NEURAL NETWORK BASED INFLOW FORECASTING FOR OPTIMUM POND OPERATION OF A RUN-OF-RIVER TYPE HYDRO PLANT

Miyanakolatanne Hewage Dhammike Wimalaratne

(178534T)

Degree of Master of Science

Department of Electrical Engineering

University of Moratuwa
Sri Lanka

May 2020

NEURAL NETWORK BASED INFLOW FORECASTING FOR OPTIMUM POND OPERATION OF A RUN-OF-RIVER TYPE HYDRO PLANT

Miyanakolatanne Hewage Dhammike Wimalaratne

(178534T)

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Electrical Engineering

Department of Electrical Engineering

University of Moratuwa

Sri Lanka

May 2020

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

I also grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

M.H. Dhammike Wimalaratne

Digitally signed by M.H. Dhammike Wimalaratne
Date: 2020.05.25 13:44:27 +05'30'

(**M.H. Dhammike Wimalaratne**)

The above candidate has carried out research for the Masters Thesis under my supervision.

Signature of the supervisor:

Lidula Nilakshi Widanagama Arachchige

Digitally signed by Lidula Nilakshi Widanagama Arachchige
Date: 2020.05.25 15:16:44 +05'30'

(**Dr. Lidula N. Widanagama Arachchige**)

The above candidate has carried out research for the Masters Thesis under my supervision.

Signature of the supervisor:

Shavindranath Fernando

Digitally signed by Shavindranath Fernando
Date: 2020.05.25 16:31:54 +05'30'

(**Eng. W.J. L. Shavindranath Fernando**)

ABSTRACT

The current practise of pond operation of Upper Kotmale Hydropower Station is studied, where management of the pond is by subjective judgements of the operator. Accurate and reliable inflow forecast makes up an important basis for optimum pond operation connected with effective spillway gate operation. This research proposes a novel technique to forecast inflow to the pond and utilise these forecasts to optimise the operation of the pond.

In the first phase of the research, an artificial neural network based Nonlinear Autoregressive eXogenous model, which is a dynamic neural network meant for time series forecasting, is used to develop the real time inflow forecasting system. Cross correlation analysis is used as feature selection for effective selection of the inputs to the Nonlinear Autoregressive eXogenous network. In the second phase, real time inflow forecast for next six hours is used to optimise the pond operation focusing on goals of shorter-term nature, such as maximising power generation, maximising pond storage and minimising spillway discharge. Multi-objective global optimisation using MATLAB "fmincon" algorithm and weighted approach of solving multi-objective problem are utilised to solve the optimisation problem. Trading-off conflicting objectives by this approach proves very effective. This optimisation approach enhances the flexibility of the operator in the decision making process resulting in achievement of efficiency in pond operation.

The results show that the Nonlinear Autoregressive eXogenous modelling is an efficient tool for inflow forecasting and MATLAB "fmincon" algorithm can be used effectively to carry out the multi-objective optimisation of run-of-river pond. Simulation studies for the past years show that there exists an opportunity for optimising run-of river ponds for generation using inflow forecast and with the use of the proposed methodology, it enhances the hydropower generation with gains of over 5% which is significant in a plant of this type.

Keywords : Artificial neural network, cross correlation, dynamic neural network, feature selection, inflow forecast, multi-objective global optimisation, Nonlinear Autoregressive Exogenous (NARX); pond operation, run-of-river, time series forecasting, ,

DEDICATION

To my wife Anusha Priyadarshani and my children Laksandi, Sithuli and Senuk Wimalaratne throughout my study. Without their patience dedication this thesis would not have been completed in this short period of time. To my parents Nita and Rohana Wimalaratne , who nurtured me and educated me and showed me the right path and introduced me to the Library in a tender age.

ACKNOWLEDGEMENTS

Foremost, I am pleased to express my sincere gratitude to my supervisor Dr. Lidula N. Widanagama Arachchige of the Department of Electrical Engineering, University of Moratuwa for the continuous support for my MSc research, for thought-provoking discussions, constructive feedback, encouragement and guidance.

I would like to thank my external supervisor, Engineer W.J.L. Shavindranath Fernando, who gave me the initial thought for this research, who often stimulated interesting and enlightening discussions during the time at Upper Kotmale Project.

This work would have not been conceivable without the assistance and advice of many other people. I want to thank the two seniors in our Department of Electrical Engineering, University of Moratuwa, Professor J.R. Lucas and Professor H.Y. Ranjith Perera, whose ideas and thoughts had a great impact on my work.

I also would like to thank my colleagues and friends in our MSc batch of the campus, and those who are in my office at Upper Kotmale Power Station. You made my working in this research more inspiring.

I also wish to express my sincere respect to the chief priest, Ven. Nindane Chandawimala thero, at “Methmuni Viharaya”, Warakapola for giving me accommodation in the temple for me to seriously embark on the research work in a calm and tranquil surrounding away from home.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANN	: Artificial Neural Network
AOF	: Aggregated Objective Function
AW_RF	: Ambewela Rainfall
CD_RF	: Calidonia Rainfall
CD_WL	: Calidonia Water Level
CEB	: Ceylon Electricity Board
FF&WS	: Flood Forecasting & Warning System
IEEE	: Institute of Electrical and Electronic Engineers
IFM	: Inflow Forecasting Model
MAD	: Mean Absolute Deviation
MSE	: Mean Square Error
NARX	: Nonlinear Autoregressive eXogenous
NE_RF	: Nuwara Eliya Rainfall
NO_WL	: Nanuoya Water Level
PACF	: Partial Auto Correlation Function
POM	: Pond Optimisation Model
PS	: Power Station
R	: Correlation Coefficient
RMSE	: Root Mean Square Error
SCC	: System Control Centre
SH_RF	: Sandringham Rainfall
TK_RF	: Talawakelle Rainfall
TK_WL	: Talawakelle Water Level
UKPS	: Upper Kotmale Power Station

1 INTRODUCTION

1.1 Background

Upper Kotmale Power Station (UKPS), is a run of river hydropower station with no irrigation requirements as can be seen in other power stations on the Mahaweli river. Operation of the pond in Talawakelle presents a significant opportunity for the plant operation engineers to manage the pond in an optimised manner so as to maximise generation while minimising spilling. Furthermore, Upper Kotmale Power Station is currently considered to be a semi-dispatchable plant. Even during the peak times, the opening and closing of spillway gates are decided by the plant operation engineers although the plant is dispatched by System Control Centre. Table 1.1 show basic plant data.

Table 1.1: Basic Plant Data at UKPS **[1]**

Item	Value	Unit
Plant Capacity	150	MW
Effective Storage of Pond	0.8	MCM
Annual Expected Generation	409	GWh
Catchment Area	317	km^2
Annual Rainfall	2000	mm
1000-year flood	2000	m^3/s
10000-year flood	3000	m^3/s

Optimisation of Hydropower Stations can be carried out from three levels namely, unit Level, plant level, and system level. At system level, it is the unit dispatch based on lowest incremental cost that achieves optimisation for the whole generating system. Whereas unit level refers to running the unit at its best efficiency point, which consumes less discharge thus saving water. Managing the pond skilfully with minimum release of water while maximising generation is the plant level optimisation.

Accurate and reliable inflow forecast makes up an important basis for optimum pond operation connected with effective spillway gate operation. Pond operation is a complex problem which involves numerous requirements including flood control and

warning, power optimisation, downstream water users, etc. Often, it becomes more complex when it comes to a small pond as in Upper Kotmale Power Station. Moreover, pond operation without a proper mechanism is subjective in nature; consequently, optimum or near optimum operation is hardly achieved with only human intervention.

1.2 Problem Statement

There are five rainfall measuring stations (Nuwara Eliya, Ambewela, Calidonia, Sandringham, Talawakelle) and two water level gauging stations (Nanuoya, Calidonia) in the catchment of Upper Kotmale pond. The overall hydro-meteorological observation network of Upper Kotmale PS is shown in Figure 1.1. The data is real time telemetered to a computer in Main Control Room. Both inflow and telemetry data is available after commissioning since 2012.

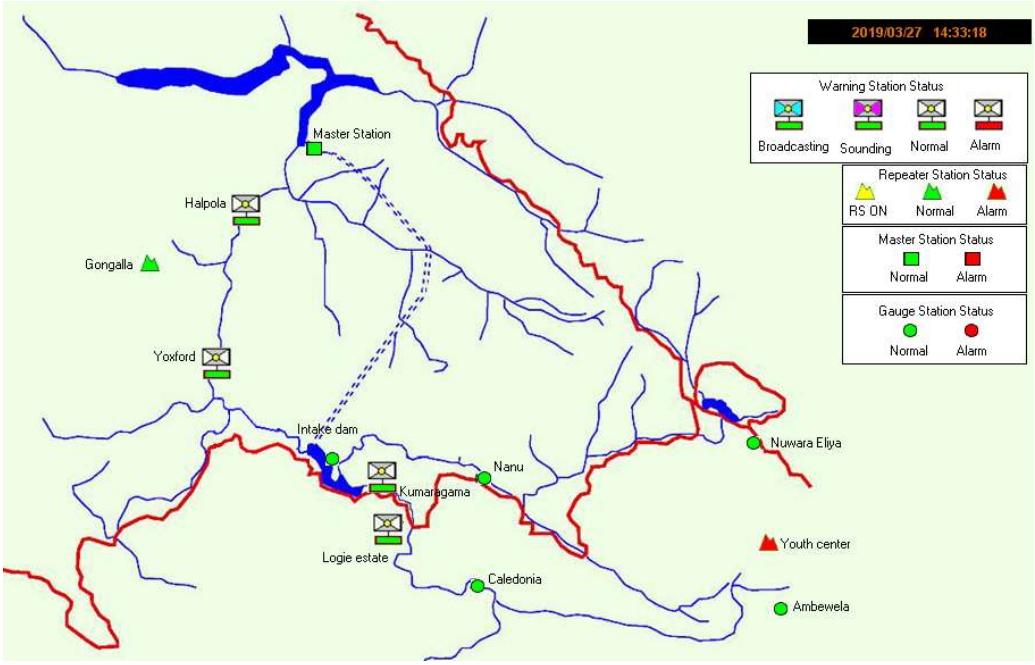


Figure 1.1: Hydro-Meteorological Observation Network of Upper Kotmale PS [2]

Present method of inflow forecasting uses hydrological modelling that involves many parameters which are dynamic in nature. Not only it requires field data such as river cross sections to be measured to calibrate the system regularly, but also it requires

detailed understanding of the underlying physical processes for such calibration. This exercise is both time consuming and costly. Hence, an alternative modelling technique is advantageous to model the system to get inflow forecast. At the same time, it will be possible to update with the data being collected with new model to make the forecast more accurate.

Reservoir at Upper Kotmale, being a small pond and not having gate operation rules, plant level optimisation made by the operator based on subjective judgements can cause unnecessary spilling; thus, wasting energy. Accurate inflow forecast will allow an effective spillway gate operation, thereby reducing spilling and use water effectively for power generation.

1.3 Objectives of the Study

The main objective of this study is to design an intelligent inflow forecast algorithm to improve the efficiency of power generation at Upper Kotmale Power Station by effective operation of spillway gates

Other specific objectives are as follows:

- To develop a correlation between rainfall forecast from reputed sources such as reliable websites and rainfall data recorded in the five rainfall gauging stations
- To analyse past operation data and develop a new model to forecast inflow to reservoir at Upper Kotmale
- To use inflow forecast to develop reservoir operation rules with the aim of minimising spill while effectively using water for power generation subject to other issues such as flood control.

1.4 Overall Model Development

In this research, two models were developed: Inflow Forecast Model using MATLAB Nonlinear Auto Regressive with eXogenous model (NARX) and Pond Optimisation Model using MATLAB Nonlinear multi-objective multivariable optimisation techniques. The outcome of the above inflow forecast model was utilised as the main

input to the pond optimisation model. The procedure and the methodology applied in this study is depicted in Figure 3.1.

Figure 1.2: Overall Methodology

1.5 Thesis Outline

Chapter 1 gives the background and problem statement of the research together with the objectives. It also gives a brief introduction to the overall model development. Chapter 2 describes the theory on development of Artificial Neural Network Model for inflow forecasting system and explains how to utilise the inflow forecast for managing of pond by optimisation. It also explains different feature selection methods available for input selection for ANN. It also describes different strategies available for multistep ahead forecasting. Chapter 3 provides the insight into development of ANN models for inflow forecasting for Upper Kotmale Hydropower Station; thus, the overall setting up of Inflow Forecasting Model (IFM). Feature selection and design of the neural network architecture for the models are described. Chapter 4 provides the mathematical formulation of Pond Optimisation Model (POM) using inflow forecast from the IFM described in Chapter 3. Chapter 5 presents the results of performance of MATLAB models of Inflow Forecasting Model and the performance of Pond Optimisation Model. Chapter 6 summarises the overall work done together with the energy gain in the study of pond optimisation using inflow forecasting; furthermore, it gives areas of future work.

2 LITERATURE REVIEW

2.1 The background

In a run-of-river hydropower plant like Upper Kotmale Power Station (PS) and Kukule Ganga[1] PS in Sri Lanka, where there are no irrigation requirements as seen in other power stations in Mahaweli river, the management of reservoir (pond) is done usually in "the way it is used to do approach" rather than using a scientifically optimised approach. Several studies suggest that reservoirs are managed using fixed or pre-defined rules, which are presented by way of tables and graphs. They further claim that these tables assist the operator in releasing water based on hydro- meteorological factors, the current level, and the time of the year [3]. However, these curves, which are designed by experience or trial and error, are not efficient. Furthermore, these rule curves are meant for long term operation and cannot be used for run-of-river type ponds where short term optimisation is important. Presence of trade-offs among hydropower generation, pond storage and downstream releases to maintain waterfall consistently pose pond management a multi-purpose problem.

A series of recent studies has indicated that short term optimisation of pond operation can be done with the forecasted information on inflows. To enhance the efficiency of pond operation, prior research has demonstrated how compelling the online operating systems are [4]. Accuracy and lead time are used as parameters to assess the quality of inflow forecasting. The accuracy is usually defined as the difference between observed and forecasted inflows and the lead time is the time interval after issuing the forecast to occur the forecasted event [5]. With the inflow forecasting assuming perfect accuracy forecasts, the benefits in terms of power generation increase with the extension of forecast lead time [6]. Previous studies have also emphasised that in spite of error in inflow forecast, it is best to utilise inflow forecasting [7].

[1] Kukule Ganga Power Station is another run-of-river hydropower station in Sri Lanka built on a tributary of the Kalu Ganga.

2.2 Inflow Forecasting Methods

2.2.1 Types of Models

A model is a set of software and tools to replicate a real physical system that is used to help predict its response and behaviour to new inputs.

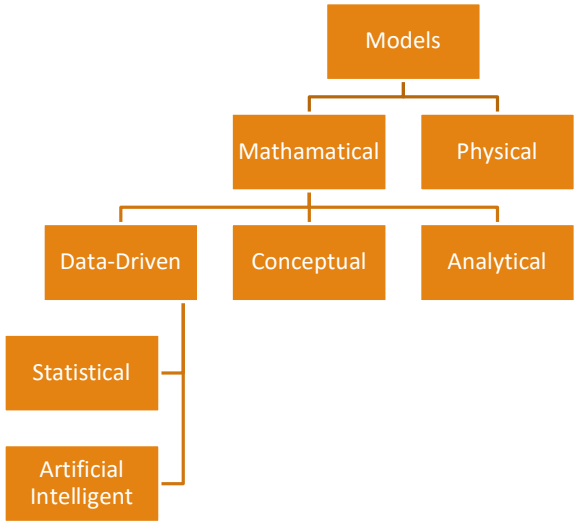


Figure 2.1: Different types of models

Figure 2.1 shows one simple classification of different types of models used for inflow forecasting in resources of water. A physical model is often a scaled down version of the original system. A mathematical model refers to models developed with equations and mathematical logic, that are used to simulate a system, and can be divided into three types as: Analytical, Conceptual and Data Driven models.

Analytical model describes a system by explicit mathematical equations. Application of analytical model is limited to the extent of the knowledge of mathematics behind the system and often applied for non-complex systems.

Models prepared using data-driven techniques utilise input/output data to discover patterns with the intention of generalising them to a larger set of data. Two types of models under data-driven types are statistical and artificial intelligence models. A conceptual model is a combination of data-driven model and analytical models. Statistical model is linked to a stochastic process, which includes both random and deterministic variables. Deterministic part is dealt with mathematical models and

random part is dealt with theory of probability and probabilistic modelling. Artificial Intelligence models include Fuzzy logic, and Artificial Neural Network that replicate real world system utilising biological concepts.

2.2.2 Artificial Neural Network (ANN)

Artificial Neural Network are models based on the structure of the human brain and are used for complicated problems of pattern recognition, clustering, regression etc. Any complicated nonlinear function can be mapped by them very easily, which is done intelligently by learning through training. It can be emphasised that ANN has the ability to learn the exact behaviour between the inputs and outputs from examples without any kind of the physical knowledge and physical involvement. ANN has been known as to recognise the fundamental behaviour between the variables although data is noisy and containing some errors [8].

The basic component of ANN is a neuron, a unit that acts two jobs of joining the inputs coming towards it (X) and comparing the joined inputs with a set threshold (α) to ascertain suitable output.

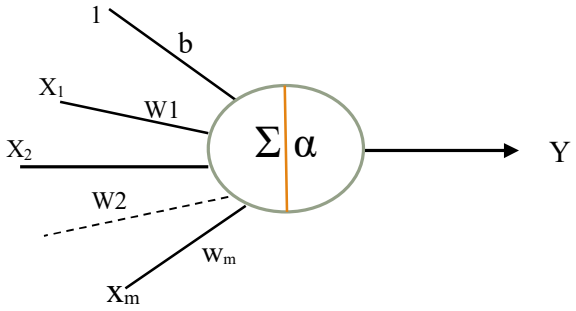


Figure 2.2: Structure of a Neuron

Figure 2.2 gives the neuron structure. The inputs to a neuron are weighted by a weight matrix, and it can have an additional bias input, too. The mathematical relation of a functional neuron is defined as in Equation (2.1) and (2.2).

$$I = W \times X + b \tag{2-1}$$

$$Y = \begin{cases} 1, & I \geq \alpha \\ 0, & I < \alpha \end{cases} \tag{2-2}$$

In the above equations, $\boldsymbol{X}$ = inputs, $\boldsymbol{W}$ = weight matrix, $\boldsymbol{b}$ = bias, $\boldsymbol{I}$ = sum of the weighted inputs, $\boldsymbol{\alpha}$ = threshold, $\boldsymbol{Y}$ = output

Processing element explained in Figure 2.2 is called a perceptron. Furthermore, a single neuron is not sufficient for solving numerous practical problems; hence, a network of perceptrons is commonly used in series or parallel, and it is called a neural network. The threshold value ($\boldsymbol{\alpha}$) in Figure 2.2 of the second half of the neuron can be substituted by a mathematical function to generalise the range of outputs which a neuron can output. It is called a transfer function, which connects an input to an output. Some of the transfer functions which are commonly in use with the artificial neurons are Linear, Log Sigmoid and Tangent Sigmoid.

The biases and weights are parameters of a network which shall be initialised before training ANN using a supervised approach, which will ultimately decide on the optimum weights and biases. The training in supervised mode of an ANN can be done using delta rule with backpropagation algorithm. Upon training, the network is ready to simulate the outputs for specific inputs associated utilising final derived biases and weights.

2.2.3 ANN Modelling Process

A step by step process was introduced to develop neural network (NN) models for hydrological applications in [9]. They are as follows:

1. Data selection: Collect appropriate and sufficient data
2. Choose appropriate predictors: Determine what to be modelled (inflow discharge)
3. Neural Net selection: Choose appropriate type of network and training algorithm
4. Data pre-processing: Feature selection to identify predictors and treat for missing data. Scaling the input before being fed.
5. Training the network
6. Use appropriate assessment criteria such as MSE.

2.2.4 Data Selection and Preparation

For the effective development of Neural Net models, adequate data shall be available. In other words, the data shall be of high quality free from errors and omissions and

available in adequate quantity. On the contrary, the presence of too many inputs types/features could lead to poor generalisation performance. If the data used has covered a broad range of more than one year, it is not required to eliminate any seasonal components from the data set and if the data selected periods are adjacent, it is not necessary to remove any long term trends or cycles. Some abnormal noise in the data can be filtered out by using an appropriate filter such as moving average filter or Butterworth filter. The Butterworth filter is more popular in ANN environment. Suitable tuning of cut off frequency (f_c) and sampling frequency (f_s) can be done by looking at the output using trial and error approach in MATLAB environment.

2.2.5 Feature Selection

Feature selection refers to the selection of appropriate inputs for the modelling of Neural Network model. There are different categories of inputs such as rainfall, water level, evaporation, their cumulative values and antecedent values of all. Not all inputs have an impact to the outputs; hence, including them arbitrary to the ANN training will weaken the generalisation performance of the ANN model. The following are three common methods for feature selection.

- Cross correlation Analysis
- Stepwise regression
- Use of Genetic Algorithms

In stepwise regression, all possible combinations of inputs are used to train an ANN network one after the other. Mean square error can be assessed for each combination and drop the combinations, which have no relation to output that giving very high MSE. This method never fails, but it consumes a lot of time making it difficult to apply for a practical case having many variables such as in inflow forecasting. Use of Genetic Algorithms also can speed up the process. However, it still consumes a lot of time for evaluation of fitness function. Therefore, cross correlation analysis is a suitable feature selection technique for most ANN applications. Several studies suggest that cross correlation techniques can be used to identify suitable antecedent values (lags) of inputs as well. In [8], cross correlation between water level, rainfall, and the lag time necessary for the system to respond was evaluated and the lag was identified as the

value producing a peak in the cross correlation diagram. It is required to check the cross correlation between each input and inflow (output) and partial autocorrelation of each input. The partial autocorrelation function measures the correlation between $\boldsymbol{y_t}$ and $\boldsymbol{y_{t+k}}$ after adjusting for the linear effects of $\boldsymbol{y_{t+1}, \ldots, y_{t+k-1}}$. The detailed cross correlation analysis was performed in this research to identify the suitable inputs and it is described in Chapter 3.

2.2.6 Data Preprocessing

Once the suitable inputs are selected, next step is to prepare the data before being fed to the ANN. Two processes frequently applied for data pre-processing are as follows:

- Standardising/Normalising
- Data Division

In standardising, the input values are rescaled to a uniformed scale. It can be [-1 1], [0 1] or [0.1 0.9]. MATLAB automatically does the standardising of inputs and it is not required to do explicitly. It is mathematically expressed by equation (2-3) [10]

$$X_n = X_{n\,min} + \frac{(X_o - X_{o\,min})}{(X_{o\,max} - X_{o\,min})} (X_{n\,max} - X_{n\,min}) \tag{2-3}$$

Where $\boldsymbol{X_o}$ and $\boldsymbol{X_n}$ denote the original and transformed data, whereas $\boldsymbol{X_{o\,max}}$ and $\boldsymbol{X_{o\,min}}$ denote the maximum and minimum values of original data, respectively. $\boldsymbol{X_{n\,min}}$ and $X_{n\,max}$ are the uniform scale defined previously, which in MATLAB is [-1 1].

Before training ANN, the data is divided into three subsets namely training set, validation set and test set. The training set is used for computing the gradient and updating the network weights and biases while validation set is utilised to monitor the error when the training is in progress. The validation error normally decreases during the training together with training set error. Conversely, when the network starts to overfitting, the error on validation data set starts to rise. At this point, the weights and biases of the network are saved right at the lowest error on the validation data set. Overfitting of network should be avoided in this way, and it is very important in ANN modelling. In overfitting, ANN tries to memorise the training samples and not learn

the underlying pattern. This technique is also called early stopping in some literature. Figure 2.3 shows an extract from MATLAB documentation, which describes how MATLAB avoids overfitting of ANN.

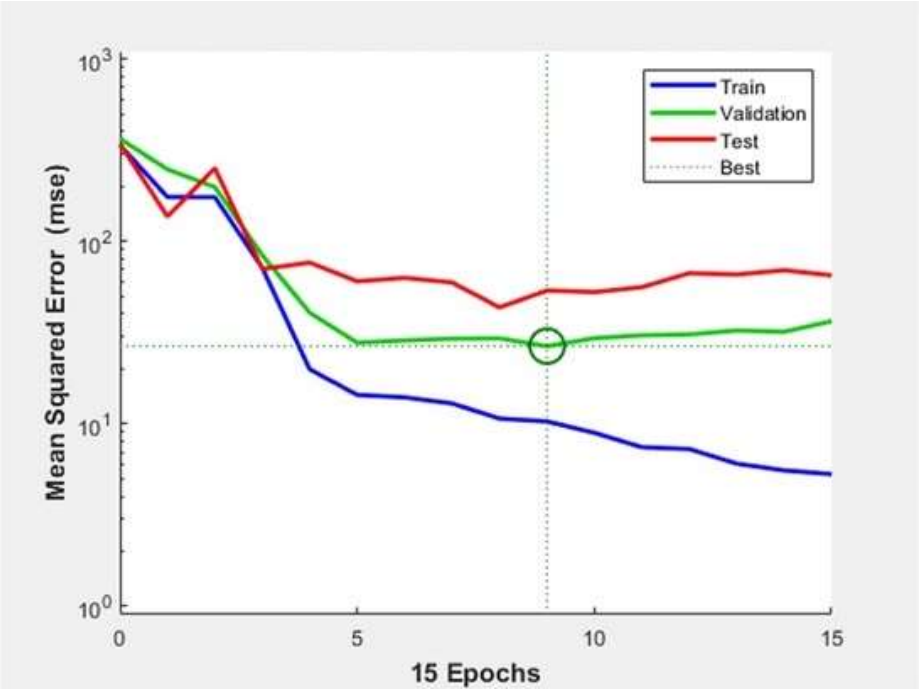


Figure 2.3: Extract from MATLAB Documentation of Overfitting

As shown in Figure 2.3, the error decrease after more iterations (epochs) of training, but start to increase on the validation data set from 9th epoch, at which time the network starts overfitting the training data. In the MATLAB setup, the training ceases after six consecutive increases in validation error, and the best performance is taken from the 9th epoch, which has the lowest validation error [11]. The test set is used to verify the performance of the model and different models are compared. As in the Figure 2.3, it is useful to plot the test set error during the training process. If the error on the test set reaches a minimum at a significantly different iteration number (epoch) than the validation set error, it might indicate a poor division of the data set. In general, data is divided as 70%, 15% and 15% for training, validation and test sets, respectively. It can be 80%, 10% and 10%, too. Data division depends on number of data points available. Higher the number of data points available, even a small percentage allocated to test error is a significant number of samples. It is very important that data division is done

in a manner that protects the patterns and relationships of time series; hence, random data division techniques were not be applied for inflow forecasting model in this research.

2.2.7 ANN Architecture

Figure 2.4 show a simple neural network with one hidden layer. It has the input and output layers. This network is called a two-layer network as number of layers are calculated including output layer for a ANN.

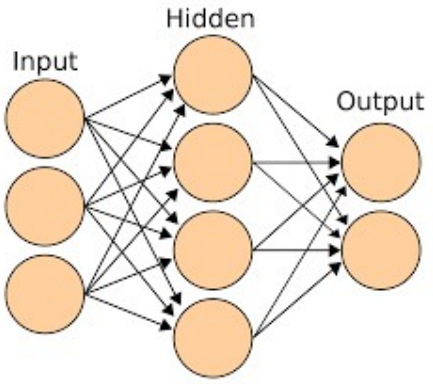


Figure 2.4: Simple neural network with one hidden layer

Input layer has three nodes, hidden layer has 4 nodes and output layer has two nodes. A heuristic approach is used in [12] to set the hidden number of nodes and to decide the final structure of the neural network. According to some literature, the number of hidden nodes be taken as the half of the input nodes [13]. Often two-layer network is suitable for most problems except for very complex models, in which more than one hidden layer is used. If the number of hidden layers is very high, then the network is said to be deep neural network; however, for the inflow forecasting, shallow neural network is sufficient. The weights shall not exceed the training samples, and a ratio of training sample to weight number up to 30 is necessary to obtain good generalisation [14].

2.2.8 Training of ANN

Neural networks are trained with a selected data set and it is generally assumed that a network does not have any a prior-knowledge about the problem before it is trained [15]. At the start of the training, the weights initialisation takes place with a set of

random values and then the weights are systematically modified by the learning algorithm with the aim of minimising, for a given input, the difference between the actual output and output of ANN. The process is terminated when this difference becomes smaller than the set values as the learning samples are repeatedly presented to the network. At this stage, the ANN is supposed to have been trained and the resulting weight vector of the properly trained network carries its knowledge about the problem. The three main training methods are :

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

For Supervised Learning, both inputs and corresponding outputs are fed to the networks. In Unsupervised learning, learning is by clustering technique and the expected output is not known beforehand and Reinforcement learning is a combination of both unsupervised learning and supervised learning. It has penalty for wrong outputs and rewards for correct outputs [11]. In many applications supervised learning is chiefly used.

The most popular method of network training algorithm has been the back-propagation learning rule [16]. Nevertheless, back-propagation has a few drawbacks including long duration of training with many iterations. Owing to the drawbacks of back-propagation, some authors have proposed more efficient rules such as the Levenberg-Marquardt rule, which is the most powerful method at present and reach to one of the best solutions in a few iterations [17].

2.2.9 Nonlinear Autoregressive EXogenous Model (NARX)

Prior knowledge of the system to be modeled will determine the choice of the ANN. In this sense, inflow forecasting being a time series and NARX neural network being a good predictor of time series, it is suitable for inflow forecasting modelling [18]. NARX could be used to model various kinds of non-linear dynamic systems. They are used in numerous applications that include time series applications [19]. The NARX is a recurrent dynamic neural network. Dynamic networks have memory in it, and they can be trained to learn time-varying or sequential patterns; hence, they are more

powerful than the networks of static counterparts. Furthermore, these networks have feedback connections across multiple network layers. To derive the full potential of NARX for time series forecast, it is useful to use its memory capability using the antecedent of true or predicted time series. There are two distinct architectures for the NARX model, open loop (series parallel) and closed loop (parallel architecture). Their equations are as shown in (2-4) and (2-5) respectively.

$$\hat{Y}(t+1) = F\begin{Bmatrix} y(t), y(t-1), \ldots . y(t-n_y), x(t+1), \\ x(t), x(t-1), \ldots x(t-n_x) \end{Bmatrix} \quad (2\text{-}4)$$

$$\hat{Y}(t+1) = F\begin{Bmatrix} \hat{y}(t), \hat{y}(t-1), \ldots . \hat{y}(t-n_y), x(t+1), \\ x(t), x(t-1), \ldots x(t-n_x) \end{Bmatrix} \quad (2\text{-}5)$$

Where function F is the network mapping function, $\widehat{\boldsymbol{Y}}(\boldsymbol{t+1})$ is the NARX output at the time t for the time t +1 (i.e. predicted value of Y for the time t+1). $\widehat{\boldsymbol{y}}(\boldsymbol{t}), \widehat{\boldsymbol{y}}(\boldsymbol{t-1}), \ldots . \widehat{\boldsymbol{y}}(\boldsymbol{t-n_y})$ are the past outputs of the NARX. $\boldsymbol{y}(\boldsymbol{t}), \boldsymbol{y}(\boldsymbol{t-1}), \ldots . \boldsymbol{y}(\boldsymbol{t-n_y})$ are the true past values of the time series, called also the desired output values. $\boldsymbol{x}(\boldsymbol{t}), \boldsymbol{x}(\boldsymbol{t-1}), \ldots \boldsymbol{x}(\boldsymbol{t-n_x})$ are the inputs of the NARX. n_x is the number of input delays and $\boldsymbol{n_y}$ is the number of output delays. If only series parallel architecture is used, it can forecast one-step ahead time series forecasting and for multi-step ahead forecast, different strategy shall be used.

2.2.10 Multistep Time Series forecasting Strategies

In time series prediction, forecasting the next time step is generally common, and it is called a one-step forecast. In the case of multiple time steps to be forecasted, it is called multi-step forecasting. The four common methods for multi-step forecasting [20].

1. Direct Multistep Forecasting
2. Recursive Multistep Forecasting
3. Direct Recursive Hybrid Multistep Forecasting
4. Multiple Output Forecasting

Direct Multistep forecasting requires to develop a separate model for every time step. In case of forecasting inflow for the next six hours, for 30 minute time steps, it is

required to develop 12 models. Equation (2-6) and (2-7) underlines the relationship of this strategy.

$$Forecast\ (t+1) = model1(obs(t-1), obs(t-2), ..obs(t-n)) \quad (2\text{-}6)$$

$$Forecast\ (t+2) = model2(obs(t-2), obs(t-3), ..obs(t-n)) \quad (2\text{-}7)$$

To have one model for every time step is an additional burden in view of computation and maintenance, particularly, as the timesteps to be forecasted increases.

Recursive Multistep Forecasting involves using a one step ahead model multiple times where the forecast for the prior time step is used as an input for making forecast on the following time step. Equation (2-7) and (2-9) underlines the relationship of this strategy.

$$Forecast\ (t+1) = model(obs(t-1), obs(t-2), ..obs(t-n)) \quad (2\text{-}8)$$

$$Forecast\ (t+2) = model(Forecast(t+1), obs(t-1), ..obs(t-n)) \quad (2\text{-}9)$$

In this case, in place of observations, forecasts are used. The recursive strategy let forecast errors to accumulate so that performance quickly degrades as the forecast time horison increases.

In Direct Recursive Hybrid Multistep Forecasting, direct and recursive strategies are combined to offer benefits of both methods. A separate model is constructed for each timestep to be forecasted, but each model use the forecasts made by models at prior time steps as input values. Equation (2-10) and (2-11) explains the relationship of this strategy.

$$Forecast\ (t+1) = model1(obs(t-1), obs(t-2), ..obs(t-n)) \quad (2\text{-}10)$$

$$Forecast\ (t+2) = model2(Forecast(t+1), obs(t-1), ..obs(t-n)) \quad (2\text{-}11)$$

Combining the recursive and direct strategies can help overcome the limitations of each.

Multiple Output Forecasting involves developing one model that is capable of forecasting the entire forecast sequence in a one-shot manner. Equation (2-12) depicts the relationship of this strategy.

$$Forecast\ (t+1), Forecast\ (t+2) = model(obs(t-1), obs(t-2), ..obs(t-n)) \quad (2\text{-}12)$$

Multiple output models can learn the dependence structure between inputs and outputs as well as between outputs; hence, these models are more complex. Which strategy is better for model is to be decided based on trial and error or by judgement.

2.2.11 Performance of ANN model

The performance of ANN model can be evaluated using statistical comparisons of predicted and observed outputs. One common such statistic is Mean Square Error (MSE). It is defined as in equation 2-13.

$$MSE = \frac{\sum_i^n((obs_i - for_i)^2)}{n} \quad (2\text{-}13)$$

Where obs_i is i[th] observed data and $\boldsymbol{for_i}$ is the i[th] forecast data and $\boldsymbol{n}$ is the number of observed values. Root Mean Square (RMSE) is the square root of MSE.

2.3 Pond Optimisation Methods

Mathematical Optimisation refers to a process of maximising or minimising objectives without violating design constraints, and regulating a set of decision variables that affect both the objectives and the design constraints [21].

2.3.1 Categories of Optimisation

There exists a considerable body of literature on the categories of optimisation. They can be classified as follows:

1. Nonlinear vs Linear
2. Unconstrained vs Constrained
3. Continuous vs Discrete
4. Multobjective vs Single
5. Multiple vs Single Minima
6. Non-deterministic vs Deterministic
7. Simple vs Complex

If the objective function or any of the constraints is a nonlinear function of the design variables, then the problem is called a nonlinear optimisation problem. If the optimisation problem has constraints, then it is called a constrained optimisation problem. If any design variable is discrete, it is called discrete optimisation, and if this discrete variable can have only one or zero then the problem is binary optimisation problem whereas design variables can have any integer, then it is an integer programming problem. Optimisation problem having several optima is referred to as

global optimisation problem. If the design variables are nondeterministic, then the problem is nondeterministic optimisation, and if the optimisation problem can be solved easily as it may be nonlinear and deterministic etc., then the problem is simple optimisation problem.

2.3.2 Multi-objective Optimisation Problem

In general, most practical problems involves trade-offs among competing objectives. Maximising power generation while maximising end pond storage together with minimising spilling is a Multi objective optimisation problem.

Several objectives are optimized simultaneously in Multiobjective optimization. The objectives are often in conflict with each other and are measured by different units. Therefore, the most important component of multiple objective problem solving is how to evaluate solutions or parameter sets, when there are two or more performance measures [22].

The structure of multi objective optimisation problem can be defined referring to two objective case (n=2) as given by (2-14) - (2-16).

$$min_x \ [\mu_1(x) \ \ \mu_2(x) \ ] \tag{2-14}$$

Subject to

$$g(x) \leq 0 \\ h(x) = 0 \tag{2-15}$$

$$x_l \leq x \ \leq x_u \tag{2-16}$$

$\boldsymbol{x}$ is the design variable vector, $\boldsymbol{g(x)}$ represents the vector of inequality constraints and $\boldsymbol{h(x)}$ is the vector of equality constraints.

The solution to the Multi-objective problem can be found from two different ways: Using Aggregated Objective Function (AOF) Method and Pareto Domination Approach. The AOF method combines different objectives into a single objective using weightages. Pareto approach is based on whether one solution is dominated by other and not on a single comparative value. Trade off among different objectives is presented by the set of Pareto optimal solutions. For each solution in this pareto solution set, an improvement in one objective cannot be achieved without compromising others.

In this research, weighted approach is used for optimisation of the pond. Maximising power generation while maximising end-pond storage at the end of period are conflicting objectives. The aggregate objective function for the two objective case can be written as in (2-17).

$$J(x) = w1 \times \mu_1(x) + w2 \times \mu_2(x) \tag{2-17}$$

Where $w1 + w2 = 1$.

MATLAB fmincon command can be used to find the solution to this multi-objective optimisation problem after converting to single objective optimisation problem with weighted approach. MATLAB fmincon command is as in (2-18) [11]:

$$\begin{aligned} & min_x\,[f(x)\,] \\ \text{Subject to} \quad & \\ & c(x) \leq 0 \\ & ceq(x) = 0 \\ & Ax \leq b \\ & Aeq\ x = beq \\ & LB \leq x\ \leq UB \end{aligned} \tag{2-18}$$

First two constraints are nonlinear constraints and next two constraints are for linear constraints. Final constraint defines the upper and lower bounds. The syntax for fmincon is as follows [11].

$$[xopt, fopt]\ =\ fmincon(\text{'}fun\text{'}, x0, A, b, Aeq, beq, LB, UB, \text{'}nonlcon\text{'}) \tag{2-19}$$

Where ***x0, A, b, Aeq, beq, LB***, and ***UB*** are the input variables which need to be defined before calling 'fmincon' function. 'fun' is the name of the function file containing the definition of **f(x),** and 'nonlcon' is the name of the function file containing nonlinear constraints if any. The variables xopt and fopt are the outputs of fmincon, where xopt is the optimum vector of variables [x_1, x_2] and fopt is the minimum value of objective function f. To maximise a function, it is required to simply perform operation of minimisation of **-f(x).**

2.3.3 Global Optimisation

Pond optimisation may have global minima and local minima; hence, it is called a multi-modal optimisation problem. The global optimisation is to find the global optima. Most of the algorithms in optimisation do not guarantee finding global minima as optimisation can easily converge to a local minima based on the starting point used.

GlobalSearch and MultiStart algorithms of MATLAB have similar approaches to finding global or multiple minima. Both algorithms start a local solver (such as fmincon) from multiple start points. GlobalSearch uses the scatter search algorithm to generate a set of trial points. GlobalSearch generates trial points within any finite bounds set within LB and UB [11]. Several authors have recognised MATLAB GlobalSearch to find the global minimum. However, in all the cases, global minimum is not guaranteed but somewhat closer. Figure 2.5 shows MATLAB peaks function which demonstrates the both local and global minimum.

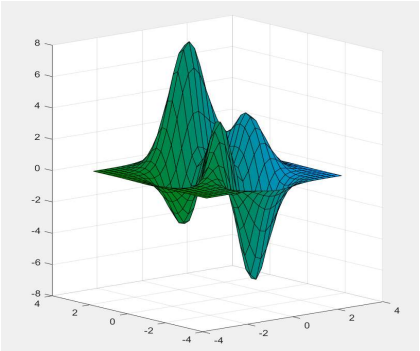


Figure 2.5: MATLAB peak function

3 INFLOW FORECASTING USING NEURAL NETWORK

3.1 Introduction

Figure 3.1 shows the overall diagram of Inflow Forecasting Model. First, raw data was screened through a process called Feature Selection through which the final inputs to the model were derived. Using the derived inputs and the output of inflow MATLAB NARX network was trained using supervised learning approach. Finally, trained network was used to forecast the inflow one step ahead. One step ahead means 0.5 hours ahead. Several of similar models trained utilised in a Direct Recursive Strategy to forecast the inflow for the multiple time steps ahead. The whole procedure of the setting up of the inflow forecast model is discussed next.

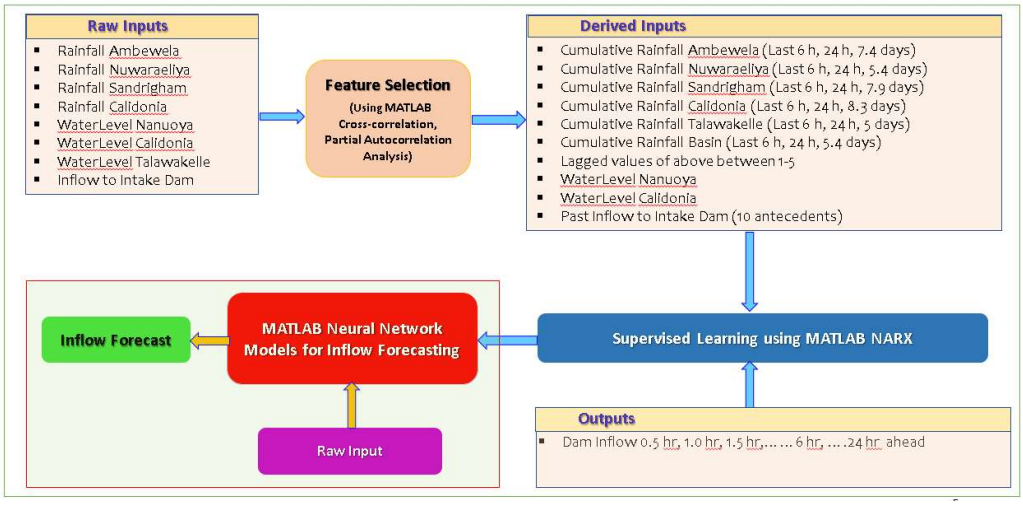


Figure 3.1: Overall Block Diagram of Inflow Forecast Model

3.2 Data Collection and Pre-processing

Routine rainfall and water level observations are made by UKPS as a part of Flood Forecasting & Warning System (FF&WS) established in the Power Station. The Power Station was commissioned in year 2012, but FF&WS was established much later in year 2014. UKPS operates a hydro-meteorological network of seven meteorological stations where rainfall and water level data are transmitted real time to the Main Control Room of the Power Station at every 10 minutes interval. Various calibration and model setups had been taken place prior to 2016 before taking over the system

from the contractor. Hence, analysis and assessment indicate that the data before 2016 are not of good quality.

3.2.1 Raw Data Selection in the Present Study

It is important to select adequate data which are specific for the modelling task. On the contrary, limited data shall be selected as much as feasible to minimize the training time and potential for overfitting of the model. Rainfall, water level and actual recorded inflow time series data from year 2016 to 2019 were collected at 30-minute interval. Figure 3.2 shows an extract of collected raw data from the Upper Kotmale Power Station.

	A	B	P	T	X	AB	AF	AJ	AN	AO	AP
1	Time	Q	NE_RF	AW_RF	SH_RF	CD_RF	TK_RF	Basin_RF	NO_WL	CD_WL(t)	TK_WL(t)
1287	10/13/17 18:00	14.9	1	1	0	1	0	0.53	1312.72	1264.78	1191.25
1288	10/13/17 18:30	15.5	0	0	0	9	5	1.89	1312.74	1264.81	1191.07
1289	10/13/17 19:00	15.8	5	7	0	5	2	2.12	1312.78	1264.84	1190.87
1290	10/13/17 19:30	15.8	3	1	0	3	0	1.27	1312.85	1264.88	1190.67
1291	10/13/17 20:00	15.8	1	1	0	1	1	0.59	1312.95	1264.92	1190.47
1292	10/13/17 20:30	16.1	0	0	0	0	1	0.22	1313.05	1264.95	1190.28
1293	10/13/17 21:00	17.1	1	1	0	1	0	0.53	1313.12	1264.98	1190.18
1294	10/13/17 21:30	18.9	0	1	0	1	4	0.54	1313.15	1265.00	1190.17
1295	10/13/17 22:00	21.0	1	0	0	0	1	0.43	1313.15	1265.03	1190.25
1296	10/13/17 22:30	22.6	0	0	0	1	0	0.32	1313.14	1265.04	1190.36
1297	10/13/17 23:00	23.6	1	2	0	4	2	1.13	1313.13	1265.06	1190.51
1298	10/13/17 23:30	24.1	1	1	0	0	1	0.43	1313.13	1265.07	1190.66
1299	10/14/17 0:00	24.3	0	0	0	1	2	0.43	1313.14	1265.09	1190.81
1300	10/14/17 0:30	24.6	5	0	0	1	6	1.70	1313.18	1265.10	1190.95
1301	10/14/17 1:00	25.0	0	1	0	1	0	0.32	1313.22	1265.10	1191.10
1302	10/14/17 1:30	25.5	0	1	0	0	1	0.22	1313.27	1265.10	1191.24
1303	10/14/17 2:00	26.0	0	0	0	0	0	0.16	1313.29	1265.11	1191.39
1304	10/14/17 2:30	26.3	0	0	0	0	0	0.16	1313.28	1265.11	1191.54
1305	10/14/17 3:00	26.4	1	0	0	2	1	0.75	1313.26	1265.11	1191.69

16_17_18

Figure 3.2: Extract of Collected Raw Data

Gauging stations, from which data was collected as in Figure 3.2, are further summarised in Table 3.1.

Table 3.1: Summery of Gauging Stations and Raw Data Sources

Station Code	Description	Unit
NE_RF	Nuwara Eliya Rainfall	mm
AW_RF	Ambewela Rainfall	mm
SH_RF	Sandringham Rainfall	mm
CD_RF	Calidonia Rainfall	mm
TK_RF	Talawakelle Rainfall	mm
Basin_RF	Basin Average Rainfall	mm
NO_WL	Nanuoya Water Level	masl
CD_WL	Calidonia Water Level	masl
TK_WL	Talawakelle Water Level	masl
Q	Inflow Discharge to Pond	m^3/s

Basin_RF, as referred in Table 3.1, is the total average rainfall of the Upper Kotmale Catchment calculated based on Theisen polygon average, and it is readily available.

3.2.2 Outliers and Missing Data

Data was plotted on a MATLAB time-plot that contains missing values and outliers, then gaps appeared on the plot where missing data existed, and outliers were easily recognisable. Furthermore, most the missing data and outliers of input raw data were treated manually with the author's engineering judgement. The inflow discharge was also applied a Butterworth filter (fc = 10Hz, fs = 100Hz, Step 2) to compensate noise errors in the inflow values particularly when low inflow discharges were calculated.

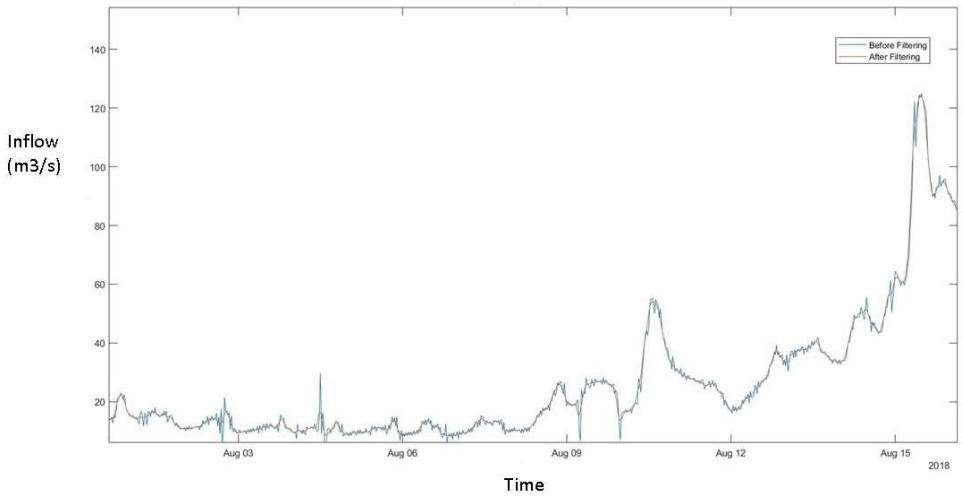


Figure 3.3: Low-Pass Butterworth Filter used for Inflow Discharge to remove noise

Figure 3.3 shows an extract from the result of inflow discharge vs time before and after applying low-pass Butterworth Filter in MATLAB.

3.3 Feature Selection

This part of the study could be considered as one of the most important and challenging process. As described in the Chapter 2: Literature Review, feature selection refers to the selection of appropriate inputs for the modelling of Neural Network based Inflow Forecast Model. The candidate data were present rainfall values of five rainfall gauging stations ($\mathbf{P_t}$), their Antecedent Rainfall Values ($\mathbf{P_{t-1}, P_{t-2},..}$), Present Water Levels of three water Level gauging stations ($\mathbf{L_t}$), their Antecedent values ($\mathbf{L_{t-1}, L_{t-2},..}$), Present Inflow discharge ($\mathbf{Q_t}$) and its Antecedent values ($\mathbf{Q_{t-1}, Q_{t-2},..}$).

The steps given next were followed in the feature selection process.

Step 1: Correlation Coefficient between each combination of the present values of input variables were calculated using MATLAB correlation matrix. Figure 3.4 shows the results of correlation coefficients (**R**) between each. Present value of rainfall means that rainfall in mm recorded between now and half hour before.

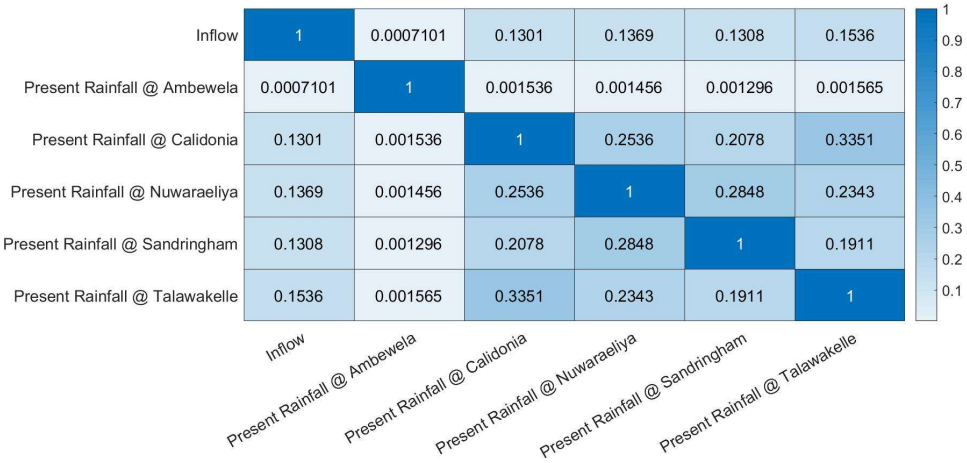


Figure 3.4: Correlation Matrix of Present Values of Input Data

The correlation matrix above suggests that there is no significant correlation between the present values of selected inputs. Moreover, the first column of this matrix gives the correlation of present rainfall of the rainfall gauging stations and the present inflow to the pond. It is also clear that inflow to the pond at any given time is not due to the

rainfall at that time but the rainfall that had received before that time. With the conclusion that present values of rainfall have no bearing on the present inflow; hence, present values of rainfall were not considered as inputs to the Neural Network model. Similarly, present values of Water Levels at Calidonia and Nanuoya also have no effect on present values of inflow; therefore, they, too, were not considered as inputs.

Step 2: Next step was to see if any correlation between antecedent values of rainfall and present inflow. In here, antecedent values refer to the point rainfall values recorded in the past. For example: rainfall before 0.5 h (t-1), 1.0 h (t-2), 1.5 h (t-3), etc. MATLAB programme was written to calculate 200 lags (1 lag = 0.5 hours) and to see any cross-correlation between the present inflow against each rainfall input. The cross correlation of present inflow and lagged values of Nuwara Eliya Rainfall is shown in Figure 3.5. The cross correlation of inflow and lagged rainfall values for each of the other rainfall gauging station are shown in Figure 3.6. One lagged rainfall means that rainfall recorded between half an hour before and one hour before. It is clear from the Figure 3.5 & 3.6 that even point values of lagged values of rainfall have no bearing with the present inflow as there is no significant cross-correlation in any of the graph in Figure 3.5 & 3.6.

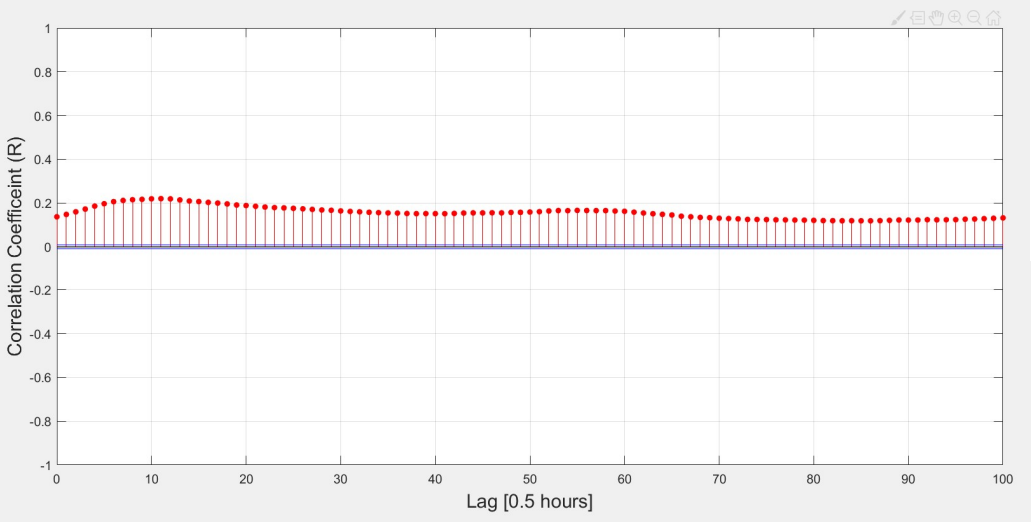


Figure 3.5: Cross Correlation between Present Inflow and Lagged Values of Nuwara Eliya RF

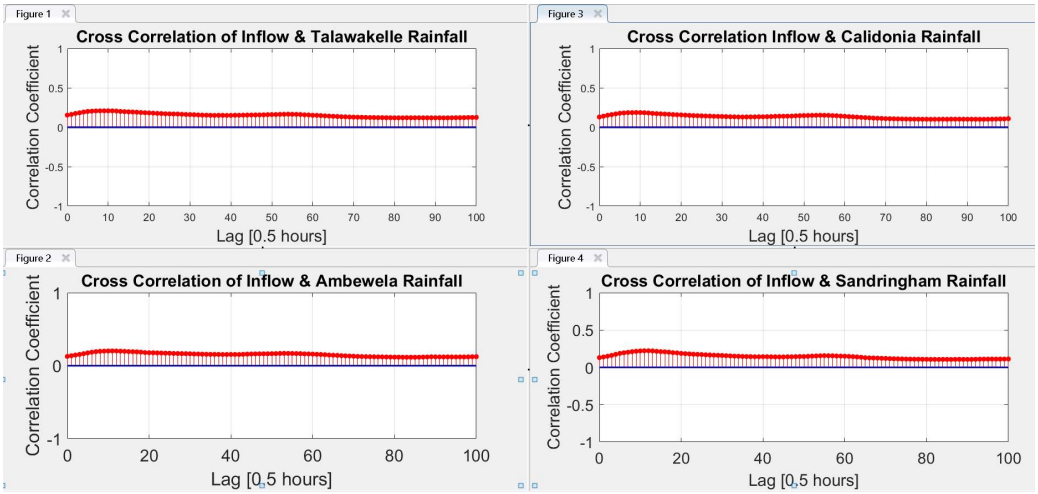


Figure 3.6: Cross Correlation between Inflow and Rainfall Values

Hence, it was concluded not to consider as inputs to the neural network model even the point lagged values of rainfall.

Step 3: Next step was to see if any correlation between antecedent values of water levels at Nanuoya and Calidonia with present inflow. Figure 3.7 shows the cross-correlation between present inflow and up to two days lagged values of Calidonia water level while Figure 3.8 shows the cross-correlation between present inflow and up to 10 hours lagged values of Nanuoya water level. It can be seen a significant correlation between lagged point values of Nanuoya water level and present Inflow. The lag time of the river in Upper Kotmale varies between 4 to 6 hours.

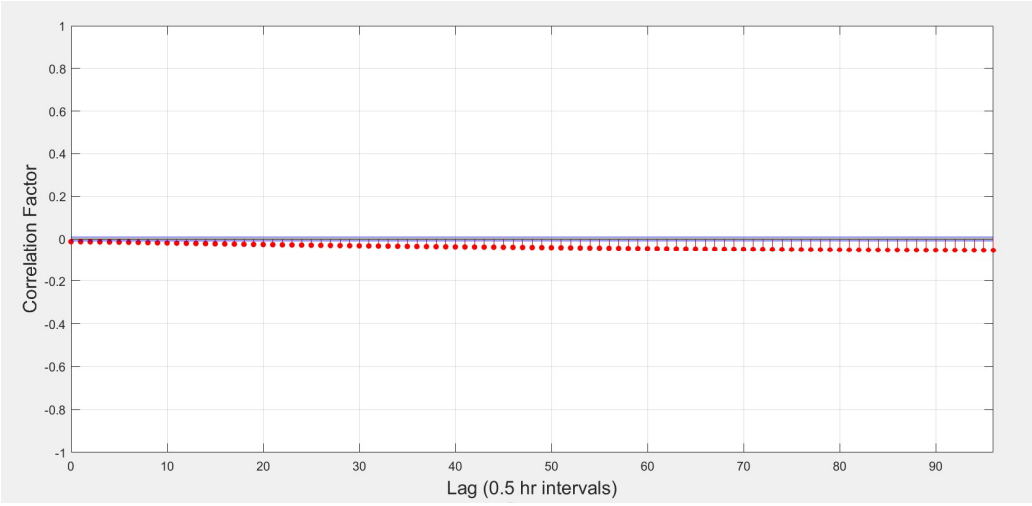


Figure 3.7: Cross-Correlation between Present Inflow and Lagged Values of Calidonia Water Level

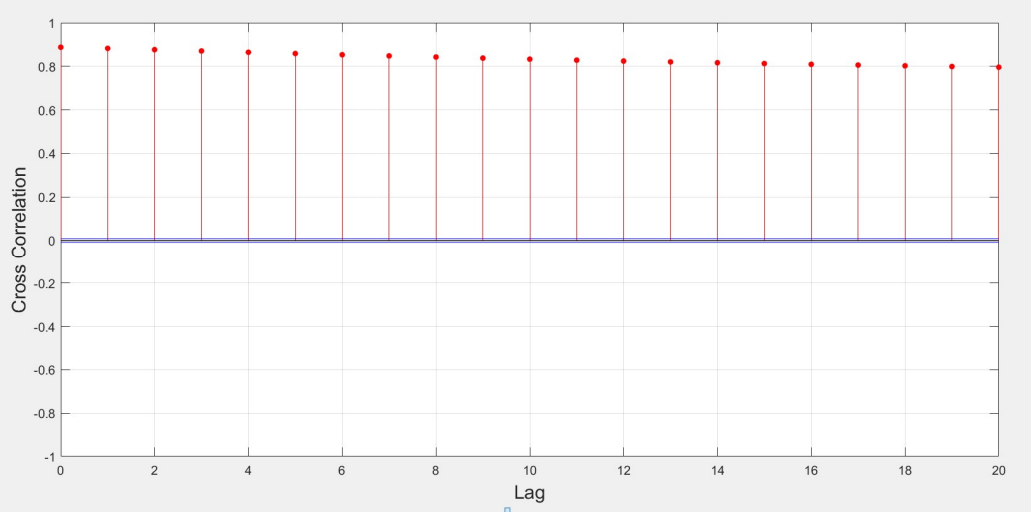


Figure 3.8: Cross-correlation between Present Inflow and Nanuoya Water Level

In order to ascertain, how many lagged values of Nanu Oya water level have the significant information, partial autocorrelation of Nanuoya water level was analysed using MATLAB as shown in Figure 3.9. Up to two lags need to be considered from Nanuoya water level.

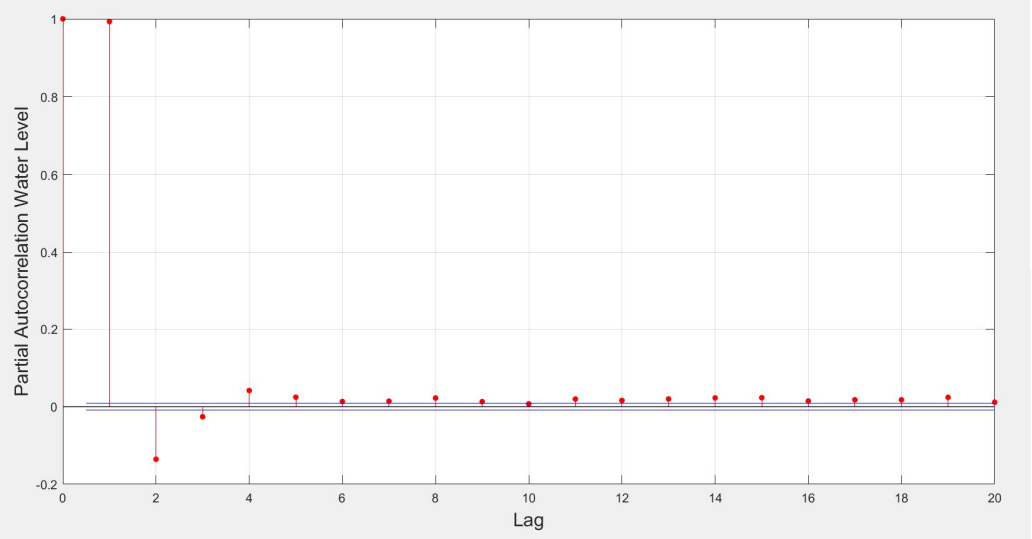


Figure 3.9: Partial Autocorrelation of Nanuoya Water Level

Step 4: With regard to rainfall, failing to spot correlation for present values of rainfall or lagged values of point rainfall with present inflow, next step was to see if any cross-correlation available between present inflow and cumulative values of rainfall of each

station. Figure 3.10 shows the cross-correlation matrix among present inflow and cumulative rainfall values. The values in first column gives the cross-correlation between present inflow and cumulative rainfall of Nuwara Eliya for different cumulative time steps.

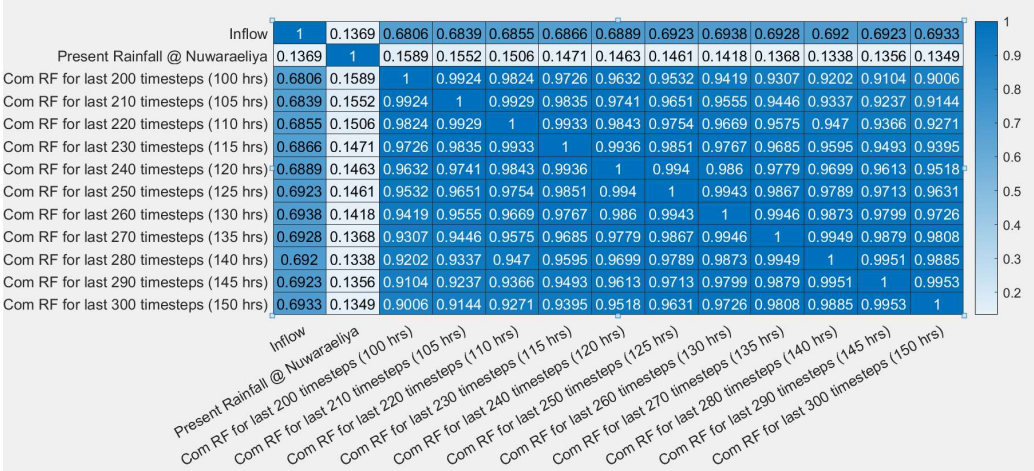


Figure 3.10: Cross Correlation among Nuwara Eliya Cumulative Rainfall(up to 300 timesteps) and Present Inflow

The highest correlation value, as in Figure 3.9, is 0.6938, and it belongs to the cumulative Nuwara Eliya rainfall for last 260 timesteps (130 hours). This means, the present inflow is highly correlated with the cumulative Nuwara Eliya rainfall for the last 5.4 days. Hence, it is taken as an input to the Neural Network model. Furthermore, similar observations were done for the other rainfall gauging values. In addition to each individual cumulative rainfall values, effect of cumulative basin average was also analysed in a similar way. Accordingly, the following rainfall variables were selected for the modelling.

- 5.4 days cumulative Nuwara Eliya rainfall
- 7.6 days cumulative Ambewela rainfall
- 7.9 days cumulative Sandringham rainfall
- 8.3 days cumulative Calidonia rainfall
- 5.0 days cumulative Talawakelle rainfall
- 5.4 days cumulative Basin average rainfall

It is clear that not the point rainfall, or lagged values of point rainfall that has an effect on inflow but the cumulative values of rainfall.

Moreover, partial autocorrelation of each of the above rainfall inputs were considered to identify the effect of antecedent cumulative rainfall values on the present cumulative values. Figure 3.11 shows the partial autocorrelation of 5.4 days cumulative Nuwara Eliya rainfall.

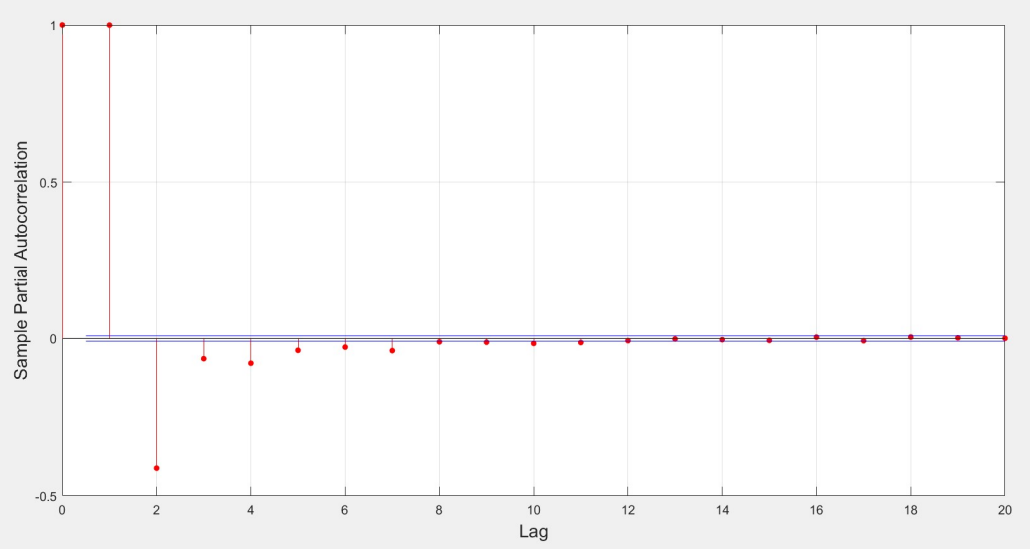


Figure 3.11: Partial-Autocorrelation of 5.4 days (260 steps) Comulative Nuwara Eliya Rainfall

Two antecedent values of this variable carry significant information content as seen in Figure 3.11. Similar analysis done for the other cumulative rainfall variables selected in this step. Furthermore, NARX model in MATLAB does not permit configuring different antecedent values for different input variables; by analysing all other graphs of other 11 models, a common 5 antecedent values were selected for all cumulative rainfall variables. In other words, if V is equal to "5.4 days cumulative Nuwara Eliya rainfall", $\mathbf{V_{t-1}, V_{t-2}, V_{t-3}, V_{t-4}, V_{t-5}}$ were also considered as inputs.

Step 5: Next, the analysis was done to identify the effect of present inflow with its antecedent values and up to how many antecedent values were to be considered. For this, partial autocorrelation was carried out as shown in Figure 3.12.

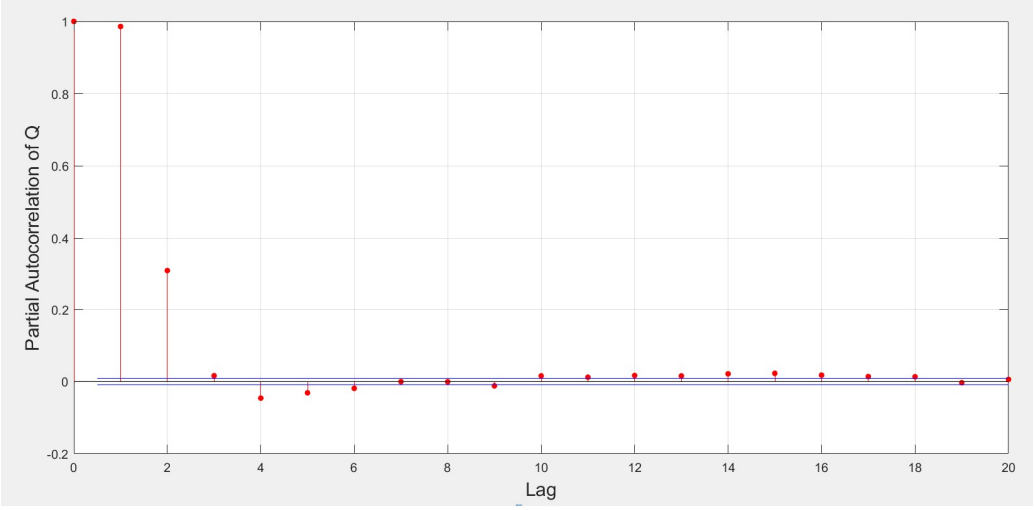


Figure 3.12: Partial autocorrelation of Inflow to the Pond

Figure 3.12 indicates that present inflow is correlated to the past inflows. Hence, at least, two antecedent values of inflows need to be considered as inputs to the model.

Step 6: Another user defined variable was created as flag to indicate the range of present inflow to the model. Table 3.2 shows values of the user defined inflow flag variable.

Table 3.2: User Defined Inflow Flag

Inflow, Q (m^3/s)	**flag**
$Q < 10$	1
$10 \leq Q < 20$	2
$20 \leq Q < 30$	3
$30 \leq Q < 40$	4
$40 \leq Q < 50$	5
$50 \leq Q < 60$	6
$60 \leq Q < 70$	7
$70 \leq Q < 80$	8
$80 \leq Q < 90$	9
$90 \leq Q < 100$	10
$100 \leq Q$	11

Step 7: In addition to the variables selected in the previous steps, 6-hour and 24-hour cumulative rainfall of each gauging station and whole basin were considered as inputs after analysing their cross-correlation significance to the present inflows. The total cross-correlation results among variables selected together with their significant lags are further summarised in Table 3.3.

Table 3.3: Cross-correlation values of cumulative rainfall values and inflow for most significant lags (most significant lag is shown in brackets)

Variable	**6 hr Cumulative**	**24 hr Cumulative**	**X days Cumulative**
Nuwara Eliya Rainfall	0.429 (lag= 4)	0.584 (lag= 1)	0.697 (lag= 0), (X= 5.4)
Ambewela Rainfall	0.407 (lag= 5)	0.668 (lag= 0)	0.670 (lag= 0), (X= 7.6)
Calidonia Rainfall	0.399 (lag= 3)	0.549 (lag= 1)	0.550 (lag= 0), (X= 8.3)
Sandringham Rainfall	0.446 (lag= 5)	0.595 (lag= 1)	0.637 (lag= 0), (X= 7.9)
Talawakelle Rainfall	0.424 (lag= 3)	0.574 (lag= 0)	0.675 (lag= 0), (X= 5.0)
Basin Rainfall	0.500 (lag= 5)	0.650 (lag= 1)	0.697 (lag= 0), (X= 5.4)

As can be seen in Table 3.3, lag-5 is the highest lag available as significant input of a variable. Hence, the input delay to the neural network model was selected as 5 for all input variables.

The list of inputs selected for the Neural network model is as follows.

1. Cumulative Rainfall in the last 6 hours of each Rainfall Gauging Station (**5 inputs)**
2. Cumulative Rainfall in the last 6 hours of the whole basin (**1 input**)
3. Cumulative Rainfall in the last 24 hours of each Rainfall Gauging Station (**5 inputs**)
4. Cumulative Rainfall in the last 24 hours of the whole basin (**1 input**)
5. Cumulative Rainfall in the last 5.4 days of Nuwara Eliya Rainfall Gauging Station (**1 input**)
6. Cumulative Rainfall in the last 7.4 days of Ambewela Rainfall Gauging Station (**1 input**)
7. Cumulative Rainfall in the last 7.9 days of Sandringham Rainfall Gauging Station (**1 input**)
8. Cumulative Rainfall in the last 8.3 days of Calidonia Rainfall Gauging Station (**1 input**)
9. Cumulative Rainfall in the last 5 days of Talawakelle Rainfall Gauging Station (**1 input**)

10. Cumulative Rainfall in the last 5.4 days of Whole Basin (**1 input**)
11. Water Level at Nanuoya Gauging Station (**1 input**)
12. Water Level at Calidonia Gauging Station (**1 input**)
13. Flag Variable to represent inflow value (1 **input**)
14. Significant Lag Values of above variables between 1 to 5
15. Past Natural Inflow to Reservoir (**10 input**)

A MATLAB programme was written to create these variables real time when their raw values are available real time. Appendix-A describes the programme.

3.4 Design of Neural Network Architecture

For half-hourly simulations and model setup, three years rainfall, water level and inflow recorded at 30-minute interval from 2016 to 2018 were used. Whole dataset was divided preserving the sequential relationships using MATLAB *dividerint* command into three sets: 80% training set, 10% validation set and 10% testing set. The raw input and output values were set to normalised to [-1, 1] in the NARX programme. The MATLAB programme for NARX modelling is attached as Appendix-A. Figure 3.13 shows the Open Loop NARX model used for modelling the inflow forecasting.

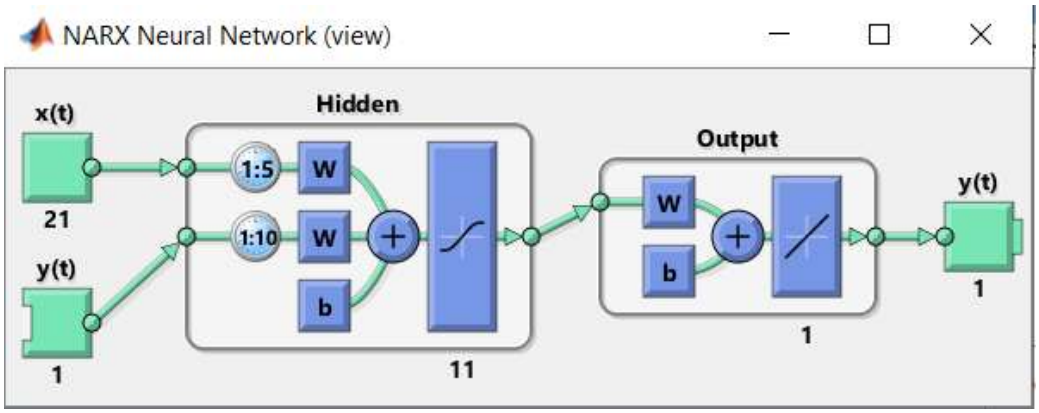


Figure 3.13: MATLAB Open Loop NARX Network used for Modelling

A heuristic approach (trial and error) was used to decide on the layer numbers, types of transfer functions used for each layer, number of hidden nodes and the

final structure of the neural network model. Mean Square Error (MSE) was used as the performance function, and same was used in the above trial and error approach. Finally, selected architecture has two layers (1 input layer, 1 hidden layer), and 11 hidden nodes and one output node. A rule of thumb of deciding number of hidden nodes, as discussed in the Chapter 2, for any neural network is to get the half of the input nodes.

Input Nodes = 21 → Half of Input Nodes = 10.5 → Hence, Hidden Nodes = 11

Basic characteristics of ANN model selected is summarised in Table 3.4.

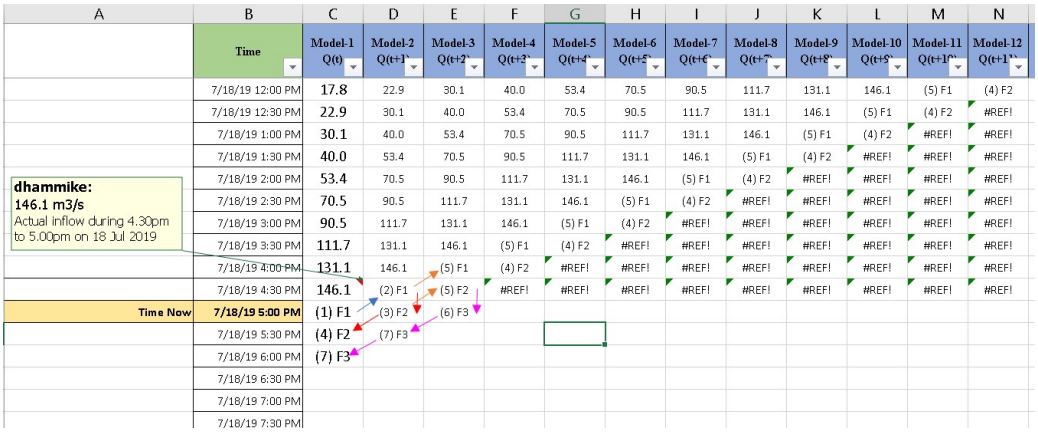
Table 3.4: Characteristics of Selected ANN Model

Characteristic	Value/Description
Number of Layers	2
Hidden Layer Neurons	11
Input Delay	5
Feedback Delay	10
Activation function of Input Layer	Tansig
Activation function of Hidden Layer	Linear
Training Algorithm	Levenberg Marquardt
Maximum Epochs	1000
Error Performance Function	MSE

Based on the feature selection, as significant lag values of input is 5, input delay for NARX network is set to 5. Similarity, feedback delay, antecedent values of inflow, was set to 10.

3.5 Multistep Ahead Forecasting

The ANN model discussed in section 3.4 is used to forecast inflow one step ahead i.e. inflow for next half an hour. To forecast multi step ahead, direct recursive hybrid strategy was used. Figure 3.14 shows the method of forecasting multistep ahead.



	Time	Model-1 Q(t)	Model-2 Q(t+1)	Model-3 Q(t+2)	Model-4 Q(t+3)	Model-5 Q(t+4)	Model-6 Q(t+5)	Model-7 Q(t+6)	Model-8 Q(t+7)	Model-9 Q(t+8)	Model-10 Q(t+9)	Model-11 Q(t+10)	Model-12 Q(t+11)
	7/18/19 12:00 PM	17.8	22.9	30.1	40.0	53.4	70.5	90.5	111.7	131.1	146.1	(5) F1	(4) F2
	7/18/19 12:30 PM	22.9	30.1	40.0	53.4	70.5	90.5	111.7	131.1	146.1	(5) F1	(4) F2	#REF!
	7/18/19 1:00 PM	30.1	40.0	53.4	70.5	90.5	111.7	131.1	146.1	(5) F1	(4) F2	#REF!	#REF!
	7/18/19 1:30 PM	40.0	53.4	70.5	90.5	111.7	131.1	146.1	(5) F1	(4) F2	#REF!	#REF!	#REF!
	7/18/19 2:00 PM	53.4	70.5	90.5	111.7	131.1	146.1	(5) F1	(4) F2	#REF!	#REF!	#REF!	#REF!
	7/18/19 2:30 PM	70.5	90.5	111.7	131.1	146.1	(5) F1	(4) F2	#REF!	#REF!	#REF!	#REF!	#REF!
	7/18/19 3:00 PM	90.5	111.7	131.1	146.1	(5) F1	(4) F2	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!
	7/18/19 3:30 PM	111.7	131.1	146.1	(5) F1	(4) F2	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!
	7/18/19 4:00 PM	131.1	146.1	(5) F1	(4) F2	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!
	7/18/19 4:30 PM	146.1	(2) F1	(5) F2	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!	#REF!
Time Now	7/18/19 5:00 PM	(1) F1	(3) F2	(6) F3									
	7/18/19 5:30 PM	(4) F2	(7) F3										
	7/18/19 6:00 PM	(7) F3											
	7/18/19 6:30 PM												
	7/18/19 7:00 PM												
	7/18/19 7:30 PM												

Figure 3.14: Multistep ahead forecasting methodology

There are 12 models for 12 timesteps ahead forecasting. The difference in each model is in the output with which the model was trained. As shown in the Figure 3.14, Model-2 was trained with output of inflow two step ahead inflow recorded in the past. The equations (3.1) and (3.2) show the case of two-step ahead forecast. Output of Equation (3.1) is fed to input of Equation (3.2).

$$Forecast(t+1) = Model1\big(observation(t) + obs(t-1) + \cdots..obs(t-n)\big) \quad (3\text{-}1)$$

$$Forecast(t+2) = Model2\big(Forecast(t+1) + obs(t) + \cdots \ldots..obs(t-n)\big) \quad (3\text{-}2)$$

Arrows and numbers in Figure 3.14 show the sequence in carrying out the multi-step ahead forecasting.

4 POND OPTIMISATION USING INFLOW FORECAST

4.1 Introduction

Using inflow forecast from inflow forecast model described in Chapter 3 was utilised to optimise the pond operation. Main objective in this pond optimisation is to maximise generation while minimising spilling all the time. This model was developed in total independence from the inflow forecast model so that any inflow forecast received from any other source could well be input to this model for optimisation. The overall block diagram of this model is shown in Figure 4.1.

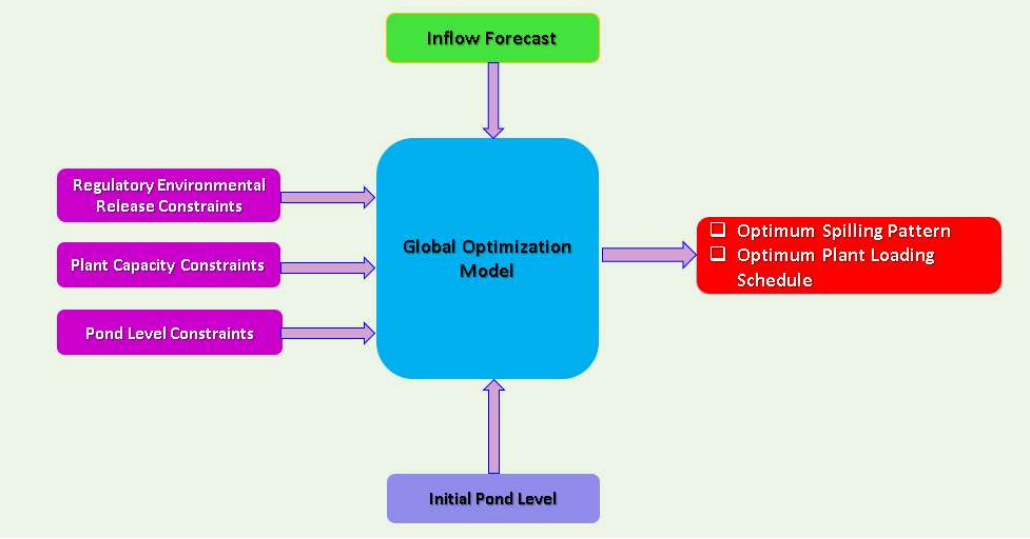


Figure 4.1: Overall Block Diagram of Pond Optimisation Model

It is expected the programme to run automatically without human intervention. Forecasted inflow for next 24 hours from Inflow Forecast models and initial water level at the start of optimisation are input to this model. Constraints such as pond capacity, environmental releases (St Claire release[2]) and Plant turbine capacity constraints are accounted in the MATLAB optimisation programme. When the programme is run, it presents optimised loading pattern for next 24 hours. Based on the results presented in the Chapter 5 for Inflow forecast models, the accuracy of the

[2] St. Claire is a waterfall on the downstream of Upper Kotmale pond in Talawakelle. It is required to release 1.33 m3/s of water from 5am to 3pm every day from Upper Kotmale pond to the St. Claire waterfall downstream.

inflow forecast diminishes after four to five hours, the average lag time of this river. Therefore, the optimisation model was also set to run in every 4 hours. Even if the optimisation could be run for less than 4-hour intervals, sufficient time was allowed for the periodical training of ANN models in Inflow Forecast models to take place.

4.2 Problem Formulation

Cross section of the pond and various parameters concerned are shown in Figure 4.2. It is noted here that the optimisation period in this study is in hours and period concern is very short term; hence, the effect of variables, such as the rate of evaporation, infiltration, Evapo-transpiration are not considered.

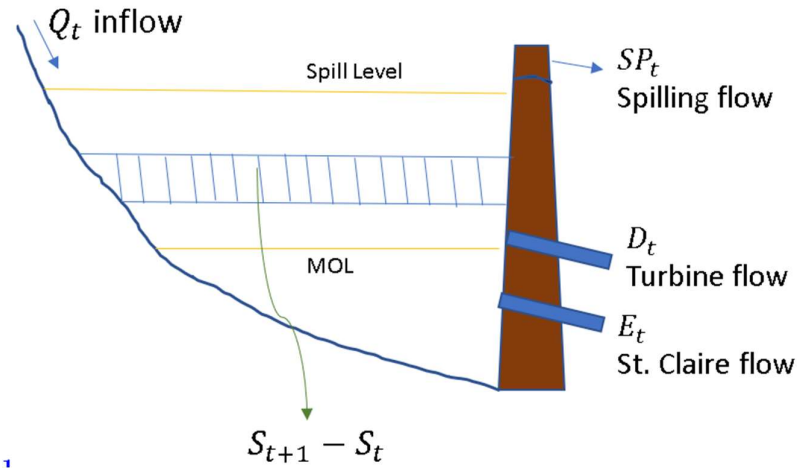


Figure 4.2: Pond Cross Section and Different Flows

The objective functions in this optimisation are:

- Maximise Turbine flow
- Minimise Spilling flow
- Maximise Pond storage

This is a multi-objective optimisation problem for which a method of Aggregate Objective Function (AOF) was used to convert multi-objective problem into single-objective problem for optimisation using MATLAB FMINCON optimisation command. FMINCON function in MATLAB is used for both non-linear and linear optimisation problems.

The decision variables and other parameters of the problem are summarised in the Table 4.1.

Table 4.1: Decision Variables and Other Parameters

Item	Unit	t=1h	t=2h	……	t=23h	t=24h
Initial Storage	m^3	S_0	X(49)		X(70)	X(71)
Inflow	m^3	I(1)	I(2)	……	I(23)	I(24)
Turbine Flow	m^3	X(1)	X(2)	……	X(23)	X(24)
Spilling	m^3	X(25)	X(26)	……..	X(47)	X(48)
Final Storage	m^3	X(49)	X(50)	……..	X(71)	X(72)

The AOF was expressed as in Equation (4-1).

$$Maximize\ J = w_1 \times D - w_3 \times SP + w_2 \times PS \tag{4-1}$$

Where,

w_1, w_2, w_3 are non-negative weight between 0 to 1 to each objective. In Equation (4-1), t refers to the period of optimisation. ***D*** is total Turbine discharge during optimisation period, ***SP*** is the total spilling required during the optimisation period, and ***PS*** is the amount of water stored in the pond during the optimisation period of 24 hours.

Appropriate values w_1, w_2, w_3 were set using trial and error by looking at the results. The optimum values for w_1, w_2, w_3 could also have been obtained using the Pareto Frontier for the three objectives. The Pareto solution using pareto frontier is explained in Chapter 3 : Literature Review.

Other variables in the objective function defined in Equation (4-1) can be expressed as follows in Equation (4-2), (4-3), (4-4).

$$D = \sum_{i=1}^{24} X(i) \tag{4-2}$$

$$SP = \sum_{i=25}^{48} X(i) \tag{4-3}$$

$$PS = \sum_{i=49}^{72} X(i) \tag{4-4}$$

The decision vector was expressed in vector form as shown in Equation (4-5).

$$X = \begin{pmatrix} X(1) \\ X(2) \\ \cdot\cdot \\ \cdot\cdot \\ X(71) \\ X(72) \end{pmatrix} \tag{4-5}$$

The remaining equations to fully describe the problem are the constraints. First, the boundary constraints for the decision variables are as following:

As plant has a minimum discharge (one unit minimum discharge) and a maximum discharge (both units in full load), the constraint (4.6) can be written. For all practical concerns, spilling can be assumed to have no upper limit. So, constraint (4-7) was written for spilling.

$$Turbine\ Min \leq X(1), X(2), \ldots\ldots X(24) \leq Turbine\ Max \tag{4-6}$$

$$0 \leq X(25), X(26), \ldots\ldots X(48) \tag{4-7}$$

The storage of reservoir, in this case the effective storage was considered, has minimum 0 and maximum $\boldsymbol{S_M}$ corresponding to MOL of 1190 masl and Spill Level of 1194 masl respectively. $\boldsymbol{S_M}$ for Upper Kotmale is 822470 m^3. Hence, the constraint 4-8 can be written as follows.

$$0 \leq X(49), X(50), \ldots\ldots X(72) \leq S_M \tag{4-8}$$

Water for St Clair (***Et***) is released as follows:

$$E(t) = \begin{cases} 0\ m^3/s, & 0000 < t \leq 0500 \\ 1.13\ m^3/s, & 0500 < t \leq 1500 \\ 0\ m^3/s, & 1500 < t \leq 2400 \end{cases} \tag{4-9}$$

Other constraints that govern the model is Pond water balance equation for each period. The inflow to the pond is used to change the reservoir volume, spilling, turbine discharge etc. There are 24 constraints for the total period of 24 hours. The set of equations are as shown in (4-10).

Period 1	$S_0 + I(1) - X(1) - X(25) = X(49)$	(4-10)
Period 2	$X(49) + I(2) - X(2) - X(26) = X(50)$	
Period 3	$X(50) + I(3) - X(3) - X(27) = X(51)$	
Period …	……	
Period 24	$X(71) + I(24) - X(24) - X(48) = X(72)$	

The MATLAB fmincon command requires all constraints to be defined as $\boldsymbol{AX < b}$, and boundary constraints as $\boldsymbol{lb \leq X < ub}$. The programme written in MATLAB is attached as Appendix-A and MATLAB fmincon command explained in the Chapter 2: Literature Review.

5 RESULTS AND ANALYSIS

5.1 Introduction

This section presents results of the feature selection, results and analysis of the Inflow Forecast Model (IFM) described in Chapter 3 and Pond Optimisation Model (POM) described in Chapter 4. As the IFM was trained using plant data from 2016 to 2018, the model performance was assessed using 2019 data. The POM was run with historical data between 2018 to 2019 assuming perfect knowledge of then future inflow for every next four hours.

5.2 Results of Feature Selection

It can be seen from feature selection that neither the point values of rainfall nor the antecedent values of point rainfall values have an impact to the inflow. It is the cumulative values of rainfall that have the influence on the future inflow. Antecedent values of cumulative rainfall have also got a significant effect on the future inflow.

5.3 Performance of Inflow Forecast Model

As there are twelve models trained for forecasting for next six hours, the performances of all the models are described in this section with various standard MATLAB plots. The progress of training the Model-1 Artificial Neural Network (ANN) is shown in Figure 5.1 and Figure 5.2 shows the regression plots of Model-1 which display the network outputs with respect to targets for training, validation, and test sets. As the data falls along a 45 degree line, where network outputs are equal to the targets, and as the fit is reasonably good for all data sets (training, validation and test) with correlation Coefficient (R value) of above 0.99, the Model-1 is a good fit for forecasting inflow for next 0.5 hours. To get additional verification of the models, error histogram was also plotted using MATLAB for all 12 models. Figure 5.3 shows the error histogram for Model-1. In this diagram, the blue bars represent training data, the green bars represent validation data, and the red bars represent testing data and it

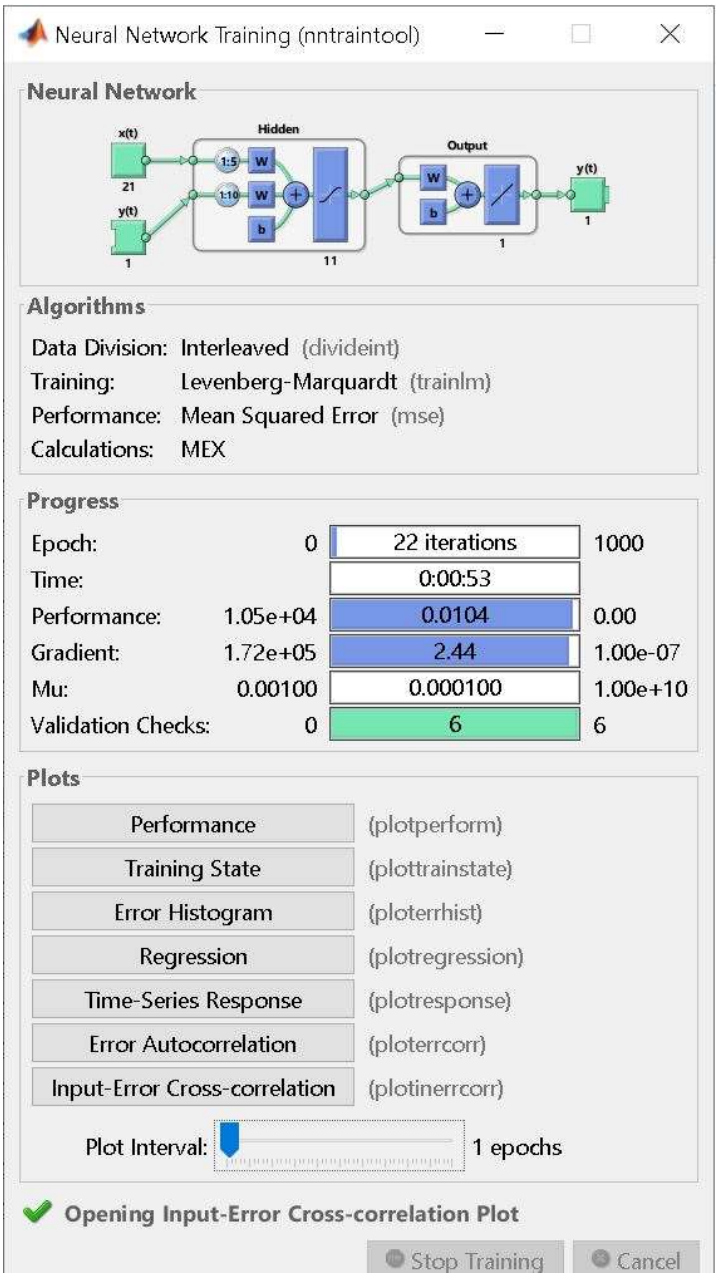


Figure 5.1: Progress Window of Training of ANN Model-1

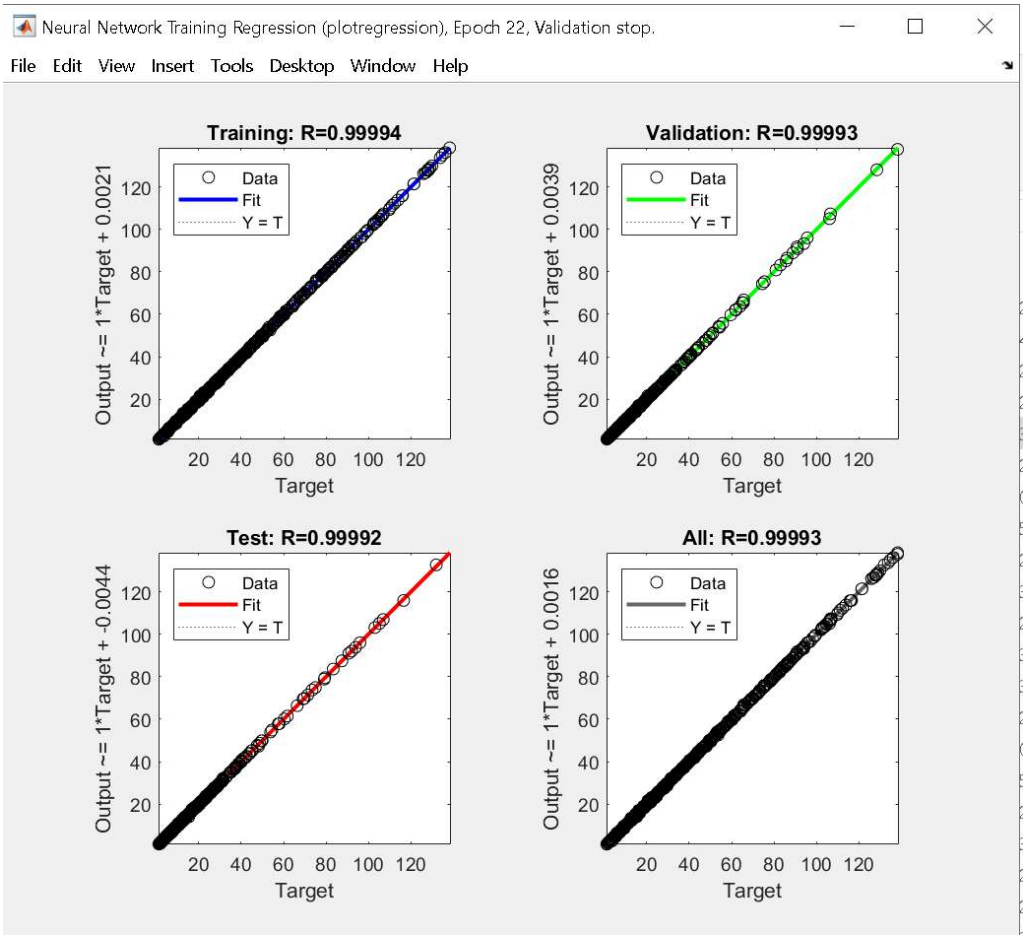


Figure 5.2: Model-1 Regression Plots

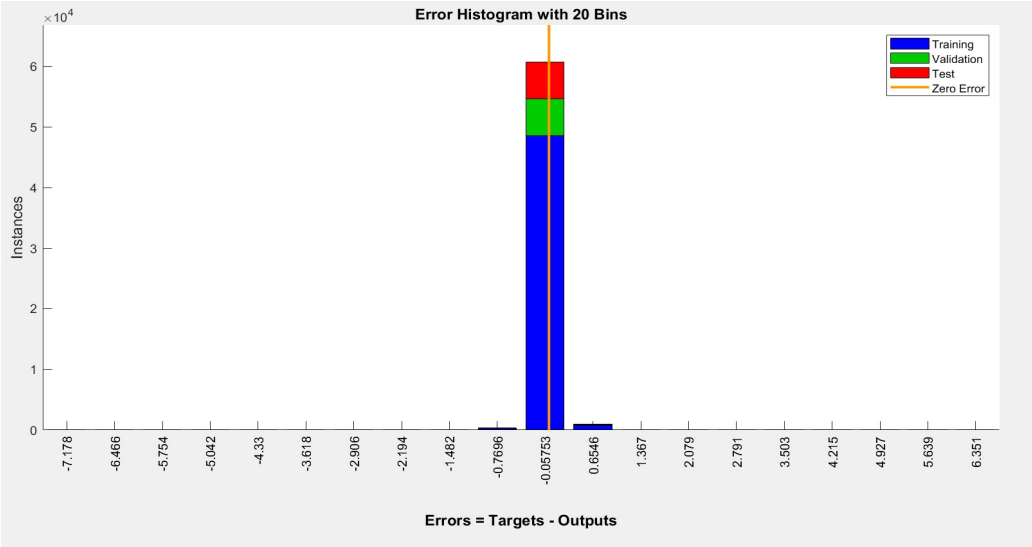


Figure 5.3: Error Histogram for Model-1

gives an indication of outliers, which are data points where the fit is significantly worse than most data. In this case, most errors fall between -1 and 1. Furthermore, the datapoints used for training the models can be deemed good as no significant outliers in any of the Models. During the training, all models were set to stop training if validation error increases to prevent models overfitting.

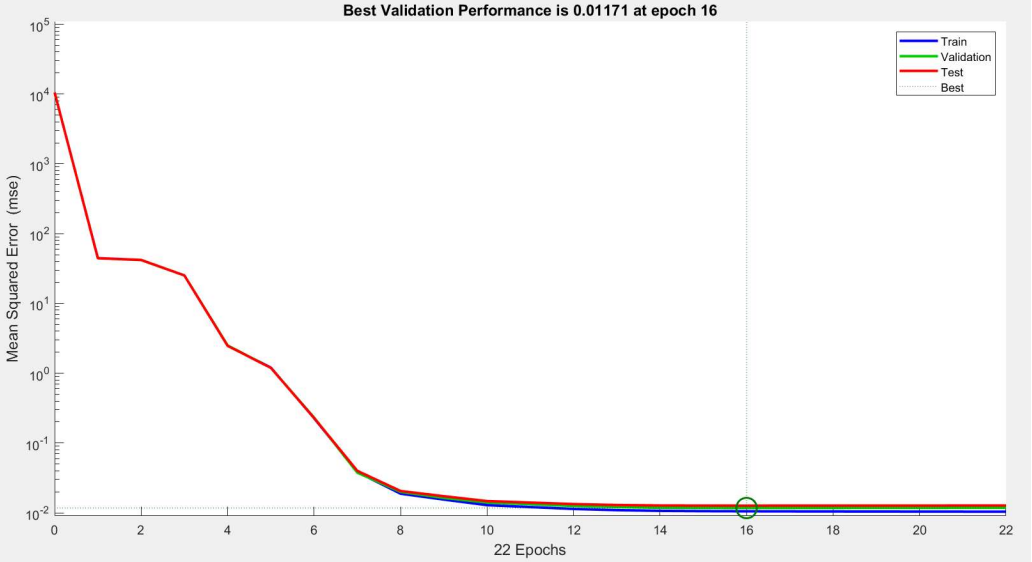


Figure 5.4: Best Validation Performance for Model-1

Figure 5.4 shows a plot of the training errors, validation errors, and test errors against each iteration as the training of the Model-1 progressed. The final Mean Square Error (MSE) is 0.0117 at iteration 16 (RMSE = 0.108 m^3/s), and it is very small. Also, it can be noted that the test error and validation error have similar characteristics. Hence, the result of model training was quite satisfactory.

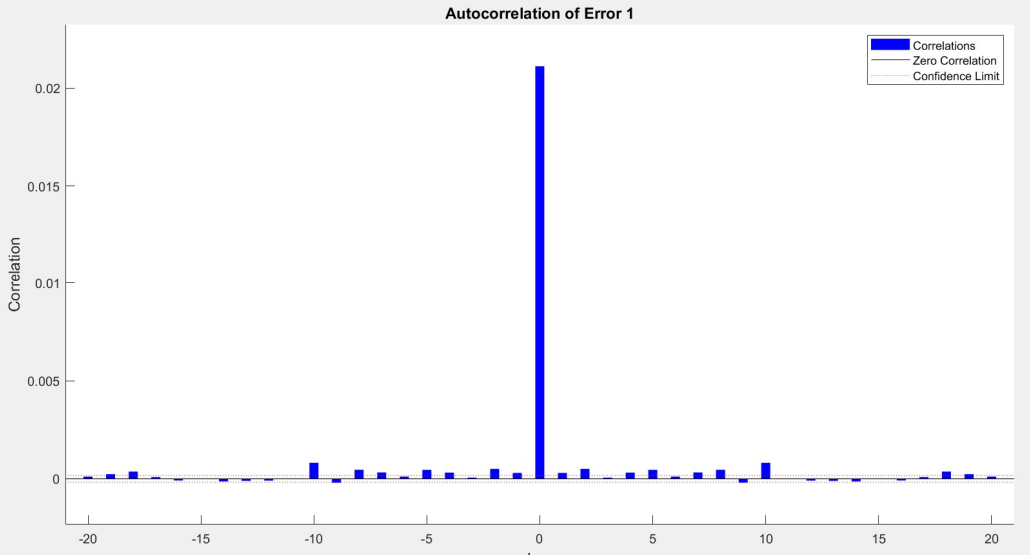


Figure 5.5: Plot of Error Autocorrelation for Model-1

Figure 5.5 shows the plot of error autocorrelation for Model-1, which explains how the error predictions are related in time. There should only be one nonzero value of the autocorrelation function for a perfect prediction model. Moreover, it should appear at zero lag. (This is the mean square error). As there is no significant correlation in the prediction errors, it could no longer be possible to improve the prediction by increasing the number of delays in the tapped delay lines. In this case, the correlations, except for the one at zero lag, fall approximately within the 95% confidence limits around zero, so the Model-1 seems to be adequate. The final plot used to validate performance was shown in Figure 5.6, which shows the plot of input-error correlation. This input-error cross-correlation plot illustrates how the errors are correlated with the input sequence and for a perfect prediction model, all the correlations should be zero. If the input is correlated with the error, then it should be possible to improve the prediction, perhaps by increasing the number of delays in the tapped delay lines. In the case of Model-1, all of the correlations fall within the confidence bounds around zero.

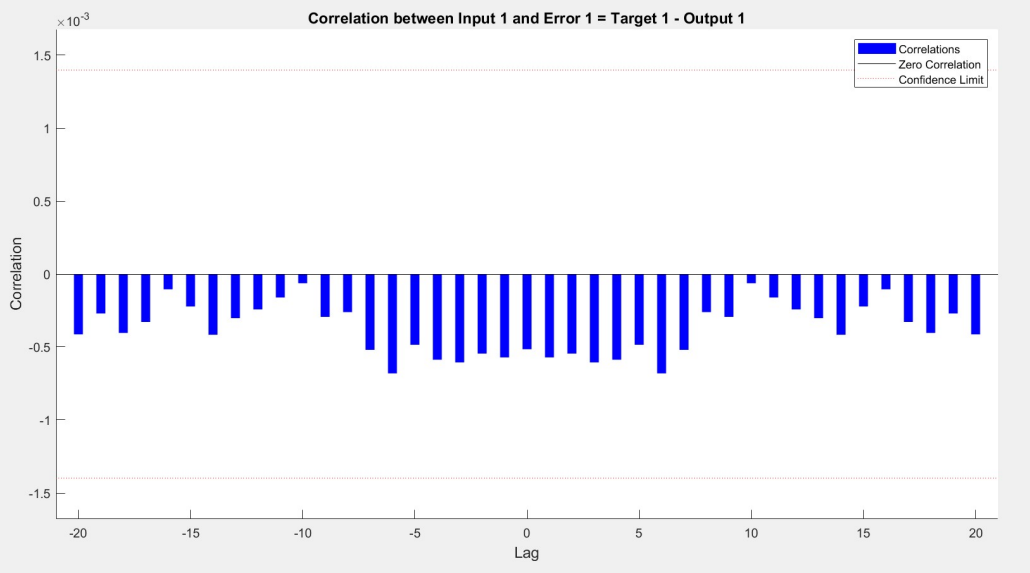


Figure 5.6: Input Error Correlation for Model-1

The results of training performance of all twelve models are summarised in Table 5.1.

Table 5.1:Summery of Statistics of Model Training for all 12 Models

ANN Model	Forecasting Period (hours)	No. of Iterations (Epoch)	RMSE (m^3/s)	R-Training	R-Validation	R-Testing
Model-1	0.5	16	0.1082	0.99994	0.99993	0.99992
Model-2	1.0	87	0.1106	0.99993	0.99993	0.99992
Model-3	1.5	11	0.1125	0.99993	0.99992	0.99992
Model-4	2.0	189	0.1130	0.99993	0.99992	0.99992
Model-5	2.5	17	0.1142	0.99993	0.99992	0.99992
Model-6	3.0	61	0.1154	0.99993	0.99992	0.99991
Model-7	3.5	263	0.1156	0.99993	0.99992	0.99992
Model-8	4.0	10	0.1160	0.99993	0.99992	0.99992
Model-9	4.5	117	0.1170	0.99993	0.99992	0.99991
Model-10	5.0	116	0.1183	0.99993	0.99992	0.99992
Model-11	5.5	76	0.1187	0.99993	0.99991	0.99992
Model-12	6.0	57	0.1207	0.99993	0.99991	0.99993

A few cases of 2019 data used to verify the forecasting accuracy of the Inflow Forecasting Model. It should be noted here, as this system is to be applied in real time

system where new data is gathered real time, it was programmed to retrain the network with new data being gathered in every 4 hours. This period could be changed in the programme as the system is in use. Table 5.7 shows the forecasting done by the models for six hours ahead from 5:00pm on 18 July 2019.

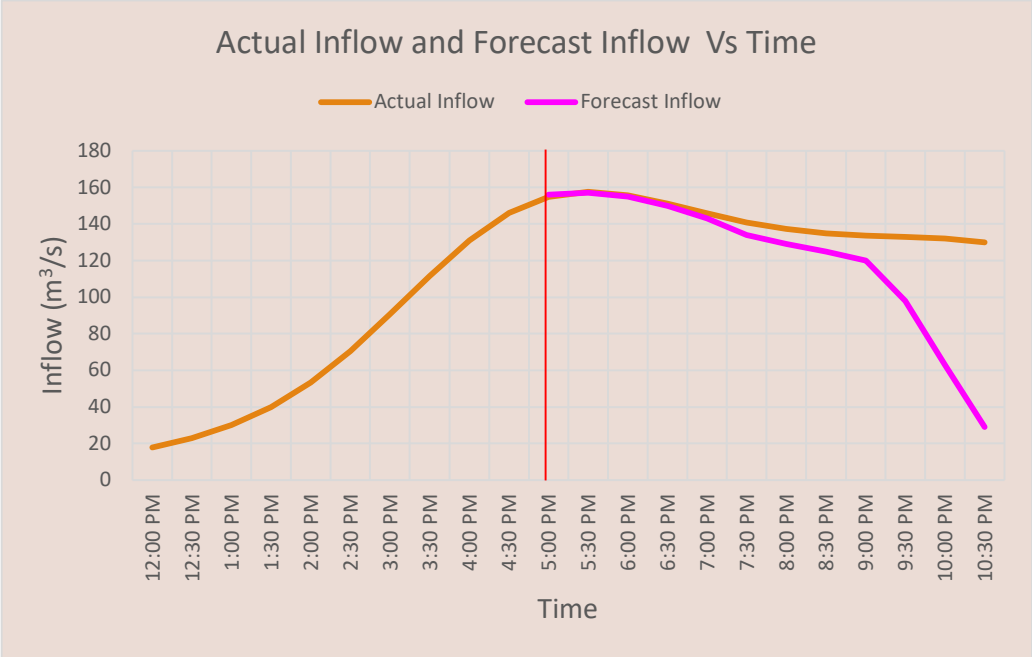


Figure 5.7: Actual Inflow and Forecasted Inflow with Time of 18 July 2019

The calculation of Root Mean Square Error and Mean Absolute Deviation for the case described in Figure 5.7 is shown in Table 5.2. Furthermore, a plot of RMSE and MAD vs time is show in Figure 5.8.

Table 5.2 : Calculation of RMSE and MAD

Forecast Horizon (hour)	Actual	Forecast	Error	Absoulute Value of Error	Square of Error	Absolute Values of Errors divded by Atual Values	number of periods	RMSE	MAD
t	At	Ft	At-Ft	|At-Ft|	$(At-Ft)^2$	|(At-Ft)/At|	n		
0.5	158	156	2.20	2.20	4.827	0.01	1	2.20	2.20
1.0	160	157	3.33	3.33	11.103	0.02	2	2.82	2.76
1.5	156	155	0.52	0.52	0.272	0.00	3	2.32	2.02
2.0	149	150	-1.00	1.00	1.007	0.01	4	2.07	1.76
2.5	143	143	0.38	0.38	0.142	0.00	5	1.86	1.49
3.0	139	134	5.21	5.21	27.151	0.04	6	2.72	2.11
3.5	136	129	7.34	7.34	53.934	0.05	7	3.75	2.86
4.0	135	125	9.57	9.57	91.657	0.07	8	4.87	3.69
4.5	134	120	13.89	13.89	192.829	0.10	9	6.52	4.83
5.0	134	98	36.16	36.16	1307.419	0.27	10	13.00	7.96
5.5	135	63	71.74	71.74	5146.338	0.53	11	24.93	13.76
6.0	134	29	104.83	104.83	10989.137	0.78	12	38.54	21.35

It can be seen from Figure 5.7, 5.8 and Table 5.2 that the forecasting error increases beyond 4 to 4.5 hours. This is typical lag time of the Upper Kotmale river.

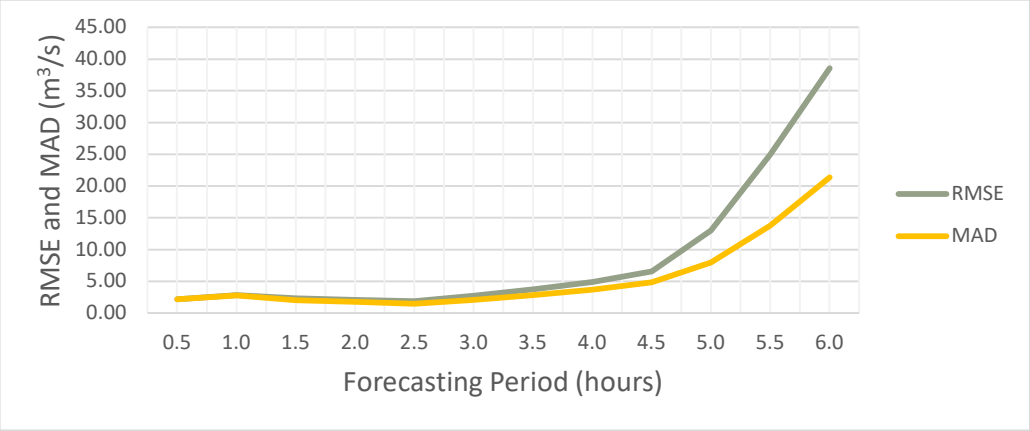


Figure 5.8 : RMSE adn MAD of Forecasting vs Forecasting Period

The result of the Inflow Forecasting model explained so far reveals that it can forecast to a reasonable accuracy for the next 4 to 4.5 hours.

5.4 Performance of Pond Optimisation

Any optimisation programme written in MATLAB can be written in three files namely, “Main”, “ObjFunc” and “ConFun” as a practice. “Main” file has the initial conditions and boundary conditions, “ObjFun” has the objective function and “ConFun” has any non-linear constraints if any. The MATLAB programme written for the Pond Optimisation is given in Appendix-A.

Using trial and error, it was found that optimum weightages that meet the multi-objectives defined are ***w1 = 0.51, w2 = 0.49, w3 =0. w1, w2, & w3***, as explained in Chapter 4, are the weightages of aggregate objective function referring to Generation, Pond Storage and Spilling. It is clear from the optimum weightings found, that maximising generation, and end storage automatically minimise spilling. Hence, one can consider the optimisation as two objective problem of maximising generation and end storage. For simulation, hourly inflow records of 2017, 2018, and 2019 were used

and optimisation was run every four hours for each year assuming perfect knowledge of the future inflow.

Figure 5.9 is the optimisation run for year 2016, which shows for every hour, average plant loading in MW and average spilling required in m^3/s and the resulting Average Level in the pond in masl for optimum operation of the pond for year 2016.

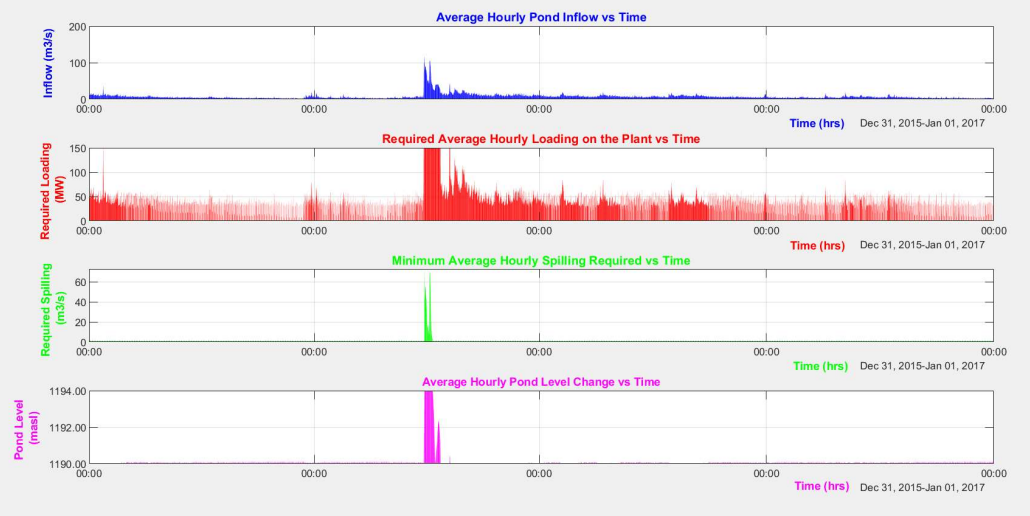


Figure 5.9: Optimisation run for year 2016

The optimised generation given by the programme and actual generation for each month in 2016 was compared and the results are shown in Table 5.4. The gain in generation in GWh and as a percentage is shown in this table.

It was observed that gain in generation of about 8% could be achieved for year 2016.

Table 5.3: Comparison of Actual and Optimised Generation with Gain for 2016

Month	**Actual Generation (GWh)**	**Optimised Generation (GWh)**	**Gain in Generation (GWh)**	**Percentage Generation Gain (%)**
January	24.62	26.95	2.33	9.5
February	10.65	11.80	1.15	10.8
March	8.76	9.49	0.73	8.3
April	10.00	10.81	0.81	8.1
May	45.54	49.19	3.65	8.0
June	32.03	34.81	2.78	8.7
July	25.63	27.27	1.64	6.4

August	22.06	23.67	1.61	7.3
September	18.96	19.75	0.79	4.2
October	11.28	13.03	1.75	15.5
November	17.77	18.37	0.6	3.4
December	7.01	8.36	1.35	19.3
Total	**243.3**	**253.5**	**19.19**	**8.2**

Furthermore, same analysis done for year 2017 and 2018. Figure 5.10 shows the results of optimisation run for the year 2017.

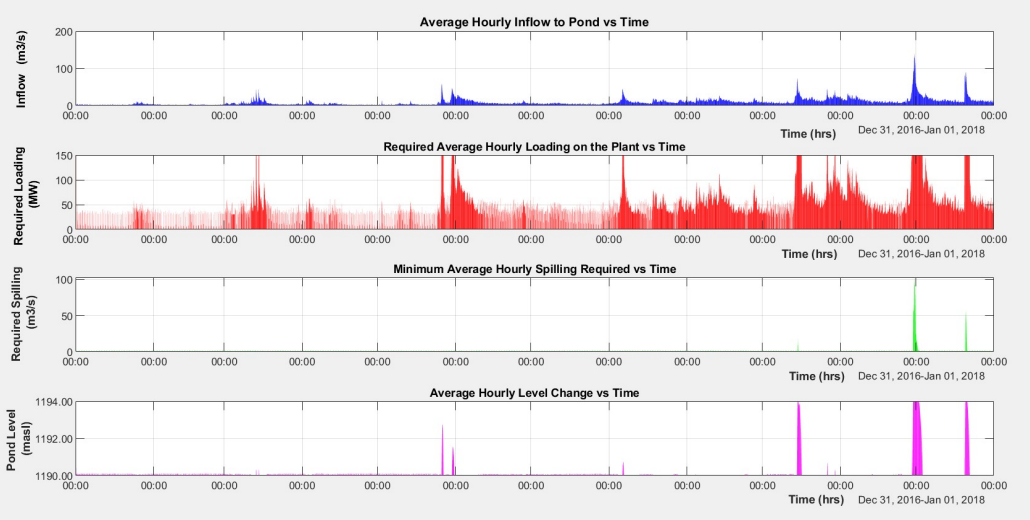


Figure 5.10: Optimisation Run for the Year 2017

The generation comparison for year 2017 is shown in Table 5.4.

Table 5.4: Comparison of Actual and Optimised Generation with Gain for 2017

Month	**Actual Generation (GWh)**	**Optimised Generation (GWh)**	**Gain (GWh)**	**Percentage Gain (%)**
January	7.17	8.23	1.06	14.8
February	3.95	4.11	0.16	4.1
March	18.41	20.09	1.68	9.1
April	8.85	9.70	0.85	9.6
May	19.01	19.17	0.16	0.8
June	26.08	28.02	1.94	7.4
July	12.29	13.22	0.93	7.6
August	31.41	32.79	1.38	4.4

September	35.44	37.41	1.97	5.6
October	46.00	47.99	1.99	4.3
November	42.32	44.31	1.99	4.7
December	51.48	54.23	2.75	5.3
Total	**302.41**	**319.27**	**16.86**	**5.6**

It can be seen from Table 5.4 that about 6% gain in generation can be achieved for year 2017. Figure 5.11 shows the results of optimisation run for the year 2018.

Figure 5.11: Optimisation Run for the Year 2018

The generation comparison for year 2018 is shown in Table 5.5.

Table 5.5: Comparison of Actual and Optimised Generation with Gain for 2018

Month	**Actual Generation (GWh)**	**Optimised Generation (GWh)**	**Gain (GWh)**	**Percentage Gain (%)**
January	18.67	20.37	1.7	9.1
February	11.83	12.66	0.83	7.0
March	12.18	12.97	0.79	6.5
April	23.62	25.84	2.22	9.4
May	57.33	60.49	3.16	5.5
June	67.39	71.72	4.33	6.4
July	53.20	56.22	3.02	5.7
August	79.74	83.07	3.33	4.2

September	47.69	49.94	2.25	4.7
October	91.91	96.68	4.77	5.2
November	60.02	63.54	3.52	5.9
December	26.01	27.54	1.53	5.9
Total	**549.59**	**581.04**	**31.45**	**5.7**

From the analysis of 2018 generation data, 5.7% gain in generation could be achieved had the pond was operated optimally.

Figure 5.12 shows a zoomed-in extract from Figure 5.9 pertaining to data of the 15 May 2016 for closer viewing. Figure 5.13 shows the comparison of actual and optimised generation with inflow on 15 May 2016.

Figure 5.12: Extract of 2016 data from 15 May 2016

As can be seen from Figure 5.13, when inflow is forecasted to have a sharp rise, Pond Optimisation Model (POM), being anticipative and proactive, increases the generation in anticipation of high inflow coming in; thus, managing the pond effectively. The operator failed to increase the generation due to not having proper inflow forecast and being judgemental.

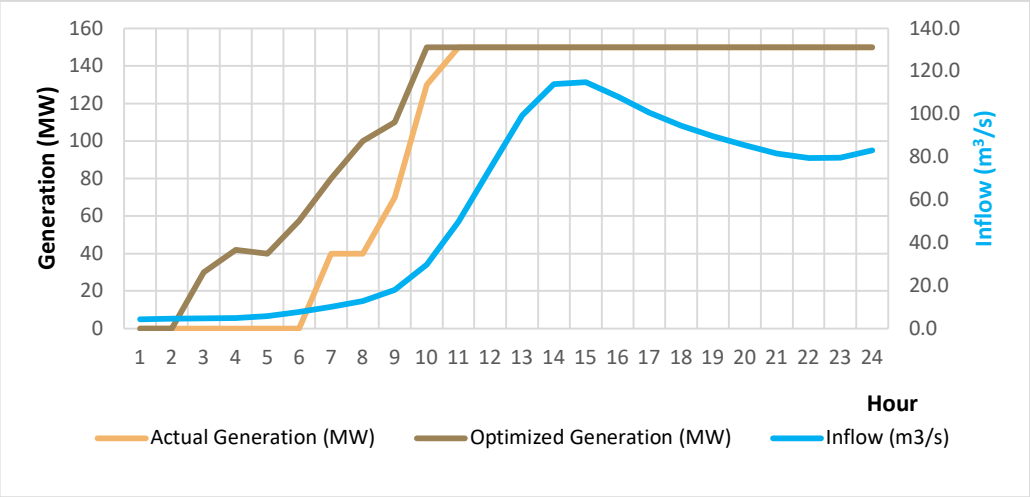


Figure 5.13: Comparison of Actual and Optimised Generation with Inflow on 15 May 2016

Figure 5.14 shows the summery of actual generation and optimised generation for three years of 2016, 2017 and 2018.

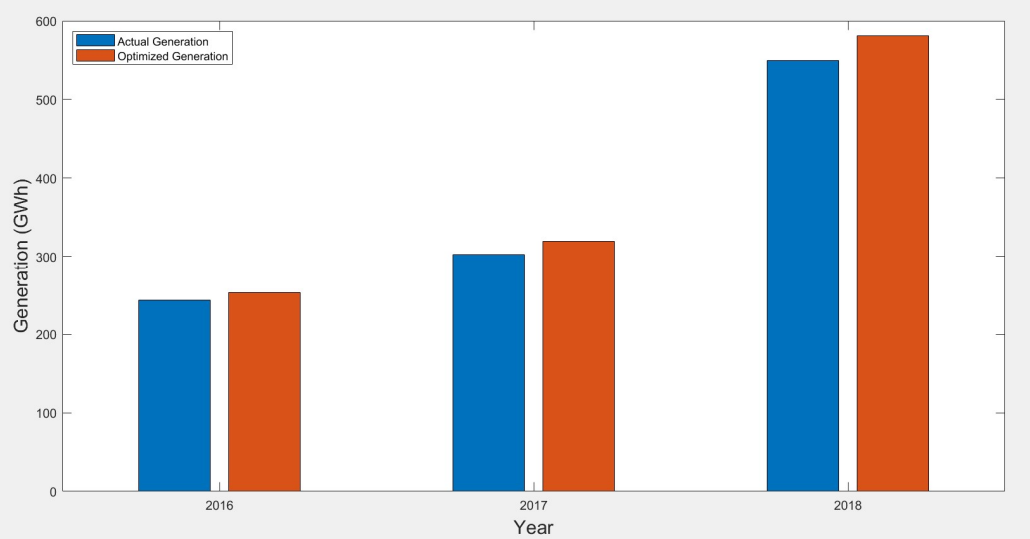


Figure 5.14: Comparison of Actual and Optimised Generation for 2016, 2017 and 2018

Next, it was analysed to see the amount of water spilled by the operator and amount of water that would have spilled if the gates were operated to meet optimum spilling required for each hour for years 2016, 2017 and 2018. Table 5.6 shows the actual and optimised spilling for the year 2016, 2017 and 2018.

Table 5.6: Comparison of Actual and Optimised Spilling in year 2016, 2017, 2018

Year	2016	2017	2018	Total
Actual Spilling (MCM)	10.39	12.19	63.957	**86.54**
Optimised Spilling (MCM)	9.48	9.73	59.74	**78.95**
Saving in Spilling (MCM)	0.91	2.46	4.22	**7.59**

It can be seen from Table 5.6 that significant amount of water can be saved by optimally operating the reservoir. The effective storage of the pond is 0.8 MCM, and water can be saved more than the effective storage for each year. It is, indeed, this saving of water which causes enhancement of generation in optimal operation cases.

5.5 Economic Evaluation of Water Saving

The gain in energy generation in this study is mainly attributable to water saving as a result of optimal reservoir operation. Therefore, to ascertain the benefit in economic terms, water values need to be considered for the reservoir. When contacted the System Control Centre (SCC) of CEB, it was pointed out that value of water depends on many factors including current storage, required irrigation releases, price of thermal unit, availability of thermal power plants, planned plant outages, inflows, etc and that a software is run every month to estimate these water values. The value of unit of electricity in Rs/kWh that were derived from water values[3] given from the software for each month from 2016 to 2018 were received from SCC of CEB as shown in the Table 5.7.

Table 5.7: Economic Water Value for each Month from 2016 to 2018

Month	Economic Water Value (Rs / kWh)		
	2016	2017	2018
January	57.94	24.71	71.35
February	47.19	28.80	58.19
March	49.96	33.97	34.20
April	64.66	32.26	35.55
May	48.12	29.07	32.23
June	20.72	51.64	26.09

[3] These water values may not be accurate in case of spilling.

July	16.11	18.43	22.98
August	17.13	19.38	24.27
September	16.71	20.13	24.45
October	18.24	20.10	24.64
November	20.26	20.16	26.65
December	21.01	20.13	28.66

The gains in generation for the three years from 2016 to 2018 valued based on above economic terms are summarised in the Table 5.8

Table 5.8: Summary of Economic Gain for years 2016 -2018

	Year			
	2016	**2017**	**2018**	**Total**
Economic Gain (Rs. million)	651	439	951	2,041

A total gain in economic terms, which is long term in nature, is around Rs. 2 billion for the three years of 2016 to 2018.

6 CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusion

Historical data of rainfall, water level and inflow for 2016, 2017, 2018 and 2019 at Upper Kotmale Power Station was collected. Then, an input and output model using MATLAB NARX was trained using the historical data to develop a model to forecast pond inflow at Upper Kotmale. After that, second model using MATLAB FMINCON was developed to utilise inflow forecast from the first model to optimise the pond in order to maximise the power generation while minimising spilling. Operator is given the optimum loading and spilling pattern of the power station without any human intervention. Models were developed such that they can be updated and trained as the new data is collected.

Using the results of present study, it is possible to characterise the pond optimisation of run-of-river hydropower plant using accurate inflow forecasting with ANN modelling and optimising the pond using that inflow forecast. Although the research done as a solution for Upper Kotmale Power Station, the methodology described could well be applied in other scenarios of forecasting and optimisation.

Results of the inflow forecasting model show that inflow to Upper Kotmale Pond could reasonably be forecasted for next 4 to 5 hours, which is approximately the average lag time of the river. ANN models could well be developed in real time applications like inflow forecasting, where accuracy and speed is crucial. Among such ANN models, Nonlinear Autoregressive network with eXogenous inputs (NARX), which is a recurrent dynamic network with feedback connections, was used in the research for inflow forecasting, and it was modelled using MATLAB. The Inflow Forecasting Model (IFM) was setup using data for the period 2016-2018 and tested for 2019 data, too.

In any ANN modelling, it is very important to select the input data set, which is enough to determine the neural network. At the same time, too many unnecessary input features could lead to poor performance. The method of selecting input in this research was discussed under feature selection mainly using correlation analysis. It was

established that cumulative inputs such as cumulative rainfall is more important than point rainfall.

It was noted that the optimisation of pond is a multi-objective optimisation problem; hence, MATLAB multi-objective optimisation algorithms used for optimisation of pond using Aggregate Objective Function (AOF). It was also noted that the appropriate weightings for the AOF were selected by trial and error. This research reveals that there exists a room for optimisation to enhance the gain in generation by more than about 5% with allowance for uncertainty in inflow forecasts.

With successful implementation of these models, operator is presented with optimum loading pattern of plant and required optimum spilling each hour for next 24 hours.

Furthermore, it can be concluded that this type of modelling and optimisation could be done for run-of-river hydro plant with small pondages such as in Upper Kotmale PS and Kukule PS in CEB.

6.2 Future Work

In this study, Direct Recursive Hybrid Strategy was used to multi-step ahead forecasting for inflow forecasting models. Other methods also can be experimented with different ANN structures.

Optimisation in this study forecasted only on plant level optimisation, reader could also incorporate unit level and system level optimisation as well to some extent. It is also possible to add some weightage on the different time slots so that units loading is given higher priority in peak time rather than in off peak time. Moreover, the optimisation in the present study does not have any sense of number of start and stops of the units, so it can be programmed in future work.

REFERENCES

[1] Electric Power Development Company, "Supplementary Report of Final Design Report of Upper Kotmale Hyusropower Project," JPower., Tokyo, Japan, Nov. 2004.

[2] Ceylon Electricity Board, "Layout Diagram of Telemetry and Warning System of Upper Kotmale Hydropower Station," Mitsubishi Corporation., Tokyo, Japan, E02-6255, Jan. 2015.

[3] R. A. Wurbs, "Reservoir System Simulation and Optimization Models," *J. Water Resources Planning and Management.,* vol. 119, no. 4, pp. 455-472, Jul. 1993.

[4] A. F. Hamlet *et al*., "Economic Value of Long-Lead Streamflow Forecasts for Columbia River Hydropower," *J. Water Resources Planning and Management.,* vol. 128, no. 2, pp. 91-101, Mar. 2002.

[5] L. W. Mays and Y. K. Tung, *Hydrosystems Engineering and Management*, 2nd ed., Colorado: Water Resources Publications, 2002.

[6] X. Dong *et al*., "Effect of flow forecasting quality on benefits of reservoir operation - a case study for the Geheyan reservoir (China)," *J. Hydrology and earth system sciences discussions.,* vol. 3, pp. 3771-3814, 2006.

[7] J. W. Labadie, "Optimal Operation of Multireservoir Systems: State-of-the-Art Review," *J. Water resources planning and Management.,* vol. 130, no. 2, pp. 93-111, Mar. 2004.

[8] M. R. Mustafa *et al*., "Artificial Neural Networks Modelling in Water Resources Engineering: Infastructure and Applications," *J. Civil and Environmental Engineering.,* vol. 6, no. 2, pp. 128-134, Jan. 2012.

[9] C. W. Dawson *et al*., "A comparative study of artificial neural network techniques for river stage forecasting," *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Montreal, Que., 2005, pp. 2666-2670 vol. 4.

[10] N. Sajikumar and B. Thandaveswara, "A non-linear rainfall-runoff model using an artificial neural network," *J. Hydrology,* vol. 21, no. 6, pp. 32-35, Mar. 1999.

[11] *Improve Shallow Neural Network Generalization and Avoid Overfitting*, [online] Available:https://in.mathworks.com/help/deeplearning/ug/improve-neural-network-generalization-and-avoid-overfitting.html.

[12] M. Campolo *et al*., "Artificial neural network approach to flood forecasting in the River Arno," *J. Hydrological Sciences,* vol. 48, no. 3, Jun. 2003.

[13] A.W. Minns and M.J. Hall, "Artificial neural networks as rainfall-runoff models," *J. Hydrology,* vol. 41, no. 3, pp. 399-417, Jun. 1996.

[14] L. L. Rogers and F. U. Dowla, "Optimization of groundwater remediation using artificial neural networks with parallel solute transport modelling," *J. Water Resources Resarch,* vol. 30, pp. 457-481, 1994.

[15] N. Karunanithi *et al*., "Neural Networks for River Flow Prediction," *J. Computer Science,* vol. 8, no. 2, pp. 201-220, Apr. 1994.

[16] D. E. Rumelhart *et al*., "Learning representations by back-propagating errors," *J. Nature,* pp. 323, 533-536, Oct. 1986.

[17] P. H. Talaee, "Multilayer perceptron with different training algorithms for streamflow forecasting," *J. Neural Comput & Applic.,* vol. 5, Mar. 2014.

[18] S. Mohanty *et al*., "Prediction of global solar radiation using nonlinear auto regressive network with exogenous inputs (narx)," *2015 39th Nat. Systems Conf. (NSC)*, Noida, 2015, pp. 1-6.

[19] E. Cadenas *et al*., "Wind Speed Prediction Using a Univariate ARIMA Model and a Multivariate NARX Model," *J. Energies ,* vol. 09, no. 109, Feb. 2016.

[20] B. Jason, *Deep Learning for Time Series Forecasting*, 1st ed. 2018.

[21] A. Messac, *Optimization in Practice with MATLAB*, USA: Cambridge University Press, 2015.

[22] S. Khu and H. Madsen, "Multiobjective calibration with Pareto preference ordering: an application to rainfall–runoff model calibration," *J.Water Resources,* vol. 41, Mar. 2005.

APPENDIX-A: MATLAB PROGRAMMES OF IFM

Inflow Forecast Model

12/21/19 7:01 AM F:\2. Results\Matlab Files\New\Run.m

```
tic

LoadData1
myDataInput
OptimizeModels
ForecastAhead

toc
```

12/21/19 7:02 AM F:\2. Results\Matlab File...\LoadData1.m

```
clear;
clc;

% Import data from spreadsheet
% Script for importing data from the following spreadsheet:

% Workbook: F:\2. Results\Final Row Data 15 Nov 2019.xlsx
% Worksheet: 16 & 17

% Auto-generated by MATLAB on 15-Nov-2019 19:51:25

% Setup the Import Options
opts = spreadsheetImportOptions("NumVariables", 10);

% Specify sheet and range
opts.Sheet = "16 to 18";
opts.DataRange = "B3:K52598";

% Specify column names and types
opts.VariableNames = ["Q1", "InF1", "InR1", "InR5", "InR9", "InR13",
"InR17", "InR21", "InL1", "InL2"];
opts.VariableTypes = ["double", "double", "double", "double",
"double", "double", "double", "double", "double", "double"];

% Import the data
tbl = readtable("F:\2. Results\Final Row Data 15 Nov 2019.xlsx",
opts, "UseExcel", false);

% Convert to output type
Q1 = tbl.Q1;
InF1 = tbl.InF1;
InR1 = tbl.InR1;
InR5 = tbl.InR5;
InR9 = tbl.InR9;
InR13 = tbl.InR13;
InR17 = tbl.InR17;
```

```
InR21 = tbl.InR21;
InL1 = tbl.InL1;
InL2 = tbl.InL2;

% Clear temporary variables
clear opts tbl

%% Import data from spreadsheet
% Script for importing data from the following spreadsheet:
%
%    Workbook: F:\2. Results\Final Row Data 15 Nov 2019.xlsx
%    Worksheet: 18
%
% Auto-generated by MATLAB on 15-Nov-2019 19:54:36

%% Setup the Import Options
clc;
clear;

opts = spreadsheetImportOptions("NumVariables", 10);

% Specify sheet and range
opts.Sheet =    "16 to 9 Apr"    ;          % "16 to 19 Aug";
%"forecast";
opts.DataRange = "B3:K57334" ;    %"B3:K62136";  %"B3:K9540";

% Specify column names and types
opts.VariableNames = ["Qnew1", "InewF1", "InewR1", "InewR5",
"InewR9", "InewR13", "InewR17", "InewR21", "InewL1", "InewL2"];
opts.VariableTypes = ["double", "double", "double", "double",
"double", "double", "double", "double", "double", "double"];

% Import the data
tbl = readtable("F:\2. Results\2019 data forecast.xlsx", opts,
"UseExcel", false);

%% Convert to output type
Qnew1 = tbl.Qnew1;
InewF1 = tbl.InewF1;
InewR1 = tbl.InewR1;
InewR5 = tbl.InewR5;
InewR9 = tbl.InewR9;
InewR13 = tbl.InewR13;
InewR17 = tbl.InewR17;
InewR21 = tbl.InewR21;
InewL1 = tbl.InewL1;
InewL2 = tbl.InewL2;

%% Clear temporary variables
clear opts tbl
```

12/21/19 7:04 AM F:\2. Results\Matlab...\myDataInput.m

```
%% Import data from spreadsheet
% Script for importing data from the following spreadsheet:
```

```
%
%    Workbook: C:\Users\Dell\Documents\Sixteen_to18 as at 13 Sep
2019.xlsx
%    Worksheet: 16_17_18
%
% Auto-generated by MATLAB on 14-Aug-2019 19:32:07


%% Data input from Excel and initial procesing
% Step 1
n =size(Q1,1); % get number of data points

InR2(n)=0; InR3(n)=0; InR4(n)=0;
InR6(n)=0; InR7(n)=0; InR8(n)=0;
InR10(n)=0; InR11(n)=0; InR12(n)=0;
InR14(n)=0; InR15(n)=0; InR16(n)=0;
InR18(n)=0; InR19(n)=0; InR20(n)=0;
InR22(n)=0; InR23(n)=0; InR24(n)=0;
InF1(n)=0;
sum1=0;sum2=0;sum3=0;
sum4=0;sum5=0;sum6=0;
sum7=0;sum8=0;sum9=0;
sum10=0;sum11=0;sum12=0;
sum13=0;sum14=0;sum15=0;
sum16=0;sum17=0;sum18=0;
for j=1:n

    % Nuwara Eliya
    if j<12+1
        sum1=0;
        for i=1:j-1
            sum1=sum1+InR1(i);
        end
        InR2(j)=sum1;
    end

    if j>12
        sum1=0;
        for i=j-12:j-1
            sum1=sum1+InR1(i);
        end
        InR2(j)=sum1;
    end

    if j<48+1              %
        sum2=0;
        for i=1:j-1
            sum2=sum2+InR1(i);
        end
        InR3(j)=sum2;        %
    end

    if j>48                %
        sum2=0;
        for i=j-48:j-1      %
            sum2=sum2+InR1(i);
        end
        InR3(j)=sum2;       %
```

```
end

if j<259+1                %
    sum3=0;
    for i=1:j-1
        sum3=sum3+InR1(i);
    end
    InR4(j)=sum3;         %
end

if j>259                  %
    sum3=0;
    for i=j-259:j-1       %
        sum3=sum3+InR1(i);
    end
    InR4(j)=sum3;         %
end

%Ambewela
if j<12+1
    sum4=0;
    for i=1:j-1
        sum4=sum4+InR5(i);
    end
    InR6(j)=sum4;
end

if j>12
    sum4=0;
    for i=j-12:j-1
        sum4=sum4+InR5(i);
    end
    InR6(j)=sum4;
end

if j<48+1               %
    sum5=0;
    for i=1:j-1
        sum5=sum5+InR5(i);
    end
    InR7(j)=sum5;         %
end

if j>48                   %
    sum5=0;
    for i=j-48:j-1        %
        sum5=sum5+InR5(i);
    end
    InR7(j)=sum5;         %
end

if j<363+1                %
    sum6=0;
    for i=1:j-1
        sum6=sum6+InR5(i);
    end
    InR8(j)=sum6;         %
end
```

```
if j>363                %
    sum6=0;
    for i=j-363:j-1     %
        sum6=sum6+InR5(i);
    end
    InR8(j)=sum6;       %
end

%Sandrigham
if j<12+1
    sum7=0;
    for i=1:j-1
        sum7=sum7+InR9(i);
    end
    InR10(j)=sum7;
end

if j>12
    sum7=0;
    for i=j-12:j-1
        sum7=sum7+InR9(i);
    end
    InR10(j)=sum7;
end

if j<48+1             %
    sum8=0;
    for i=1:j-1
        sum8=sum8+InR9(i);
    end
    InR11(j)=sum8;        %
end

if j>48                 %
    sum8=0;
    for i=j-48:j-1      %
        sum8=sum8+InR9(i);
    end
    InR11(j)=sum8;       %
end

if j<380+1              %
    sum9=0;
    for i=1:j-1
        sum9=sum9+InR9(i);
    end
    InR12(j)=sum9;        %
end

if j>380                %
    sum9=0;
    for i=j-380:j-1     %
        sum9=sum9+InR9(i);
    end
    InR12(j)=sum9;       %
end
```

```
    sum13=0;
    for i=j-12:j-1
        sum13=sum13+InR17(i);
    end
    InR18(j)=sum13;
end

if j<48+1              %
    sum14=0;
    for i=1:j-1
        sum14=sum14+InR17(i);
    end
    InR19(j)=sum14;        %
end

if j>48                %
    sum14=0;
    for i=j-48:j-1     %
        sum14=sum14+InR17(i);
    end
    InR19(j)=sum14;       %
end

if j<240+1             %
    sum15=0;
    for i=1:j-1
        sum15=sum15+InR17(i);
    end
    InR20(j)=sum15;        %
end

if j>240               %
    sum15=0;
    for i=j-240:j-1    %
        sum15=sum15+InR17(i);
    end
    InR20(j)=sum15;       %
end

%Basin
if j<12+1
    sum16=0;
    for i=1:j-1
        sum16=sum16+InR21(i);
    end
    InR22(j)=sum16;
end

if j>12
    sum16=0;
    for i=j-12:j-1
        sum16=sum16+InR21(i);
    end
    InR22(j)=sum16;
end

if j<48+1              %
    sum17=0;
```

```
        for i=1:j-1
            sum17=sum17+InR21(i);
        end
        InR23(j)=sum17;        %
    end

    if j>48                 %
        sum17=0;
        for i=j-48:j-1      %
            sum17=sum17+InR21(i);
        end
        InR23(j)=sum17;       %
    end

    if j<261+1              %
        sum18=0;
        for i=1:j-1
            sum18=sum18+InR21(i);
        end
        InR24(j)=sum18;        %
    end

    if j>261                %
        sum18=0;
        for i=j-261:j-1      %
            sum18=sum18+InR21(i);
        end
        InR24(j)=sum18;       %
    end

    if Q1(j)<10
        InF1(j)=1;
    elseif Q1(j)<20
        InF1(j)=2;
    elseif Q1(j)<30
        InF1(j)=3;
    elseif Q1(j)<40
        InF1(j)=4;
    elseif Q1(j)<50
        InF1(j)=5;
    elseif Q1(j)<60
        InF1(j)=6;
    elseif Q1(j)<70
        InF1(j)=7;
    elseif Q1(j)<80
        InF1(j)=8;
    elseif Q1(j)<90
        InF1(j)=9;
    elseif Q1(j)<100
        InF1(j)=10;
    else
        InF1(j)=11;
    end


end
```

```
if isrow(InR2)==1 InR2=InR2'; end
if isrow(InR3)==1 InR3=InR3'; end
if isrow(InR4)==1 InR4=InR4'; end
if isrow(InR6)==1 InR6=InR6'; end
if isrow(InR7)==1 InR7=InR7'; end
if isrow(InR8)==1 InR8=InR8';end
if isrow(InR10)==1 InR10=InR10'; end
if isrow(InR11)==1 InR11=InR11';end
if isrow(InR12)==1 InR12=InR12'; end
if isrow(InR14)==1 InR14=InR14'; end
if isrow(InR15)==1 InR15=InR15'; end
if isrow(InR16)==1 InR16=InR16'; end
if isrow(InR18)==1 InR18=InR18'; end
if isrow(InR19)==1 InR19=InR19'; end
if isrow(InR20)==1 InR20=InR20'; end
if isrow(InR22)==1 InR22=InR22'; end
if isrow(InR23)==1 InR23=InR23'; end
if isrow(InR24)==1 InR24=InR24'; end
if isrow(InF1)==1 InF1=InF1';end


Q2=lagmatrix(Q1,-1);
Q3=lagmatrix(Q1,-2);
Q4=lagmatrix(Q1,-3);
Q5=lagmatrix(Q1,-4);
Q6=lagmatrix(Q1,-5);
Q7=lagmatrix(Q1,-6);
Q8=lagmatrix(Q1,-7);
Q9=lagmatrix(Q1,-8);
Q10=lagmatrix(Q1,-9);
Q11=lagmatrix(Q1,-10);
Q12=lagmatrix(Q1,-11);



%%
InputData=    [InF1';...

InR2';InR4';InR6';InR8';InR10';InR12';InR14';InR16';InR18';InR20';In
R22';InR24';...
    InR3';InR7';InR11';InR15';InR19';InR23';...
    InL1';InL2'];



inputSeries = tonndata(InputData,true,false);
targetSeries{1} = tonndata(Q1',true,false);
targetSeries{2} = tonndata(Q2',true,false);
targetSeries{3} = tonndata(Q3',true,false);
targetSeries{4} = tonndata(Q4',true,false);
targetSeries{5} = tonndata(Q5',true,false);
targetSeries{6} = tonndata(Q6',true,false);
targetSeries{7} = tonndata(Q7',true,false);
targetSeries{8} = tonndata(Q8',true,false);
targetSeries{9} = tonndata(Q9',true,false);
targetSeries{10} = tonndata(Q10',true,false);
targetSeries{11} = tonndata(Q11',true,false);
targetSeries{12} = tonndata(Q12',true,false);
```

```
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
%trainFcn = 'trainbr';

% Create a Nonlinear Autoregressive Network with External Input
inputDelays = 1:5;  % 1:5 input delay is the best
feedbackDelays = 1:10; % 1:15 is the best
hiddenLayerSize = 11; %11 the best

save
%%

% Data Filtering   : final selection
Step 2
fc = 10;
fs = 100;

[b,a] = butter(2,fc/(fs/2));
dataIn = Qnew1;
dataOut = filter(b,a,dataIn);
dataOut=lagmatrix(dataOut,-3);
title('Inflow to Upper Kotmale vs Time')
plot(TimeV,dataIn,'r');
hold on;
plot(TimeV,dataOut);
xlabel('Time')
ylabel('Inflow (m3/s)')
legend({'Before Filtering','After Filtering'})


%%
12/21/19 7:04 AM  F:\2. Results\Matla...\OptimizeModels.m

load MSE;


for m= 1:1    %1:12
    tic
    for i=1:1  %  5
        %net{m} = init(net{m});
        net{m} =
narxnet(inputDelays,feedbackDelays,hiddenLayerSize,'open',trainFcn);
        net{m}.layers{1}.transferFcn = 'tansig';
        net{m}.layers{2}.transferFcn = 'purelin';

        % Prepare the Data for Training and Simulation
        [inputs,inputStates,layerStates,targets] =
preparets(net{m},inputSeries,{},targetSeries{m});

        %net1.divideFcn = 'divideblock';
        net{m}.divideFcn = 'divideint';
        net{m}.divideParam.trainRatio = 80/100;
        net{m}.divideParam.valRatio = 10/100;
        net{m}.divideParam.testRatio = 10/100;

        % Choose a Performance Function
```

```
        % For a list of all performance functions type: help
nnperformance
        net{m}.performFcn = 'mse';  % Mean Squared Error

        % Choose Plot Functions
        % For a list of all plot functions type: help nnplot
        net{m}.plotFcns = {'plotperform','plottrainstate',
'ploterrhist', ...
            'plotregression', 'plotresponse', 'ploterrcorr',
'plotinerrcorr'};

        % Train the Network
        %net.trainParam.showWindow=false;  %hide training window
        [net{m},tr,A,Es] =
train(net{m},inputs,targets,inputStates,layerStates);

        % Test the Network
        outputs = net{m}(inputs,inputStates,layerStates);
        errors = gsubtract(targets,outputs);
        performance = perform(net{m},targets,outputs);

        ts        = cell2mat(targets);
        MSE00         = var(ts,1);
        %R2(1) = 1-mse(Es)/MSE00
        % View the Network
        %view(net1);

        %  TS = size(targets1,2);
        %
plot(1:TS,cell2mat(targets1),'b',1:TS,cell2mat(outputs1),'r')

        if performance<MSE(m)
            MSE(m)=performance;
            R2(m)=1-mse(Es)/MSE00 ;
            fprintf('m = %d MSE = %d ', m,performance);
            pause(0.5);
            fprintf(' Done. \n');
            MODEL{m}=net{m};
            save MODEL;
            save MSE;
        end
    end
    toc
end
```

12/21/19 7:05 AM F:\2. Results\Matl...\ForecastAhead.m

```
%% Direct-Recursive Hybrid Strategies
%
% prediction(t+1) = model1(obs(t-1), obs(t-2), …, obs(t-n))
% prediction(t+2) = model2(prediction(t+1), obs(t-1), …, obs(t-n))
%
% 1. Use train data to train model1
% 2. Predict t+1 for all train data
% 3. Use predicted t+1 plus train data to train model2
```

```
%%
load MODEL;

n =size(Qnew1,1); % get number of data points

%InR2 = ComNE last 6hr RF
%InR3 = ComNE last 24hr RF
%InR4 = ComNE last 5.4days RF
InewR2(n)=0; InewR3(n)=0; InewR4(n)=0;
InewR6(n)=0; InewR7(n)=0; InewR8(n)=0;
InewR10(n)=0; InewR11(n)=0; InewR12(n)=0;
InewR14(n)=0; InewR15(n)=0; InewR16(n)=0;
InewR18(n)=0; InewR19(n)=0; InewR20(n)=0;
InewR22(n)=0; InewR23(n)=0; InewR24(n)=0;
InewF1(n)=0;
sum1=0;sum2=0;sum3=0;
sum4=0;sum5=0;sum6=0;
sum7=0;sum8=0;sum9=0;
sum10=0;sum11=0;sum12=0;
sum13=0;sum14=0;sum15=0;
sum16=0;sum17=0;sum18=0;
for j=1:n

    % Nuwara Eliya
    if j<12+1
        sum1=0;
        for i=1:j-1
            sum1=sum1+InewR1(i);
        end
        InewR2(j)=sum1;
    end

    if j>12
        sum1=0;
        for i=j-12:j-1
            sum1=sum1+InewR1(i);
        end
        InewR2(j)=sum1;
    end

    if j<48+1              %
        sum2=0;
        for i=1:j-1
            sum2=sum2+InewR1(i);
        end
        InewR3(j)=sum2;        %
    end

    if j>48                %
        sum2=0;
        for i=j-48:j-1     %
            sum2=sum2+InewR1(i);
        end
        InewR3(j)=sum2;      %
    end

    if j<259+1             %
```

```
    sum3=0;
    for i=1:j-1
        sum3=sum3+InewR1(i);
    end
    InewR4(j)=sum3;        %
end

if j>259                  %
    sum3=0;
    for i=j-259:j-1      %
        sum3=sum3+InewR1(i);
    end
    InewR4(j)=sum3;       %
end

%Ambewela
if j<12+1
    sum4=0;
    for i=1:j-1
        sum4=sum4+InewR5(i);
    end
    InewR6(j)=sum4;
end

if j>12
    sum4=0;
    for i=j-12:j-1
        sum4=sum4+InewR5(i);
    end
    InewR6(j)=sum4;
end

if j<48+1              %
    sum5=0;
    for i=1:j-1
        sum5=sum5+InewR5(i);
    end
    InewR7(j)=sum5;        %
end

if j>48                  %
    sum5=0;
    for i=j-48:j-1      %
        sum5=sum5+InewR5(i);
    end
    InewR7(j)=sum5;       %
end

if j<363+1               %
    sum6=0;
    for i=1:j-1
        sum6=sum6+InewR5(i);
    end
    InewR8(j)=sum6;        %
end

if j>363                 %
    sum6=0;
```

```
    for i=j-363:j-1     %
        sum6=sum6+InewR5(i);
    end
    InewR8(j)=sum6;      %
end

%Sandrigham
if j<12+1
    sum7=0;
    for i=1:j-1
        sum7=sum7+InewR9(i);
    end
    InewR10(j)=sum7;
end

if j>12
    sum7=0;
    for i=j-12:j-1
        sum7=sum7+InewR9(i);
    end
    InewR10(j)=sum7;
end

if j<48+1             %
    sum8=0;
    for i=1:j-1
        sum8=sum8+InewR9(i);
    end
    InewR11(j)=sum8;       %
end

if j>48               %
    sum8=0;
    for i=j-48:j-1     %
        sum8=sum8+InewR9(i);
    end
    InewR11(j)=sum8;      %
end

if j<380+1            %
    sum9=0;
    for i=1:j-1
        sum9=sum9+InewR9(i);
    end
    InewR12(j)=sum9;       %
end

if j>380              %
    sum9=0;
    for i=j-380:j-1     %
        sum9=sum9+InewR9(i);
    end
    InewR12(j)=sum9;      %
end

%Calidonia
if j<12+1
    sum10=0;
```

```
    for i=1:j-1
        sum10=sum10+InewR13(i);
    end
    InewR14(j)=sum10;
end

if j>12
    sum10=0;
    for i=j-12:j-1
        sum10=sum10+InewR13(i);
    end
    InewR14(j)=sum10;
end

if j<48+1              %
    sum11=0;
    for i=1:j-1
        sum11=sum11+InewR13(i);
    end
    InewR15(j)=sum11;        %
end

if j>48                %
    sum11=0;
    for i=j-48:j-1     %
        sum11=sum11+InewR13(i);
    end
    InewR15(j)=sum11;      %
end

if j<400+1             %
    sum12=0;
    for i=1:j-1
        sum12=sum12+InewR13(i);
    end
    InewR16(j)=sum12;        %
end

if j>400               %
    sum12=0;
    for i=j-400:j-1    %
        sum12=sum12+InewR13(i);
    end
    InewR16(j)=sum12;      %
end
%Talawakelle
if j<12+1
    sum13=0;
    for i=1:j-1
        sum13=sum13+InewR17(i);
    end
    InewR18(j)=sum13;
end

if j>12
    sum13=0;
    for i=j-12:j-1
        sum13=sum13+InewR17(i);
```

```
    end
    InewR18(j)=sum13;
end

if j<48+1              %
    sum14=0;
    for i=1:j-1
        sum14=sum14+InewR17(i);
    end
    InewR19(j)=sum14;        %
end

if j>48                %
    sum14=0;
    for i=j-48:j-1     %
        sum14=sum14+InewR17(i);
    end
    InewR19(j)=sum14;       %
end

if j<240+1              %
    sum15=0;
    for i=1:j-1
        sum15=sum15+InewR17(i);
    end
    InewR20(j)=sum15;        %
end

if j>240                %
    sum15=0;
    for i=j-240:j-1     %
        sum15=sum15+InewR17(i);
    end
    InewR20(j)=sum15;       %
end

%Basin
if j<12+1
    sum16=0;
    for i=1:j-1
        sum16=sum16+InewR21(i);
    end
    InewR22(j)=sum16;
end

if j>12
    sum16=0;
    for i=j-12:j-1
        sum16=sum16+InewR21(i);
    end
    InewR22(j)=sum16;
end

if j<48+1              %
    sum17=0;
    for i=1:j-1
        sum17=sum17+InewR21(i);
    end
```

```
        InewR23(j)=sum17;        %
    end

    if j>48                 %
        sum17=0;
        for i=j-48:j-1      %
            sum17=sum17+InewR21(i);
        end
        InewR23(j)=sum17;       %
    end

    if j<261+1              %
        sum18=0;
        for i=1:j-1
            sum18=sum18+InewR21(i);
        end
        InewR24(j)=sum18;        %
    end

    if j>261                %
        sum18=0;
        for i=j-261:j-1     %
            sum18=sum18+InewR21(i);
        end
        InewR24(j)=sum18;       %
    end

    if Qnew1(j)<10
        InewF1(j)=1;
    elseif Qnew1(j)<20
        InewF1(j)=2;
    elseif Qnew1(j)<30
        InewF1(j)=3;
    elseif Qnew1(j)<40
        InewF1(j)=4;
    elseif Qnew1(j)<50
        InewF1(j)=5;
    elseif Qnew1(j)<60
        InewF1(j)=6;
    elseif Qnew1(j)<70
        InewF1(j)=7;
    elseif Qnew1(j)<80
        InewF1(j)=8;
    elseif Qnew1(j)<90
        InewF1(j)=9;
    elseif Qnew1(j)<100
        InewF1(j)=10;
    else
        InewF1(j)=11;
    end


end

if isrow(InewR2)==1 InewR2=InewR2'; end
if isrow(InewR3)==1 InewR3=InewR3'; end
if isrow(InewR4)==1 InewR4=InewR4'; end
```

```
if isrow(InewR6)==1 InewR6=InewR6'; end
if isrow(InewR7)==1 InewR7=InewR7'; end
if isrow(InewR8)==1 InewR8=InewR8';end
if isrow(InewR10)==1 InewR10=InewR10'; end
if isrow(InewR11)==1 InewR11=InewR11';end
if isrow(InewR12)==1 InewR12=InewR12'; end
if isrow(InewR14)==1 InewR14=InewR14'; end
if isrow(InewR15)==1 InewR15=InewR15'; end
if isrow(InewR16)==1 InewR16=InewR16'; end
if isrow(InewR18)==1 InewR18=InewR18'; end
if isrow(InewR19)==1 InewR19=InewR19'; end
if isrow(InewR20)==1 InewR20=InewR20'; end
if isrow(InewR22)==1 InewR22=InewR22'; end
if isrow(InewR23)==1 InewR23=InewR23'; end
if isrow(InewR24)==1 InewR24=InewR24'; end
if isrow(InewF1)==1 InewF1=InewF1';end


Qnew2=lagmatrix(Qnew1,-1);
Qnew3=lagmatrix(Qnew1,-2);
Qnew4=lagmatrix(Qnew1,-3);
Qnew5=lagmatrix(Qnew1,-4);
Qnew6=lagmatrix(Qnew1,-5);
Qnew7=lagmatrix(Qnew1,-6);
Qnew8=lagmatrix(Qnew1,-7);
Qnew9=lagmatrix(Qnew1,-8);
Qnew10=lagmatrix(Qnew1,-9);
Qnew11=lagmatrix(Qnew1,-10);
Qnew12=lagmatrix(Qnew1,-11);


%%
inputNS = [InewF1';...
    InewR2';InewR3';InewR4';InewR6';InewR7';InewR8';...
    InewR10';InewR11';InewR12';InewR14';InewR15';InewR16';...
    InewR18';InewR19';InewR20';InewR22';InewR23';InewR24';...
    InewL1';InewL2'];


targetNS1 = Qnew1'; targetNS2 = Qnew2'; targetNS3 = Qnew3';
targetNS4 = Qnew4';
targetNS5 = Qnew5'; targetNS6 = Qnew6'; targetNS7 = Qnew7';
targetNS8 = Qnew8';
targetNS9 = Qnew9'; targetNS10 = Qnew10'; targetNS11 = Qnew11';
targetNS12 = Qnew12';

inputNS = tonndata(inputNS,true,false);

targetNS1 = tonndata(targetNS1,true,false); targetNS2 =
tonndata(targetNS2,true,false);
targetNS3 = tonndata(targetNS3,true,false); targetNS4 =
tonndata(targetNS4,true,false);
targetNS5 = tonndata(targetNS5,true,false); targetNS6 =
tonndata(targetNS6,true,false);
targetNS7 = tonndata(targetNS7,true,false); targetNS8 =
tonndata(targetNS8,true,false);
targetNS9 = tonndata(targetNS9,true,false); targetNS10 =
tonndata(targetNS10,true,false);
```

```
targetNS11 = tonndata(targetNS11,true,false); targetNS12 =
tonndata(targetNS12,true,false);

Forecast(1:12)=0;


%%
numTimesteps = size(inputNS,2);
knownOutputTimesteps = 1:numTimesteps;
x = inputNS(1,knownOutputTimesteps);

%% Forecast 30min ahead
t1 = targetNS1(1,knownOutputTimesteps);
% nets1 = removedelay(MODEL{1});
% adaptFcn   =  net1.adaptFcn;
% adaptParam =  net1.adaptParam;

[Xs1,Xis1,Ais1,ts1] = preparets(MODEL{1},x,{},t1);
%net1=init(MODEL{1});
% net1.trainparam.min_grad=1e-30;

%
%[net1  Ys Es Xf Yf tr]  = adapt(net1,Xs1,ts1,Xis1,Ais1);
[net1] = train(MODEL{1},Xs1,ts1,Xis1,Ais1);

nets1 = removedelay(net1);
[Xs1,Xis1,Ais1,ts1] = preparets(nets1,x,{},t1);
Ys1 = nets1(Xs1,Xis1,Ais1);
Forecast(1)=Ys1{end}

%% Forecast 1hour ahead

t2 = targetNS2(1,knownOutputTimesteps);
t2{end}=Forecast(1);
%t2{end}=158;
%
%  nets2 = removedelay(MODEL21111NET2);

[Xs2,Xis2,Ais2,ts2] = preparets(MODEL{2},x,{},t2);
% net2=init(MODEL21111NET2);
% net2.trainparam.min_grad=1e-30;


[net2] = train(MODEL{2},Xs2,ts2,Xis2,Ais2);

nets2 = removedelay(net2);
[Xs2,Xis2,Ais2] = preparets(nets2,x,{},t2);
Ys2 = nets2(Xs2,Xis2,Ais2);
Forecast(2)=Ys2{end}
%% Forecast 1.5hour ahead

t3 = targetNS3(1,knownOutputTimesteps);
t3{end}=Forecast(2);
t3{end-1}=Forecast(1);

% nets3 = removedelay(MODEL21111NET3);

[Xs3,Xis3,Ais3,ts3] = preparets(MODEL{3},x,{},t3);
```

```
%net3=init(MODEL21111NET3);
% net3.trainparam.min_grad=1e-30;

[net3] = train(MODEL{3},Xs3,ts3,Xis3,Ais3);

nets3 = removedelay(net3);
[Xs3,Xis3,Ais3] = preparets(nets3,x,{},t3);
Ys3 = nets3(Xs3,Xis3,Ais3);
Forecast(3)=Ys3{end}

%% Forecast 2.0hour ahead

t4 = targetNS4(1,knownOutputTimesteps);
t4{end}=Forecast(3);
t4{end-1}=Forecast(2);
t4{end-2}=Forecast(1);

%nets4 = removedelay(MODEL21111NET4);
[Xs4,Xis4,Ais4,ts4] = preparets(MODEL{4},x,{},t4);
% net4=init(MODEL21111NET4);
% net4.trainparam.min_grad=1e-30;
[net4] = train(MODEL{4},Xs4,ts4,Xis4,Ais4);

nets4 = removedelay(net4);
[Xs4,Xis4,Ais4] = preparets(nets4,x,{},t4);
Ys4 = nets4(Xs4,Xis4,Ais4);
Forecast(4)=Ys4{end}


%% Forecast 2.5hour ahead

t5 = targetNS5(1,knownOutputTimesteps);
t5{end}=Forecast(4);
t5{end-1}=Forecast(3);
t5{end-2}=Forecast(2);
t5{end-3}=Forecast(1);

%nets5 = removedelay(MODEL21111NET5);
[Xs5,Xis5,Ais5,ts5] = preparets(MODEL{5},x,{},t5);
% net5=init(MODEL21111NET5);
% net5.trainparam.min_grad=1e-30;
[net5] = train(MODEL{5},Xs5,ts5,Xis5,Ais5);

nets5 = removedelay(net5);
[Xs5,Xis5,Ais5] = preparets(nets5,x,{},t5);
Ys5 = nets5(Xs5,Xis5,Ais5);
Forecast(5)=Ys5{end}


%% Forecast 3.0hour ahead

t6 = targetNS6(1,knownOutputTimesteps);
t6{end}=Forecast(5);
t6{end-1}=Forecast(4);
t6{end-2}=Forecast(3);
t6{end-3}=Forecast(2);
t6{end-4}=Forecast(1);
```

```
%nets6 = removedelay(MODEL21111NET6);
[Xs6,Xis6,Ais6,ts6] = preparets(MODEL{6},x,{},t6);
%net6=init(MODEL21111NET6);
% net6.trainparam.min_grad=1e-30;
[net6] = train(MODEL{6},Xs6,ts6,Xis6,Ais6);

nets6 = removedelay(net6);
[Xs6,Xis6,Ais6] = preparets(nets6,x,{},t6);
Ys6 = nets6(Xs6,Xis6,Ais6);
Forecast(6)=Ys6{end}

%% Forecast 3.5hour ahead

t7 = targetNS7(1,knownOutputTimesteps);
t7{end}=Forecast(6);
t7{end-1}=Forecast(5);
t7{end-2}=Forecast(4);
t7{end-3}=Forecast(3);
t7{end-4}=Forecast(2);
t7{end-5}=Forecast(1);

%nets7 = removedelay(MODEL21111NET7);
[Xs7,Xis7,Ais7,ts7] = preparets(MODEL{7},x,{},t7);
% net7=init(MODEL21111NET7);
% net7.trainparam.min_grad=1e-30;
[net7] = train(MODEL{7},Xs7,ts7,Xis7,Ais7);

nets7 = removedelay(net7);
[Xs7,Xis7,Ais7] = preparets(nets7,x,{},t7);
Ys7 = nets7(Xs7,Xis7,Ais7);
Forecast(7)=Ys7{end}

%% Forecast 4.0hour ahead

t8 = targetNS8(1,knownOutputTimesteps);
t8{end}=Forecast(7);
t8{end-1}=Forecast(6);
t8{end-2}=Forecast(5);
t8{end-3}=Forecast(4);
t8{end-4}=Forecast(3);
t8{end-5}=Forecast(2);
t8{end-6}=Forecast(1);

%nets8 = removedelay(MODEL21111NET8);
[Xs8,Xis8,Ais8,ts8] = preparets(MODEL{8},x,{},t8);
% net8=init(MODEL21111NET8);
% net8.trainparam.min_grad=1e-30;
[net8] = train(MODEL{8},Xs8,ts8,Xis8,Ais8);

nets8 = removedelay(net8);
[Xs8,Xis8,Ais8] = preparets(nets8,x,{},t8);
Ys8 = nets8(Xs8,Xis8,Ais8);
Forecast(8)=Ys8{end}

%% Forecast 4.5hour ahead

t9 = targetNS9(1,knownOutputTimesteps);
t9{end}=Forecast(8);
```

```
t9{end-1}=Forecast(7);
t9{end-2}=Forecast(6);
t9{end-3}=Forecast(5);
t9{end-4}=Forecast(4);
t9{end-5}=Forecast(3);
t9{end-6}=Forecast(2);
t9{end-7}=Forecast(1);

%nets9 = removedelay(MODEL21111NET9);
[Xs9,Xis9,Ais9,ts9] = preparets(MODEL{9},x,{},t9);
%  net9=init(MODEL21111NET9);
% % net9.trainparam.min_grad=1e-30;
[net9] = train(MODEL{9},Xs9,ts9,Xis9,Ais9);

nets9 = removedelay(net9);
[Xs9,Xis9,Ais9] = preparets(nets9,x,{},t9);
Ys9 = nets9(Xs9,Xis9,Ais9);
Forecast(9)=Ys9{end}

%% Forecast 5.0hour ahead

t10 = targetNS10(1,knownOutputTimesteps);
t10{end}=Forecast(9);
t10{end-1}=Forecast(8);
t10{end-2}=Forecast(7);
t10{end-3}=Forecast(6);
t10{end-4}=Forecast(5);
t10{end-5}=Forecast(4);
t10{end-6}=Forecast(3);
t10{end-7}=Forecast(2);
t10{end-8}=Forecast(1);

% %nets10 = removedelay(MODEL21111NET10);
[Xs10,Xis10,Ais10,ts10] = preparets(MODEL{10},x,{},t10);
%  net10=init(MODEL21111NET10);
% % net10.trainparam.min_grad=1e-30;
[net10] = train(MODEL{10},Xs10,ts10,Xis10,Ais10);

nets10 = removedelay(net10);
[Xs10,Xis10,Ais10] = preparets(nets10,x,{},t10);
Ys10 = nets10(Xs10,Xis10,Ais10);
Forecast(10)=Ys10{end}

%% Forecast 5.5hour ahead

t11 = targetNS11(1,knownOutputTimesteps);
t11{end}=Forecast(10);
t11{end-1}=Forecast(9);
t11{end-2}=Forecast(8);
t11{end-3}=Forecast(7);
t11{end-4}=Forecast(6);
t11{end-5}=Forecast(5);
t11{end-6}=Forecast(4);
t11{end-7}=Forecast(3);
t11{end-8}=Forecast(2);
t11{end-9}=Forecast(1);
```

```
% %nets11 = removedelay(MODEL21111NET11);
[Xs11,Xis11,Ais11,ts11] = preparets(MODEL{11},x,{},t11);
%  net11=init(MODEL21111NET11);
% % net11.trainparam.min_grad=1e-30;
[net11] = train(MODEL{11},Xs11,ts11,Xis11,Ais11);

nets11 = removedelay(net11);
[Xs11,Xis11,Ais11] = preparets(nets11,x,{},t11);
Ys11 = nets11(Xs11,Xis11,Ais11);
Forecast(11)=Ys11{end}

%% Forecast 6.0hour ahead

t12 = targetNS12(1,knownOutputTimesteps);
t12{end}=Forecast(11);
t12{end-1}=Forecast(10);
t12{end-2}=Forecast(9);
t12{end-3}=Forecast(8);
t12{end-4}=Forecast(7);
t12{end-5}=Forecast(6);
t12{end-6}=Forecast(5);
t12{end-7}=Forecast(4);
t12{end-8}=Forecast(3);
t12{end-9}=Forecast(2);
t12{end-10}=Forecast(1);

% %nets12 = removedelay(MODEL21111NET12);
[Xs12,Xis12,Ais12,ts12] = preparets(MODEL{12},x,{},t12);
%  net12=init(MODEL21111NET12);
% % net12.trainparam.min_grad=1e-30;
[net12] = train(MODEL{12},Xs12,ts12,Xis12,Ais12);

nets12 = removedelay(net12);
[Xs12,Xis12,Ais12] = preparets(nets12,x,{},t12);
Ys12 = nets12(Xs12,Xis12,Ais12);
Forecast(12)=Ys12{end}

%%

Forecast(1:12)
```

12/21/19 7:06 AM F:\2. Results\M...\Discharge to Power.m

```
function [P] = Discharge_to_Power(q, DamLevel
,TailraceLevel,SyncUnits)
% q : Turbine Discharge (m3/s)
% DamLevel : Prsent Dm Level (masl)
% TailraceLevel : Present TailraceLevel (masl)
% SyncUnits : Number of synchronized Units

A = DamLevel - TailraceLevel;
C = SyncUnits;

if A<490 && C==1;
```

```
    a11 = 0.00000719;      a12 = 0.00000699;  b11 = -0.00066977;
b12 = -0.00067913;
    c11 = 0.22347804;      c12 = 0.22235891;  d11 = 1.48000195;
d12 = 1.48062322;

    %a*x^3 + b*x^2 + c*x + d
    a = (490-A)*a11 + (A-487)*a12;
    b = (490-A)*b11 + (A-487)*b12;
    c = (490-A)*c11 + (A-487)*c12;
    d = (490-A)*d11 + (A-487)*d12 -3.0*q;

    x = roots([a b c d]);
    x(imag(vpa(x))~=0) = [];
    P=x;

elseif A>=490 && C==1;

    a13 = 0.00000675;      a12 = 0.00000699;  b13 = -0.00068570;
b12 = -0.00067913;
    c13 = 0.22091472;      c12 = 0.22235891;  d13 = 1.48250266;
d12 = 1.48062322;

    %a*x^3 + b*x^2 + c*x + d
    a = (493.73-A)*a12 + (A-490)*a13;
    b = (493.73-A)*b12 + (A-490)*b13;
    c = (493.73-A)*c12 + (A-490)*c13;
    d = (493.73-A)*d12 + (A-490)*d13 -3.73*q;

    x = roots([a b c d]);
    x(imag(vpa(x))~=0) = [];
    P=x;


elseif A<490 && C==2;

    a21 = 0.00000996;      a22 = 0.00000960;  b21 = -0.00085761;
b22 = -0.00083571;
    c21 = 0.22692553;      c22 = 0.22551339;  d21 = 1.47198975;
d22 = 1.47409836;

    %a*x^3 + b*x^2 + c*x + d
    a = (490-A)*a21 + (A-487)*a22;
    b = (490-A)*b21 + (A-487)*b22;
    c = (490-A)*c21 + (A-487)*c22;
    d = (490-A)*d21 + (A-487)*d22 -3.0*q;

    x = roots([a b c d]);
    x(imag(vpa(x))~=0) = [];
    P=x;

else A>=490 && C==2;

    a23 = 0.00000919;      a22 = 0.00000960;  b23 = -0.00081176;
b22 = -0.00083571;
    c23 = 0.22394178;      c22 = 0.22551339;  d23 = 1.47548317;
d22 = 1.47409836;

    %a*x^3 + b*x^2 + c*x + d
```

```
    a = (493.73-A)*a22 + (A-490)*a23;
    b = (493.73-A)*b22 + (A-490)*b23;
    c = (493.73-A)*c22 + (A-490)*c23;
    d = (493.73-A)*d22 + (A-490)*d23 -3.73*q;

    x = roots([a b c d]);
    x(imag(vpa(x))~=0) = [];
    P=x;

end
```

12/21/19 7:06 AM F:\2. Results\Matlab ...\LevelStorage.m

```
function [LS] = LevelStorage(flag,LevelStorage)

% flag : if flag =1  then function returns Storage given input as
Level
%        if flag =0  then function returns Level given input as
Storage
% Storage is above 1190 level storage

X = LevelStorage;
a = 3.213; b= -72.56; c= 650.4; d= -2958; e=7326; f= -9374; g=17760;
h= 151500;


if flag == 1 ;
    Le = (X-1190);


    f = a*Le^8 + b*Le^7 + c*Le^6 + d*Le^5 + e*Le^4 + f*Le^3 + g*Le^2
+ h*Le;
    LS =f;
else
    % given storage(above 1190) find Level
    % X = a*X^8 + b*X^7 + c*X^6 + d*X^5 + e*X^4 + f*X^3 + g*X^2 +
h*X;

    x = roots([a b c d e f g h -X]);
    B = x(real(x) >= 0 & imag(x) == 0);
    LS=B+1190;

end
```

APPENDIX-B: MATLAB PROGRAMMES OF POM

Pond Optimization Model

12/21/19 7:07 AM F:\2. Results\Matlab Files\New\main1.m

```
clc; clear;

%% Setup the Import Options
opts = spreadsheetImportOptions("NumVariables", 2);

% Specify sheet and range
opts.Sheet = "2018 Vertical";
opts.DataRange = "A2:B8749";

% Specify column names and types
opts.VariableNames = ["Time", "Inflow"];
opts.VariableTypes = ["datetime", "double"];

% Import the data
tbl = readtable("F:\2. Results\Matlab
Files\New\OptimizedResults.xlsx", opts, "UseExcel", false);

% Convert to output type
Time = tbl.Time;
Inflow = tbl.Inflow;

% Clear temporary variables
clear opts tbl

%%

%% Input Inflow data  (m3/s)
numInflow = size(Inflow,1);
optimizeperiod = 4;

P(numInflow)=0;
PP(numInflow)=0;
SP(numInflow)=0;
SPSP(numInflow)=0;
S(numInflow)=0;
SS(numInflow)=0;
%%

for k=1:optimizeperiod:numInflow-20

    [a,b1] = getData1(k,Inflow,Time);
    for m=1:24
        inflow(m)=b1(m);
    end

    TimeHour = hour(a(1));
```

```
    % Input inflow convert to m3
    for j=1:24
        I(j) = inflow(j)*3600;
    end
    %%

    %% Initial values
    InitialLevel= 1192.38;
    s0 = LevelStorage(1,InitialLevel);   % Initial Storage (m3)
1193.14 masl
    sm = 822470;  % Maximum Storage (m3)  1194.00 masl
    x0 = [26000*ones(24,1);zeros(24,1);s0*ones(24,1)];  % Initial
Values

    %% Linear Parameters
    A=[]; b=[];


    %%
    Aeq = [1 zeros(1,23) 1 zeros(1,23) 1 zeros(1,23)];
    Aeq = [Aeq ; zeros(1,1) 1 zeros(1,22)   zeros(1,1) 1 zeros(1,22)
-1 1 zeros(1,22)];
    Aeq = [Aeq ; zeros(1,2) 1 zeros(1,21)   zeros(1,2) 1 zeros(1,21)
zeros(1,1) -1 1 zeros(1,21)];
    Aeq = [Aeq ; zeros(1,3) 1 zeros(1,20)   zeros(1,3) 1 zeros(1,20)
zeros(1,2) -1 1 zeros(1,20)];
    for jj=5:24
        Aeq = [Aeq ; zeros(1,jj-1) 1 zeros(1,24-jj)   zeros(1,jj-1)
1 zeros(1,24-jj)    zeros(1,jj-2) -1 1 zeros(1,24-jj)];
    end

    %
    beq = s0 + I(1);
    beq = [beq ; I(2)];
    for mm=3:24
        beq = [beq ; I(mm)];
    end

    %% Boundry Constarints
    LB = [25295*ones(24,1);zeros(48,1)];

    if (TimeHour <= 5)
        LB = [25295*ones(24,1);zeros(5+1-
TimeHour,1);4788*ones(10,1);zeros(10-1+TimeHour-1,1);zeros(24,1)];
% must have positive flow
    elseif (TimeHour >= 15)
        LB = [25295*ones(24,1);zeros(14+5+1-
TimeHour,1);4788*ones(10,1);zeros(TimeHour-15,1);zeros(24,1)];     %
must have positive flow
    elseif (TimeHour >= 6)
        LB = [25295*ones(24,1);4788*ones(10-
TimeHour+5,1);zeros(14,1);4788*ones(TimeHour-5,1);zeros(24,1)];
% must have positive flow
    end


    UB = [128160*ones(24,1);Inf*ones(24,1);822470*ones(24,1)];
```

```
    %% Weight initialization of AOA of objective function
    % AOA = Aggregrade Objective Function
    %w1 = 0.89; w2 =0.11;
    w1 = .51; w2 =0.49;
    %% FMINCON
    options =
optimset('MaxFunEval',Inf,'MaxIter',3000,'Algorithm','interior-
point','Display','off');

    %[x, fval] = fmincon(@objfun1,x0,A,b,Aeq,beq, LB,
UB,[],options,w1,w2);

    %% if Global Search is used
    % Comment FMINCON line above before run for global search
    f1 = @(x)objfun1(x,w1,w2);
    problem.options = options;
    problem.solver = 'fmincon';
    problem.objective = f1;
    problem.x0 = x0;
    problem.A = A;
    problem.b = b;
    problem.Aeq = Aeq;
    problem.beq = beq;
    problem.lb = LB;
    problem.ub = UB;
    problem.nonlcon =[];

    gs = GlobalSearch;
    [x, fval] = run(gs, problem);

    %%
    for i= 1 :72
        if i<=24
            P(i) = 4.269663*x(i)/3600;
            PP(k-1+i)=P(i);
        elseif i<= 48
            Sp(i-24) = x(i)/3600;
            SpSp(k-1+i-24)=Sp(i-24);
        else
            S(i-48) = LevelStorage(0,x(i));
            SS(k-1+i-48)=S(i-48);
        end
    end

    for q=1:24
        T(k-1+q)=4.249292*X(q)/3600;
        Sp(k-1+q)=Y(q)/3600;
        S(k-1+q)=Z(q);
        DL(k-1+q)=LevelStorage(0,S(k-1+q));
        if k ==1
            AvgLevel(k-1+q)=(InitialLevel+DL(k-1+q))/2;
        else
            AvgLevel(k-1+q)=(DL(k-1+q-1)+DL(k-1+q))/2;
        end
    end

    pause(0.5);
```

```
    fprintf('Completion : %d', k *100 /numInflow );
    fprintf(' Done. \n');

end
%%
%inflow = [i1;i2;i3;i4];
%plotmine(inflow,Sp,P,S);
plotmine(Inflow,SpSp,PP,SS);
plotbar(Time,Inflow,Sp,T,AvgLevel);

%%
```

12/21/19 7:07 AM F:\2. Results\Matlab Files\...\objfun1.m

```
function [f] = objfun1(x,w1,w2)

% TS = x(5)+x(6)+x(7)+x(8);  % spilling
D =
x(1)+x(2)+x(3)+x(4)+x(5)+x(6)+x(7)+x(8)+x(9)+x(10)+x(11)+x(12)...

+x(13)+x(14)+x(15)+x(16)+x(17)+x(18)+x(19)+x(20)+x(21)+x(22)+x(23)+x
(24);   % generation

SP =
x(25)+x(26)+x(27)+x(28)+x(29)+x(30)+x(31)+x(32)+x(33)+x(34)+x(35)+x(
36)...

+x(37)+x(38)+x(39)+x(40)+x(41)+x(42)+x(43)+x(44)+x(45)+x(46)+x(47)+x
(48);  % Spiling

PS =
x(49)+x(50)+x(51)+x(52)+x(53)+x(54)+x(55)+x(56)+x(57)+x(58)+x(59)+x(
60)...

+x(61)+x(62)+x(63)+x(64)+x(65)+x(66)+x(67)+x(68)+x(69)+x(70)+x(71)+x
(72);  % storage


f = -w1*D-w2*PS+0*SP; %FINAL
```

12/21/19 7:08 AM F:\2. Results\Matlab Files\New\main3.m

```
%% Alternative Optimzation programme for main1 programme

clc; clear;

%% Setup the Import Options
opts = spreadsheetImportOptions("NumVariables", 2);

% Specify sheet and range
opts.Sheet = "2017 Vertical";
opts.DataRange = "A2:B8761";

% Specify column names and types
opts.VariableNames = ["Time", "Inflow"];
```

```
opts.VariableTypes = ["datetime", "double"];

% Import the data
tbl = readtable("F:\2. Results\Matlab
Files\New\OptimizedResults.xlsx", opts, "UseExcel", false);

% Convert to output type
Time = tbl.Time;
Inflow = tbl.Inflow;

% Clear temporary variables
clear opts tbl

%% Input Inflow data  (m3/s)
numInflow = size(Inflow,1);
optimizeperiod = 4;

T(numInflow)=0;
Sp(numInflow)=0;
S(numInflow)=0;
DL(numInflow)=0;
AvgLevel(numInflow)=0;

%% Initial values
InitialLevel= 1193.63;
S0 = LevelStorage(1,InitialLevel);
Smax = 822470;
%x0 = [25416*ones(1,1);zeros(1,1);S0*ones(1,1)];
x0 = [zeros(1,1);zeros(1,1);S0*ones(1,1)];
X(1:24)=0; Y(1:24)=0; Z(1:24)=0; I(1:24)=0;
Z0=S0;
%%
for k=1:optimizeperiod:numInflow-20
    if k>1
        Z0=S(k-1);
    end

    [a,b1] = getData1(k,Inflow,Time);
    %%
    for m=1:24
        I(m)=b1(m)*3600;
    end
    TimeHour = hour(a(1));

    %% Linear Parameters
    A=[]; b=[];
    Aeq = [1  1  1 ];
    %%
    for j=1:24
        %01 Iteration

        if j==1
            beq = Z0+I(1) ;   %s0+I1-sm;
        else
            beq = Z(j-1)+I(j);
        end
        %% Boundry Constarints
```

```
        %LB = [25416*ones(1,1);zeros(1,1);zeros(1,1)]; UB =
[127080*ones(1,1);Inf*ones(1,1);822470*ones(1,1)];
        LB = [zeros(1,1);zeros(1,1);zeros(1,1)]; UB =
[127080*ones(1,1);Inf*ones(1,1);822470*ones(1,1)];
        options =
optimset('MaxFunEval',Inf,'MaxIter',3000,'Algorithm','interior-
point','Display','off');
        [R, fval] = fmincon(@objfun3,x0,A,b,Aeq,beq, LB,
UB,[],options);



        Y(j)= R(2);
        Z(j)= R(3);
        if  R(1)<25416              %I(j)<36000
            if TimeHour<18
                Z(j)=Z(j)+R(1);
                X(j)=0;
            elseif TimeHour>21
                Z(j)=Z(j)+R(1);
                X(j)=0;
            else
                X(j)= R(1);
            end

            if R(2)>0
                X(j)= X(j)+R(2) ;
                Y(j)=0;
            end

        else
            X(j)= R(1);
        end

        if (TimeHour+j)<16
            if (TimeHour+j)>5
                if R(2)< 4788
                    Y(j)=4788;

                    if Z(j)>4788-R(2)
                        Z(j)=Z(j)-(4788-R(2));
                    end
                end
            end
        end


    end



    for q=1:24
        T(k-1+q)=4.249292*X(q)/3600;
        Sp(k-1+q)=Y(q)/3600;
        S(k-1+q)=Z(q);
        DL(k-1+q)=LevelStorage(0,S(k-1+q));
        if k ==1
            AvgLevel(k-1+q)=(InitialLevel+DL(k-1+q))/2;
```

```
        else
            AvgLevel(k-1+q)=(DL(k-1+q-1)+DL(k-1+q))/2;
        end
    end

    %%
    pause(0.5);
    fprintf('Completion : %d', k *100 /numInflow );
    fprintf(' Done. \n');
end

%plotmine(Inflow,SpSp,PP,SS);
plotbar(Time,Inflow,Sp,T,AvgLevel);

%%
```

12/21/19 7:08 AM F:\2. Results\Matlab Files...\getData1.m

```
function [T24,inflow24] = getData1(startIndex,Inflow,Time)

T24 = Time(startIndex:startIndex+23,1);
inflow24 =Inflow(startIndex:startIndex+23,1);

end
```

12/21/19 7:09 AM F:\2. Results\Matlab Files\...\objfun3.m

```
function [f] = objfun3(y)

TG = 0.51*y(1)+0.49*y(3);    % generation

f=-TG;
```

12/21/19 7:09 AM F:\2. Results\Matlab Files\...\plotbar.m

```
function plotbar(Time,Inflow,Sp,P,DL)

%figure;

%% Top plot of Spilling data
ha(1)=subplot(4,1,1);
bar(Time,Inflow,'b');
ylim([0 200]);
grid on
title('Average Hourly Inflow to Pond vs Time');
ylabel('Avg Hourly Inflow (m3/s)');
xlabel('Time (hrs)');
xtickformat('HH:mm');

ha(2)=subplot(4,1,2);
bar(Time,P,'r');
```

```
ylim([0 150]);
grid on
title('Required Average Hourly Loading on the Plant vs Time');
ylabel('Avg Hourly Loading (MW)');
xlabel('Time (hrs)');
xtickformat('HH:mm');

ha(3)=subplot(4,1,3);
bar(Time,Sp,'g');
grid on
%ylim([0 200]);
title('Minimum Average Hourly Spilling Required vs Time');
ylabel('Optimum Avg Hourly Spilling (m3/s)');
xlabel('Time (hrs)');
xtickformat('HH:mm');

ha(4)=subplot(4,1,4);
bar(Time,DL,'m');
grid on
title('Average Hourly Level Change vs Time');
ylim([1190.00 1194.00]);
ytickformat('%.2f')
ylabel('Avg Hourly Level (masl)');
xlabel('Time (hrs)');
xtickformat('HH:mm');


linkaxes(ha, 'x');
```

12/21/19 7:10 AM F:\2. Results\Matlab Files...\plotmine.m

```
function plotmine(Inflow,Sp,P,DL)
%function plotmine(inflow,AP,OP,AL,OL,OS,AS,Time)


% Create a new figure
figure;

%% Top plot of Spilling data
ha(1)=subplot(3,1,1);
yyaxis right
plot(Inflow);
ylim([0 100]);

ylabel('inflow (m3/s)');
hold on;
yyaxis left
plot(Sp,'-b');
ylim([0 100]);

xlabel('Time (hrs)');

ylabel('Required Spilling (m3/s)');
title('Minimum Spilling of the Upper Kotmale Plant');

%% Top plot of Plant Loading
ha(2)=subplot(3,1,2);
```

```
%%
plot(P,'-r');
ylim([0 152]);

% Add labels
xlabel('Time (hrs)');

ylabel('Plant Loading (MW)');
title('Optimum Plant Loading of the Upper Kotmale Plant');
% Add legend in upper left (NorthWest) corner
% legend('Turbine flow','Spill flow','Location','NorthWest');


%% Bottom plot of Level Data
ha(3)=subplot(3,1,3);

plot(DL,'-g');
ylim([1190.00 1194.00]);

xlabel('Time (hrs)');

ylabel('Reservoir Level (masl)');
title('Reservoir Level of the Upper Kotmale Plant');

linkaxes(ha, 'x');
% %
```