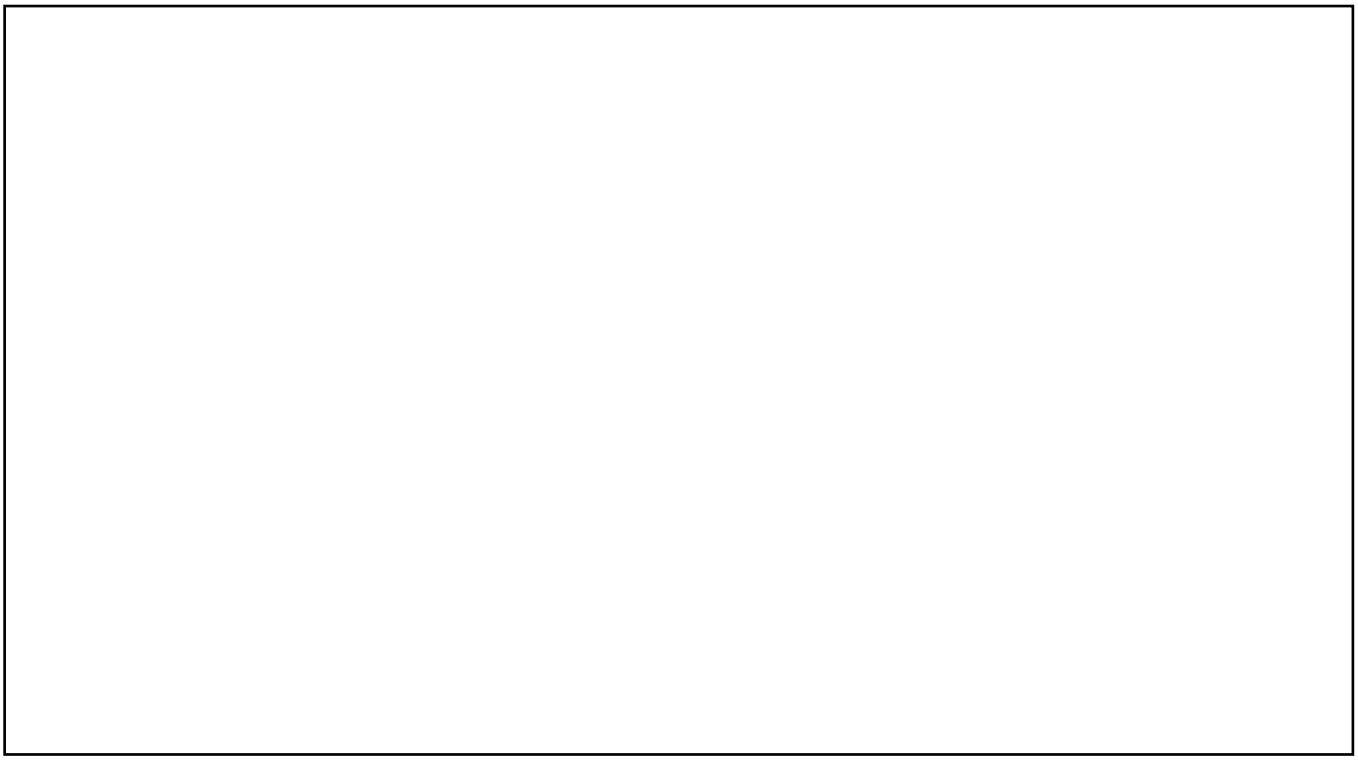


Fig. 4.5h: Comparison of Observed and Predicted Discharges of August 1998 in Model Calibration

Fig. 4.6a: Predicted and Observed Inflow Hydrographs of August 1999 in Model Verification

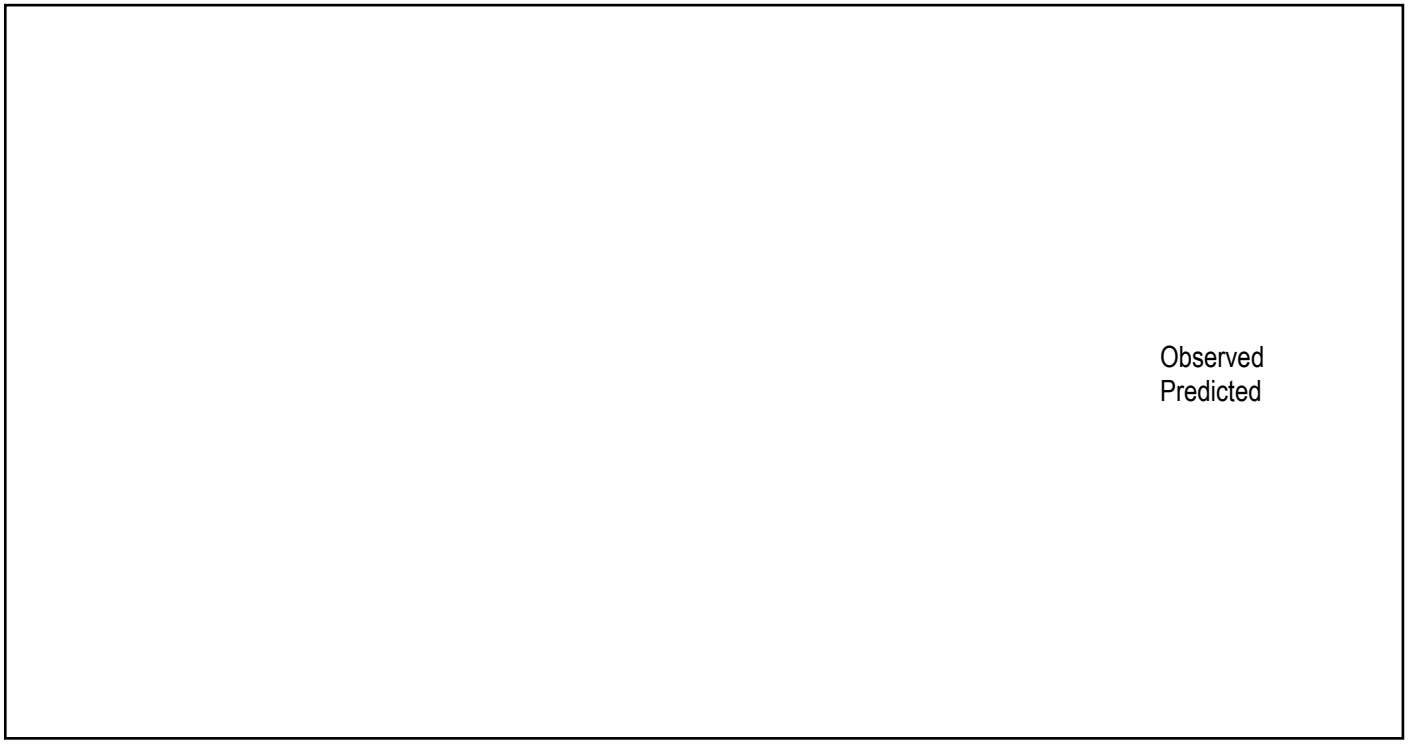


Fig. 4.6b: Predicted and Observed Inflow Hydrographs of August 2000 in Model Verification

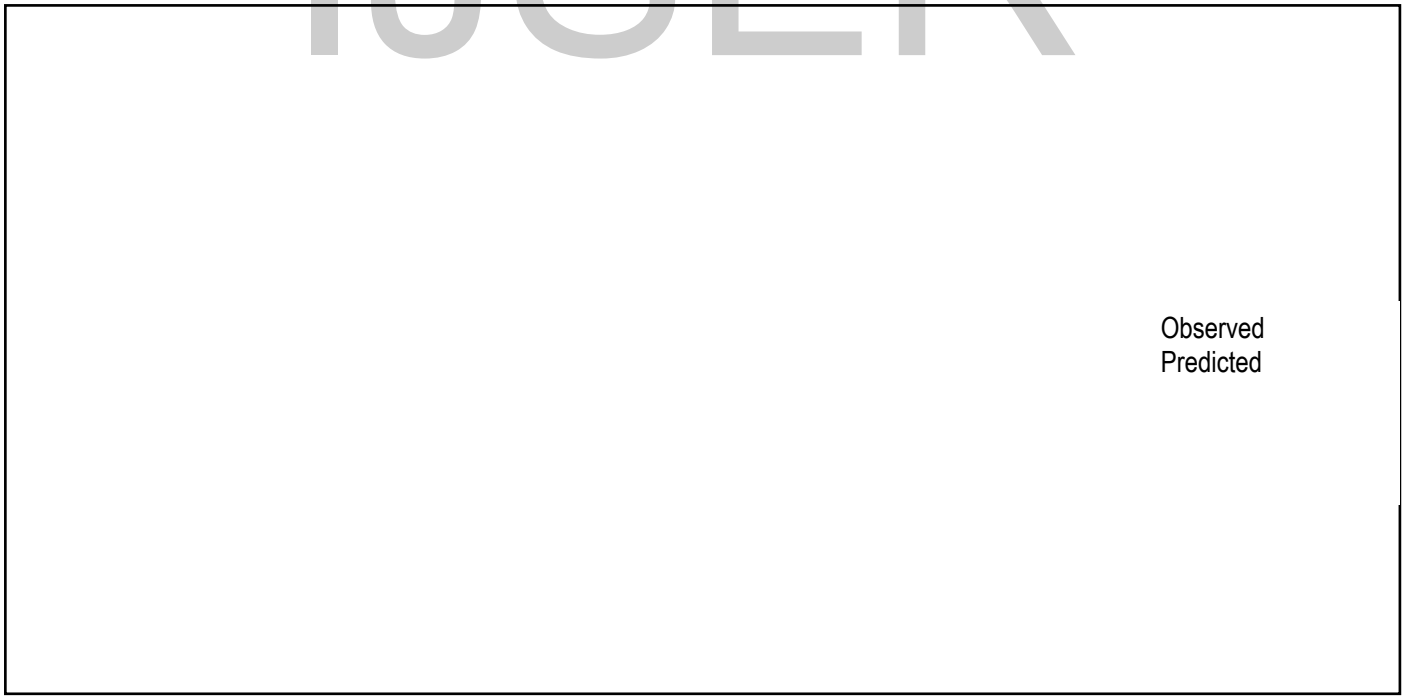


Fig. 4.6c: Predicted and Observed Inflow Hydrographs of August 2001 in Model Verification

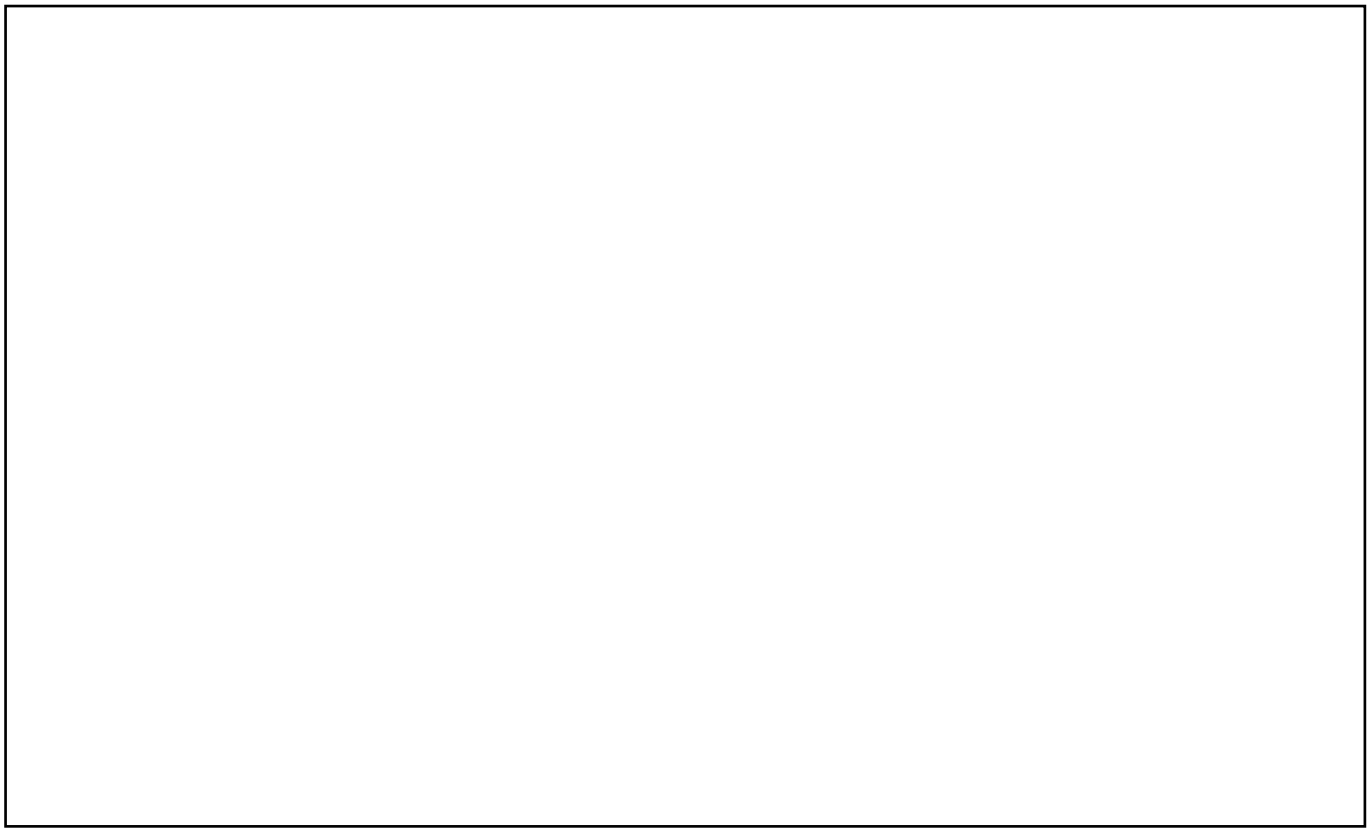


Fig. 4.7a: Comparison of Observed and Predicted Discharges of August 1999 in Model Verification

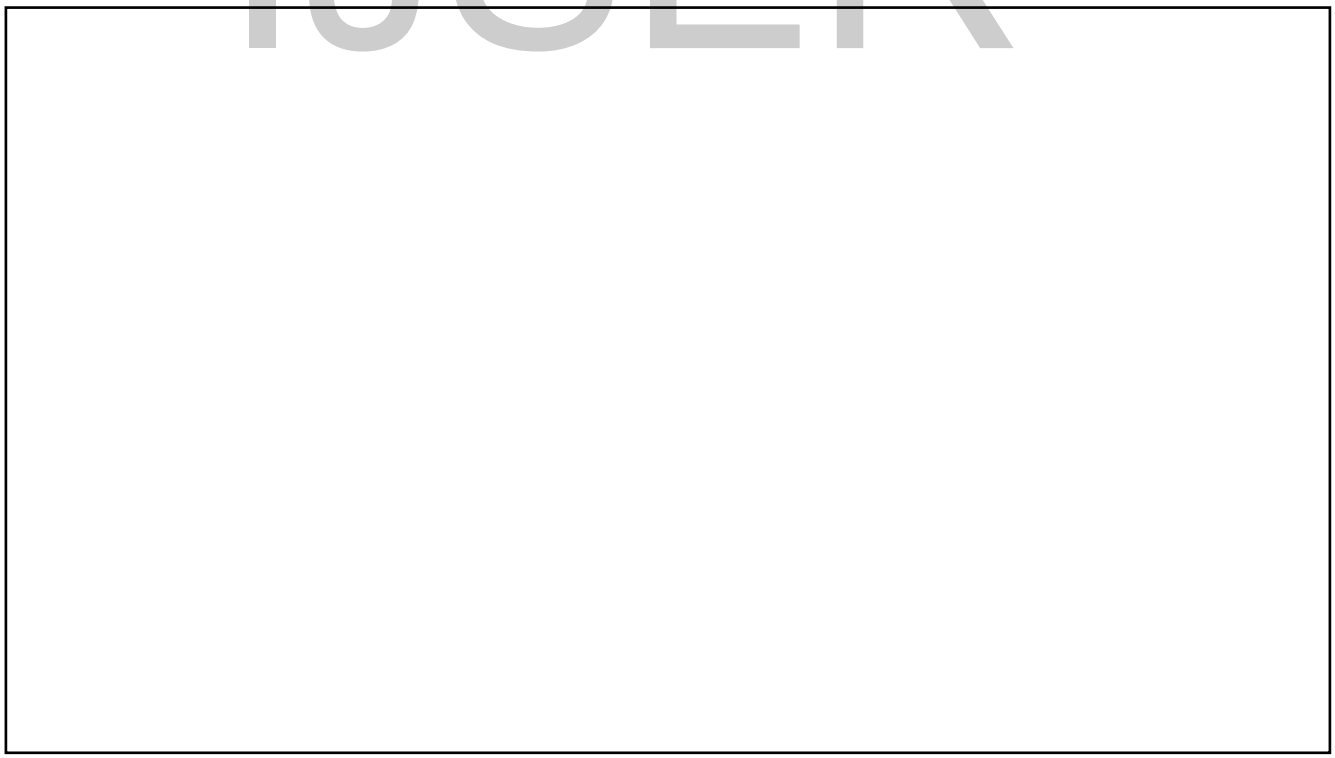


Fig. 4.7b: Comparison of Observed and Predicted Discharges of August 2000 in Model Verification

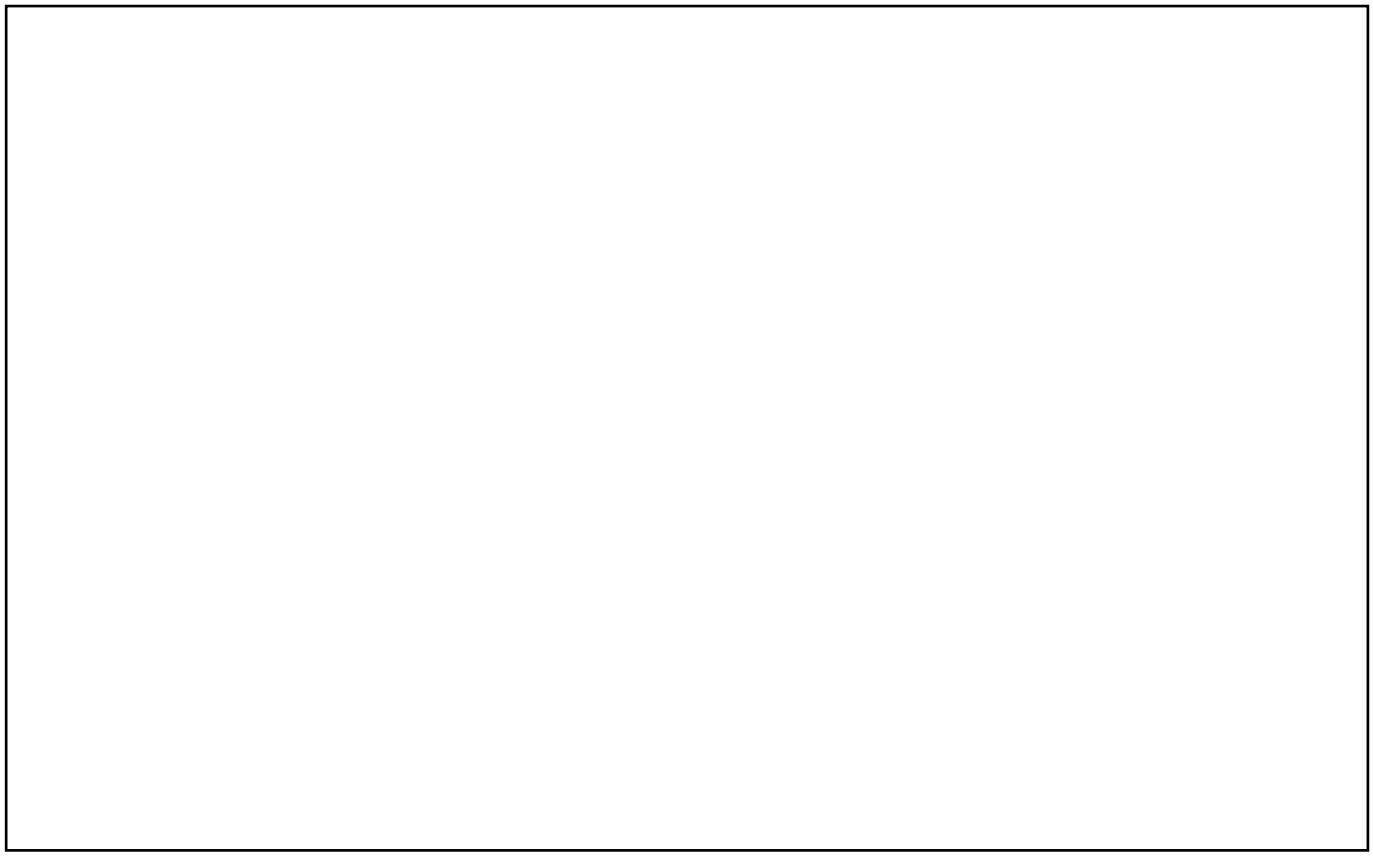


Fig. 4.7c: Comparison of Observed and Predicted Discharges of August 2001 in Model Verification

4.1.3 Results of Model Performance Assessment

4.1.3.1 Model Performance Assessment for Data Calibration (1991-1998)

The perfomance of the model was evalauted for the data calibration period (1991 – 1998).

From Equation 3.22, the mean absolute error (MAE) is determined as:

$$MAE = \frac{8.759\ X\ 10^{-3}}{1224} = \mathbf{7.156\ X\ 10^{-6}}$$

From Equation 3.23, the mean squared relative error (MSRE) is calculated as:

$$MSRE = \frac{\frac{8.180\ X\ 10^{14}}{4.460\ X\ 10^{18}}}{1224} \equiv \frac{8.180\ X\ 10^{14}}{4.460\ X\ 10^{18}} X\ \frac{1}{1224} = \mathbf{1.4984\ X\ 10^{-7}}$$

From Equation 3.24, the coefficient of determination (r^2),

$$r^2 = \left[\frac{(4.054\ X\ 10^{18})}{\sqrt{(3.765\ X\ 10^{18})(4.07\ X\ 10^{18})}}\right]^2 = \left[\frac{4.054\ X\ 10^{18}}{\sqrt{1.657\ X\ 10^{37}}}\right]^2$$

$$r^2 = \frac{4.054\ X\ 10^{18}}{4.071\ X\ 10^{18}} = (\mathbf{0.9688})^2$$

r =0.9688

4.1.3.2 Model Performance Assessment for Data Verification (1991-2001)

The perfomance of the model was evalauted for the data verification period (1991 – 2001).

From Equation 3.22, the mean absolute error (MAE) is determined as:

$$MAE = \frac{3.200\ X\ 10^{-3}}{459} = \mathbf{1.478\ X\ 10^{-5}}$$

From Equation 3.23, the Mean squared relative error (MSRE) is calculated as:

$$MSRE = \frac{\frac{8.811\ X\ 10^{13}}{1.673\ X\ 10^{18}}}{459} \equiv \frac{8.811\ X\ 10^{13}}{1.673\ X\ 10^{18}} X\ \frac{1}{459} = \mathbf{1.1478\ X\ 10^{-7}}$$

From Equation 3.24, the coefficient of determination (r^2) can be obtained as:

$$r^2 = \left[\frac{(1.520\ X\ 10^{18})}{\sqrt{(1.412\ X\ 10^{18})(1.526\ X\ 10^{18})}}\right]^2 = \left[\frac{1.422\ X\ 10^{18}}{\sqrt{2.155\ X\ 10^{36}}}\right]^2$$

$$= \frac{1.422\ X\ 10^{18}}{1.468\ X\ 10^{18}} = \mathbf{(0.9688)^2}, \quad \boldsymbol{r = 0.9688}$$

From the results shown in the graphs and calculations, a satisfactory prediction can be obtained in the study since the r^2 is sufficiently high and close to 1, and the MSRE is adequately low and approximates to 0. The values of MAE in model calibration and verification are far less than the relevant mean value of the observed data. The high scores of r^2 indicate that all the models present the "best" performance according to the standard given by Dawson et al. (2007). From the diagrams in Figures 4.5a-h and 4.7a-c, it is obvious that both the low values and the high values are close to the exact fit line and this result suggests that there is no evidence of overestimation or underestimation occurring during the model prediction. The outcome of the modelling implies that the training procedures are successful without "overtraining" or "local minimum" and the proposed models have powerful generalization abilities for out-of-sample prediction.

The coefficient of determination (r^2), the mean absolute error (MAE), the mean squared relative error (MSRE), for the calibration and the verification data set,

are given in Table 4.5. The notation (Q_R/ MLP-BP-ANN: 1-7-1/0.9957) from (Table 4.5) means that the best architecture of the specific MLP-BP-ANN model is composed of one input layer with one input node, one hidden layer with seven nodes and one output layer with one output variables, with value of coefficient of determination equals to 0.9688. According to the results of Table 4.6, it could be seen that the difference in the r^2, MAE and MSRE obtained using the test data was not markedly different from that obtained using the training data, this means that there was no overfitting. Also, the results of Table 4.5 show a good performance of the chosen MLP-BP-ANN model for predicting the Dadin-Kowa reservoir daily inflow. Table 4.6 depicted the percentage error in daily peak flow estimates for the MLP-BP-ANN model during the calibration and verification years. The low percentage error in daily peak flow estimates imply that the MLP-BP-ANN model is able to predict the peak flows with reasonable accuracy.

Table 4.5: Results for Model Error Measures for the Calibration and Verification Data sets

Q_R/ MLP-BP-ANN: 1-7-1/0.9957			
Data	**R**	**MAE×10^{-5}**	**MSRE×10^{-7}**
Calibration (1991-1998)	**0.9688**	**0.7156**	**1.4984**
Verification 1999-2001	**0.9688**	**1.1478**	**1.1478**

Table 4.6: Percentage Error in Daily Peak Inflow Estimation for the Model during the Calibration and Verification Years

Calibration years			
Date	**Peak flow (m^3/sec)**		**Error (%)**
	Historical	**MLP-BP-ANN**	
15th August 1991	**810.2**	**798.5**	**-1.42**
27th June 1992	**925.9**	**845.7**	**-8.66**
13th July 1993	**1382800000.0**	**1388800000.0**	**0.43**
14th August 1994	**926.0**	**926.0**	**0.00**
14th September 1995	**599.0**	**679.0**	**13.36**
13th August 1996	**926.0**	**910.0**	**-1.73**
8th August 1997	**810.0**	**810.0**	**0.00**
17th August 1998	**1505.0**	**1611.0**	**7.04**
Verification years			
5th May 1999	**1620.0**	**1900.3**	**17.30**
9th August 2000	**2315.0**	**2123.1**	**-8.29**
13th August 2001	**2315.0**	**2489.4**	**7.53**

An analysis to assess the potential of the chosen MLP-BP-ANN model to preserve the statistical properties of the historic inflow revealed that the inflow series predicted by the MLP-BP-ANN model reproduced the first three statistical moments (i.e., mean, standard deviation and skewness) for the calibration and verification years. The comparisons were also made by using the paired t-test with the two-sided tabular value (a=0.05) and the 45-degree line test. The computed t-values and the slopes of the chosen MLP-BP-ANN model, for the calibration and verification data sets, are given in Table 4.7. The computed t-values of the chosen MLP-BP-ANN model were less than two-sided tabular t-values, for calibration and verification data sets (Table 4.7). These results imply that there were also no significant differences between the observed and the predicted value.

Table 4.7: T-value, Two-sided Tabular Value (a=0.05) and Slope of the Model for the Calibration and Verification Data Sets

QR/MLP-BP-ANN: 1-7-1/0.9957			
Sample size	**t-value**	**Two-sided tabular value (a=0.05)**	**Slope (o)**
Calibration (1991-1998)	**0.7564**	**1.9665**	**45.98**
Verification (1999-2001)	**1.0258**	**1.9665**	**44.24**

The observed values and the predicted values yielded slopes close to 45 degrees, for the calibration and verification data sets (Table 4.7). It can be observed that the MLP-BP-ANN model tended to make an angle of 45 degrees with the axes, meaning that there was no significant difference between the observed and the predicted values. The good predictions on these data sets demonstrated the adequacy and the potential of the chosen MLP-BP-ANN model for predicting daily reservoir inflow. This clearly demonstrated the ability of the chosen MLP-BP-ANN model to predict very well daily inflow values, into Dadin-kowa reservoir. Consequently, the MLP-BP-ANN models seem promising for predicting daily reservoir inflow. As regards the accuracy, the model provided good accuracy for short time horizon forecast which however decreased when longer time horizons were considered and this was particularly true for the rising phase of the flood wave where a systematic underestimation was observed. This temporal limit is coherent with that detected by other authors using similar data-driven models applied to basins with similar extension to that considered in this study (e.g., Campolo et al., 1999, 2003; See and Open-shaw, 1999; Solomatine and Dulal, 2003), and this limit is certainly due to the fact that

no information or forecast of rainfall is considered available within the time spell ahead with respect to the instant when the forecast is performed.

4.2 Discussion on Reservoir Inflow Predicting Software

On the basis of the rainfall-inflow models developed, an integrated Web-Based Daily Reservoir Inflow Predicting Software (WBDRIPS) was implemented within java technology framework. It was conformed to the J2EE criterion and a number of Java techniques, such as Applet, Servlet, Swing, JDBC and JSP, were used during the implementation. By the aid of this system, web-based prediction of daily reservoir inflow was automatically accomplished in time every day after the (Quantitative Precipitation Forecast) QPF was released. The software interface was composed of three main function modules, namely internet connection, download of forecasted rainfall and inflow predicting modules, respectively.

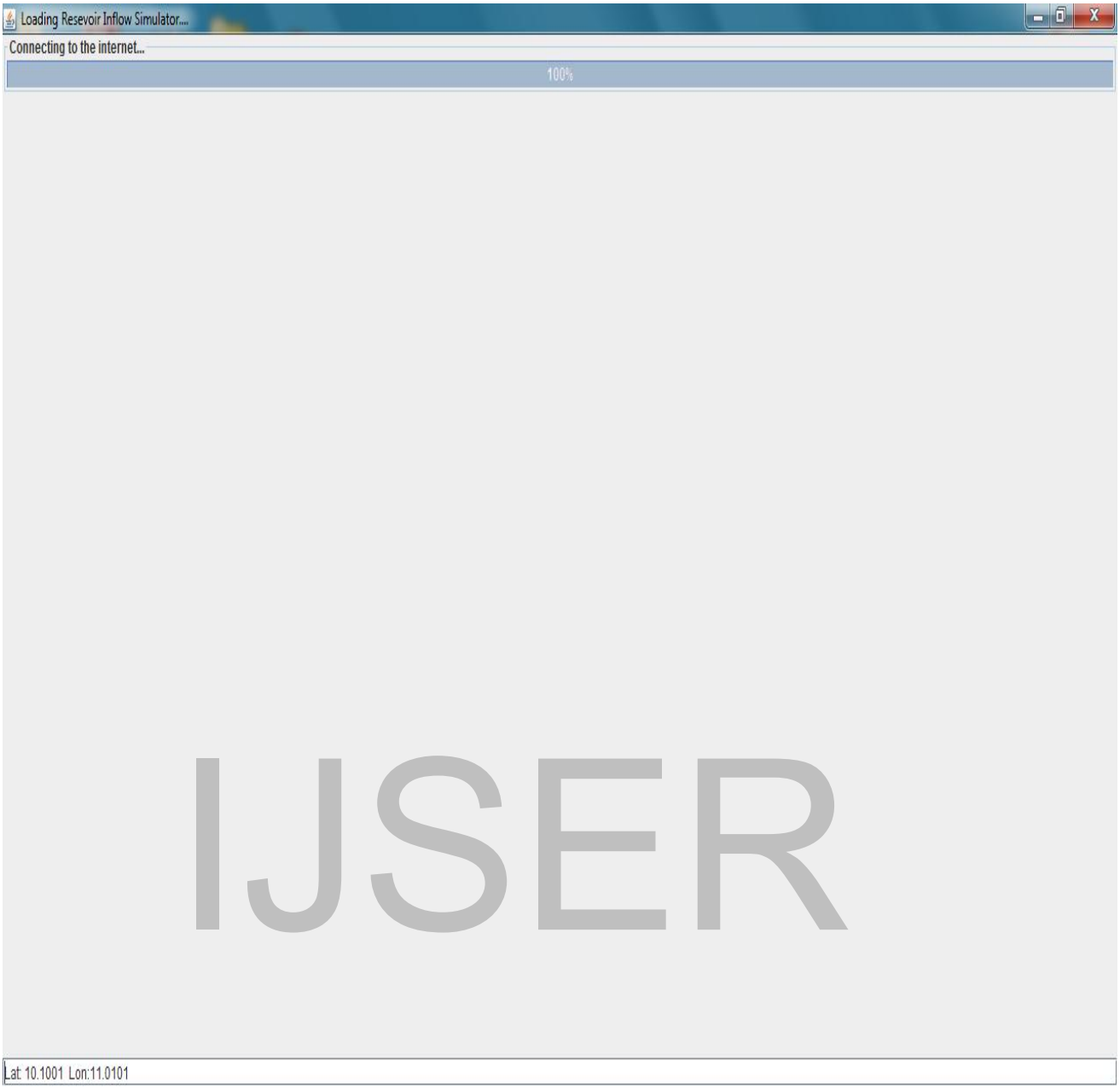


Fig. 4.8: The Internet Connection Module

The inflow predicting software is the final result of this thesis; the software can make predictions of the daily inflow into the Dadin-kowa reservoir as well as other reservoirs located in any semi-arid region anywhere in the world. The only thing that constantly has to be updated is the database containing the predicted

rainfall. The software has an interface which is easy to understand and use. The interface can be seen in Figure 4.8.

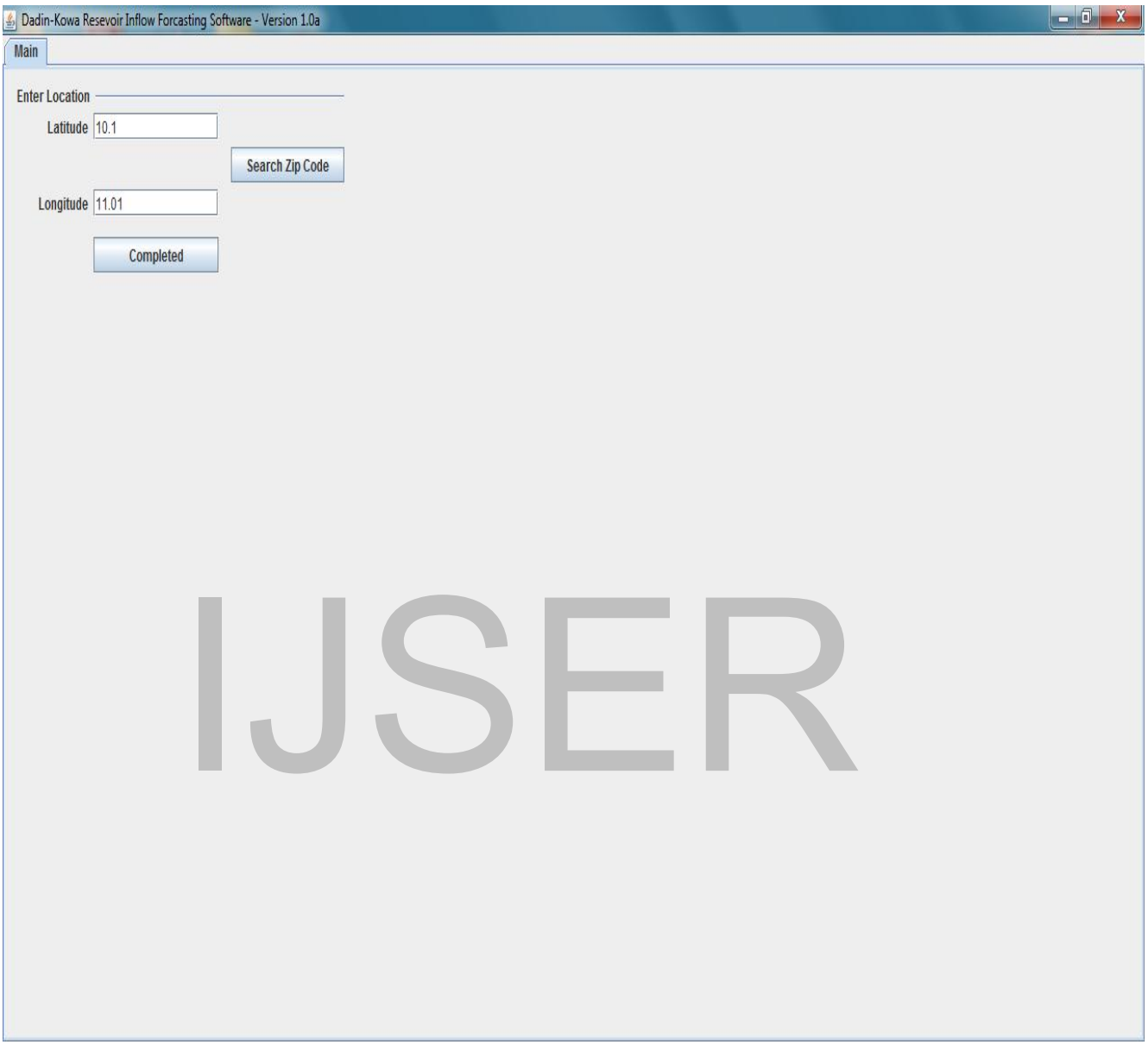


Fig. 4.9: The Predicted Rainfall (QPF) Download Module

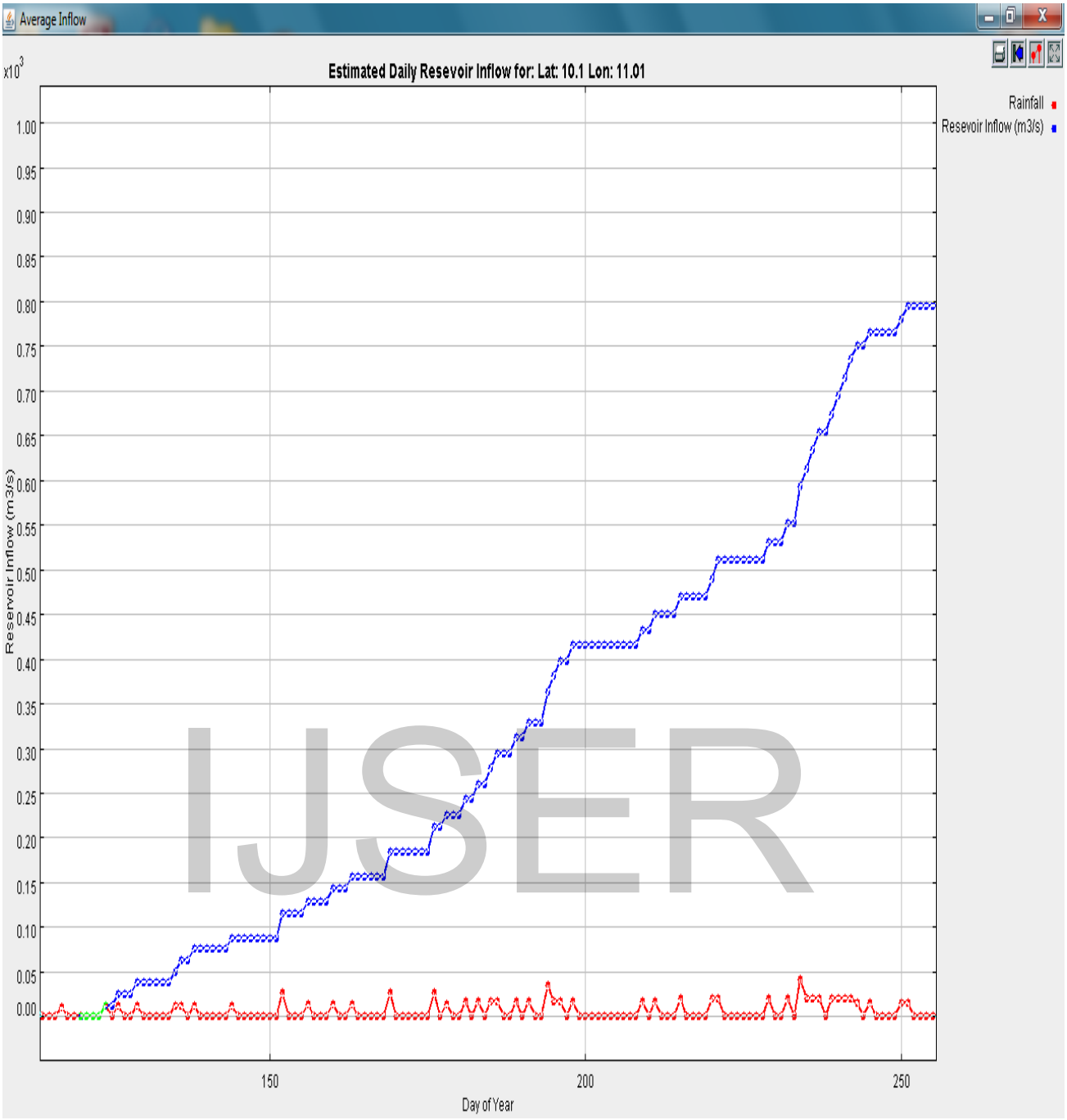


Fig. 4.10: Inflow Predicting Module

By selecting the week for which the simulation should account for, selecting the day for which it should be run, the inflow prediction was presented (Fig. 4.10). The inflow predicting software can be operated by filling in the geometrical coordinates of the location for which the inflow prediction is being made (Fig.

4.10). Then, the inflow prediction would be shown as soon as the "completed button" is clicked (Fig. 4.10). A detailed Java Program codes for the software development can be found in Appendix D.

Some shortcomings needed to be addressed in the developed software for practical use.

(a) Rainfall-inflow Modelling: The Model showed performances which remain satisfactory up to 6-day ahead for the basin considered, which indeed is too short for operational flood prediction. This latter observation highlights the need to feed these models with further information, with respect to the time of prediction, relevant to the future rainfall amount and its spatial and time distribution. This further information should arrive mainly from radar data, possibly elaborated by another model dedicated to the rainfall forecast, which would allow the time spell of prediction to be extended to useful operational time intervals.

The design of the chosen system for short-term reservoir inflow forecast led to appreciable results. However results may be considerably improved if available atmospheric parameters such as temperature, humidity, rainfall and brightness were included as exogenous inputs to the MLP model.

Finally, it can be observed that the models here considered, for their very nature, are "blind" with respect to specific physical information: they only deal with data (e.g., rainfall and inflow) which indirectly contain the "integral" behavior of the system to be modelled. As a consequence, since the study

presented was performed with reference to a basin of a semi-arid region, it is expected that similar results can be obtained when other basins located in similar regions are considered. On the contrary, if basins located in humid regions were considered, different inputs and structures may be selected and identified, so the relative performances of the models may change. Further analyses are currently being developed with reference to catchments in humid regions to analyze these aspects.

(b) Computer Resources: The computational power required of the analysis tools required a large amount of memory in the user's machine to work effectively. To increase the number of calculations, one must increase the computational power of the computer. Also, a real-time rendering system such as the one proposed required sufficient memory in order to work properly. The server-side can be equipped with a powerful CPU and a large amount of memory for minimal cost per user. Internet Explorer with the applet loaded consumed 20MB. When displaying data, the memory usage varied between 25 and 35MB. Because of Java's garbage collection feature, the Java VM slowly increased memory consumption until it reaches 35MB at which point it automatically frees 10MB and started again at 25MB. The database engine employed in this prototype was a free lightweight relational database management system. MSQL is designed for the fast retrieval of small amounts of data. It does not offer full database functionality that other database vendors currently support. Therefore, the database engine in the prototype should be

replaced with database designed for fast retrieval of large amounts of data, especially if this data were to be retrieved from a Web site.

(c) Bandwidth: Modem speed is a crucial limiting factor for a home-PC user accessing predicting information over the Internet. To remove this problem, the user at the query site required a high-bandwidth link. New internet technologies will continue to change how information is created, distributed, and maintained. These include advances in Web protocols and languages and increase in access to the internet. Integrated Services Digital Network (ISDN) telephone lines, cable-TV modems, Digital Subscribers Lines (DSL), and other high-speed internet connection services will become available and affordable to most home-PC users.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

The future precipitation is the dominant source of uncertainty in hydrological prediction (Krzyszofowicz, 1995). So any research on integrating QPF into hydrological models is significant and promising for improving the quality of hydrological prediction. Artificial neural network brings more as it further approaches the human approximation. This study combined the QPF into ANNs models to perform daily reservoir inflow prediction. The aim of this work is to show the relevance of designing web-based reservoir inflow predicting software. The developed tool predicts the Dadin-kowa reservoir inflow by using the Multi Layer Perceptron ANN architecture which is a powerful tool for modeling time series prediction. The system gives the user the choice for training and verification of the appropriate MLP structure for Dadin-Kowa reservoir inflow via a web interface.

5.1 Conclusion

In this work, Multilayer Perceptron Back Propagation Artificial Neural Network (MLP-BP-ANN) models were developed for predicting daily inflow values into Dadin-Kowa reservoir. The Artificial Neural Network approach becomes more explicit and can be adopted for any reservoir daily inflow prediction. The experimental results indicate that these models can extend the predicting lead-times with a satisfactory goodness of fit.

As regards the accuracy, the model provided good accuracy for short time horizon forecast which however decreased when longer time horizons were considered and this was particularly true for the rising phase of the flood wave where a systematic underestimation was observed. This temporal limit is coherent with that detected by other authors using similar data-driven models applied to basins with similar extension to that considered in this study (e.g., Campolo et al., 1999, 2003; See and Open-shaw, 1999; Solomatine and Dulal, 2003), and this limit is certainly due to the fact that no information or forecast of rainfall is considered available within the time spell ahead with respect to the instant when the forecast is performed. As per the application of this study, integrated web-based daily reservoir inflow predicting software, implemented within java technology framework, was developed and was successfully used for daily inflow predictions of Dadin-kowa reservoir which is operated by the Upper Benue River Basin Development Authority in Nigeria.

5.2 Recommendations

Mobile Web-based predicting allows a user to access large amount of data on a small hand-held device. With the advent of low-cost, easy-to-use, and field-ready phone devices, mobile web-based prediction will become more cost effective and more widely deployed. Mobile web-based prediction allows the following (Spencer, 2000): provision of a more durable/robust computer-based medium; more easily pre-packaged/organized daily jobs into work-order packets for field execution and additional data set availability to assist operators

in making decisions in the field. With the advent of Wireless Application Protocol (WAP) and Universal Mobile Telecommunications System (UMTS) technologies, it is possible that web-based modelling can be built into every mobile telephone. Significant progress has been made since the late 1980s towards the development of mobile communication concepts, systems, and networks. UMTS is a third-generation mobile communication system currently being developed in Europe. It defines a European standard enabling global usage of personal mobile communications regardless of the surrounding environment. These observations highlight the need for the development of mobile Web-based predicting using small hand-held device.

5.3 Contributions to Knowledge

This research has demonstrated a method of developing a modelling tool with a high level of integration with environmental models in a WBS. This coupling provides functions that were not available in traditional simulation models.

An important objective was to find ways to access a large amount of information and application tools. The approach taken has a number of significant advantages over traditional methods. It overcame the difficulties of data access and participation by developing interactive pages, users can access and analyze various hydrological and environmental data from computers at public libraries, schools or their homes. Another benefit of the approach was that the interactive modelling and analysis of environments were highly visual.

The developed software is a WBS effectively linked with real-time hydrological model. This research demonstrated that Java programming for linking hydrological model was possible using the Web, communication tools, and cost-effective communication equipment. The other contributions to knowledge are summarized briefly as follows:

i. Development of lumped rainfall-inflow model capable of automatic simulation and calibration from online database of quantitative precipitation forecasts.

ii. Combination of Back Propagation and Levenberg-Marguardt algorthms for rainfall-inflow data training in Multi Layered Perceptron ANN using sigmoid transfer function.

iii. Integration of inflow predicting model into user-friendly software that can couple effectively with the web.

REFERENCES

Abboth, M. B., Bathurst, J.C, Cunge, J.A and Rasmussen, J. (1986). An Introduction to the European Hydrological System: History and Philosophy of Physically-based Distributed Modelling System. J. Hydrol. 87, 45-59.

Abnous, R. and Khoshafian, S. (1990). Object-orientation: Concepts, Languages, Databases and User Interfaces. New York: John Wiley and sons Inc, pp. 50-58

Abrahart, R. J., See, L. and Kneale, P.E. (1998). New Tools for Neurohydrologists: using Network Prunning and Model Breeding Algorithm to Discover Optimum Inputs and Architectures. [On–line]. Available: http://www.geog.port. ac.uk/geocamp/geo p8/20/ge, 20html.

Abrahart, R.J. and See, L. (1998). Multi-model Data Fusion for River-flow Forecasting: An Evaluation of Six Alternative Methods Based on Two Contrasting Constraints. Hydrol. Earth Syst Sci. 6, 655-670.

Adeni, H and Hung S.L. (1995). Machine Learning Neural Networks Genetic Algorithms and Fussy Systems New York; Wiley Press, pp. 90-115.

Alfredsen, K. (1999). An Object-Oriented Framework for Application Development and Integration in Hydroinformatics. Phd. Thesis. Dept. of Hydraulic and Environmental Engineering. Trondhein Norwegian University of Science and Technology.

Alfredsen, K. (2000). Simulation of Human Impacts on Flood Regimes. The Design of an Object-oriented Integration Framework. Proceedings of Hydrodynamics 2000 (on CD-ROM). The University of Iowa, USA.

Alfredsen, K. and Saether, B. (2000). An Object-oriented Application Framework for Building Water Resources Information and Planning Tools Applied to the Design of Flood Analysis System. Environmental Modelling and Software. 15, 215-224.

Alvisis, R., Prince, L., and Soares, S. (2006). Long-term Hydropower Scheduling Based on Deterministic Nonlinear Optimization and Annual Inflow Forecasting Models. Procs. Of the Powertech Conference, Madrid. pp 1-8.

Andreson, M. L., Karvas, M. L and Mierzwa, M.D. (2001). Probablistic Ensemble Forecasting with EI Nino Southern Oscillation (ENSO). Journal of Hydrology 249, 134-146.

Bardossy, A. and Duckstein, L. (1995). Knowledge – based Classification of Circulation Patterns for Stochastic Precipitation Modelling. Proc. Stochastic and Statistic Methods in Hydrol Environ. Eng., Iowa Water 100.

Baxes, G. (1994). Digital Image Processing Principles and Application. News York: John Wiley and Sons, Inc., pp. 70-85.

Behrendt, H. and Opitz, D. (2000). Retention of Nutrients in River Systems: Dependence on Specific Runoff and Hydraulic Load. Hydrological, 410: 111-122.

Bertes, N.and Grafer, I. (1969). Investigation of Short-term Rainfall Prediction Method by a Translation Model. Proc. 28th Japanese conf. on Hydraulic: JSCE pp 423-428.

Bertino, E. and Martino, L. (1993). Object-oriented Database Systems. Concepts and Archtectures. MA: Addision-Wesley publishing company Inc., pp. 30-45.

Beven, K.J. and Binley, A.M. (1992). The Future of Distributed Models. Model Calibration and Uncertainty Prediction. Hydrol. Processes. 6, 279-298.

Beven, K.J. and Kirby, M.J. (1979). A Physically-based, Variable Contributing Model of Basin Hydrology. Hydrological Sciences Bulletin. 24, 43-69.

Binge, M. (2000). Surveying GIS. A Practical Geographical Information Sciences Guide for Survey Professionals [On-line]. Available- http://www.pobonline-com/CDA/Article Information/ features/ BNP_ features_item/0.2338, 17058,00html (Accessed Dec. 2010).

Blundon, W. (1996). Java and Active X: Java wins on the Internet but the Intranet rules. Java world. Available online at : http://www.javaworld.com/jw-09-1996/jw-09-blunder. html (Accessed Mar 2011)

Bogglid, C.E., Kundby, C. J., Kundsen, M.B and Starzer, W, (1999). Snowmelt and Runoff Modeling on Artic Hydrological Basin in West Greenland. Hydrological Processes. 13, 1989-2002.

Booch, G. (1994). Object-Oriented Analysis and Design with Applications. CA: Benjamin/ Cummings Publishing Company, pp. 17.

Borah, D.K. and Bera, M. (2004). Watershed-scaled Hydrologic and Nonpoint-source Pollution Models. Review of Applications. Transactions of the ASAE 47(3), 1539-1559.

Borah, D.K., Bera, M. and Xia, R. (2004). Storm Event flow and Sediment Simulations in Agricultural Watersheds using DWSM: Transactions of the ASAE. 47 (5) 1539-1559.

Bowden, G.T., Maier, H.R. and Danzy, G.C. (2002). Optimal Division of Data for Neural Networks Models in Water Resources Applications Water Resour, Res 38 (2), 2.1-2.11.

Bray, M. (2002). Identification of Support Vector Machines for Runoff Modelling. M. Eng. Thesis Department of Civil Engineering. University of Bristol, k., Breusch, T. and Pagan, (1979). A Simple Test of Heteroscedastic and Random Coefficient Variation. Econometrical. Vol. 47, 1287-1294.

Brodie, M. and Stonebraker, M. (1995). Migrating Legacy Systems. Morgan Kanufmann Series in Data Management Systems. Jim.Gray series (Ed). Morgan Kanufirann Publishers, CA. pp. 100-115.

Budd, T. (1991). An Introduction to Object-oriented Programming. Reading, M.A: Addison- Wasley Publishing Company. Pp. 78-84.

Bundock, M.S. and Raper J.F. (1991). GIS Customization: From Tools to Efficient Working Systems. Proc. Mapping Awareness 91. Conferenc 6-8 February, 1991. London, Uk. pp 101-114.

Burke, L.I. (1991). Clustering Characterization of Adaptive Resonance. Neural Network. 4(4), 485-491.

Burrough, P. A. (1997). Environmental Modelling with Geographical Information Systems. In kamp, 2. (Ed) Innovations in GIS, 4 Fourth National Conference on GIS Research, UK. GISRUK). London: Taylor and Francis pp 150-157.

Campolo, M., Andreussi, P. and Soldali, A. (1999). River Flood Forecasting with a Neural Network Model. Water Resour, Res 35(4), 1191-1197.

Chance, A. G., Richard, A. G. and Theriult, D. (1999). Small World GIS. An Overview of Small World Magic. Environmental Monitroing, pp. 52-68.

Chang, D. and Harkey, D. (1998). Client-server Data Access with Java and XML. New York: John Wiley and Sons. Pp. 40-43.

Chappell, D. (1996). Component Software Meets the Web: Java Applets Vs Active X Controls. Chappel Associates [On-line]. Available: http://www. Chappell asso.com/Java ActiveX. html. (Accessed Jan 2010).

Chatfield, C. (1996). The Analysis of Time Series: An Introduction. New York: Chapman and Hall. Pp. 71-84.

Chang, N.B. and Chen, H.W. 2000. Prediction Analysis of Solid Waste Generation Based on Grey Fuzzy Dynamic Modelling Resources and Recycling. Journal of Mathematical Computing, Vol. 29, pp. 1-18.

Chew, C.Y., Moore, L.W. and Smith, R.H. (1991). Hydrological Simulation of Tennessce's North Reel Foot Greek Watershed Research. Journal of the Water Pollution Control Federation, 63(1), 10-16.

Chi, Y. (1999). An Integration Architecture for Large Scale Applications Involving Workflow, Data Exchange and knowledge Basis. PhD. Thesis, Arizona State University.

Choan, U. and Tucci, T. (2001). Multivariate Analysis; Methods and Application. New Work: John Wiley and Sons. Pp. 47-51.

Chopra, P.N. (1996). Geological Mapping on the Web- the Australian Geological Survey Organisation Introduces On-line Customizable Maps. GIS USER, 19, pp 35-39

Coffee, P. (1996). Java, Active X Under a Microscope. ZNET. [On-line]. Available:http://icq.2dnet.com/devhead/Stories/articles/0,4413, 1600418100.html (Accessed Jan., 2011).

Collier, C. G. (1994). Objective Methods of Forecasting Precipitation Using Weather Radar Data. In Almeda-Teixeira, M.E., Fantechi, R., Moore, R., Silva, V.M. (Ed). Advances in Radar Technology. European Commission Report EUR 14334 EN

Collis, L. (2001). Training Feed Forward Network with the Marquardt Algorithm. IEEE Transactions Neural Networks, 5, 989-993.

Correia, F. N., Saraiva, M.G. and Ramos, I. (1997). GIS-based Flood Analysis and Flood Plain Management. Water Resources Management. Dordrecht: Kluwer Academic Publishers, Pp 229-249

Coulibaly, P., Anctil, F. and Bobee, B. (1999). Neural Network-based Water Inflow Forecasting. Control Engineering Parctice. 6(5), pp. 593-600.

Cybenko, G. (1989). Approximation by Superposition of a Sigmoid Function. Math. Control Signals.. 2, 303-314.

Daconta, M.C. (1996). Java for C/C ++ Programmers. John Wiley and Sons. New York NY. Pp. 31-38.

Dandy, G. and Mainer, H. (1996). Use of Artificial Neural Network for Real Time Forecasting of Water Quality. Proc. Int. Conf. Water Resource. Environ Res. Vol II October 29-31. Kyoto, Japan.

Davis, C and Medycki-Scott, D. (1994). GIS Usability Based on User View. International Journal of Geographical Information System (IJGIS). London: Taylor and Franeis. 8(3). 175-189.

Davis, C, A. (1994). Object-Oriented GIS in Practice. Urban and Regional Information Association pp 786-795.

Davis, M.K. and Maidment, D. (2000). Object Oriented Modelling of Rivers and Watershed in Geographical Information System. CRWR. Online Report No 2007.

Dawdy, D. R. and Bergman, J.M. (1969). Effect of Rainfall Variability on Streamflow Simulation. Water Resources. 5 (8), 958-966.

Dawson, C.W., Abrahart R.J, and See, L.M. (2007). Hydrotest : A web-based Toolbox of Evaluation Metrics for the Standardized Assessment of Hydrological Forecasts, Environ Modell, Soft Ware 22, 1034-1052.

Dia, U., Evans, L. and Shank, M. (1997). Internet Interactive GIS and Public Empowerment. Proceedings of Integrating Spatial Information Technologies for Tommoro. GIS'97 World Inc, February 17-20 Vancover Canada. pp 555-559.

Dibike, Y. B., Solomatine, D. and Abbott, M. (2001). On the Encapsulation of Numerical Hydraulic Models in Artificial Neural Network. Journal of Hydraulic Research Vol. 37 No2, pp 147-161.

Dolling, K. and Vares, H. (2003). Neural Networks for River Flow Prediction. J. Comp. Civil Engng ASCE 8(2), 201-220.

Domino,C.(2003).[Online].Available:http://www./ibin.com/servers/esener/isenes/dominos/products/html.

Druce, D.J. (2001). Insights From a History of Seasonal Inflow Forecasting with a Conceptual Hydrologic Models. Journal of Hydrology. 249, 102-112.

Eckel, B. (1998). Thinking in Java. Prentice Hall, Upper Saddle River, N.J. E-book [On-line].Available: http; //www.mindview. net/books/TIJ/ (Accessed, Jul, 2010).

Elmasri, R. and Narathan, S. (1994). Fundamentals of Database Systems. The Benjamin/Cummings Publishing Inc: CA. pp. 21-26.

Faures, J. M, Goodrich, D.C., Woolhiser, D. A. and Sorooshian, S. (1998). Impact of Small-scale Rainfall Variability on Runoff Simulation. Journal of Hydrology. 173, 309-326.

Federal Ministry of Internal Affairs (FMIA). (2012). Map of Nigeria. Queen Press. Abuja.

Finger, F., Reed, D. and Stikeleether, J. (1996). Next Generation Computing Distributed Objects for Business SIGS. New York, NY. Pp. 51-52.

Finnoff, W., Hergert, F. and Zimmermann, H.C. (1993). Improving Model Selection by Nonconvergent Methods. Neural Networks 6, 771-783.

Frede, H.G., Bach, M., Fohrer, N. and Breuer, L. (2002). Inter-disciplinary Modelling; the Significance of Soil Functions. Journal of Plant Nutrition and Soil Science. 165, 460-497.

Free BSD (2003) [On-Line]. Available: http://www.free bsd.org.

Freer, J., Beuan, K. J. and Ambrosia, B. (1996). Bayesian Estimation of Uncertainty in Runoff Prediction and the value of Data: An Application of GLUE Approach. Water Resour. Res 32 (7). 2161-2173.

French, M.N., Krajaswski, W.F. and Cuykendall, R.R. (1992). Rainfall Forecasting in Space and Time, using a Neural Network. Journal of Hydrology 137, 1-31.

Gamma, F., Helm, R., Johnson, R. and Vissides, J. (1995). Design Patterns Elements of Reusable Object-oriented Software. Reading, MA: Addison-Wesley Inc. pp. 61-64.

Gongbing, A.A., Wichard, J.D and Ogorzalak, M. (2003). Time series Prediction with Enremble models in IEEE International conference on Neural Network Conference Proceedings. Vol 2, pp 1625-1630. IEEE Press: New York.

Goodchild, M. (1992). Integrating GLS and spatial data Analysis: Problems and possibilities International Journal of Geographical Information Systems. London: Taylor and Francis (6) pp 407-432.

Gosling, J. and McGilton, H. (1996). Java Language Environment: A white paper, Sun Microsystems. Inc. [On-Line] Available: http://java .sun.com/does/white/langenv. (Accessed. March, 2010).

Gosling, J., Joy, B. and Steele, G. (1996). The Java Language Specification. Sun Microsystems, Inc. CA: Addission-Wesley. Pp. 60-61.

Gotz, R., Steiner, B., Sievers, S., Friesel, P., Roch, K., Schworer, R. and Haag, F. (1998). Dioxin, dioxin-like PCBSS and Organotin Compounds in the River Elbe and the Hamburg Harbour. Identification of Sources. Water Sci. Technol., 37(6-7) 207-215.

Grand, M. and Kundsen, J. (1997). Java Fundamental classes References Sebastopol; O'Reilly and Associates Inc., Sebastopol, CA, pp 34-56.

Hagan, K.T. and Menhaj, M.B. (1994).Training Feedforward Networks with the Marguardt Algorithm. IEEE Trans-Neural Networks 5(6), 989-993

Hagget, C. (1998). An Integrated Approach to Flood Forecasting and Warning in England and Wales. Journal of Water and Environmental Management 12 (16), 425-432.

Haklay, M. (2001). Public Environmental Information Systems. Challenges and Perspective. PhD Thesis, Department of Geography, University of London, London UK, pp 89-100.

Hall, M. J. and Minns, A. W. (1993). Rainfall-runoff modelling. As a Problem in Artificial Intelligence: Experience with Neural Network. Proc. 4th National Hydrology Simposium. British Hydrological Society, Gidiff. pp 5.51-5.57.

Hamilton, G., Fisher, M. and Cattel, R. (1998). JDBC Database Access with Java:A Tutorial and Annoted Reference. Reading, MA: Addison Wesley. Pp. 154-160.

Han, D., Clackie, I.D., Karbassoun, D., Lawry J. and Krauskopf, B. (2002). River flow Modelling using Fuzzy Decision Trees. Water Resources Management. 16, 431-445

Han, D., Kwong, T. and Li, S. (2006). Uncertainties with Real time Flood Forecasting with Neural Networks. Hydrol. Process. 10.1002/hyp. 6184.

Harnedy, S. (1996). Web-based Management for the Enterprise. New Jersey: Prentice Hall. Pp. 210-213.

Hardy, P.G. (1999). Active Object Techniques for the Production of Multiple Map and Geodata Products from a Spatial Database. Cambridge: Laser Scan Ltd, pp 78-87.

Harlin, J. (1991). Development of a Process-oriented Calibration Scheme for the HBV Hydrological Model. Nordic Hydrology 22(1), 15-36

Hatterman, M., Adeli, H. and Hung, S.I. (2005). Machine Learning Neural Networks. Genetic Algorithms and Fuzzy System. New York: Wiley. Pp. 110.

Herbst, M., Hardelauf, H., Harms, R., Vanderbought, J. and Vereecken, H. (2005). Pesticide Fate at Regional Scale. Development of an Integrated Model Approach and Application. Physics and Chemistry of the Earth, 30 (8-10), 542-549.

Hernandez, M., Miller, S. N., Goodrich, D., Goff, B. F., Kepner, W. G., Edmonds, C. M. and Jones, K. B. (2000). Modelling Runoff Response to Land

Cover and Rainfall Spatial Variability in Semi-Arid Watershed. Environmental Monitoring and Assessment 64, 355-398.

Herring, J. (1991). TIGRIS: A Data Model for an Object-Oriented Geographic Information System. Computer and Geosciences, 18(4). pp 443-452.

Heywood, M. I. and Zineir-Hey wood A.N. (2002). Dynamic Page-based Linear Genetic Programming. IEEE Transaction on Systems. Man and Cybernetics-part B, Cybernetics 32 (3), 380-388.

Hipel, K.W., Meleod, A. I., Panu, U. S. and Singh, V.P. (1994). Stochastic and Statistical Methods in Hydrology and Environmental Engineering. Vol. 3. UK; Southampton: WIT Press, 77-85.

Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer Feedforward Networks are Universal Approximators. Neural Networks 2(5), 359-366.

Horstman, D., Young. P. and Taman, N. (1984). Recursive Estimation and Time-series Analysis. Springer-verlag; New York. Pp. 38.

Hunter, J. (1998). Java Servlet Programming. Sebastopol, CA: O'Relly and Associates Inc. pp. 134-140.

Hunter, J. And Crawford, W. (1998). Java Servlet Programming. O' Relly and Associates inc. Sehastopol, CA. pp. 15.

IAB (2003). Internet Architecture Board [On-line]. Available; http: //www.iab.org/lab.

Ibeje, A.O., Agunwamba, J. C. and Okoro, B.C. (2012). Allocation of Water Resources in Semi-arid Region of Northern Nigeria (Poster Presentation). In: Integrated Water Resources Management; Hartwig Steusloff, ed. Conference Proceedings at International Conference Karlsruche, IWRM Karlsruche 2012, 12-22 November, 2012 at Karlsruche Convention Centre, Germany. ISBN 978-3-8396-0478-6.

Ireson, A., Makropoulos, C. and Maksimovic, C. (2006). Water resources Modelling Under Data Scarcity:Coupling MIKE BASIN and ASM Groundwater Model. Water Resources Management. 20, 567-590.

Irvine, K. and Eberhardt, A. (1992). Multi-capative Seasonal ARIMA Models for Lake Erie and Lake Ontario Water Levels. J. Am. Water Resour. As, 28, 385-346.

Jain, S. K. and Sringvasulu, D. K. (2004). Application of ANN for Reservoir Inflow Prediction and Operation. Journal of Water Resources Planning Management. 125(5) 263-271.

James, W. P. (1991). Radar-assisted Real-time Flood Forecasting. Journal of Water Resources Planning and Management, ASCE 119(1), 32-44.

Jankowski, P., Nyerges T., Smith, A.,Moore, T.J. and Horvath, E. (1997). Spatial Group Choice: A SDSS Tool for Collaborative Spatial Decision-making International Journal of Geographical Information Science. In Fisher, P., Clerk, K.C and Lees, B. (Ed). London: Taylor and Francis 11(6), pp 577-602.

Jaquin, D. I. and Shamseldin, V.O . (2006). Rainfall-runoff Models Using Artificial Neural Networks for Ensemble Streamflow Prediction. Hydrological Processes. J. Hydrol., 199; pp. 278-281.

Julien, P. V., Sahafian, B. and Ogben, F. L. (1995). Raster-based Hydrologically Modelling of Spatially Varied Surface Runoff. Water Resources Bulletin 31(3). 523-536.

Karimi, H. and Blais, M. (1997). Overcoming the Time Complexity Barrier of Routing for Real Time GIS- Application. A parallel Routing Algromthm. Journal of Geographical Information and Decision Analysis. London. Taylor and Francis 1 (2), 144-150.

Keeley, M., Chang, M. and Boyer, G. (2000). Estimates of Low Flows Using Watershed and Climate Parameters. Water Resources Research 13, 997-1001.

Keeman, V. (2001). Learning and Soft Computing. Support Vector Machines Neural Networks and Fuzzy Logic Models. Cambridge, MA: MIT Press. Pp. 48-60.

Kelly, K.S and Krzysofowicz. R. (2000). Precipitation uncertainty processor for probabilistic River stage forecasting. Water Resources Res 36 (9), 2643-2653.

Kienzle, S.W. and Schulze, R.E. (1992). A Simulation Model to Assess the Effect of Afforestation on Groundwater Resources in Deep Sandy Soils. Water SA, 18(4), 265-72.

Kjerne, D. and Ducker, H. (1986). Modelling Cadestral Spatial Relationships Using Object-oriented Language. In Marble, B. (Ed). Second International Symposium on Spatial Data Handling. Seattle, WA.

Klocking, B. and Haberlandt, U. (2002). Impact of Land Use Changes on Water Dynamics-A Case Study in Temperate Meso and Macroscale River Basin. Physics and Chemistry of the Earth-27, 619-629

Kohonen, T.(1990). The self-organising Map Proc. IEEE 79(9) .1464-1480

Koza, J. R. (2004). [On-line]. Available: http://genetic–programming.org. last update: 16 September 2004.

Kramar, D. (1996). The Java platform: A White Paper. Sun Microsystem. Inc. [on-line]. Available: fip//.ftp.java soft.com/docs/papers/java platform. Pdf (Accessed Jun., 2009).

Krause, P. (2002). Quantifying the Impact of Land Use Changes on the Water Balance of Large Catchments Using the J2000 Models. Physical and Chemistry of the Earth, 27 (9-10), 663-673.

Krutsch, K.F., Cargo, D.S. and Howlett, V. (2001). Professional Java Custom UI Components. Wrox Press Inc. Chicago, IL. Pp. 30.

Krysanova, V., Muller-Wehfeil, D. I. and Becker, A. (1998). Development and Testing of a Spatially Distributed Hydrological Water Quality Model for Mesoscale Watershed. Ecological modelling. 106 (2-3), 261-289.

Krzyszofowicz, R. (1995). Bayensian Theory of Probabilistic Forecasting via Vereministic Hydrologic Model. Water Resour Res 35(a) 2739-2750.

Krzyszofowicz, R. (2001). The Case of Probabilistic Forecasting in Hydrology: Journal of Hydrology. 249, 2-9.

Krzyszofowicz, R. (2007). Bayesian System for Probabistic River Stage Forecasting. J. Hydrol 268, 16-40.

Kuczera, G. and Parent, E. (1998). Monte Carlo. Assessment of Parameter Uncertainty in Conceptual Catchment Models. The Metropolis Algorithm. Journal of Hydrology 211, 69-85.

Kundsen, J., Thomsen, A. and Refsgard, J.C. (1986), WATBAL. A Semi-distributed Physically-based Hydrological Modelling System. Nord Hydrol. 17, 415.

Lardet, P. and Obled, L. (1994). Real-time Flood Forecasting Using a Stochastic Rainfall Generator. Journal of Hydrology. 162, 391-408.

Larson, P.M. (1980). Industrial Application of Fuzzy Logic Control. Int. J Man. M 12 (1), 3-10.

Lee, J. G. and Heaney, J. P. (2003). Estimation of Urban Imperviousness and its Impact on Storm Water Systems. J. Water Resour Plann. Manage. 129, 419-426.

Lee, R. (2001). Java versus Active X: The Marketing War is on, but which offers the better Technology. Sun World [On-line]. Available: http.//sunsite uakon.sk.wor/donline/swol-09-1996/swot-09-active x html. (Accessed Jul-2010).

Legatees, D.R. and McCabe, G.J. (1999).Evaluating the Use of "Goodness of Fit" Measures in Hydrologic and Hydroclimatic Model Validation. Water Resour.Res., 35, 233-241.

Lemany, L. and Perkins, C. (1997). Teach Yourself Java 1.1 in 21 days. Indianapolis: Sams.net. Inc. pp. 30.

Lin, J. Y., Cheng, C. T. and Chan, K.W. (2006). Using Support Vector Machines for Long-Term Prediction. Hydrology Sc. J. 51, 599-612.

Liong, A. T. and Sivapragasn, M. (2002). Runoff Simulation Using Artificial Neural Networks. Proceedings of Neural Networks. Proceedings of Twenty-fifth Congress of International Association for Hydraulic Research Spread Lectures, Technical Session A, flood and Drought I, 517-522.

Liong, S.V., Lim, W.H and Paudyal, G. N. (2000). River Stage Forecasting in Bangladesh; Neural Network Approach. J. Comput Civil Eng. 14, 1-8.

LSWS. (1999). Internet page of LUND SPACE WEATHER CENTER (LSWS). [On-Line] available. http://nastol.astro . in.se/._henrik/neuralnet3-html.

Lyon, D. (1995). Using Stochastic Petri Nets for Real-time Nth-order Stochastic Composition. Computer Music Journal, Winter 19(4), 13-22.

Makkeasorn, P. C., Suchheer, K. P., Raumen, D. M. and Ramesastri, K. S. (2005). A Neurofuzzy Computing Techniques for Modelling Hydrological Time Series. J. Hydrol. 29 (2), 52-66.

Mambani, k. (1974). A Fast Learning Algorithm for Parsimonius Fuzzy Logic Systems. Fuzzy Sets and Systems 126, 337-351.

Masters, T. (1995). Advanced Algorithms for Neural Networks: A C++, MIT Press Cambridge MA, pp. 318-362.

Mathworks(2005).www.mathworks.com/access/helpdesk/help/toolbox/nnet/adapt107.html. (Accessed Jan, 2012).

Miller, R. C., Guertin, D. P. and Hailman, P. (2004). Information Technology in Watershed Management Decision Making. J. Am. Water Resour. Assoc. 40(2). 347-357.

Milne, P., Milton, S. and Smith, J. L. (1993). Geographical Object-oriented Database – A Case Study. International Journal of Geographical Information System. London: Taylor and Francis. 7(1), 39-55.

Minns, M. (1998). Metro Determinstc Use of Artificial Neural Networks for Prediction of Water Quality Parameters. Water Resour Res 32 (4), 1013-1022.

MMS Manual (1998). The Modular Modelling System (MMS): User Manual [On-line].Available:http://wwwbrr.cr.usgs.gov/projects/SWprecip_ runoff/mms/(accessed-July, 2010).

Montanari, A. and Brath, C. (2004). Large Sample Behaviour of Generalized Likelihood Uncertainty Estimation (GLUE) in Assessing the Uncertainty of Rainfall-runoff Simulation. Water Resour. Res. 41.

Moore, R.J. (2002), Aspects of uncertainty Reliability and Risk in flood Forecasting Systems in Corporating Weather Radar. In Borgadi, J. and Kundzewic, Z.W. (Ed.). Risk, Reliability, Uncertainty and Robustress of Water Resources Systems. Cambridge University Press. Pp. 84.

More, J.J. (1977). The Levenberg-Marguardt Algorithm: Implementation and Theory; Numerical Analysis, G.A. Watson ed., Lecture Notes in Mathematics 630; Springer, New York, 105-106.

Morgan, S. (2000). Deterministic Distributed Mathematical Model. New York Publishers, pp. 57.

Morten, S. and Vefsnmo, S. (1997). An Object Oriented framework for Creating Models in Hydrology. ACM SIGPLAN Notices.

Motovilov, P., Rosso, R., Cadavid. L.G. and Salas, J.O. (1999). Forecasting of Short-term Rainfall Using ARMA Models. J. Hydrol. 144, 193-211.

Muleta, S., Yohannes F. and Rashid, S.M. (2006). Soil Erosion Assessment of Lake Alemoya Catchment, Ethiopia. Land Degradation and Development, 71,331-341.

Naden, P. S. (1992). Spatial Variability in Flood Estimation for large Catchments. The Exploitation of Channel Network Structure. Hydrol. Sc J. 37 (1). 53-71.

Nayak, P. (2008). Fuzzy Computing Techniques for Modeling Hydrological Time Series. Journal of Hydrology. 291, 52-66

NCGIA (1996). GIS and Society. The Social Implications of how People, Space and Environment are represented in GIS. Scientific Report for the Initiative-19. Specialist Meeting available online at: http://www.ucsb.edu/reserach/i17/spec report. htm#RTFToC1

Nijssen, B., Haddalend, I. and Lettenmaier, D. P. (1997). Point Evaluation of a Surface Hydrology Model for BOREAS Journal of Geophysical Research, 102 (D24), 29, 367-29

Nussbaum, P., Schreiba, T. and Schaffrath, C. (1996). Nonlinear Time Sequence Analysis. Int. J. Bifurcation Chao.I(3), 521-547.

Nwezeh, K. (2001). "FG to Spend $32m On Gombe Dam Project". This Day. Retrieved 2010-05-23, pp15.

Ogben, F. L, Shanf, H. O., Senerath, U. S., Smith, J. A., Baeck, M.L. and Richardson, J. R. (2000). Hydrometeorological Analysis of the Fort Collins. Colorado Flash Flood of 1997, Journal of Hydrology. 228, 82-100.

Ogbu, M. (2010). "Gombe Rural Communities Enjoy Potable Water -Isa". Daily Independent. Retrieved 2010-05-23, pp 8.

Orfali, R., Hakey, D. and Edwards, J. (1996). The Essential Client-serve Survival Guide. Canada: John Wiley and Sons, Canada. Pp. 72.

Ozgu-Kisi, M. (2007). A Non-linear and Stochastic Response Surface Method for Bayesian Estimation of Uncertainties, Non-linear Process Geophy. II: 2-44.

Parson, S. (1999). Development of an Internet Watershed Education Tool (Internet) for the Spring Greek Watershed of Central Pennsylvania. A PhD Thesis in Agricultural and Biological Engineering. The Pennsylvania State University.

Peng, Z. P. (1997). An Assessment of the Development of Internet GIS. 1997 ESR, International User Conference. Eric Inc. [On-Line]. Available:http://www.gisdevelopment.net/aars/2000ts7/gdi012.shtm .(Accessed April, 2010).

Rajat, K., Asefa, T. and Azouz, A. D. (2000). An Exploration of Artificial Neural Network Rainfall-runoff Forecasting Combined with Wavelet Decomposition. Journal of Environmental Engineering and Science 3, S121-S128.

Raper, J. and Livingstone, D. (1993). High Level Coupling of GIS and Environmental Process Modelling. Second International Conference/Workshop on Integrating Geographical Information Systems and Environmental Modelling. September. Breckenridge, London. Pp. 87-93.

Refsguard, J. C. (1997). Validation and Intercomparison of different Updating Procedures for Real Time Forecasting. Nord Hydrol. 128, 65-84.

Ricert, M.A. (1996). Apache Server, Survival Guide. Sams. Net publishing Inc.

Rode, M. A. and Lindenschmidt, K. K. (2001). Distributed Sediment and Phosphorous Transport Modelling on Medium-sized Catchment in Central Germany Physics and Chemistry of the Earth (b) 26(7-8). 635-640.

Rodngues, l. H. (2001). Building Imaging Application with Java Technology: Using AWT Imaging, Java 2D and Java Advanced Imaging (JAI). Boston: Addison-Wesley Longman. Pp. 58-61.

Romanowicz, R. J. (2007). Databased Mechanistic Models for low flows Implications for the Effects of Climate Change. Journal of Hydrology. 336, 74-83

Ropke, B., Bach, M. and Frede, H. G. (2004). DRIPS- A DSS for Estimating the Input Quantity of Pesticides for German River Basins. Environmental modelling and Software a (11). 1021-1028.

Rosenthal, W. D., Srinivason, R. and Arnold, J. G. (1995). Alternative River Management Using a Linked GIS-Hydrology Model. Transactions of the ASAE, 38(3). 783-790.

Rumellhert, D.E., McClelland, J.L. and the PDP Research Group. (1986). Parallel Recognition in Modern Computers. In Proceeding: Explorations in the Micro Structure of Cognition. Vol. Foundation, MIT Press/Bradfood Books, Cambridge Mass. Pp 54.

Rundra, R.P., Dickinson, W.T. and Arnold, J.G. (1985). Application of the CREAMS Model in Southern Ontario Conditions. Transactions of ASAE, 28(4):1233-1240.

Salazar, J. E., Caravajah, L. F., Mesa, O. J., and Smith, R. (1998). Modelos de Prediccion de Caudales Mensuales Considerando Anomalia Climaticas. Ponencia XVIII Congresso Latinoam-De Hidraulica Oaxaca Mexico. pp 525-533.

Schimming, C. G., Melte, R., Reiche, E. W., Schrauter, J. and Wetzel, H. (1995). Measurement Budgets Model Evaluation. Journal of Plant Nutrition and Soil Science. 158(4), 313-322.

Schwartz, R., Olson, E. and Christiansen, T. (1997). Learning Perl on Win 32 Systems. O' Really and Associates Inc: Sabastopol CA. pp. 118.

See, L. And Open-shaw, S. (1999). Applying Soft Computing Approaches to River Level Forecasting Hydrol Sci J. 44(5). 763-778.

Semineno, M. (1997). Microsoft Debuts Security Program to Address Active X Issues PC WEER Magazine. [On-Line] Available: http://www.zonet.com/eweek/news/0217/10msoft.html. (Assessed June, 2010).

Shamseldin, A.Y. (1997). Application of A Neural Networks Technique to Rainfall-Runoff Modelling. J. Hydrol., 199, pp 272-294

Singh, V.P. and Woolhiser, D.A. (2002). Mathematical Modelling of Watershed Hydrology. ASCE J. Hydrol. Eng. 7(4), 270-292).

SIS (2003). [On-line]. Available: http://www.servertec. Com/products/html.

Solomatine, D.P. and Dulal, K.N. (2003). Model Trees as An Alternative to Neural Networks in Rainfall-Runoff Modelling. Hydrol. Sci J., 48(3), 399-411.

Song, D., Heywood, M.I. and Zinar-Heywood, A.N (2003). A Linear Genetic Programming Approach to Intrusion Detection. In Cantu-Paz, E. et al., (Eds). GECCO 2003 LNCS Vol. 2724. Berlin: Springer-Verlag. Pp2325-2336.

Spencer, B. (2000), Mobile GIS, Geo Solutions, Geo Asia Pacific, Adams Business Media, Available on line at:http://www.geoplace.com/asiapac/2000/0700/0700sol.asp. (Accessed May 2002).

Srinivasan, S. (1997). Advanced Perl Programming. Sebastopol, CA: O'Relly and Associates Inc., pp. 120-121.

Stron, D. (1999). More on Active X Versus Java Security, Are You Secure? Web Developer Magazine. [On-line] Available at : http://webdeveloper.com/security_java active_html. (Accessed Feb 2010).

Sun, C., Neale, U. and McDonnel, J. (2007). "The Potential of Using ANN in Estimation of Snow Water Equivalent from SSM/I Data", Proc., Engng. Hydrol.,ASCE, New York,pp 234-244.

Takagi, T. and Sugeno, M. (1985). Fuzzy Identification of Systems and its Application and Control. IEEE Transactions on Systems, Man and Cybernetics, 15(1), 116-132.

Taylor, K., Cameron, M. and Haines, J. (1998). An Integrated Information System on the Web for Catchment Management ACM GIS 98, November, 1998, Washington D.C. pp. 220.

Thiery, P. and Amraoui, R. (2001). Local Probabilistic Neural Networks in Hydrology, Phys-Chem. Earth (B). 26(1), 9-14

Thirumaleh, K. and Deo, M.C. (1998). Hydrological Forecasting Using Neural Networks. Journal of Hydrological Engineering 5(2), 180-181.

Timawus, M. (2010). "Jonathan Needs More Than Good Luck". Daily Trust. Retrieved 2010-05-23, pp10.

Todini, E. (2004). Rainfall-runoff Modelling: Past, Present and Future Journal of Hydrology. 100, 342-352

Todini, E. and Di Bacco, M. (1995). A Bayesian Approach to Rainfall Modelling. Proc. Statistical and Bayesian Methods in Hydrological Sciences. An International Conference in Honour of Jaques Bernier-UNESCO, Delft, Netherland.

Traister, R. (1993). Mastering C-pointers, Tools for Programming Power. Book News Inc, New York, pp. 48-50.

Tucci, C. E., Collischonn, M., Dia, P.S. and Clarke, R.T. (2002). Adaptive Forecasting with a Conceptual Rainfall-runoff Model Hydrological Forecasting. Proc. Oxford Symposium, Geneva. IAHS No 129, 425-454.

Upper Benue River Basin Development Authority. (2009). Gombe State, Nigeria.

Van Deussen, W.P. and Kwadijk, J. C. (1993). RHINE FLOW: An Intergraded GIS Water Balance Model for the River Rhine in Application of Geographic Information, System in Hydro GIS 1993. In Kavar, K. and Nachtnebel, H.P. (Eds) IAHS publication No 211.

Vapnik, V. (1995). Statistical Learning Theory. New York: Springer Verlag. Pp. 43.

Vapnik, V. (1998). Statistical Learning Theory. New York: John Wiley and Sons. pp. 38-45.

Venners, B. (1996). Inside the Java Virtual Machine. New York :McGraw-Hill Professional Publishing. 2nd Edition. Pp. 97-100.

Villanueva, A.O., Vieux, B.E. and Bedient, P. B. (2001). Assessing Urban Hydrologic Prediction Accuracy through Event Reconstruction. Journal of Hydrology. 299, 217-236.

Villanueva, A.O., Zamanillo, E.D. and Tucci, C.E (1987). Previsado De Zazao Para Irrigacao. VII Symposia Brasilerio De Resources Hidricos Anasis Vol. 1, 536-549.

Waleed, A. (2003). Approaches to Developing A Web-based GIS Modelling Tool : for Application to Hydrological Nowcasting. A PhD Thesis, Department of Civil Engineering, King's College, University of London. Pp 100-134.

Wall, L., Christiansen, T and Orwant, J. (2000). Programming Perl, O'Relly and Associates, Inc. Sebastopol. CA. pp. 48.

Wang, L. (1997). Network-Oriented GIS: Integrating GIS with the Internet. URISA 1997. Annual Conference Proceedings, Toronto, Canada, July 20-24. 1997. CD-Rom.

Wang, S.H., Huggins, D.G., Frees, L., Volkman, C.G., Lim, N.C., Baker, D.S., Smith, V. and de Noyelles Jr. F. (2005). An Integrated Modelling Approach to Total Watershed Managements: Water Quality and Watershed Assessment of Cheney Reservoir, Kansas, USA. Water, Air and Soil Pollution. 164(1-4), 1-19.

Weinthal, E., Vengosh, A., Marel, A., Gutierrez, A., Marel, A., and koppmann, W. (2005). The water crisis in the Gaza strip: Prospects for Resolution. Groundwater 43(5), pp 653-660.

WHO (World Meteorologic Organization) (1986). Inter-Comparison of Models of Snow melt Runoff, Geneva, Switzerland, pp. 646.

Wilson, C. B., Valdes, J. B. and Rodriguez, I. (1979). On the Influence of Spatial Distribution of Rainfall on Storm Runoff Water Resour res. 15(1), 321-328.

WMS Reference (1999). WMS Watershed Modelling System Reference Manual. Brigham Young University-Environmental Modelling, Research laboratory, pp 56.

Wood, E. F., Lettemaier, D., Liang, X., Nijssen, B. and Wetzel, S.W., (1985). Hydrological Modelling of Continental Scale Basin. Annual Review of Earth and Planetary Sciences. 25, 279-300.

Worboys, M.F., Mason, K.T. and Dawson, B.R.P. (1993). The Object-based Paradigm for Geographical Database System: Modelling Design and Implementation Issues. Georapgical Information Handling-research and Application. Paul Matter (Ed) Chichester; Wiley and Sons publishing company. Pp 91-102.

Yamaltu Local Government Area. (1999). Potrait of Dadin-Kowa L.G.A, Gombe State, Nigeria.

Yan, J. and Haan, C. J. (1991). Multi-obiective Parameter-estimation for Hydrologic Models-Weighing of Errors., Transaction of the ASAE 34(1), 135-141.

Yarnal, B., Lakhtakia, M. N., Yu, Z., White, R. A., Pollard, D., Miller, D. A. and Lapenta. W. M. (2000). A Linked Meterological and Hydrological Model System: The Susquenhanns River Basin Experiment (SRBEX). Global and Planetary change, 25 (1-2), 149-161.

Yoon, K. S., Yoo, K. H., Soileau, J. M. and Touchton, J. T. (1992). Simulation of Sediment and Plant Nutrient Losses by the Creams Water Quality Model. Water Resources Bulletin. 28, 1013-1021.

Young, R.A., Onstead, C.A. and Bosch, D. D. (1989). AGNES-A Nonpoint Sources Pollutant Model for Evaluating Agricultural Watersheds. Journal of Soil and Water Conservation. 44(2), 168-173.

Yourdon, E. (1994). Object-oriented Systems Design: An Integrated Approach. New Jersey: Yourdon press. Pp. 84-86.

Yu, P. S., Chan, S. T. and Chang, I. F. (2006). Support Vector Regression for Real Time Flood Stage Forecasting. Journal of Hydrology. Vol. 328, 704-716.

Yu, X .and Schwartz, L. (1996). Water Problem and Opportunites in the Hydrological Science in China. Hydrol Sci J. 46(6), 907-921.

Yu, X. and Schwartz, L. (1998). A System Approach to Real-time Hydrological Forecasts in Watershed. Water Int. 27 (1) 87-97.

Yu-chi, W., Haan, D., Pao-shan, Y. and Cluckie, I. D. (2006). Comparative Modelling of Two Catchments in Taiwan and England. Hydrol Processes. 20, 4335-4349,

Zadeh, L.A. (1973). The Concept of Linguistic Variable and its Application to Approximate Reasoning. Memorandum ERL-M411, Berkeley. Pp. 38.

Zealand, C.M, Burn, D. H. and Simononic, S.P. (1999). Short-term Streamflow Forecasting using Artificial Neural Networks. Journal of Hydrology 214, 32-48.

Zhang, G., Patuwo, B.E. and Hu, M.Y. (1998). Forecasting with Artificial neural Networks: the State of the Art. International Journal of Forecasting. Vol. 14, 35-62.

Zheng, Y. and Smith, T. (2003). Space-time Variability of Rainfall and Extreme Flood Response in the Manomonee River Basin. Wiscosin. J. Hydrometer, 4, 506-517.

Zhou, D., Schwan, K., Eisenhouer, G. and Chen, Y. (2001). JECHO-Interactive High performance computing with Java event channels-Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS, 2001). Iowa.

Zhu, M.L. and Fugita, M. (1994). Comparison Between Fuzzy Reasoning and Neural Network Methods for Forecasting Runoff Discharge. Journal of Hydro science Hydraulic Engineering 12(12), 131-141.

Zurada, R. J. (1992). The Xinanjiang Model Applied in China. Journal of Hydrology. 13, 371-381.

APPENDICES

Appendix A: Dandin-Kowa Dam Project Features

DAM

Type: Earth and Rockfill (1×10^6 m^3)

Height: 42meters Above Deepest Foundation

Length of Crest: 520meters

Width at Crest: 230meters

SPILLWAY

Location: Left Abutment

Type: Gated Overflow Crested-open chute Bucket

Design Discharge: 1,1110m^3/s (at reservoir flood level elevation 2490m)

Gates: 3 Radial gates 6m× 8m

RESERVOIR

Maximum Flood Level Elevation: 249m^3

Live Storage Capacity: 1.77× $10^9 m^3$

Surface Area Elevation: 247300mkm^2

Maximum Supply Level Elevation: 239m

FLOOD

Peak Inflow: 3160m^3/s

Peak Outflow: 1110km/s

POWER HOUSE

Location: Left Bank

Installed Capacity: 2 17MW

Type of Turbines: Francis Vertical Shaft

Generators: Umbrella Type

Generator Rating: 22MVA at 0.85 power factor

Discharge at Full Load: 66m^3/s

IRRIGATION

Outlet: One at the Right Bank

Inlet Invert Level Elevation: 233.50meters

Designed discharge: 10m^3/s

Canal Type Length: Trapezoidal Lined 2.5Km Long

Appendix B: Maps of Study Area

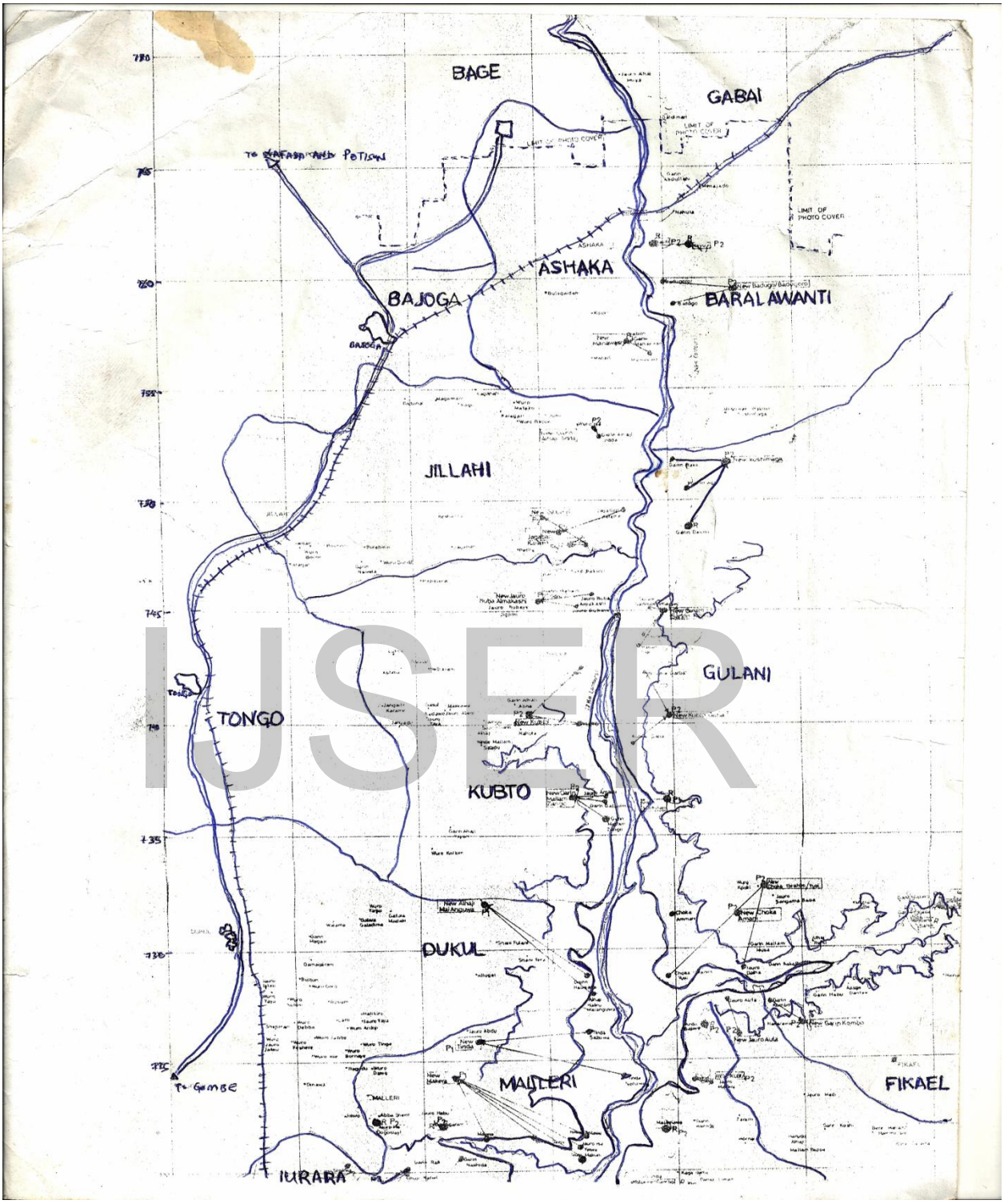


Figure B.1: Towns Located Along the Gongola River

(Source: Upper Benue River Basin Development Authority, 2009)

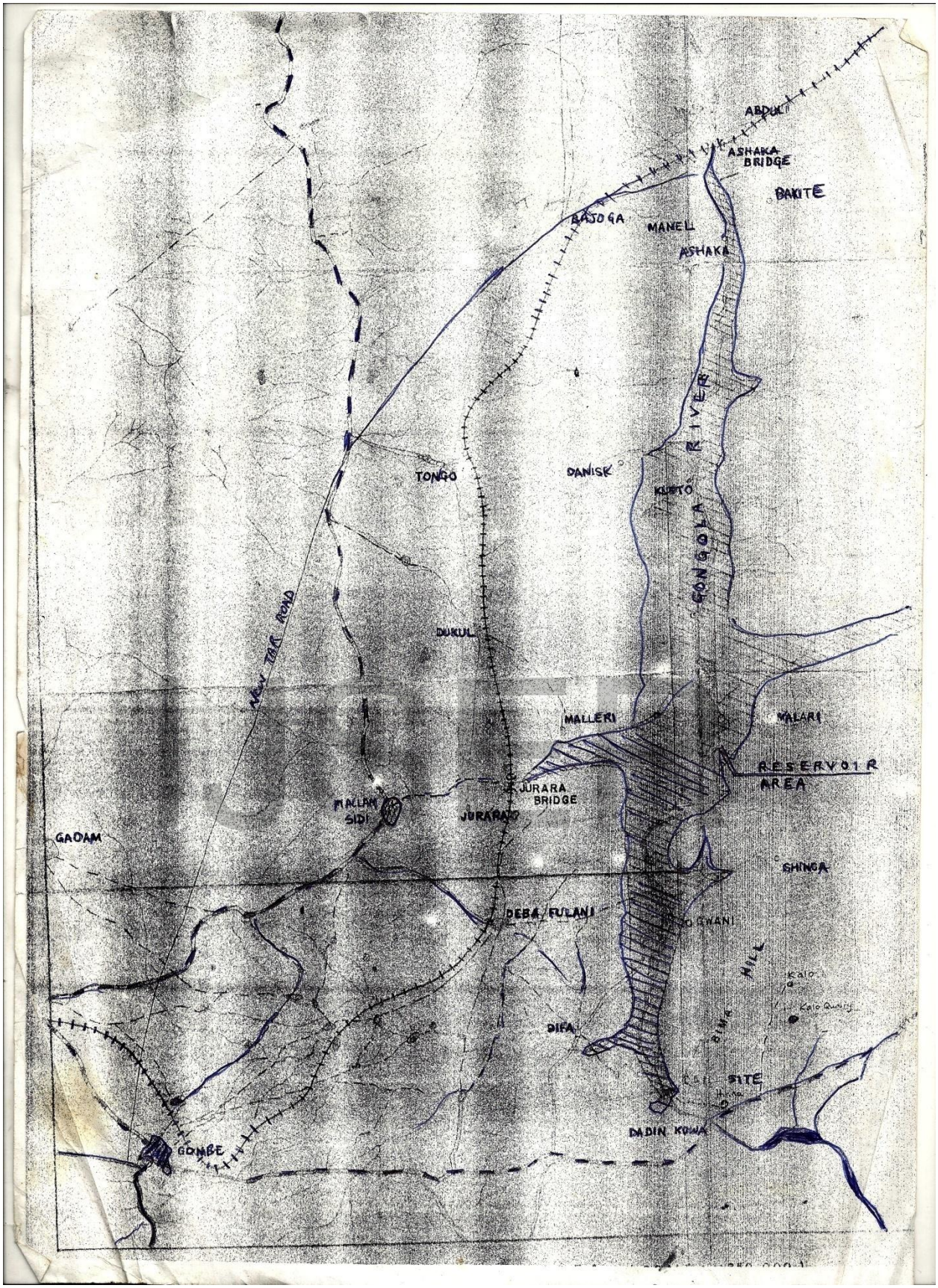


Figure B.2: Location of Dadin-Kowa Reservoir at Gongola River

(Source: Upper Benue River Basin Development Authority, 2009)

Appendix C: System Connections

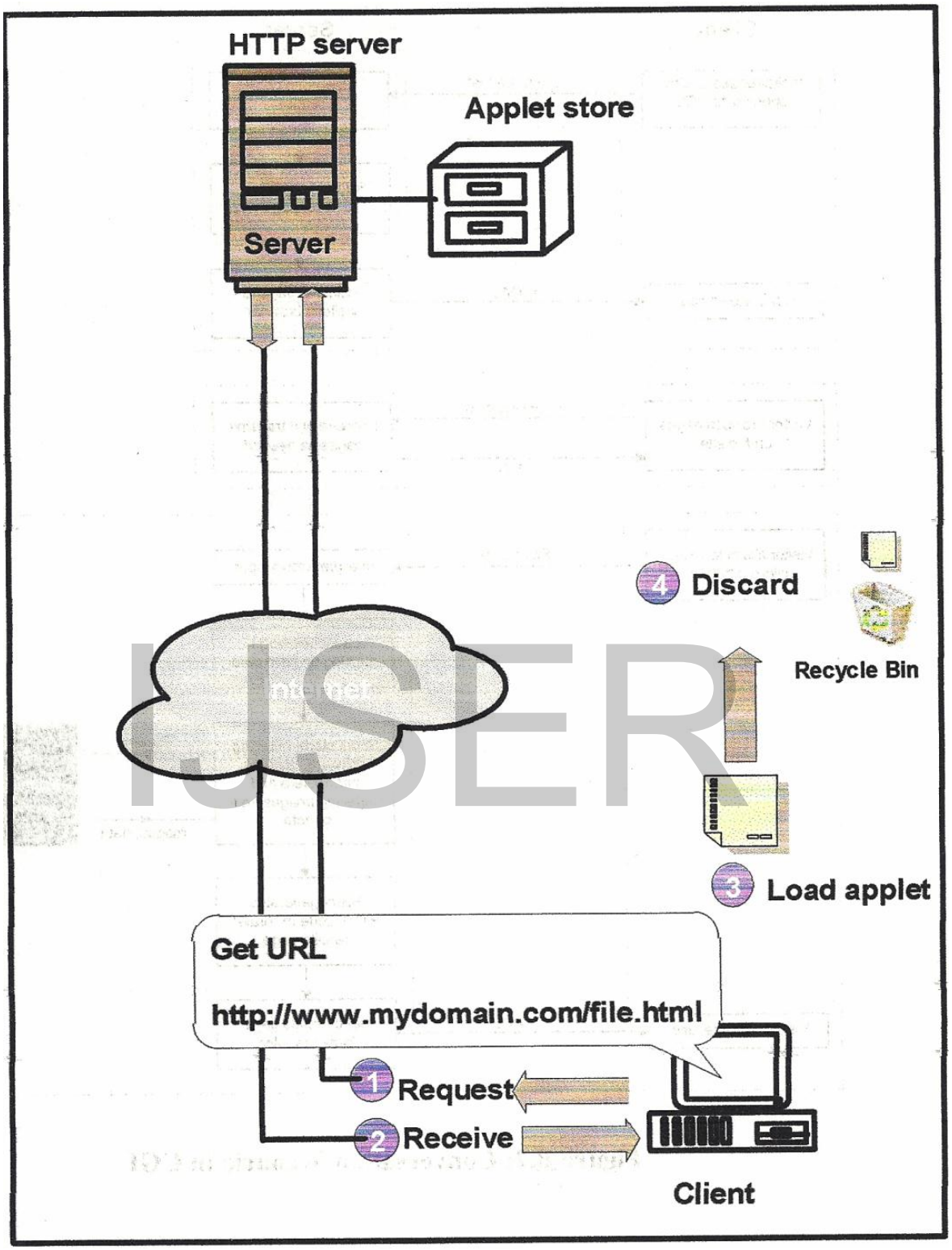


Figure C.1: Applet Mechanism
(Source: Waleed, 2003)

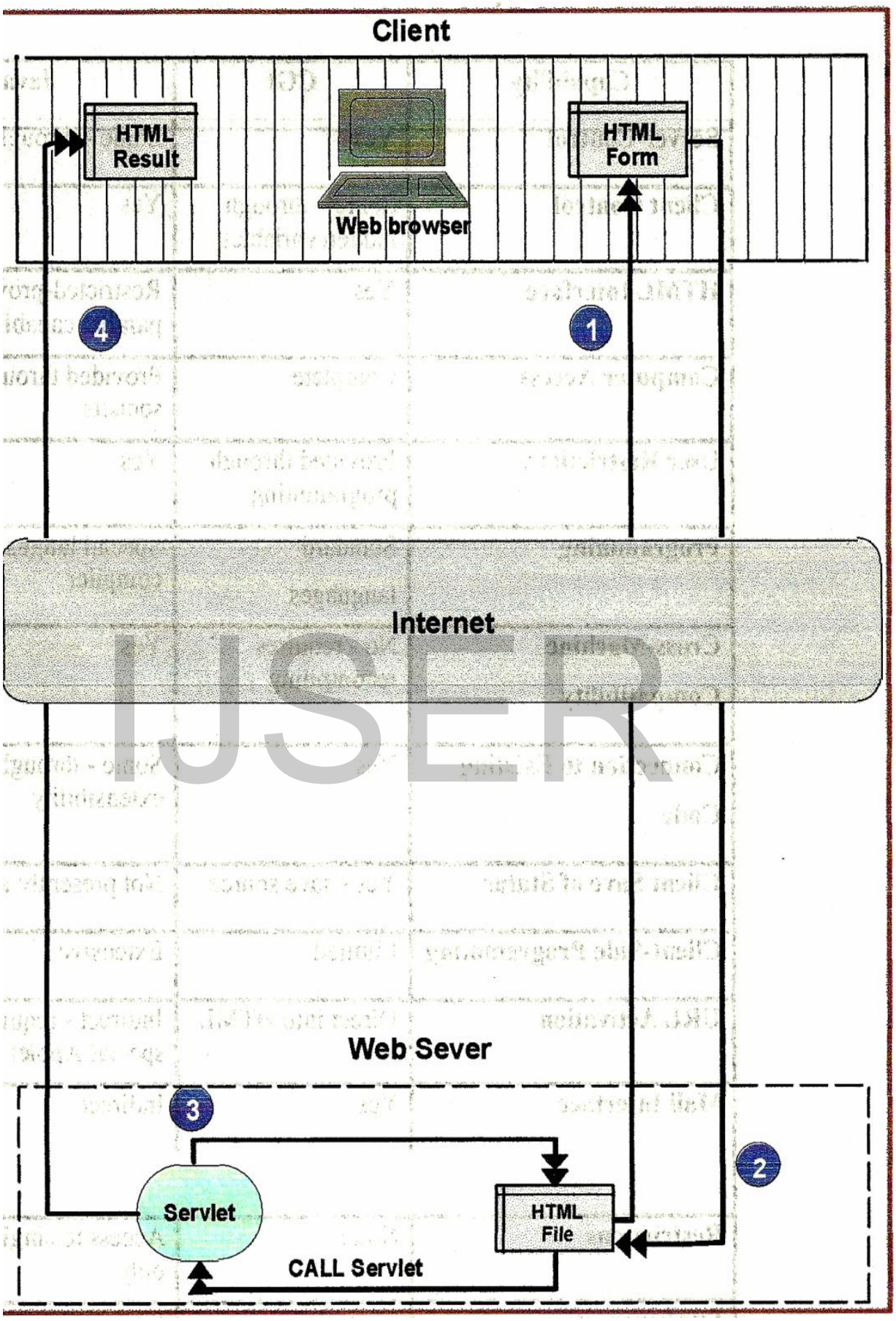


Figure C.2: The Servlet Mechanism (Source: Waleed, 2003)

Appendix D: Java Codes

Code D1: LivePlot

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JFrame;
import javax.swing.SwingUtilities;

import ptolemy.plot.Plot;
import ptolemy.plot.PlotLive;

public class LivePlot extends Plot {
    /** Construct a plot for live, animated output display.
     *  Configure the title, axes, points style, and persistence.
     */


      static public final Plot plot = new Plot();

    public LivePlot() {

        setXRange(0, 365);
        setXLabel("Day of Year");
        setTitle("Estimated Daily Inflow for: Lat:
     "+MainOptions.lat.getText()+" Lon: "+MainOptions.lon.getText());

        setYLabel("Reservoir inflow. (m3/s)");
        setPointsPersistence(10000);
        setMarksStyle("dots");
        setSize(700,300);
        addPoint(0, 0,0, false);

    }


    ////////////////////////////////////////////////////////////////
    ////                       public methods                   ////

    /** Add points to the plot.  This is called by the base class
     *  run() method when the plot is live.
     */
    public synchronized void addPoints() {
        // You could plot multiple points at a time here
        // for faster response, but in our case, we really need
        // to slow down the response for visual aesthetics.
        addPoint(0, Main.day, Main.rain[(int)Main.day], false);


        try {
            Thread.sleep(1);
        } catch (InterruptedException e) {
        }
    }


    public static void main(String[] args) {
        // Run this in the Swing Event Thread.
        Runnable doActions = new Runnable() {
                public void run() {
                    try {


                        JFrame frame = new JFrame("Average Inflow");

                        frame.getContentPane().add("Center", plot);
                        frame.setSize(900,480);

                        frame.show();
```

```
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                        plot.setButtons(true);
                        plot.setTitle("Estimated Average Rainfall");

                       plot.setYRange(0, 5000);
                        plot.setXRange(120, 249);
                        plot.setXLabel("Day of Year");
                        plot.setTitle("Estimated Daily Inflow for: Lat:
    "+MainOptions.lat.getText()+" Lon: "+MainOptions.lon.getText());


                        // for interval days:
                        plot.addXTick("0",0);
                        plot.addXTick("50",50);
                        plot.addXTick("100",100);
                        plot.addXTick("150",150);
                        plot.addXTick("200",200);
                        plot.addXTick("250",250);
                        plot.addXTick("300",300);
                        plot.addXTick("350",350);
                        plot.addXTick("400",400);



                        plot.setYLabel("Reservoir Inflow (m3/s)");
                        plot.setPointsPersistence(10000);
                        plot.setMarksStyle("dots");
                        plot.setSize(800,400);
                        plot.addLegend(0,"Rainfall");


                      //  frame.pack();
                    } catch (Exception ex) {
                        System.err.println(ex.toString());
                        ex.printStackTrace();
                    }
                }
            };

        try {
            SwingUtilities.invokeAndWait(doActions);
        } catch (Exception ex) {
            System.err.println(ex.toString());
            ex.printStackTrace();
        }
    }

    ////////////////////////////////////////////////////////////////
    ////                         private variables                ////

    /** @serial Value being plotted */
    private double _count = 0.0;
}
```

Code D2: MainOptions

```
import java.awt.Component;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.URL;

import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import com.jgoodies.forms.builder.PanelBuilder;
import com.jgoodies.forms.factories.Borders;
import com.jgoodies.forms.layout.CellConstraints;
import com.jgoodies.forms.layout.FormLayout;

public class MainOptions {

    public static boolean Latlon = false;

    static String _lat="", _lon="", _yr="", _elev="";
    static JTextField lat;
    static JTextField lon;
    static JTextField elev;
    static JTextField yr;
    static JLabel city;
    static JSlider percentage = new JSlider();
    static JLabel perlabel = new JLabel();

    // output options...
    static JCheckBox tmax = new JCheckBox("Maximum Temeprature");
    static JCheckBox tmin= new JCheckBox("Mimimum Temeprature");
    static JCheckBox statusoutput= new JCheckBox("Show Caluclation
     Status");

    // other options...
    static JCheckBox randtemp= new JCheckBox("Include Random Diurnal
     Temperature Effect");

    static JCheckBox Evap= new JCheckBox("Evaporation Estimate (mm/day)");


    JButton btn_c = new JButton("Completed");
    JButton btn_hlp = new JButton ("Help");
    JButton btn_exit = new JButton ("Exit");

    private JButton  b_zip;


    public static void main(String[] args) {


        JFrame frame = new JFrame();
        frame.setTitle("Dadin-Kowa Resevoir Inflow Forcasting Software -
     Version 1.0a");


     frame.setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);

        JComponent panel = new MainOptions().buildPanel();

        frame.getContentPane().add(panel);

        frame.pack();

        frame.setVisible(true);


    }


    public JComponent buildPanel() {
```

```
    JTabbedPane tabbedPane = new JTabbedPane();
    tabbedPane.putClientProperty("jgoodies.noContentBorder",
Boolean.TRUE);


    // only display main options (geographical settings....

    tabbedPane.add("Main", buildLatLonPanel());

    //   tabbedPane.add("Vertical",   buildVerticalSizesPanel());
   //   tabbedPane.add("File Output",   buildPanel2());

    return tabbedPane;
}
private JComponent buildLatLonPanel() {


    lat       = new JTextField();
    lon        = new JTextField();
     city = new JLabel(" ");
    randtemp = new JCheckBox();


    elev      = new JTextField();
    yr        = new JTextField();
    b_zip   = new JButton ("Search Zip Code");

    btn_c.addActionListener(new Completed());
    btn_hlp.addActionListener(new HelpDialog());
    btn_exit.addActionListener(new PrgExit());


    percentage.setMaximum(400);
    percentage.setMinimum(1);
    percentage.setValue(100);
    percentage.setLabelTable(percentage.createStandardLabels(50));

    percentage.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {

        if (percentage.getValue()>100)
perlabel.setText(percentage.getValue()+"");
        if (percentage.getValue()<100)
perlabel.setText(percentage.getValue()+" ");
        if (percentage.getValue()<10)
perlabel.setText(percentage.getValue()+"  ");

        // change event??

        }
  });



    FormLayout layout = new FormLayout(
            "right:max(40dlu;pref),"+
            "3dlu, 70dlu, 7dlu, "
          + "right:max(40dlu;pref),",//  3dlu, 70dlu",

           "p, 3dlu, p, 3dlu, p, 3dlu, p, 9dlu, "
          + "p, 3dlu, p, 3dlu, p, 3dlu, p, 9dlu, "
          + "p, 3dlu, p, 3dlu, p, 3dlu, p");

    PanelBuilder builder = new PanelBuilder(layout);


    builder.setDefaultDialogBorder();

    builder.addSeparator("Enter Location");
    builder.nextLine(2);
```

```
        builder.addLabel("Latitude");         builder.nextColumn(2);
        builder.add(lat);
        builder.nextLine(2);

        builder.add(city);           builder.nextColumn(2);
                          builder.nextColumn(2);    builder.add(b_zip);

        b_zip.addActionListener(new zipcodesearch());

        builder.nextLine(2);

        builder.addLabel("Longitude");           builder.nextColumn(2);
        builder.add(lon);
        builder.nextLine(2);

//        builder.addSeparator("Other Parameters");
//        builder.nextLine(2);
//        builder.addLabel("Random Temp:");builder.nextColumn(2);
//        builder.add(randtemp);
//        builder.nextLine(2);
//
//        builder.addLabel("Elevation (in meters)");
     builder.nextColumn(2);
//        builder.add(elev);
//        builder.nextLine(8);
//
        builder.nextColumn(2);

        builder.add(btn_c);  builder.nextColumn(2);
        // builder.add(btn_hlp);  diabeled help button for now...


                          return builder.getPanel();


    }

    /**
     * Builds the content pane.
     *
     * @return the built panel
     */
    public JComponent buildPanel2() {



        FormLayout layout = new FormLayout(
                "right:max(40dlu;pref), 3dlu, 70dlu",
                "p, 3dlu, p, 3dlu, p, 3dlu, p, 9dlu, "
              + "p, 3dlu, p, 3dlu, p, 3dlu, p, 3dlu, p");
        layout.setRowGroups(new int[][] { { 3, 13, 15, 17 } });
        PanelBuilder builder = new PanelBuilder(layout);
        builder.setDefaultDialogBorder();

        CellConstraints cc = new CellConstraints();
        int row = 1;

        builder.addSeparator("Select Outputs",     cc.xyw (1, row++, 3));
        row++;
        builder.add(tmax,    cc.xy  (1, row++));
        builder.nextLine();

        builder.add(tmin,cc.xy(1,row++));
        builder.nextLine();
        row++;

        //builder.add(RH,cc.xy(1,row++));
        builder.nextLine();row++;

        builder.add(Evap,cc.xy(1,row++));
        builder.nextLine();row++;
```

```
        builder.addSeparator("Comments",  cc.xyw (1, row++, 3));
        builder.nextLine();row++;



        return builder.getPanel();
    }

    /**
     * Builds and returns a combo box for materials.
     *
     * @return a combo box for different materials
     */
    private JComboBox createMaterialComboBox() {
        return new JComboBox(new String[] {"C45E, ReH=600",
                                           "Bolt Material, ReH=800"});
    }
    /**
     * Builds and returns a combo box for ice classes.
     *
     * @return a combo box for a bunch of ice classes
     */
    private JComboBox createIceClassComboBox() {
        return new JComboBox(new String[] { "E", "E1", "E2", "E3", "E4"
     });
    }


    /**
     * Builds and returns a combo box for output frequency.
     *
     * @return a combo box for save intervals...
     */

    private JComboBox createIntervalComboBox() {
        return new JComboBox(new String[] { "Minute", "Hourly", "Daily"});
    }



    class Completed implements ActionListener {
        public void actionPerformed(ActionEvent e) {

              _lat  = lat.getText();
              _lon  = lon.getText();
              _yr   = yr.getText();
              _elev = elev.getText();



              if ( _lon.equals("") || _lat.equals("") )
              {

                  JOptionPane.showMessageDialog(null, "Please enter all
     the required site data.", "Error", 0);

              }
              else
                  {
                System.out.println("completed... ");
                System.out.println("lat:"+_lat+":");
                System.out.println("lon:"+_lon+":");
                System.out.println("yr:"+_yr+":");
                System.out.println("elev:"+_elev+":");
                System.out.println("Random temp :"+randtemp.isSelected());
                Main.DoRandomTemps = randtemp.isSelected();


                MainOptions.Latlon = true;
                  }
```

```
        }
    }


    class HelpDialog implements ActionListener {
        public void actionPerformed(ActionEvent e) {

       // URL index =
     ClassLoader.getSystemResource("helpfile/index.html");
        // display help dialog...
      //  new HelpWindow("SolarCalc Help Screen", index);

    }


}


 class PrgExit implements ActionListener {
    public void actionPerformed(ActionEvent e) {


        // Exits Program


        System.exit(0);

}}
```



```
    private JComponent buildVerticalSizesPanel() {
        FormLayout layout = new FormLayout(
            "pref, 12px, pref",
            "pref, 12px, 45px, 12px, min, 12px, pref");
        // Create a panel that uses the layout.
        JPanel panel = new JPanel(layout);

        // Set a default border.
        panel.setBorder(Borders.DIALOG_BORDER);
        // Create a reusable CellConstraints instance.
        CellConstraints cc = new CellConstraints();

        // Add components to the panel.
        panel.add(new JLabel("new JTextArea(10, 40)"), cc.xy(3, 1));
        panel.add(new JLabel("45px"),     cc.xy(1, 3));
        panel.add(new JLabel("Min"),      cc.xy(1, 5));
        panel.add(new JLabel("Pref"),     cc.xy(1, 7));
        panel.add(createTextArea(10, 40), cc.xy(3, 3));
        panel.add(createTextArea(10, 40), cc.xy(3, 5));
        panel.add(createTextArea(10, 40), cc.xy(3, 7));

        return panel;
    }
    private JComponent createTextArea(int rows, int cols) {
        return new JScrollPane(new JTextArea(rows, cols),
                    ScrollPaneConstants.VERTICAL_SCROLLBAR_NEVER,
                    ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
    }
    // Helper Classes
     ************************************************************
    /**
     * Creates and returns a button that can have predefined minimum
     * and preferred sizes. In the constructor you can specify or omit
     * the minimum width, height and preferred width/height.
     */
//    private static class SpecialSizeButton extends JButton {
```

```
//
//         private final Dimension fixedMinimumSize;
//         private final Dimension fixedPreferredSize;
//
//         private SpecialSizeButton(
//             String text,
//             Dimension fixedMinimumSize,
//             Dimension fixedPreferredSize) {
//             super(text);
//             this.fixedMinimumSize   = fixedMinimumSize;
//             this.fixedPreferredSize = fixedPreferredSize;
//             //putClientProperty("jgoodies.isNarrow", Boolean.TRUE);
//         }
//
//         public Dimension getMinimumSize() {
//             Dimension d = super.getMinimumSize();
//             return new Dimension(
//                 fixedMinimumSize.width  == -1
//                     ? d.width
//                     : fixedMinimumSize.width,
//                 fixedMinimumSize.height == -1
//                     ? d.height
//                     : fixedMinimumSize.height);
//         }
//
//         public Dimension getPreferredSize() {
//             Dimension d = super.getPreferredSize();
//             return new Dimension(
//                 fixedPreferredSize.width  == -1
//                     ? d.width
//                     : fixedPreferredSize.width,
//                 fixedPreferredSize.height == -1
//                     ? d.height
//                     : fixedPreferredSize.height);
//         }
//     }
}
```

Code D3: ReadNewPrecip

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.StringTokenizer;
import java.util.zip.GZIPInputStream;


public class ReadNewPrecip {

    public static boolean readavgprecip (double lat2, double lon2) {

        String data[] = new String[150];
        int count;
      //  System.out.println("Starting Search: Lat: "+lat2+" Lon: "+lon2);

        String fileName = "PRECIP_NEW.gzip";

        URL fl = ZipSearch.class.getResource(fileName);

        if (fl == null) {
            throw new RuntimeException("Average Precip file not found!");
        }

        boolean success = false;
        BufferedReader in = null;

        try {
            in = new BufferedReader(new InputStreamReader(new
    GZIPInputStream(fl.openStream())));
        } catch (IOException e) {
            e.printStackTrace();
        }
        boolean continuesearch=true;
        int loops=0;

        String a, b, s = null;
        StringBuffer sb = new StringBuffer();

                try {
                    while ((s = in.readLine()) != null &&
    continuesearch==true) {
                        loops++;


                // separate string "b":
                StringTokenizer st = new StringTokenizer(s, ",  ");
                count = 0;

                while (st.hasMoreElements()) {

                    data[count] = (st.nextToken());
                    count++;
                }

                // convert first two values to lat & lon:
                double lo = Double.valueOf(data[0]).doubleValue();
                double la = Double.valueOf(data[1]).doubleValue();


                // check for match ??
                if (la == lat2 && lo == lon2)
                {
         //         System.out.println("Match Found !!! Lat:"+la+"
    Lon:"+lo);
                continuesearch=false;
```

```
                // assign array values:
                success = true;
                for (int yy = 1; yy <= (int) 13; yy++)
                {
                   // read monthly values:
                    RainSIM.avgMonthlyPrecip[yy] =
Double.valueOf(data[yy+1]).doubleValue();
    //            System.out.println("Row:"+lat2+" Col:"+lon2+" Month
"+yy+"--> "+data[yy+1]);
                }
                }

                 }
                } catch (IOException e) {
                    e.printStackTrace();
                }

        return success;
}




}
```

Code D4: Spline

```
import java.text.*;

public class Spline
{
  int n, last_interval;
  double x[], f[], b[], c[], d[];
  boolean uniform, debug;

  public Spline(double xx[], double ff[])
  {
    // Calculate coefficients defining a smooth cubic interpolatory
     spline.
    //
    // Input parameters:
    //   xx  = vector of values of the independent variable ordered
    //         so that  x[i] < x[i+1]  for all i.
    //   ff  = vector of values of the dependent variable.
    // class values constructed:
    //   n   = number of data points.
    //   b   = vector of S'(x[i]) values.
    //   c   = vector of S"(x[i])/2 values.
    //   d   = vector of S'''(x[i]+)/6 values (i < n).
    //   x = xx
    //   f = ff
    // Local variables:
    double fp1, fpn, h, p;
    final double zero = 0.0, two = 2.0, three = 3.0;
    DecimalFormat f1 = new DecimalFormat("00.00000");
    boolean sorted = true;

    uniform = true;
    debug = false;
    last_interval = 0;
    n = xx.length;
    if(n<=3)
    {
      System.out.println("not enough points to build spline, n="+n);
      return;
    }
    if(n!=ff.length)
    {
      System.out.println("not same number of x and f(x)");
      return;
    }
    x = new double[n];
    f = new double[n];
    b = new double[n];
    c = new double[n];
    d = new double[n];
    for(int i=0; i<n; i++)
    {
      x[i] = xx[i];
      f[i] = ff[i];
      if(debug) System.out.println("Spline data x["+i+"]="+x[i]+",
     f[]="+f[i]);
      if(i>=1 && x[i]<x[i-1]) sorted=false;
    }
    // sort if necessary
    if(!sorted)
    {
      if(debug) System.out.println("sorting");
      dHeapSort(x, f);
    }

    // Calculate coefficients for the tri-diagonal system: store
    // sub-diagonal in b, diagonal in d, difference quotient in c.
    b[0] = x[1]-x[0];
    c[0] = (f[1]-f[0])/b[0];
    if(n==2)
```

```
    {
       b[0] = c[0];
       c[0] = zero;
       d[0] = zero;
       b[1] = b[0];
       c[1] = zero;
       return;
    }
    d[0] = two*b[0];
    for(int i=1; i<n-1; i++)
    {
       b[i] = x[i+1]-x[i];
       if(Math.abs(b[i]-b[0])/b[0]>1.0E-5) uniform = false;
       c[i] = (f[i+1]-f[i])/b[i];
       d[i] = two*(b[i]+b[i-1]);
    }
    d[n-1] = two*b[n-2];

    // Calculate estimates for the end slopes.  Use polynomials
    // interpolating data nearest the end.
    fp1 = c[0]-b[0]*(c[1]-c[0])/(b[0]+b[1]);
    if(n>3) fp1 = fp1+b[0]*((b[0]+b[1])*(c[2]-c[1])/
                  (b[1]+b[2])-c[1]+c[0])/(x[3]-x[0]);
    fpn = c[n-2]+b[n-2]*(c[n-2]-c[n-3])/(b[n-3]+b[n-2]);
    if(n>3) fpn = fpn+b[n-2]*(c[n-2]-c[n-3]-(b[n-3]+
                  b[n-2])*(c[n-3]-c[n-4])/(b[n-3]+b[n-4]))/(x[n-1]-x[n-
     4]);

    // Calculate the right-hand-side and store it in c.
    c[n-1] = three*(fpn-c[n-2]);
    for(int i=n-2; i>0; i--)
       c[i] = three*(c[i]-c[i-1]);
    c[0] = three*(c[0]-fp1);

    // Solve the tridiagonal system.
    for(int k=1; k<n; k++)
    {
       p = b[k-1]/d[k-1];
       d[k] = d[k]-p*b[k-1];
       c[k] = c[k]-p*c[k-1];
    }
    c[n-1] = c[n-1]/d[n-1];
    for(int k=n-2; k>=0; k--)
       c[k] = (c[k]-b[k]*c[k+1])/d[k];

    // Calculate the coefficients defining the spline.
    h = x[1]-x[0];
    for(int i=0; i<n-1; i++)
    {
       h = x[i+1]-x[i];
       d[i] = (c[i+1]-c[i])/(three*h);
       b[i] = (f[i+1]-f[i])/h-h*(c[i]+h*d[i]);
    }
    b[n-1] = b[n-2]+h*(two*c[n-2]+h*three*d[n-2]);
    if(debug) System.out.println("spline coefficients");
    for(int i=0; i<n; i++)
    {
      if(debug) System.out.println("i="+i+", b[i]="+f1.format(b[i])+",
     c[i]="+
                          f1.format(c[i])+", d[i]="+f1.format(d[i]));
    }
  } // end Spline


  public double spline_value(double t)
  {
    // Evaluate the spline s at t using coefficients from Spline
     constructor
    //
    // Input parameters
    //   class variables
    //   t = point where spline is to be evaluated.
```

```
    // Output:
    //   s = value of spline at t.
    // Local variables:
    double dt, s;
    int interval; // index such that t>=x[interval] and t<x[interval+1]

    if(n<=1)
    {
      System.out.println("not enough points to compute value");
      return 0.0; // should throw exception
    }
    // Search for correct interval for t.
    interval = last_interval; // heuristic
    if(t<x[0])
    {
      System.out.println("requested point below Spline region");
      return 0.0; // should throw exception
    }
    if(t>x[n-1])
    {
      System.out.println("requested point above Spline region");
      return 0.0; // should throw exception
    }
    if(t>x[n-2])
       interval = n-2;
    else if (t >= x[last_interval])
       for(int j=last_interval; j<n&&t>=x[j]; j++) interval = j;
    else
       for(int j=last_interval; t<x[j]; j--) interval = j-1;
    last_interval = interval; // class variable for next call
    if(debug) System.out.println("interval="+interval+", x[interval]="+
                                 x[interval]+", t="+t);
    // Evaluate cubic polynomial on [x[interval] , x[interval+1]].
    dt = t-x[interval];
    s = f[interval]+dt*(b[interval]+dt*(c[interval]+dt*d[interval]));
    return s;
  } // end spline_value

  public double integrate()
  {
    double suma, sumb, sumc, sumd;
    double dx, t;
    if(n<=3)
    {
      System.out.println("not enough data to integrate");
      return 0.0;
    }
    if(!uniform)
    {
      if(debug) System.out.println("non uniform spacing integration");
      t = 0.0;
      for(int i=0; i<n-1; i++)
      {
        dx = x[i+1]-x[i];
        t = t + (f[i]+(b[i]/2.0+(c[i]/3.0+dx*d[i]/4.0)*dx)*dx)*dx;
      }
      return t;
    }
    // compute uniform integral of spline fit
    suma = 0.0;
    sumb = 0.0;
    sumc = 0.0;
    sumd = 0.0;
    for(int i=0; i<n; i++)
    {
      suma = suma + d[i];
      sumb = sumb + c[i];
      sumc = sumc + b[i];
      sumd = sumd + f[i];
    }
    dx = x[1]-x[0]; // assumes equally spaced points
    t = (sumd+(sumc/2.0+(sumb/3.0+dx*suma/4.0)*dx)*dx)*dx;
```

```
    if(debug) System.out.println("suma="+suma+", sumb="+sumb+",\n sumc="+
                                 sumc+", sumd="+sumd);
    return t;
  } // end integral

  static void dHeapSort(double key[], double trail[])
  {
    int nkey = key.length;
    int last_parent_pos = ( nkey - 2 ) / 2;
    int last_parent_index = last_parent_pos;
    double tkey, ttrail;
    int index_val;

    if(nkey <= 1) return;
    for(index_val=last_parent_index; index_val>=0; index_val--)
        dremake_heap( key, trail, index_val, nkey-1);

    tkey = key[0];
    key[0] = key[nkey-1];
    key[nkey-1] = tkey;
    ttrail = trail[0];
    trail[0] = trail[nkey-1];
    trail[nkey-1] = ttrail;

    for(index_val=nkey-2; index_val>0; index_val--)
    {
      dremake_heap(key, trail, 0, index_val);
      tkey = key[0];
      key[0] = key[index_val];
      key[index_val] = tkey;
      ttrail = trail[0];
      trail[0] = trail[index_val];
      trail[index_val] = ttrail;
    }
  } // end dHeapSort

  static void dremake_heap(double key[], double trail[], int parent_index
     ,int last_index)
  {
    int last_parent_pos = ( last_index - 1 ) / 2;
    int last_parent_index = last_parent_pos;
    int l_child;
    int r_child;
    int max_child_index;
    int parent_temp = parent_index;
    double tkey, ttrail;

    while(true)
    {
      if( parent_temp > last_parent_index ) break;
      l_child = parent_temp * 2 + 1;
      if( l_child == last_index)
      {
        max_child_index = l_child;
      }
      else
      {
        r_child = l_child+1;
        if(key[l_child]>key[r_child])
        {
          max_child_index = l_child;
        }
        else
        {
          max_child_index = r_child;
        }
      }
      if(key[max_child_index]>key[parent_temp])
      {
        tkey = key[max_child_index];
        key[max_child_index] = key[parent_temp];
        key[parent_temp] = tkey;
```

```
        ttrail = trail[max_child_index];
        trail[max_child_index] = trail[parent_temp];
        trail[parent_temp] = ttrail;
        parent_temp = max_child_index;
      }
      else
      {
        break;
      }
    }
  } // end dremake_heap

  public static void main (String[] args) // no args expected
  {
    // FUNCTION:  Driver for using cubic interpolatory spline routines.

    int n = 14;

    double xx[] = new double[n];
    double ff[] = new double[n];
    double s, t;
    DecimalFormat f1 = new DecimalFormat("00.00000");
    // Assign values for data arrays x and f.
    System.out.println("Spline.java function definition");

   for(int i=0; i<n; i++)
    {
      // assign month data to x---> number of days (xx[i])
      double day = 15.5;
      double doy=0;

      double leapday=0.25;

                if(i==1)
                    doy=day;

                else if(i==2)
                    doy=31+day;

                else if(i==3)
                    doy=31+28+leapday+day;

                else if(i==4)
                    doy=31+31+28+leapday+day;

                else if(i==5)
                    doy=30+31+31+28+leapday+day;

                else if(i==6)
                    doy=31+30+31+31+28+leapday+day;

                else if(i==7)
                    doy=30+31+30+31+31+28+leapday+day;

                else if(i==8)
                    doy=31+30+31+30+31+31+28+leapday+day;

                else if(i==9)
                    doy=31+31+30+31+30+31+31+28+leapday+day;

                else if(i==10)
                    doy=30+31+31+30+31+30+31+31+28+leapday+day;

                else if(i==11)
                    doy=31+30+31+31+30+31+30+31+31+28+leapday+day;

                else if(i==12)
                    doy=30+31+30+31+31+30+31+30+31+31+28+leapday+day;


                xx[i]=doy;
```

```
      ff[i] = Math.pow(Math.cos(xx[i]), 4);
      System.out.println("i="+i+", x[i]="+f1.format(xx[i])+
                         ", f(x[i])="+f1.format(ff[i]));
    }
    // Calculate coefficients defining the cubic spline.
    Spline s11 = new Spline(xx, ff);
    System.out.println("interpolated values");


    for(int i=0; i<365; i++)
    {
      t = i;
      s = s11.spline_value(t);
      System.out.println("x="+f1.format(t)+",  f(x)="+f1.format(s));
    }


  } // end main
} // end class Spline
```

Code D5: ZipSearch

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.zip.GZIPInputStream;
import java.util.zip.GZIPOutputStream;

public class ZipSearch {

   // private static String fileName = "src/sourcecode/avgtemp.gzip";
     private static String fileName = "ZIP_CODES.gzip";
         static double lat = 0, lon = 0;
    static String city;
    static String county;

    public static void main(String[] args) {
        int zip = 56267;

        int y;


       compress();

//
//        try {
//          y=ReadPrecip.readdeltatemps(null);
//    } catch (IOException e1) {
//          // TODO Auto-generated catch block
//          e1.printStackTrace();
//    }
//


    }

    private static void compress(){
        try {
            // Create the GZIP output stream
            GZIPOutputStream out = new GZIPOutputStream(new
     FileOutputStream(fileName));

            // Open the input file
            File inFile = new File("src/source/precip.clim");
            if(!inFile.exists()){
                throw new RuntimeException(inFile.toString() + " not
     found!");
            }
            FileInputStream in = new FileInputStream(inFile);

            // Transfer bytes from the input file to the GZIP output
     stream
            byte[] buf = new byte[1024];
            int len;
            while ((len = in.read(buf)) != -1) {
                out.write(buf, 0, len);
//                System.out.println("Writing 1k");

            }
            in.close();

            // Complete the GZIP file
            out.finish();
            out.close();
        } catch (IOException e) {
        }
    }
```

```
    static void zipsearch(int zipc) throws IOException  {

        URL fl = ZipSearch.class.getResource(fileName);
        if (fl == null) {
            throw new RuntimeException("Zip code file not found!");
        }

        BufferedReader in = null;

        in = new BufferedReader(new InputStreamReader(new
    GZIPInputStream(fl.openStream())));

        String s;


        while ((s = in.readLine()) != null) {
            String zipString = s.substring(1, 6);

            int zip = Integer.parseInt(zipString);

            if (zipString.charAt(0)=='0') {
                zipString = zipString.substring(1,zipString.length());

            }

            if (zipc==Integer.parseInt(zipString)) {
                //                System.out.println("Found desired zip! "
    + s);
                s = s.replaceAll("\"", "");
                String[] lineData = s.split(",");

                lat = Double.parseDouble(lineData[1]);
                lon = Double.parseDouble(lineData[2]);
                city = lineData[3] + ", " + lineData[4];
                county = lineData[5];
                return;
            }
        }
        throw new RuntimeException("Invalid Zip Code, not found in file");
    }

}
```

Code D6: Main

```
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintStream;
import java.text.DecimalFormat;
import java.util.StringTokenizer;
import java.util.Timer;

import javax.swing.BorderFactory;
import javax.swing.JComponent;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JProgressBar;
import javax.swing.JTextField;
import javax.swing.WindowConstants;
import javax.swing.border.Border;


public class Main {
      public static boolean DoRandomTemps = false;
     static double rain[]=new double[400];


    public static double day = 0, s = 0;
    public static JTextField statusline = new JTextField(45);
    public static JProgressBar progressBar = new JProgressBar();

    public static void main(String[] args) throws NumberFormatException,
     IOException, InterruptedException {
        Timer timer = new Timer();

        // Get filename:
        DecimalFormat place3 = new DecimalFormat("0.000");
        DecimalFormat pl2 = new DecimalFormat("0.00");

        // Display main Screen ???

        JFrame fframe = new JFrame("Loading Resevoir Inflow
     Simulator....");
        fframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container content = fframe.getContentPane();

        progressBar.setValue(0);
        progressBar.setStringPainted(true);
        Border border = BorderFactory.createTitledBorder("Connecting to
     the internet...");
        progressBar.setBorder(border);

        content.add(progressBar, BorderLayout.NORTH);


        content.add(statusline, BorderLayout.SOUTH);

        fframe.setSize(300, 100);
 // center dialog window on the screen:
        Dimension screenSize =
     Toolkit.getDefaultToolkit().getScreenSize();
        screenSize.height = screenSize.height/2;
        screenSize.width = screenSize.width/2;
        int y = screenSize.height - ((int)fframe.getSize().getHeight()/2);
        int x = screenSize.width - ((int)fframe.getSize().getWidth()/2);
```

```
        fframe.setLocation(x, y);
        fframe.setVisible(true);

        // read in diurnal temperatures...
        // reading status in GUI ??

        // data has been read into array --> diurnaltemp[lat][lon][month]

        fframe.setVisible(false);
        boolean flag = true;

        while (flag)
        {
            String g = null;
            JFrame frame = new JFrame();
            frame.setTitle("Dadin-Kowa Resevoir Inflow Forcasting Software
    - Version 1.0a");
            frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            JComponent panel = new MainOptions().buildPanel();
            frame.getContentPane().add(panel);
            frame.pack();

            // center dialog window on the screen:

            screenSize = Toolkit.getDefaultToolkit().getScreenSize();
            screenSize.height = screenSize.height / 2;
            screenSize.width = screenSize.width / 2;
            y = screenSize.height - ( (int)frame.getSize().getHeight()/2);
            x = screenSize.width -((int)frame.getSize().getWidth()/2);;

            frame.setLocation(x, y);

            frame.setVisible(true);

            while (MainOptions.Latlon == false) {
                // wait for dialog to be ready
                Thread.sleep(10);

            }

            frame.hide();


            /// ----> after input of Lat/lon data:
            System.out.println("Finished Waiting Loop....");

           MainOptions.Latlon = false;

            double lat = Double.parseDouble(MainOptions.lat.getText()) +
    0.0001;
            double orglon = Double.parseDouble(MainOptions.lon.getText())
    + 0.0001;
            fframe.show();


            rain = RainSIM.getRain(lat, orglon, 0, 100);

            double lon;

            // routine could be separated here with passed in values of
    lat and lon...

            // output lat + long..

            statusline.setText("Lat: " + lat + "  Lon:" + orglon);

            // average lat/long to 5 degree steps...
            // convert row --> lat  ::  lat = (-0.5*row)+90.25
            // lat -- row is below:

            // convert lon to 0-360 versus -180 to 180
```

```
        // send data to Read Avg Temperature Program with calc lt and
ln


        LivePlot.main(args);

       // average temperatures found...


        for (int dy = 1; dy < 366; dy++) {
            day = dy;
            LivePlot.plot.addPoint(0, day, rain[dy], true);

        }


        File file;
        statusline.setText("Waiting for user input");

        // query user if want to output daily temperatrue file ?
        fframe.setVisible(false);


        int optionPane = JOptionPane.showConfirmDialog(null, "Do you
wish to view the Daily Inflow "
                + "for:\n" + "  Lat :  " + pl2.format(lat) + "  Lon:
" + pl2.format(orglon));


        // cumulative file...

        if (optionPane == JOptionPane.YES_OPTION) {
              LivePlot.plot.setYRange(0,rain[399]);
              LivePlot.plot.setTitle("Estimated Daily Resevoir Inflow
for: Lat: "+MainOptions.lat.getText()+" Lon:
"+MainOptions.lon.getText());

              LivePlot.plot.repaint();


              double cr=0;
              LivePlot.plot.addLegend(1,"Resevoir Inflow (m3/s)");

              for (int dy = 120; dy < 366; dy++) {
                 day = dy;
                 cr=cr+rain[dy];

                 LivePlot.plot.addPoint(1, day, cr, true);

             }

        }



        optionPane = JOptionPane.showConfirmDialog(null, "Do you wish
to save the estmiated "
                + "daily inflow for:\n" + "  Lat :  " +
pl2.format(lat) + "  Lon:  " + pl2.format(orglon));


        // OPEN OUTPUT FILE..
        if (optionPane == JOptionPane.YES_OPTION) {
            fframe.setVisible(true);
            statusline.setText("Waiting for user filename input");
            int kf = 0;
            //       outputs daily temperature file...
            JFileChooser chooser = new JFileChooser();
            chooser.setDialogTitle("Select Model Output File");
            int result = chooser.showSaveDialog(panel);
            if (result == JFileChooser.CANCEL_OPTION)
```

```
                System.exit(0);

            file = chooser.getSelectedFile();

            if (file.exists()) {
                // Check if overwrite file...
                int n = JOptionPane.showConfirmDialog(null, "Do you
wish to overwrite the file\n"
                        + file.getAbsolutePath(), "Output File
Exists", JOptionPane.YES_NO_OPTION);

                if (n == JOptionPane.NO_OPTION) {
                    // Not OK to overwrite. keep flag =1
                    kf = 1;
                }
                if (n == JOptionPane.YES_OPTION) {
                    // File OK to overwrite -- then set flag to 0
                    kf = 0;
                }

            }

            else {
                // if file does not exist then set flag to 0
                kf = 0;
            }



            // Open file for output -- nested in try block

            FileOutputStream f = null;

            f = new FileOutputStream(file);
            // attach a PrintStream to it so we can print in text form
            PrintStream pf = new PrintStream(f);

            progressBar.setValue(0);
            progressBar.setStringPainted(true);
            border = BorderFactory.createTitledBorder("Writing Model
Output File...");
            progressBar.setBorder(border);
            statusline.setText("Writing file...."+file.getPath());
            for (int dy = 1; dy < 366; dy++) {

                pf.println(dy + ","
                        + (rain[dy]) // max temp (C)
                         );
                                }
            statusline.setText("Closing file:
"+file.getAbsoluteFile());
            pf.close();
            f.close();

            progressBar.setValue(100);

            Thread.sleep(1000);



        }

        statusline.setText("Finished.");

        Thread.sleep(2000);
        fframe.hide();
        flag=false;


    }
```

```
    }

    private static String ChkSpaces(String g) {

        // routine to eliminate spaces in front of fortran foramtted
     integers...
        String s;
        s = g;

        for (int k = 0; k < g.length(); k++) {
            if (g.startsWith(" "))
                s = g.substring(k);

        }
        return s;
    }

}
```

Code D7: RainSIM

```
import java.io.IOException;



public class RainSIM {

      static boolean RainCalc = false;
      static int precip[][][]=new int[400][750][13];
      static int wetday[][][]=new int[400][750][13];
      static double rain []= new double [400];
     static double avgMonthlyPrecip[] = new double[15];
      static double cumrain[]=new double [400];

      /**
       *
       *  getRain -- Function to return estimated daily rainfall amounts:
       *  parameters (latitude, longitude, new yearly amount(mm),
     percentage increase/decrease (0-?, 100% default))
       *  * Defaults are for newrainamount = 0 (use database records) and
     percentage =100 (no modification of rainfall amounts).
       *
       *
      */
       public static double[] getRain (double lat, double orglon, double
     newrainamount, double percentage){
        int y;
           // user changeable features:: ???
        // newrainamount --- user selected new total rain amount.
        //  percentage - scaling of average yearly rainfall...


         // Reads Precipitation Records:

                    if (newrainamount!=0) {percentage = 100.0;}

              try {
                          y=ReadWetDayPrecip.read_wetdays(null);
                    } catch (IOException e1) {
                          // TODO Auto-generated catch block
                          e1.printStackTrace();
                    }


                     // For Location:

                    RainCalc=true;



             double lon;

                    // routine could be separated here with passed in values
     of lat and lon...

                    // output lat + long..

                    //System.out.println("Lat: " + lat + "  Lon:" + orglon);

                    // average lat/long to 5 degree steps...
                    // convert row --> lat  ::  lat = (-0.5*row)+90.25
                    // lat -- row is below:

                    // convert lon to 0-360 versus -180 to 180

                    if (orglon < 0) {
                        lon = orglon + 360.0;
                    } else {
```

```
                    lon = orglon;
                }

                // convert to row col identifiers:

                int lat2 = (int) ((-2.0 * lat) + 180.5);
                int lon2 = (int) ((lon * 2) + 0.25);

                double lt = (-0.5 * lat2) + 90.25;
                double ln = (0.5 * lon2) - 0.25;

                // convert from 0-360-- -180+180

                if (ln > 180) {
                    ln = ln - 360;
                }


               ReadNewPrecip.readavgprecip(lt, ln);



               double raintotal = 0;

                int doy = 0, day=0;
                int leapday = 0;

                raintotal = avgMonthlyPrecip[13];

                if (newrainamount ==0) newrainamount=raintotal;

                for (int i=1;i<=12;i++)
                {
                    // Displays average Precip and Day for Morris:
                //    System.out.println("Avg Precip: Month
(mm/month):"+i+" "+avgMonthlyPrecip[i]+  " Wet
Days:"+wetday[lat2][lon2][i]/10.0);

                    for (int
t=1;t<=((wetday[lat2][lon2][i]/10.0)+0.5);t++)
                        {
                              day = (int) (Math.random()*31.0);

                               if (i == 1) {
                                   doy = day;

                               } else if (i == 2) {
                                   doy = 31 + day;

                               } else if (i == 3) {
                                   doy = 31 + 28 + leapday + day;

                               } else if (i == 4) {
                                   doy = 31 + 31 + 28 + leapday + day;

                               } else if (i == 5) {
                                   doy = 30 + 31 + 31 + 28 + leapday +
day;

                               } else if (i == 6) {
                                   doy = 31 + 30 + 31 + 31 + 28 +
leapday + day;

                               } else if (i == 7) {
                                   doy = 30 + 31 + 30 + 31 + 31 + 28 +
leapday + day;

                               } else if (i == 8) {
                                   doy = 31 + 30 + 31 + 30 + 31 + 31 +
28 + leapday + day;

                               } else if (i == 9) {
```

```
                                    doy = 31 + 31 + 30 + 31 + 30 + 31 +
31 + 28 + leapday + day;

                                } else if (i == 10) {
                                    doy = 30 + 31 + 31 + 30 + 31 + 30 +
31 + 31 + 28 + leapday + day;

                                } else if (i == 11) {
                                    doy = 31 + 30 + 31 + 31 + 30 + 31 +
30 + 31 + 31 + 28 + leapday + day;

                                } else if (i == 12) {
                                    doy = 30 + 31 + 30 + 31 + 31 + 30 +
31 + 30 + 31 + 31 + 28 + leapday + day;
                                }


                            double amt =
avgMonthlyPrecip[i]/(wetday[lat2][lon2][i]/10.0)*newrainamount/raintot
al*(percentage/100.0);
                            rain[doy]+=amt;
                //                System.out.println("Rain on Day:"+day+
" amount: "+ amt);


                        }
                                    }



                for (int t=1;t<366;t++)
                {
                        rain[t]=(int)(rain[t]*100.0);
                        rain[t]=rain[t]/100.0;
                // commenting out print lines.. 11-15-07

                //    System.out.print("NewCalcDay:\t"
+t+"\t"+rain[t]+"\t");
                        cumrain[t]=cumrain[t-1]+rain[t];
                //    System.out.println("Cumulative Rain:\t"
+t+"\t"+cumrain[t]);

                }

                rain[399]=cumrain[365];


                //System.out.println("Total Rain(cm):"+raintotal);

                return rain;


 }

  public static void main(String[] args) {

        double lat = 45.595;
         double orglon = -95.92;
         double lon=0;
        double tt[]=new double [400];


        tt = getRain(lat, orglon, 0, 100);


         for (int t=1;t<366;t++)
        {
        System.out.print("NewCalcDay:\t" +t+"\t"+tt[t]+"\t");
              cumrain[t]=cumrain[t-1]+rain[t];
        System.out.println("Cumulative Rain:\t" +t+"\t"+cumrain[t]);
```

```
            }
             System.out.println("Cumulative Rain:\t" +tt[399]);
            // calculates new rain for new inputed total rain:
            // NewYearlyRainfall(lat, orglon, 400);


        }


}
```

Code D8: ReadWetDaytPrecip

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.StringTokenizer;
import java.util.zip.GZIPInputStream;


public class ReadWetDayPrecip {


      public static void main() {
        //
        System.out.println("Start");

      }
    public static int read_wetdays(String args) throws IOException {

        // defines variables:

        String data[] = new String[150];
        int count;
        String fileName = "WETDAY.gzip";
        String a, b, s;

        URL fl = ZipSearch.class.getResource(fileName);
        if (fl == null) {
            throw new RuntimeException("Wet-Day Frequency File not
     found!");
        }

        BufferedReader in = null;

        in = new BufferedReader(new InputStreamReader(new
     GZIPInputStream(fl.openStream())));

        StringBuffer sb = new StringBuffer();

        // read in first two lines of data...

        a = in.readLine();
        b = in.readLine();

        // separate string "b":
        StringTokenizer st = new StringTokenizer(b, ", ");
        count = 0;

        //     parse the CSV to individual values in data[]
        while (st.hasMoreElements()) {
            count++;
            data[count] = (st.nextToken());
        }
//       System.out.println("Opening Data File :");
//       System.out.println(a);
//       System.out.println(b);

       // retrieve variables in 2nd line of data file:
        double grid_sz = Double.parseDouble(data[1]);
        double xmin = Double.parseDouble(data[2]);
        double xmax = Double.parseDouble(data[3]);
        double ymin = Double.parseDouble(data[4]);
        double ymax = Double.parseDouble(data[5]);
        double n_cols = Double.parseDouble(data[6]);
        double n_rows = Double.parseDouble(data[7]);
        double months = Double.parseDouble(data[8]);

   //     System.out.println("ymin:" + ymin + "xmax:" + xmax);
```

```
        // read rest of file...parse into array
        for (int mth = 1; mth <= (int) months; mth++) {
            Main.progressBar.setValue(mth/12*100);
            Main.progressBar.repaint();
            try {
                  Thread.sleep(10);
            } catch (InterruptedException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            for (int hh = 1; hh <= (int) n_rows; hh++) {
                s = in.readLine();
                for (int yy = 1; yy <= (int) n_cols; yy++) {
                    int begin = (yy * 5) - 5;
                    int end = (begin + 5);
                    String g = s.substring(begin, end);

            // main array for data is created in Main class
                   int wd = (int) Double.valueOf(g).doubleValue();
                   if (wd<-900)
                   { wd=85; // default number of wetdays...
                   }

                    RainSIM.wetday[hh][yy][mth] =wd;


                }
            }
        }

            return 0;
    }
}
```

Code D9: Zipcodesearch

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;

import javax.swing.JOptionPane;

import java.io.IOException;

import javax.swing.JOptionPane;


public class zipcodesearch implements ActionListener {
     public static boolean DO_MULTIPLE_LOOKUPS = false;


    public void actionPerformed(ActionEvent e) {
      // TODO Auto-generated method stub
       int zip; // zip code to look up in database...

            String intStr; // String version of the input number.

            boolean lp = true;

            //... read numbers until CANCEL and/or empty input.
            while (lp) {
                intStr = JOptionPane.showInputDialog(null, "Enter Zip
    Code:");

                if (intStr == null || intStr.equals("")) {
                    //Exit loop on empty input.
                    //Exit loop on Cancel/close box.
                    break;
                }



                if (intStr.charAt(0)=='0') {
                    intStr = intStr.substring(1,intStr.length());

                }
                zip = Integer.parseInt(intStr);  // Convert string to
    int

                try {
                    ZipSearch.zipsearch(zip);

              //   JOptionPane.showMessageDialog(null, "Found Zip:
    " + zip + "\nLatitude :       " + ZipSearch.lat
              //           + "\nLongnitude:  " + ZipSearch.lon +
    "\nLocated in:   " + ZipSearch.city);

                // output debugging tools...
                System.out.println("Zip Code: " + zip + "Found");
                System.out.print("Found Zip:" + zip + "\nLatitude :" +
    ZipSearch.lat + "\nLongnitude:" + ZipSearch.lon
                        + "\nLocated in:" + ZipSearch.city);
                MainOptions.lat.setText(""+ZipSearch.lat);
                MainOptions.lon.setText(""+ZipSearch.lon);
                MainOptions.city.setText(ZipSearch.city);

                lp = DO_MULTIPLE_LOOKUPS;
                } catch (NumberFormatException e1) {
                    JOptionPane.showMessageDialog(null, "Please enter
    a 5 digit number");
                } catch (IOException e2) {
                    JOptionPane.showMessageDialog(null, "Lookup
    Failed!\n" + e);
                } catch (RuntimeException e3) {
                    JOptionPane.showMessageDialog(null,
    e3.getMessage());
```

```
                    }

            }


    }

    /**
     * @param args
     */
    public static void main(String[] args) {
      // TODO Auto-generated method stub

    }

}
```