ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **23**, Number T, 2020, T57-T77

System Identification Using Long Short Term Memory Recurrent Neural Networks for Real Time Conical Tank System

Febina CHRISTUDAS[1] and Angeline VIJULA DHANRAJ[2]

[1, 2] Department of Instrumentation and Control Systems Engineering,

PSG College of Technology, 641 004Coimbatore, India

Email: cfb.ice@psgtech.ac.in[1], dav.ice@psgtech.ac.in[2]

Abstract. An essential prerequisite for designing model-based controllers is an accurate mathematical model. System Identification is a classical approach for getting a mathematical model from experimental data. Neural Networks are extensively deployed for the identification of systems as they are efficient function approximators. This paper proposes an intelligent technique for modelling a nonlinear Conical Tank System (CTS) using neural networks. Long Short Term Memory Recurrent Neural Networks (LSTM-RNN) are used for modelling the real- time CTS using input-output data. It is shown that LSTM-RNN models are effective in modelling in comparison with empirical models.

Keywords: Artificial neural networks, LSTM, nonlinear system, predictive models, RNN, system identification.

1. Introduction

System modelling is an imperative way of exploring and understanding the physical processes and is the very basis for model based control design for predicting the future behaviour of dynamic plants. Most of the existing real time systems are nonlinear and obtaining a mathematical model for a nonlinear system is a herculean task [1]. Several methods were tried and tested in control field to model real systems. TP (Tensor Product) model-based transformation technique was put into force to model the nonlinear magnetic levitation system along with further research to reduce the approximation errors [2]. An expedient method of identification of non-deterministic process is black box modelling, which uses input and output experimental data sets. The fuzzy based PID controller design using an empirical model was effectively demonstrated for telesurgical robot application. The model was built using the data sets [3]. In any control application, optimum performance is achieved based on the designed plant model which aids in reducing the modelling errors, minimizing the plant/model mismatch as well as increasing the robustness of the system. Literature reveals that empirical modelling approach using artificial neural networks (ANN) was way more remarkable and apt for model-based controller design because of its universal approximation capability. The intelligent models developed using ANN proved to be accurate, reliable, faster and efficient in prediction, modelling and control in medical environment [4].

Later multilayer neural networks were effectively employed for control as well as system identification of nonlinear dynamical systems using back propagation (static & dynamic) methods to adjust the model parameters [5], [6]. ANN model using back propagation algorithm was implemented in model predictive control design to determine the future control inputs [7]. The main benefits of Multilayer-Perceptron (MLP) is found to be its reduced sensitivity to the selection of neural model size. Nonlinear modelling using ANN is considered to guarantee good results but with a limiting factor that the number of nodes and hidden layers needs to be defined

in advance. Several training algorithms were introduced for training the neural networks such as Levenberg-Marquardt, back-propagation, conjugate gradient, ERNEAD algorithms [8]. Gradient descent method when used in training processes resulted in local minima problems. During back propagation the neural network optimization gets stuck in the local minima because of noise that is present in the data. [9].

Considering the downsides in Multilayer Perceptron (MLP) neural architecture, several research works were carried out in Recurrent Neural Networks (RNN) which had good ability to learn the nonlinearity present in time series data. RNN employing back propagation through time algorithm (BPTT) was applied to nonlinear dynamical systems, and the derivatives obtained through this algorithm was used for system identification [10], [11]. But the main drawback is that RNN suffers from vanishing gradient problem.

Later, changes were executed in the neural architecture by increasing the number of layers. Optimal predictive models were built using the training algorithm for deep architectures [12]. Hidden layers were increased instead of nodes and making it easier to unearth a deeply dense connected network [13]. As proficient knowledge is a bare necessity in identifying the nonlinear regressive models, deep neural networks were employed which were comparatively easier to train [14]. Outcomes displayed that system identification using deep neural networks is effective in estimating the models from the available input-output data set, even if noise is present in the data. The LSTM (Long Short Term Memory), which is basically a type of RNN architecture excelling in deep learning using gated units was predominantly used to evade the vanishing gradient problem that exists in conventional RNN networks [15]. System modelling using convex based LSTM neural networks was discussed and was proved that it works much satisfactorily than RNN conventional architecture [16]. Most of the nonlinear systems are being identified using ANN, which demands the use of multiple layers in the neural network and also encounters vanishing gradient problem but realizing the network with LSTM architecture overcomes such problems.

Overfitting is one of the discerning concerns for neural architectures where the performance on the train set is good and continues to improve for an overfit model. The train loss slopes down but the performance on the validation set improves to a vantage point and then starts to degrade. To address this issue, a regularization methodology using the dropout layer is introduced in the neural network architecture which minimizes the possibility of overfitting as it works on decreasing the inter-dependencies between the nodes resulting in self-sustenance [17]. The RNN and LSTM architectures deploying the dropout layer, effectively increased the performance of the model for many applications like speech recognition, machine translation, handwriting recognition [18], [19], [20] but not implemented in control tasks as per the literature analysis. In this paper, LSTM architecture is proposed to obtain the best fit model of laboratory CTS of nonlinear dynamics from its real-time data sets that could be implemented in the model-based controller design for optimal results. This work also proposes using the dropout layer in the LSTM architecture for modelling dynamics in the field of control to suppress overfitting.

To assess the efficacy of the proposed method, the empirical models obtained by three other established techniques from the literature are compared. The models used for comparison are the transfer function (TF) model, nonlinear ARX (autoregressive exogenous) model and NARX (Nonlinear autoregressive exogenous) neural network model. The paper is systematized as follows: Section 2 briefs about the system identification procedure. An idea about the nonlinear ARX model is given in Section 3. Section 4 is about the model identification using NARX neural network. Section 5 elaborates system identification using the RNN-LSTM model. In Section 6, the results are penultimated for different empirical models and conclusion is given in Section 7.

2. System identification

In every modelling task, the subsequent paramount steps have to be executed:

- Collection of data set.
- Selection of proper model set and structure.
- Experimental design.
- Parameter estimation using observations.

- Model validation and evaluation.

Models are categorized into different types based on system characteristics. In parametric models, the model structure is determined based on the physical insight whereas in nonparametric models it is based on the frequency characteristics and impulse response. Selection of model set is determined by the prior information. In experiment design, different excitation signals are tried to excite a system so that sufficient information about the system can be acquired from the available data sets. It is assumed that the system implements the function *f:* $R^N \rightarrow R$ mapping.

The input-output measurement data relationship is given in

$$y(j) = f(x(j)) + n(j), \tag{1}$$

where *n(j)* is the observation noise. The mapping of the system is approximated in some sense by mapping the model $\hat{f}$, and model also implements *f:* $R^N \rightarrow R$ mapping:

$$y_{Mod}(j) = \hat{f}\ (x(j), \Theta), \tag{2}$$

where y_{Mod} is the model output and parameter vector of the model structure is denoted by Θ. The set of observations or measured data $\{x(j), y(j)\}_{j=1}^{P}$ about the system is the essential part for parameter estimation. Parameter estimation is the method of fitting the observations by adjusting the model parameters using a criterion function which is mainly influenced by the prior information. Criterion function ($E(\Theta)$) is the error function between the system output y and the model output y_{Mod}. It is defined as the measure of the quality of the model:

$$E(\Theta) = E\ (y - y_{Mod}(\Theta, M^d)), \tag{3}$$

$$M^d = \{x(j), y(j)\}_{j=1}^{P}, \tag{4}$$

where M^d is the set of measured data pairs, and also:

$$E(\Theta) = \frac{1}{2}\sum_{j=1}^{P}(\ y(j) - y_{Mod}\ (j))^2\ , \tag{5}$$

$$E_{emp}(\Theta) = \frac{1}{P}\sum_{j=1}^{P}(\ y(j) - y_{Mod}(\mathrm{j}))^2, \tag{6}$$

where $E_{emp}(\Theta)$ is the empirical risk which is the average of squared error between the observations and the process model output. Depending on the parameters which are to be estimated, different methods are used namely least square estimation (LS), weighted parameter estimation, Bayes estimation, and Maximum Likelihood (ML) estimation. The quadratic criterion function is commonly applied as it needs only measured input and output data of the plant. Model validation is the ultimate phase of system identification.

Black box modelling (Empirical modelling) approach is used when the prior information about the system is not known other than the observations to build the physical model. The mapping of the black box model is described as $y_{Mod}(j) = \hat{f}\ (x(j), \Theta)$, where $j=1,2.....P$ and Θ is the parameter vector, which is given as $\Theta = [\alpha_1\, \alpha_2\, \ldots\ldots\, \alpha_q]^T$. The general form of this relationship is described as a sum of basis functions $\{g_j(.)\}_j^N$ in terms of

$$y_{Mod}(k) = \sum_{j=1}^{N} \alpha_j\ g_j(x(k)). \tag{7}$$

There are several basis function sets applied in identification where the mapping of input-output measurement data of the system can be approximated by Volterra series, exponentials, polynomials, Fourier series, Taylor expansion, etc. Different empirical models and their architecture are described in the following sections.

3. Nonlinear ARX model

One of the black box model variants is the nonlinear ARX (autoregressive exogenous) model considered for identification of nonlinear systems which consist of nonlinearity estimator and model regressors as shown in Fig. 1. The linear function and nonlinear function of nonlinear estimator act on the designed regressors to predict the model output *y*. Regressors are the delayed outputs *{y(t-1), y(t-2),..}* and delayed inputs *{u(t-1), u(t-2),..}*.The orders and delays of the model are to be specified beforehand. The regressors are mapped to the model output *y* using the nonlinear estimator block.

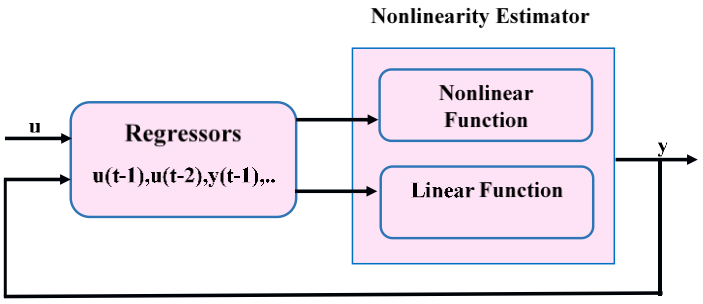

Fig. 1. Structure of the nonlinear ARX model

These regressors will act as an input to the linear as well as the nonlinear function block of nonlinearity estimator. The function *f(x)* is

$$f(x) = M^T (r_x - r_m) + d + g(N (r_x - r_m)), \quad (8)$$

where r_x is the vector of regressors and r_m is the mean of the regressors r_x . M is the model parameter and d denotes the scalar offset. The linear function block output is $M^T (r_x - r_m) + d$ and $g(N (r_x - r_m))$ denotes the nonlinear function block output. The projection matrix N helps in conditioning the calculations. The function $f(x)$ mainly depends on the choice of nonlinearity estimators. Different networks namely wavelet networks, tree partition networks, multilayer neural networks are the available nonlinearity estimators. To model nonlinear complex behaviour, a flexible nonlinear function called wavelet network is used. Wavelet frames are considered as universal approximators for nonlinear system identification. The Levenberg-Marquardt (LM) algorithm is used for solving nonlinear equations for their ability to converge quickly to a solution of the nonlinear least-squares problem A set of n nonlinear equations with m vectors of unknown parameters $(x_1, x_2, \ldots, x_m)$ is given as follows:

$$f_i (x_1, x_2, \ldots, x_m) = 0, i = 1 \ldots n, \quad (9)$$

where $f_i(x)$ is the *i-th* equation. It is assumed that $f(x)=0$, $f(x)=(f_1(x), \ldots, f_n(x))$, where $f: R^m \rightarrow R^n$, $f_1(x), \ldots, f_n(x)$ are differentiable functions of a vector x.

The aim is to find the estimate $\hat{x}$ that minimizes $\|f(x)\|$,

$$\|f(x)\| = (f_1(x)^2 + \ldots\ldots\ldots + f_n(x)^2). \quad (10)$$

The optimality condition is given as

$$\Delta\|f(x)\|^2 = 0. \quad (11)$$

Solving the nonlinear least-squares problem is difficult than solving the linear equation. The nonlinear least- squares problems are well resolved using Levenberg-Marquardt (LM) algorithm [21] as it is an epitome to act more like Gauss-Newton algorithm (GNA) when the parameters are away from the optimal value and also act like gradient-descent method when the parameters are in close proximity to the optimal value [22]. The nonlinear ARX model is built using the real time data of conical tank system using MATLAB and is briefly explained in Section 6.2. Among several empirical modelling approaches with its own drawbacks and advantages, identification using artificial neural networks plays an imperative role, for their intact skill of learning and noble problem solving capability that mimics the neurobiological system.

4. Empirical modelling using NARX neural network

System identification using neural networks are effective in identifying and controlling the nonlinear dynamical system. The adaptive nature of the learning process is the significant feature of the neural networks as it acquires knowledge from its environment. Neural network architecture using a single hidden layer is efficient and serves to be universal function approximator with an adequate degree of accuracy on network learning ability using back propagation (BP), radial basis function and regression neural network (RNN) [23] [24]. ANN also finds its significance in several other applications like pattern recognition [25]. The neural

statistical downscaling and fuzzy statistical downscaling soft computational models were proposed for forecasting the rainfall over the state as it could anticipate the environment conditions [26]. The architecture of the neural network includes an input layer, next is the hidden layer and finally the output layer as depicted in Fig. 2. A neuron model consists of an activation function (nonlinear) and a combiner (linear) that calculate the scalar product of neuron input vector and weight vector represented by *x* and *w* respectively:

$$h = \sum_{n=0}^{N} w_n x_n = w^T x. \tag{12}$$

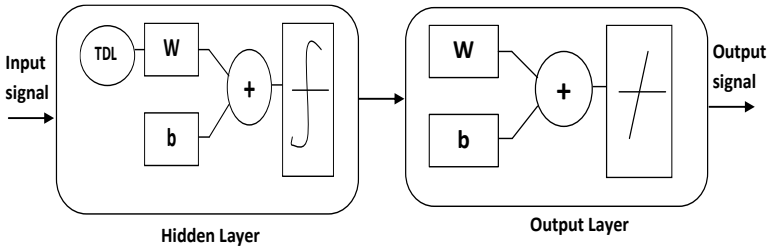

Fig. 2. Neural network architecture

Multilayer Perceptron (MLP) belongs to the class of feed-forward ANN. In MLP neural network, the error signal is calculated by comparing the estimated outputs with actual outputs and the error signal is propagated backwards through the network to minimize it. The static nonlinear mapping between the inputs and the outputs is

$$y_{Mod}(k) = f(u(k)), \tag{13}$$

where the output depends on the input *u(k)* at the same time step *k*.

The main task is to construct the models for the dynamical systems where output of the system at a given time instant relies on the present inputs as well as the previous output of the system. RNN is a simple architecture same as MLP, additionally having a memory to remember the information sequence. It utilizes the self-feedback of neurons of the hidden layer part. RNN is used to model nonlinear dynamical systems whose phase space dynamics are defined by the locally stable nodes [27].

The dynamical systems are considered to be the systems with memory. The storage elements (tapped delay lines) are used in the static network which helps to store the past values of the input. In feed forward architecture, tapped delay lines (TDL) are applied in the input part of all the neurons in the network. The *n-th* neuron output in layer *p* is

$$y_n^{(p)}(\mathrm{k}) = \mathrm{g}\left(\sum_j w_{nj}^{(p)^T} \; x_j^{(p)}(k)\right), \tag{14}$$

where $w_{nj}^{(p)} = \left[w_{nj,0}^{(p)}, w_{nj,1}^{(p)}, \ldots\ldots, w_{nj,M_n^{(p)}}^{(p)}\right]^T$ is the *j-th* filter coefficient vector in the *p-th* layer for node *n.*

The filter's input vector is framed from the regressed outputs of the *j-th* neuron of the former layer:

$$y_j^{(p-1)}(\mathrm{k}) = x_j^{(p)}(k) = \left[x_j^{(p)}(k), x_j^{(p)}(k1), \ldots\ldots, x_j^{(p)}(k - M_n^{(p)})\right]^T. \tag{15}$$

TDL are not only used in the input path but also in the output path to form a dynamic neural feedback architecture. This is considered as global feedback. The static neural networks with some universal approximation property can be framed as a general dynamic nonlinear neural modelling architecture. Local feedback is used where the output of the neuron(s) are used as inputs to the different or same neurons. The dynamic neural network can also be constructed by combining feed forward and feedback architectures. Similar to linear dynamic black box modelling, nonlinear system identification can also be framed. A nonlinear model architecture makes use of a regressor vector [28]. The model output y_{mod} is the parametrized function of the regressor vector φ:

$$y_{Mod}(k) = f(\Theta, \varphi(k)). \tag{16}$$

The regressor can be designed using the past given inputs, using the past model outputs or by using the past system outputs. One such architecture designed using past given inputs and the past system outputs is the NARX (Nonlinear autoregressive exogenous) model. NARX neural model is often called as a series & parallel model. NARX model exploits the nonlinear mapping capability of MLP that uses feedback from the neurons of the output layer to the input layer of the neural network, finds a significant role in many applications [29]. NARX model is well employed as it provides better predictions for system identification as it uses additional input that is stored in previous values of *y(k)* [30], [31], [32]. NARX neural model uses a input *u(k)* that is applied to the TDL memory of m_u units and a single output *y(k)* which is fed to the input part through another TDL memory of m_y units as shown in Fig. 3.

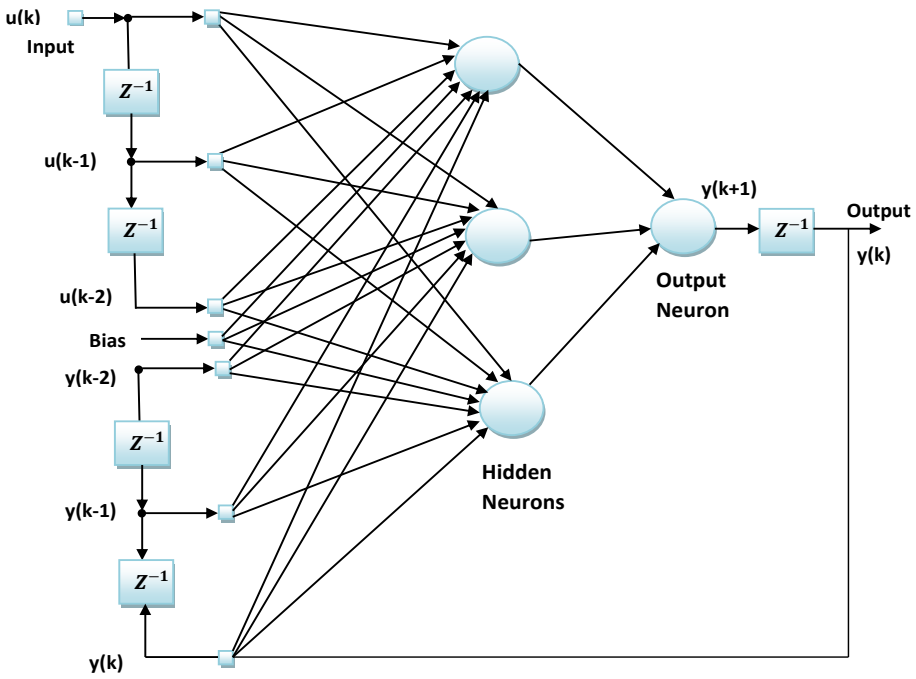

Fig. 3. NARX recurrent neural network

The input layer of MLP is fed with the contents of the two tapped delay lines memories. The signal *u(k)* is the current state of the model input and *y(k+1)* is the corresponding state of the model output. The signal vector that acts on the input layer of MLP comprises of exogenous inputs (i.e. current and the past values of the input) and regressed values of the output as shown in equation (17), *y(k)* and *u(k)* are the measurable output and input, m_u and m_y are the last value of input and output respectively:

$$y(k) = \varphi\,[y(k-1), y(k-2), \ldots\ldots, y(k-m_y), u(k-1), u(k-2), \ldots\ldots, u(k-m_u)]. \quad (17)$$

Vector $\widehat{y_p}(k)$ is the estimated output of the NARX neural model and $y_p(k)$ is the actual output of the system as shown in Fig. 4. The difference between $y_p(k)$ and $\widehat{y_p}\,(k)$ is the error signal. The error signal, in turn, adjusts the free parameters (weights & biases) of the neural network to reduce the squared variance between the plant output and NN model output efficiently and is computed over the entire training sample. Mean square error (MSE), as well as the regression values, is determined to develop the best model fit suitable for the controller design. MSE is the average squared variance between the targets and the outputs and should be minimum (closer to zero). Regression (R) determines the correlation between targets and the outputs which should be closer to one. If R is closer to one, we can assume that the model predicted is closer to the actual data set.

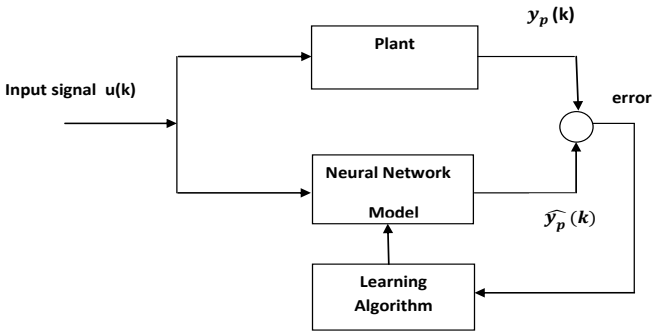

Fig. 4. Plant identification

Dynamic networks are considered to be the sequential networks implementing nonlinear mapping between the available input-output experimental data sequences. The model structure as well as the model size has to be determined. The objective of the training is the reduction of squared error resulting from the elements of the resulting error sequences. The total error E_{tot} is

$$E_{tot} = \sum_{k=1}^{N} E^2(k), \tag{18}$$

where *E(k)* is the error output of the dynamic neural network at time step *k*.

The length of the sequence is represented by *N*. The performance of the neural network must be validated to determine optimal stopping time while training. A set of test data which is not used in training is required to check the generalization capability of the model. Cross-validation plays a significant role in neural network modelling as it uses different data sets for validation purposes to estimate how the model is expected to perform.

A neural model with the optimal model complexity is attained at the lowest value of the test error. The modelling error or the mean squared error M_{emp} is defined as the sum of the squared errors:

$$M_{emp}(\Theta) = \frac{1}{P}\sum_{k=1}^{P}(E(k))^2, \tag{19}$$

$$M_{emp}(\Theta) = \frac{1}{P}\sum_{k=1}^{P}(y(k) - y_{Mod}(k))^2, \tag{20}$$

where $E(k) = y(k) - y_{Mod}(\Theta, \varphi(k))$, the regressor vector is termed as φ.

The total number of hidden neurons and length of the TDL which deploys the nonlinear mapping has to be properly selected. The proper model order is very essential for dynamic system modelling. Identifying the proper order for the unknown nonlinear dynamic systems was proposed by a heuristic method [33]. This approach mainly depends on the data utilized for training and the continuity property of the nonlinear functions. The order of the model is determined by the index

$$l^{(n)} = \left(\prod_{k=1}^{p} \sqrt{n}\ l^{(n)}(k)\right)^{1/p}. \tag{21}$$

where $l^{(n)}(k)$ is the *k-th* biggest Lipschitz quotient. The number of input variables is represented by *n* whereas *p* denotes the positive number. The Lipschitz index can be well applied to NARX models, where the order of feedback and feed-forward paths can be determined:

$$y_{Mod}(k)=f\ [y(k-1), y(k-2), \ldots\ldots, y(k-L),\ u(k-1), u(k-2), \ldots\ldots, u(k-M)]. \tag{22}$$

The Lipschitz index $l^{(n)} = l^{(L+M)}$, where *M* denotes the feed-forward order values and *L* represents the feedback order values. Therefore, this methodology is advantageous to estimate the order of the model but very sensitive to observation noise. The main disadvantage in RNN is that it suffers from vanishing gradient issues, as the BPTT method is used on the error signal for generating weights. In this method, the error signal decays exponentially as time elapses.

For a recurrent cell, *x(k)* is

$$x(k) = \phi\,[\, w(k)\, x(k-1)], \tag{23}$$

x(k) unfolded into *n* steps is

$x(k\text{-}n) = \phi\,[\, w(k)\, x(k\text{-}n\text{-}1)]$, (24)

and $y_{Mod}\,(k) = x(k)$, if $y(k)$ is the desired output. The squared output error is given in

$J(k) = \frac{1}{2}\,[\, y_{Mod}(k) - y(k)\,]^2 = e_0^2(k)$. (25)

The gradient descent is

$w(k+1) = w(k) - \eta \frac{\partial J(k)}{\partial w(k)}$, (26)

where $\frac{\partial J(k)}{\partial w} = e_0(k)\,\phi'\,x(k\text{-}1)$. For n steps it leads to

$w(k+1) = w(k) - \eta \sum_j^n e_j(k)\, x(k\text{-}j)$, (27)

$e_j(k) = e_{j-1}(k)\, w(k)\, \phi' = e_0(k) \prod_{j=1}^{n} [w(k)\, \phi']$. (28)

From equation (28), it can be concluded that if $\| w(k)\, \phi' \| > 1$, then it is a gradient distend and *if* $\| w(k)\, \phi' \| < 1$, the gradient of the error vanishes. For long term dependencies, RNN networks use their back connections to memorize the recent inputs structure, due to this the back propagated errors become immeasurably high in time or vanishes which results in complexity in computation of slow varying weights [34]. To overcome this problem of vanishing gradient, a method was proposed where RNN was used with gated units, called LSTM [35]. The procedure involved in developing the NARX neural model for the conical tank system is elaborated in Section 6.3. The following section deals with the identification of the system using the LSTM algorithm.

5. RNN-LSTM model

Recent research has proved that deep neural networks are effective in modelling highly nonlinear real-time systems and are effective model estimators for the available experimental input and output data set. Long-Short Term Memory (LSTM) is a kind of RNN that uses gate techniques and recurrent mechanisms. It allows the neural model to learn and recall the information for a long duration. LSTM has various advantages over feedforward multilayer networks as well as RNN in time series modelling. The LSTM network has good predictive capability compared to CNN-LSTM [36]. LSTM neural network also finds its applications in the field of control. The speed prediction model is developed for the automation of paving ships [37]. LSTM neural network is used to predict the control action for the micro grid to increase the efficiency of power systems [38]. The LSTMs were intended to alleviate the vanishing gradient problem while back propagating through time. It trains RNN to ensure continual error stream through CEC (constant error carousels) within multiplicative gate units. The layer is activated only when $\| w(k)\, \phi' \| \approx 1$.

5.1. LSTM structure

LSTM structure comprises of a set of recurrently linked subnets (memory blocks). Each memory block contains memory cells and distinctive gates namely input gate, a unique forget gate and the output gate where the informative data is permitted to enter through these gates and provides good control over what is added or removed from memory in the hidden layer part. The LSTM structure is shown in Fig. 5. These gates can protect and control the cell state. The data remains in the cell till the gate is closed and once the gate is opened the data can be used after a long duration of time:

$r(k) = \sigma\,(w_r\,(k)[y(k\text{-}1),u(k)])$, (29)

where σ represents sigmoid activation function which is used by each gate to scale every value of the gate vector to remain within 0 and 1 value. The input signal and output signal is represented by $u(k)$ and $y(k\text{-}1)$ respectively, $r(k)$ is the input given to the cell state and $w_r\,(k)$ is considered as the weight for forget gate in terms of

$i(k) = \sigma\,((w_i\,(k)[y(k\text{-}1),u(k)])$, (30)

$\tilde{x}(k) = tanh\,(w_x(k)[y(k\text{-}1),u(k)])$, (31)

where $i(k)$ and $\tilde{x}(k)$ are inner states, $w_i\,(k)$ and $w_x\,(k)$ are the weights for input gate. The extent to which a value withstands in the cell is controlled by forget gate.

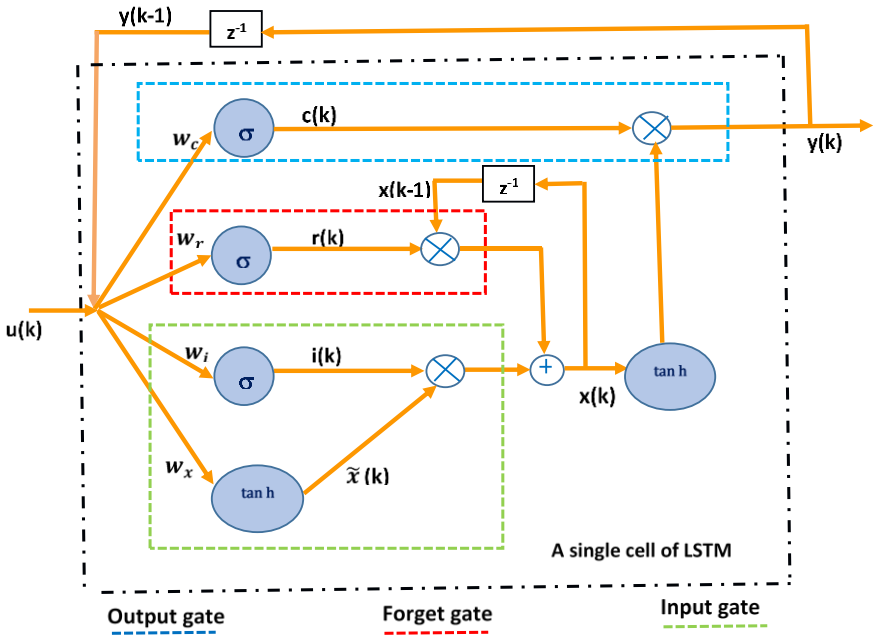

Fig. 5. The LSTM structure

The degree to which a fresh value enters the cell is controlled by input gate

$$c(k) = \sigma\ ((w_c(k)[y(k-1),u(k)]), \tag{32}$$

$$y(k)= c(k)\ tanh(x(k)), \tag{33}$$

where $w_c(k)$ is the weight for output gate and $c(k)$ is the inner state. The level of cell state that is added to the hidden state is controlled by output gate.

The updated cell state is

$$x(k) = r(k)\ x(k-1) + i(k)\ \tilde{x}(k), \tag{34}$$

where $x(k-1)$ and $r(k)$ are multiplied means to forget and add $\{ i(k)\ \tilde{x}(k)\}$. Substituting,

$$x(k)=\sigma(w_r(k)[y_{Mod}(k-1),u(k)])x(k-1)+ \sigma\ ((w_i(k)[\ y_{Mod}(k-1),u(k)])$$

$$tanh(w_x(k)[\ y_{Mod}(k-1),u(k)]), \tag{35}$$

$$y_{Mod}(k)=\sigma\ (w_c(k)[\ y_{Mod}(k-1),u(k)])\ tanh(x(k)). \tag{36}$$

The weights w_i, w_f, w_c and w_x are to be updated such that neural network output $y_{Mod}(k)$ converge to $y(k)$ which is the system output:

$$\min_{w_i,w_c,w_x,w_r} [\ y_{Mod}(k)-y(k)]^2, \tag{37}$$

$$y_{Mod}(k)=M[y(k-1),..,y(k-n),u(k),u(k-1),..,u(k-m)], \tag{38}$$

where n and m are the regression order for output $y(k)$ and input $u(k)$ respectively, and M is the model structure.

The weights are updated and learning law for one LSTM cell are

$$w_c(k+1) = w_c(k)- \eta\sum_{j=1}^{N} e_c(k)[\ y_{Mod}(kj),u(kj)], \tag{39}$$

$$w_r(k+1) = w_r(k) - \eta\sum_{j=1}^{N} e_r(k)[\ y_{Mod}(k-j), u(k-j)], \tag{40}$$

$$w_i(k+1) = w_i(k) - \eta\sum_{j=1}^{N} e_i(k)[\ y_{Mod}(k-1), u(k)], \tag{41}$$

$$w_x(k+1) = w_x(k) - \eta\sum_{j=1}^{N} e_x(k)[\ y_{Mod}(k-1), u(k)], \tag{42}$$

where $e_c(k) = e_{c-1}(k)\ w_c(k)\sigma'\phi(x(k))$, $e_{c1}(k) = e_0$, $e_r(k) = e_{r-1}(k)\ w_r(k)\sigma'\ x(k-1)$, $e_i(k) = e_{i-1}(k)\ w_i(k)\sigma'\phi\ \ (w_x(k)\ [y_{Mod}(k-1),\ u(k)])$, $e_x(k) = e_{x-1}(k)\ w_x(k)\phi'\ \sigma\ (w_i(k)\ [y_{Mod}(k-1),$

u(k)]).

The error remains in the LSTM unit cells when the errors are propagated back from the output layer. The constant error carousels (CEC) continuously provide constant error to each of the multiplicative gate units. Thus by using LSTM, vanishing gradient issue is solved by using gates units through which the memory of past states is controlled efficiently. LSTM effectively maintains long term memory which makes it easy to get accurate predictions. LSTM model serves to be best when compared to BP (Back-propagation) neural network model [39]. Speed of convergence was also found to be better for LSTM when compared to Bayesian networks, ANFIS-Grid and MLP neural networks [40].

5.2. The LSTM algorithm

LSTM algorithm works in two stages: forward phase and backward phase. The flowchart for LSTM model is shown in Fig. 6. In forward phase, the weights are randomly initialized to the nodes of each layer and values are assigned for learning rate, number of hidden neurons, and activation function. The number of neurons for each input and output layers is chosen. The weighted sum of neurons (input) through activation function is computed. Finally, the difference between NN model output and target output is calculated to determine the RMSE (root mean square error) value.

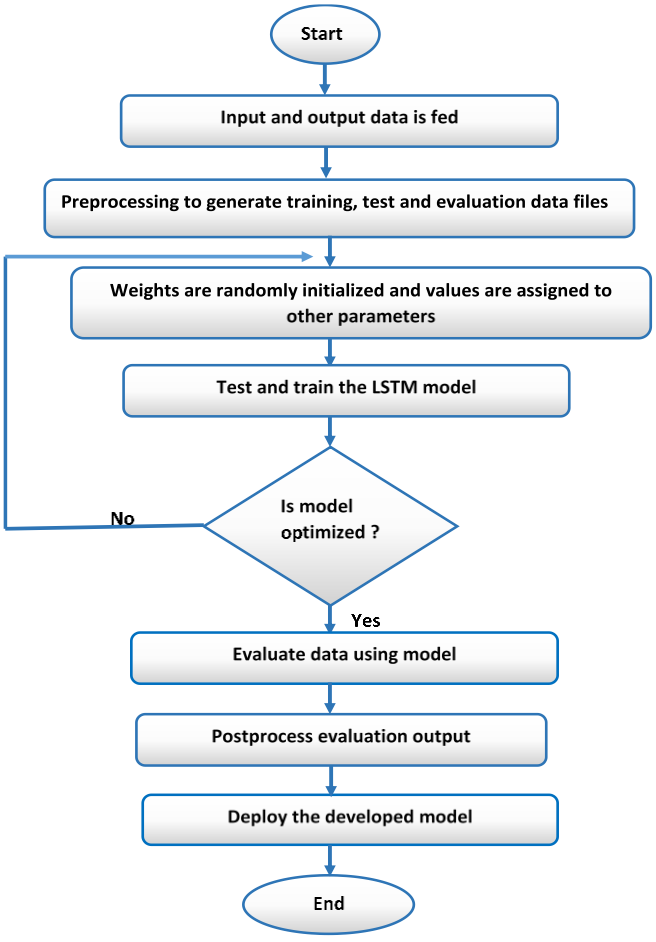

Fig. 6. Flowchart of LSTM algorithm

In backward phase, the weights are adjusted by back-propagating them in the reverse direction that is from the output layer to the input layer. In LSTM neural architecture, each node is connected to itself as well to the other node. The output is calculated and compared for the

current instant *t* and previous instant *(t-1)*. The optimum value is kept in memory and other values are removed. The error function is calculated. The program gets terminated when the minimum error is attained or when the iterations of the epoch get completed.

The real-time data set obtained from the real system is split into training data set and test data set. The LSTM model is developed by utilizing the training data set and predictions are made based on the test data set using the model. The persistence estimation on test data set attains an error which provides a tolerable lower bound of the performance on the test data set. Next is the model evaluation / validation. Every time step of the test data is estimated one at a time. A model is used to predict for the time step, then the expected value is made accessible to the model to predict for the next time step. All estimations on the test data set are collected and the performance of the model can be summarized based on the calculated root mean squared error (RMSE).

The data has to be transformed, before fitting the LSTM model to the data set to make predictions. Trends are removed from the data set. The data set is rescaled to be between 1 and -1 so that it gets access to the hyperbolic tangent activation function of the model. More training epochs are used to build the LSTM model. ADAM optimization algorithm is used to fit the model and mean squared error (MSE) function is calculated.

The identification algorithm not only takes multiple steps but also includes many files like time series data files, training data/testing data/evaluation data files, model output and estimated time series file. The training of the model takes place iteratively, by constantly tuning the parameters of the learning algorithm and finally deploying the developed RNN-LSTM model for optimal controller design. Section 6.4 describes the identification of the conical tank system using RNN-LSTM algorithm.

6. Results and discussion

The real-time laboratory conical tank system which is modelled in this study is shown in Fig. 7. The total height (H) and top radius (R) of the tank is 52 cm and 24 cm respectively. The volumetric inflow rate to the tank is 0-1000 lph. The level of the tank is the controlled variable and the voltage applied to the pump is the manipulated variable which in turn regulates the inflow rate of the tank.

Fig. 7. Laboratory conical tank system

A step change is applied to the input voltage of the pump and the open-loop responses of the conical tank in terms of level are recorded for two different operating points. The output voltage (0-5V) from the NI DAQ-6221 relates to the corresponding output level (0 cm – 50 cm) of the liquid in the conical tank. Using this experimental input/output data, four different empirical models (TF model, NARX model, NARX NN model, and RNN-LSTM model) are

developed with the help of MATLAB software.

6.1. The transfer function model

The transfer function (TF) model is built using the real-time data set using MATLAB/Simulink. Initialization is done by using the N4SID method to handle the initial state of the system. The continuous transfer function model for both the regions of the conical tank system under nominal conditions is shown in Table 1. Fig. 8 depicts the responses of the conical tank system, where voltage output (V) corresponds to the tank level is plotted against time (sec). It is observed that response of the TF model follows the response of experimental data recorded for both the operating regions.

Table 1. Transfer function for both the operating zone

Operating region	Transfer function
Zone 1 Range(4.5 to 34 cm)	$\frac{1}{294.377s+1}$
Zone 1 Range(34.1 to 50 cm)	$\frac{1.495}{593.1198s+1}$

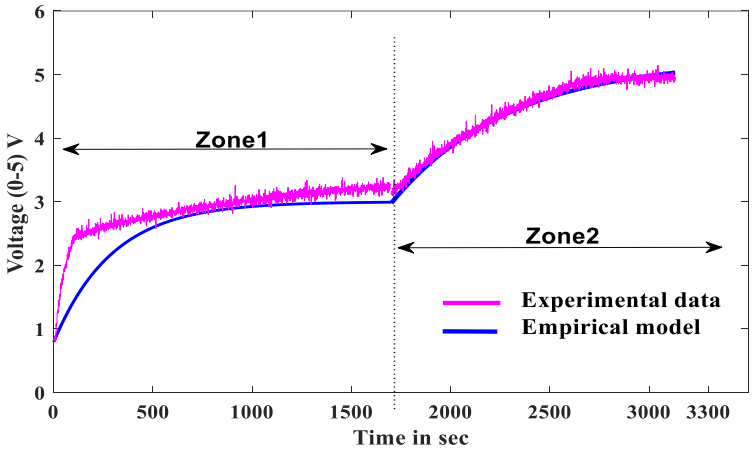

Fig. 8. Open-loop response of the conical tank system

6.2. The nonlinear ARX model

Based on the theory presented in Section 3, a nonlinear ARX model is built using the real time experimental data from the conical tank system using MATLAB/Simulink software. Figs. 9 and 10 show the nonlinear ARX model output with an accurate fit to the data for both the zones respectively, where the measured output follows the predicted output.

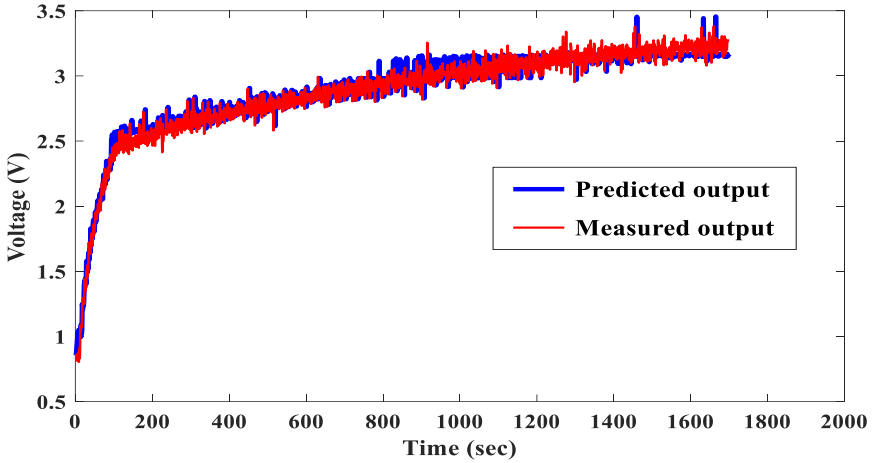

Fig. 9. Nonlinear ARX model for zone 1

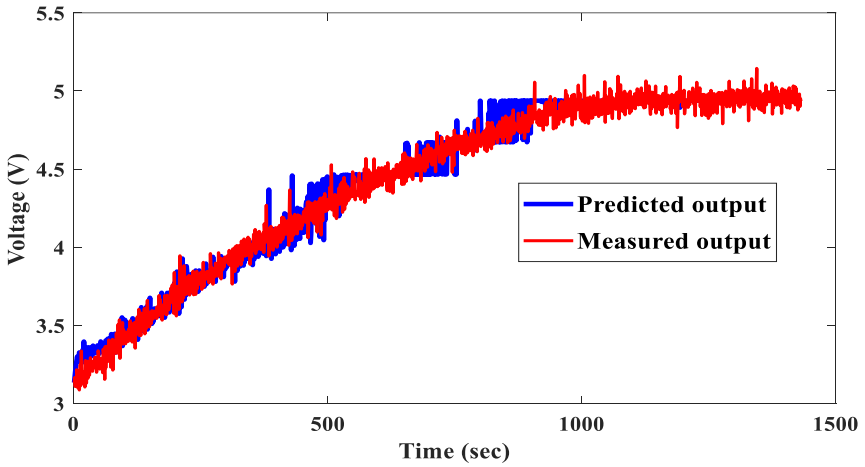

Fig. 10. Nonlinear ARX model for zone 2

The regressors and model order are changed in the trial and error method to get the best model which accurately describes the process dynamics. Orders and delays are specified to predefine the regressors of the model. To predict the current output, the number of past output terms (n_a) and past input terms (n_b) are selected as 1 and 2 respectively. The transport delay (n_k) is selected as one. The nonlinear autoregressive equation is given as $y(t) = [f(y(t-1), u(t-1), u(t-2))]$ for the two input delays selected for both the zones. The nonlinearity estimator used in this work is the wavelet network. LM (Levenberg-Marquardt) algorithm is used for iterative minimization of the objective function.

6.3. The NARX neural network model

Based on the theoretical concepts given in section 4, the NARX neural network model is built using experimental data set. The input and output real-time experimental data obtained from the conical tank system (zone1) is of (1X 1700) matrix. The procedure of training, validation, and testing is being done. Target time steps of 1700 are randomly divided as 70% target time steps (1190 samples) for training, 15% (255 samples) for validation and the remaining 15% (255) for testing. Similarly, the input and output real-time experimental data obtained is of (1X 1431) matrix for zone 2. Target time steps of 1431 are randomly divided as 70% target time steps (1001 samples) for training, 15% (215 samples) for validation and the rest of 15% for the testing procedure. NARX model time response plots are shown in Figs. 11 and 12 for both the zones.

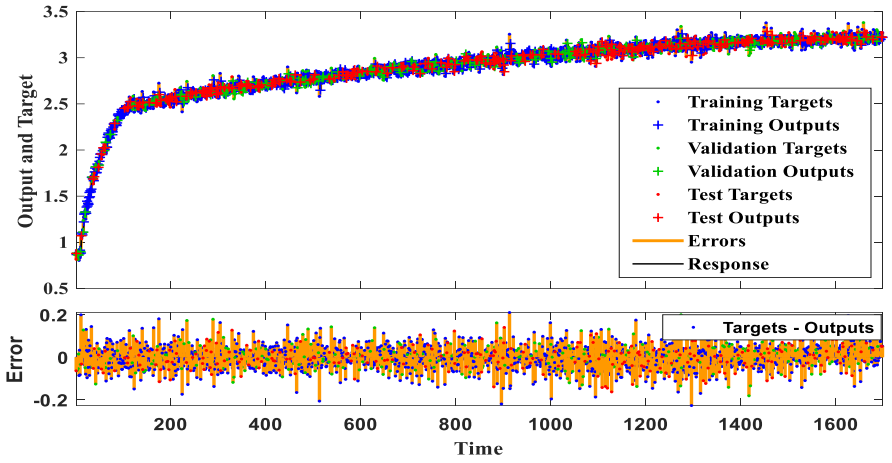

Fig. 11. Time-series response of the chosen NARX network for zone 1

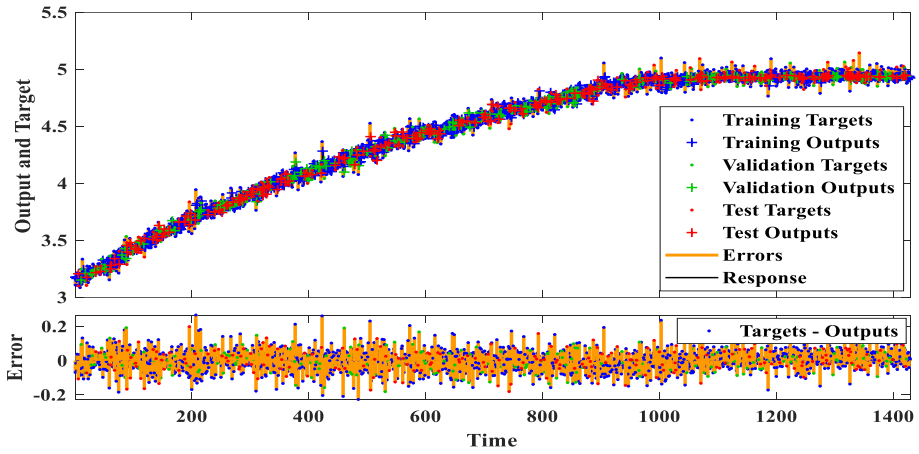

Fig. 12. Time-series response of the chosen network for zone 2

The number of hidden neurons and delays are repeatedly changed such that the regression value and MSE for all the 3 steps (training, validation, and testing) are one and zero respectively. The proposed work uses LM algorithm, as its learning capability is fast and has very good convergence behavior by gradually increasing the hidden neurons. The flexibility of neural network increases when the hidden neurons are increased in number. The procedure is repeated for developing the best model for both the zones of the conical tank. Error histograms with 20 bins for zone 1 and 2 are shown in Fig. 13 and Fig. 14, respectively. Bins are the vertical bars where each of them represents the samples from the data set in a particular bin. Best NARX neural model fit is framed using 20 hidden neurons and 2 delays for zone 1 and 10 hidden neurons and 2 delays for zone 2 and RMSE value for both the models are calculated.

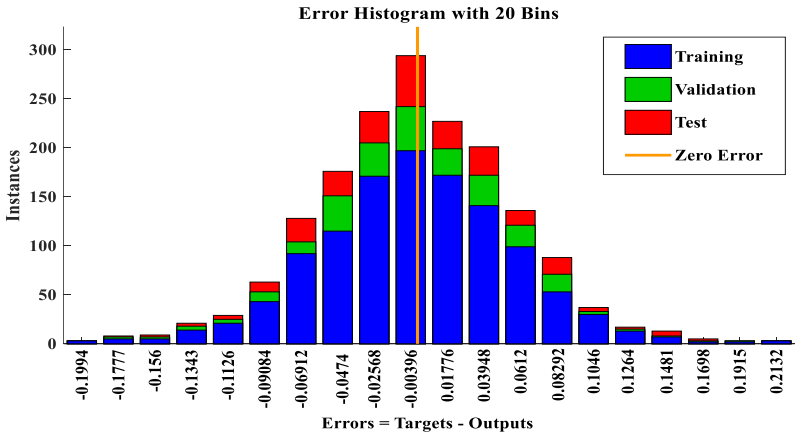

Fig. 13. Error histogram for zone 1

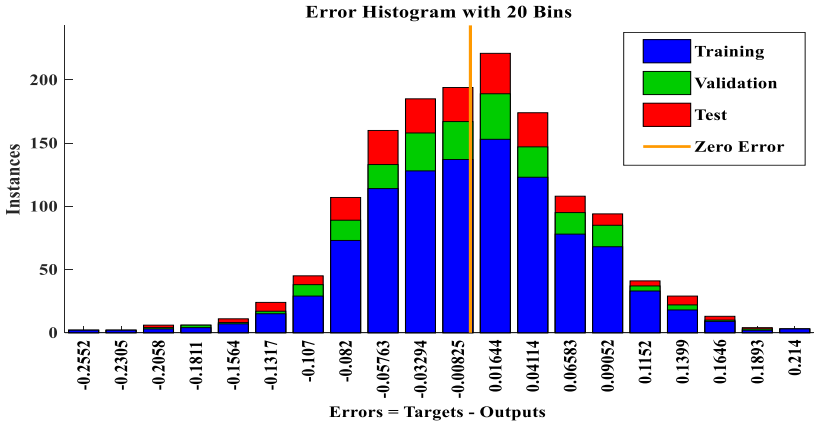

Fig. 14. Error histogram for zone 2

6.4. System Identification using RNN-LSTM model

LSTM model is built for both the zones based on the concepts briefly explained in Section 5. LSTM network is trained to estimate the values of future time steps, where the training sequences are shifted by one-time step. LSTM network learns to estimate the value of the succeeding time step at every time step of the input sequence. LSTM network can also be trained to predict the values for multiple time steps in the future.

Real-time input-output data sets (1700 samples) for zone 1 and (1431 samples) for zone 2 are segregated as training data and testing data. For the first operating zone, 1530 samples (90 % of the data) are used for training and remaining 170 samples (10 % of the data) are used for testing. Similarly for zone 2, 1288 samples for training and 143 samples for testing are used. The training data is to be standardized to get zero mean and unit variance to avoid the training process from deviating and to get a better fit. LSTM layer for zone 1 uses 150 hidden units and the learning rate being 0.00036. Zone 2 uses 200 hidden units with a learning rate of 0.0002. The learning rate can be constant or set to drop periodically throughout the training process. In this work, the learning rate is periodically set to drop as it helps the training to converge. The loss function drops fast for a smaller value of learning rate. Initial learning rate was kept as 0.009 for zone 1 and 0.001 for zone 2. Learn rate drop factor is set to 0.2 for both the zones.

L_2 norm-based gradient threshold method is employed. The gradient threshold is specified as 1 to avoid the gradients from exploding for both the zones. Maximum values for epochs are set for both the zones so to get RMSE and validation loss so that it reaches a satisfactory steady-state value. Epochs for zone 1 is set as 100 and 150 for zone 2. The solver

used for optimization is ADAM. LSTM regression network is trained for both the operating zones of the conical tank process. Overfitting is one of the major issues, where the train loss slopes down and the validation loss also slopes down, reaches an inflection point and again starts to rise. So this results in more training epochs and training model could be stopped at this inflection point. To avoid overfitting, a regularization method employing a dropout layer is included in the LSTM architecture. A dropout layer is incorporated before the fully connected layer which has an output of same size as that of number of outputs in training data and dropout layer is applied to recurrent and input connections of the memory units for both the zones of the conical tank. It was observed that the inclusion of dropout layer increased the performance of the model.

For multiple time steps future prediction, the time steps are predicted one at a time and state of the network is updated at every prediction. The test data is standardized as it was done for training data. The training progress is monitored in MATLAB as shown in Figs. 15 and 16 for both zones which reports the RMSE value calculated from the standardized data. During training, the network is validated. The training process stops when the validation loss starts decreasing to the smallest loss value. The LSTM network starts to converge at 50 epochs and later the convergence process remains to be stable for zone 1. For zone 2, the LSTM network starts to converge after about 40 epochs. Though the convergence process fluctuates lightly to some extent, the overall loss is decreasing which exemplifies the efficacy and robust generalization capability of the LSTM algorithm using the dropout layer. Speed of convergence is good for LSTM compared to RNN. LSTM faces limitations for very long sequences of data but not in this application of the conical tank system of a short sequence of task. Hence speed of convergence is efficiently good using a LSTM model.

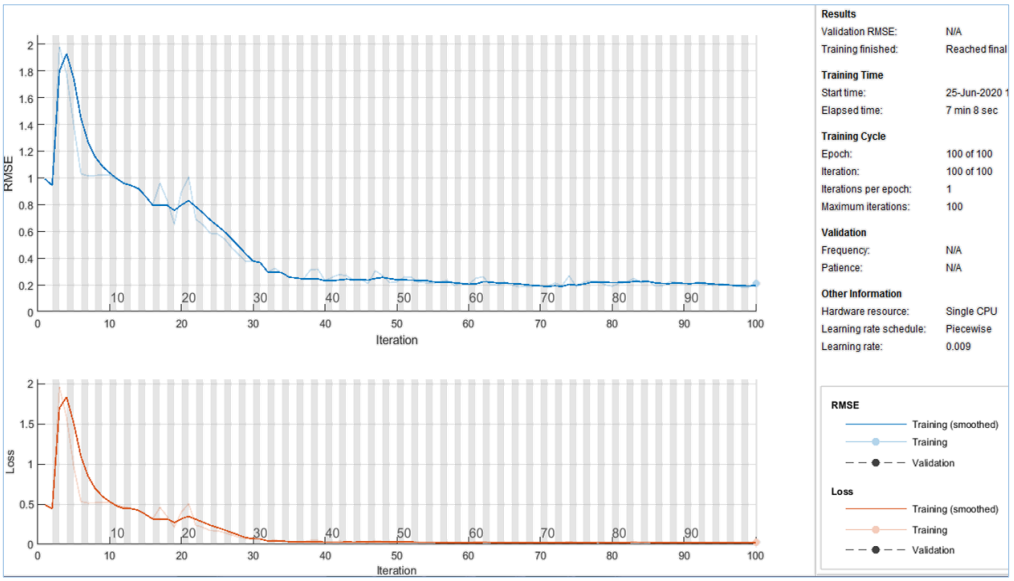

Fig. 15. Training progress plot for zone 1 (RNN-LSTM model)

The predictions will be more exact when the state is updated using observed values rather than using the predicted values. Figs. 17 and 18 show the response where the training data is compared with the predicted values for zone 1 and 2 respectively. Figs. 19 and 20 show the response of the test data in comparison with predicted data and RMSE value for both the zones respectively.

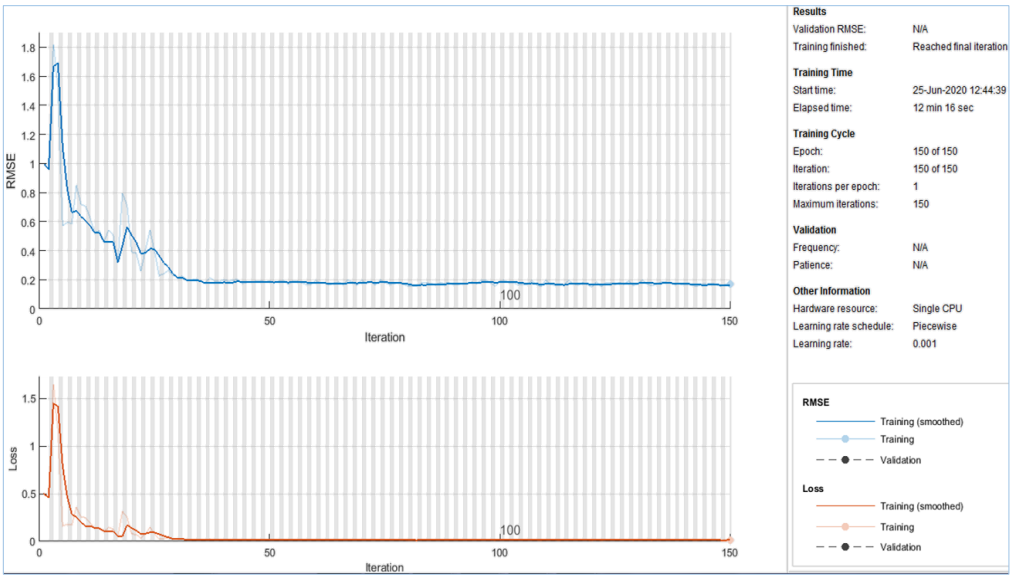

Fig. 16. Training progress plot for zone 2 (RNN-LSTM model)

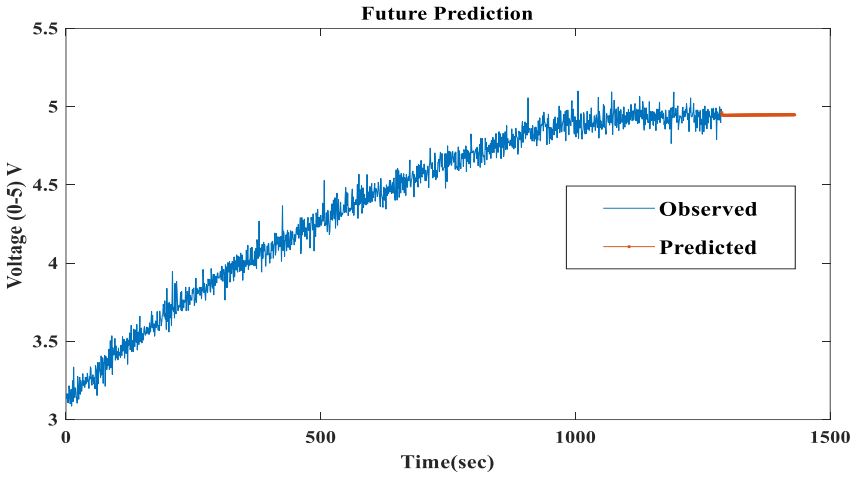

Fig. 17. Plotting training time series with predicted values for zone 1 (RNN-LSTM model)

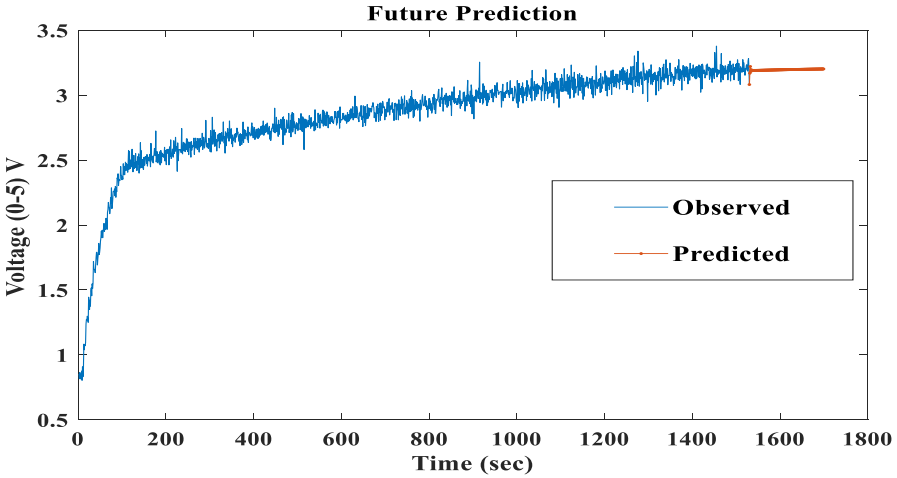

Fig. 18. Plotting training time series with predicted values for zone 2 (RNN-LSTM model)

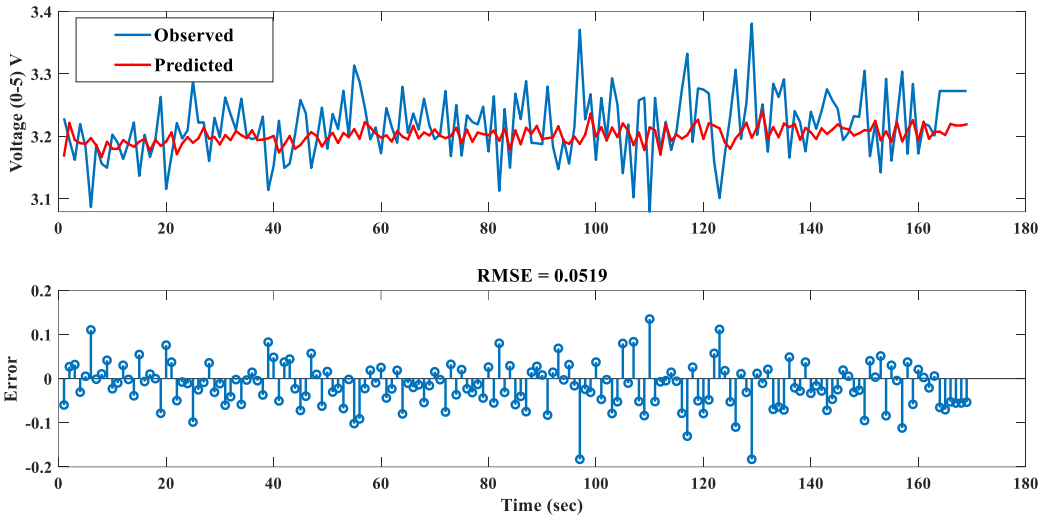

Fig. 19. Comparing the predicted values with test data for zone 1 (RNN-LSTM model)

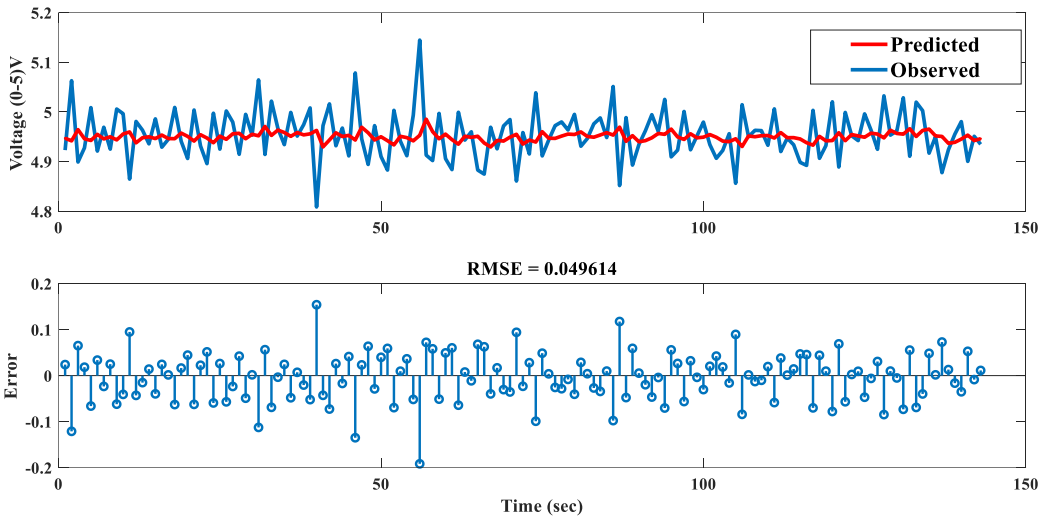

Fig. 20. Comparing the predicted values with test data for zone 2 (RNN-LSTM model)

The validity of the identified system models is confirmed by plotting the output of the models with actual system response. The validation of all the empirical models is compared based on RMSE (Root Mean Square Error) values as shown in Table 2.

Table 2. Validation of empirical models based on RMSE

Empirical Models	Zone 1	Zone 2
Transfer Function Model	0.1177	0.0942
NARX model	0.075670	0.078447

NARX Neural Model	0.063205	0.068095
RNN- LSTM Model	0.0519	0.049614

Figs. 21 and 22 show the performance comparison chart based on RMSE for zone 1 and zone 2, respectively. It is observed that the LSTM model is with the least RMSE value and hence it is comparatively efficient in giving the best fit model of the system.

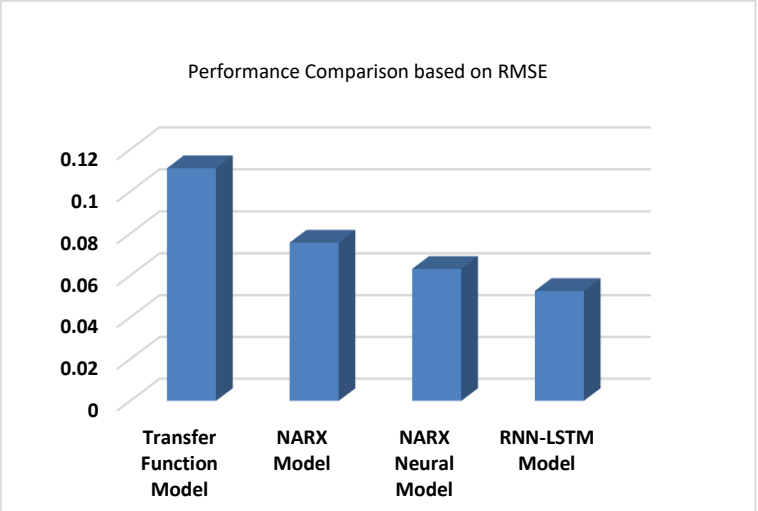

Fig. 21. Performance comparison based on RMSE for zone 1

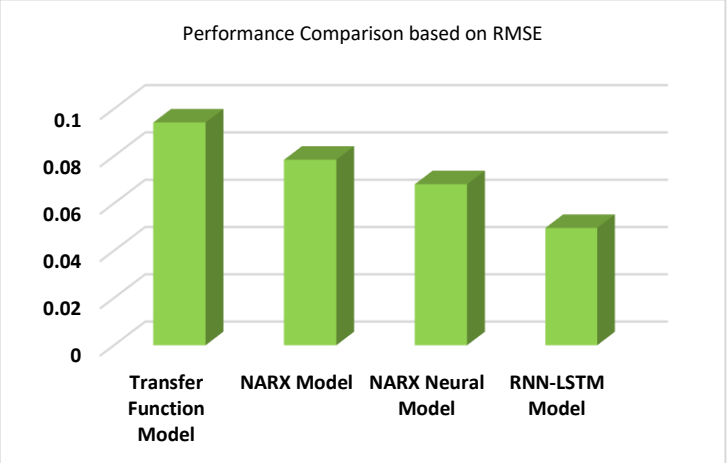

Fig. 22. Performance comparison based on RMSE for zone 2

7. Conclusion

Modelling of nonlinear dynamical systems is one of the major concerns if we go for model-based controller design. Several empirical methods are adopted for the identification of such processes. In this paper, modelling based on RNN-LSTM neural architecture is built using the experimental data for a real-time conical tank system.

Memory structure of LSTM with gated units gives an upper hand over the vanishing gradient problem that exist in conventional RNN structure and introducing a dropout layer in

LSTM architecture conquers the problem of overfitting. Through model validation, it has been empirically demonstrated that LSTM based identification has a vintage efficiency for modelling nonlinear dynamical systems in contrast to other empirical methods like the TF model, nonlinear ARX model, and NARX neural model with less RMSE in both the operating zones. Hence LSTM identification algorithm can be diligently and overwhelmingly used for modelling the conical tank system in controller design.

References

[1] B. ZHOU, X.-J. LU, S. TANG, and Z.-Q. ZHENG, *Nonlinear system identification and trajectory tracking control for a flybarless unmanned helicopter: theory and experiment*, Nonlinear Dynamics, vol. 96, pp. 2307-2326, 2019.

[2] E.-L. HEDREA, R.-E. PRECUP, and C.-A. BOJAN-DRAGOS., *Results on tensor product-based model transformation of magnetic levitation systems*, Acta Polytechnica Hungarica, vol. 16, no. 9, pp. 93-111, 2019.

[3] T. HAIDEGGER, L. KOVÁCS, R.-E. PRECUP, S. PREITL, B. BENYÓ, and Z. BENYÓ, *Cascade control for telerobotic systems serving space medicine*, IFAC Proceedings Volumes, vol. 44, no. 1, pp. 3759-3764, Aug. 2011.

[4] A. ALBU, R.-E. PRECUP, and T.-A. TEBAN, *Results and challenges of artificial neural networks used for decision-making in medical applications*, Facta Universitatis, Series: Mechanical Engineering, vol. 17, no 4, pp. 285-308, 2019.

[5] K. S. NARENDRA and K. PARTHASARATHY, *Identification and control of dynamical systems using neural networks*, IEEE Transactions on Neural Networks, vol. 1, pp. 4-27, 1990.

[6] K. S. NARENDRA and K. PARTHASARATHY, *Neural networks and dynamical systems*, International Journal of Approximate Reasoning, pp.109-130, 1992.

[7] M. J. WILLS, G. A. MONTAGUE, C. DI MASSIMO, M. T. THAM, and A. J. MORRIS, *Artificial neural networks in process estimation and control*, Automatica, vol. 28, pp. 1181-1187, 1992.

[8] J. RUIZ-RANGEL and C. J. ARDILA HERNANDEZ., *ERNEAD: Training of artificial neural networks based on a genetic algorithm and finite automata theory*, International Journal of Artificial Intelligence, vol. 16, no. 1, pp. 214-253, 2018.

[9] K. S. NARENDRA and K. PARTHASARATHY, *Gradient methods for optimization of dynamical systems containing neural networks*, IEEE Transactions of Neural Networks, pp. 252-262, 1991.

[10] P. J. WERBOS., *Backpropagation through time: What it does and how to do it*, Proceedings of the IEEE, vol. 78, no. 10, pp. 1550-1560, 1990.

[11] P. E. RUMELHART and G. E. HINTON, *Learning representatives by backpropagating errors*, Nature, vol. 323, no. 9, pp. 533-536, 1986.

[12] Y. BENGIO, P. LAMBLIN, D. POPOVICI, and H. LAROCHELLE, *Greedy layer-wise training of deep networks*, Advances in Neural Information Processing Systems (NIPS'06), pp. 153 160, 2007.

[13] G. E. HINTON, S. OSINDERO, and Y. W. TEH., *A fast learning algorithm for deep belief nets*, Neural Computation, vol. 18, no. 7, 2006.

[14] O. OGUNMOLU, X.-J. GU, S. JIANG, and N. GANS, *Nonlinear system identification using deep dynamic neural networks*, American Control Conference, May 2017.

[15] J. GONZALEZ and W. YU, *Nonlinear system modelling using LSTM neural networks*, IFAC Conference on Modelling, Identification and Control of Nonlinear Systems, pp. 485-489, 2018.

[16] Y. WANG, *A new concept using LSTM neural networks for dynamic system identification*, American Control Conference IEEE, USA, 2017.

[17] N. SRIVASTAVA, G. HINTON., A. KRIZHEVSKY, I. SUTSKEVER, and R. SALAKHUTDINOV, *Dropout: a simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.

[18] G. YARIN and Z. GHAHRAMANI, *A theoretically grounded application of dropout in recurrent neural networks*, International Conference on Neural Information Processing Systems, 2016.

[19] W. ZAREMBA, I. SUTSKEVER, and O. VINYALS, *Recurrent neural network regularization*, ICLR Conference, 2015.

[20] V. PHAM, T. BLUCHE, C. KERMORVANT, and J. LOURADOUR, *dropout improves recurrent neural networks for handwriting recognition*, International Conference on Frontiers in Handwriting Recognition, 2014.

[21] M. K. TRANSTRUM and J. P. SETHNA, *Improvements to the Levenberg-Marquardt algorithm for nonlinear least squares minimization*, Journal of Computational Physics, 2012.

[22] H. P. GAVIN, *The Levenberg-Marquardt Algorithm for Nonlinear Squares Curve-Fitting Problems*, 2019.
[23] K. HORNICK, M. STINCHCOMBE, and H. WHITE, *Multilayer feed forward networks are universal approximators*, Neural Networks, vol. 2, no. 5, pp. 359-366, 1989.
[24] M. C. VALVERDE, E ARAUJO, and H. CAMPOS VELHO, *Investigation of neural networks for function approximation*, Procedia Computer Science, vol. 17, pp. 586-594, 2013.
[25] K. K. SARMA., *Neural network based feature extraction for Assamese character and numeral recognition*, International Journal of Artificial Intelligence, vol. 2, no. S9, pp. 37-56, 2009.
[26] M. C. VALVERDE, E. ARAUJO, and H. CAMPOS VELHO, *Neural network and fuzzy logic statistical downscaling of atmospheric circulation-type specific weather pattern for rainfall forecasting*, Applied Soft Computing, vol. 22, pp. 681- 694, 2014.
[27] J. J/ HOPFIELD, *Neural networks and physical systems with emergent collective computational abilities*, Proceedings of the National Academy of Sciences of the United States of America, vol. abs/1406.1078, 2014.
[28] J. SJOBERG, Q. ZHANG, L. LJUNG, A. BENVENISTE, B. DELYON, P-Y GLORENNEC, H. HJALMARSSON, and A. JUDITSKY, *Non-linear black-box modelling in system identification: a unified overview*, Automatica, pp. 1694-1724, 1995.
[29] C. FEBINA and D. ANGELINE VIJULA, *Model based controller design using real time neural network model and PSO for conical tank system*, Control Engineering and Applied Informatics, vol. 22, no. 3, pp. 13-24, 2020.
[30] Z. BOUSSAADA, O. CUREA, A. REMACI, H. CAMBLONG, and N. MRABET BELLAAI, *A Nonlinear Autoregressive Exogenous (NARX) neural network model for the prediction of the daily direct solar radiation*, Energies, vol. 11, 620, pp. 1-2, 2018.
[31] D. A. KUMAR and S. MURUGAN, *Performance analysis of NARX neural network back propagation algorithm by various training functions with tracking signal approach for time series data*, International Journal of Engineering Sciences and Research Technology, vol. 3, no. 4, pp. 312-326, 2018.
[32] R. SARKAR, S. JULAI, S. HOSSAIN, W. T. CHONG, and M. RAHMAN, *A comparative study of nar and narx neural network for long-term wind speed forecasting in Malaysia*, Mathematical Problems in Engineering, 2019.
[33] X. HE and H. ASADA, *A new method for identifying orders of input-output models for nonlinear dynamic systems*, Proceedings of the American Control Conference, San Francisco, CA, USA, pp-2520-2523, 1993.
[34] Y BENGIO, P. SIMARD, and P. FRASCONI, *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks, doi: 10.1109/72.279181, 1994.
[35] S. HOCHREITER and J. SCHMIDHUBER, *Long short-term memory*, Neural Computation, vol. 9, no. 8, pp. 1735-1740, 1997.
[36] S. DHANANJAY KUMAR and D. P. SUBHA, *Prediction of depression from EEG signal using long short term memory*, Proceedings of the 3rd International Conference on Trends in Electronics and Informatics (IEEE), 2019.
[37] J.-H. HU, J.-M. GUO, and B.-Q. DING, *Automatic control of paving ships based on LSTM*, Proceedings of 34th Youth Academic Annual Conference of Chinese Association of Automation (YAC) IEEE, 2019.
[38] A. G. KUMAR, M. R. SINDHU, and S. S. KUMAR, *Deep neural network based hierarchical control of residential microgrid using LSTM*, Proceedings of TENCON IEEE, 2019.
[39] K.-Q. YANG M. BI, Y. LIU, and Y.-X. ZHANG, *LSTM-based deep learning model for civil aircraft position and attitude prediction approach*, Proceedings of the 38th Chinese Control conference IEEE, 2019.
[40] J. HERNANDEZ, D. LOPEZ, and N. VERA, *Primary user characterization for cognitive radio wireless networks using long short-term memory*, International Conference of Distributed Sensor Networks, vol. 14, no. 11, 2018.