INTELLIGENT IDENTIFICATION AND ADAPTIVE CONTROL OF NONLINEAR DYNAMICAL SYSTEMS

A Thesis submitted
In Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

by

SHOBANA.R
(2K20/PhD/EE/509)

Under the Supervision of

Dr. Bhavnesh Jaint
Department of Electrical Engineering,
Delhi Technological University

Dr. Rajesh Kumar
Department of Electrical Engineering,
NIT Kurukshetra

DEPARTMENT OF ELECTRICAL ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042. India

December 2024

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE'S DECLARATION

I, Shobana.R hereby certify that the work which is being presented in the thesis entitled **"Intelligent Identification and Adaptive Control of Nonlinear Dynamical Systems"** in partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy**, submitted in the Department of Electrical Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from January 2021 to November 2024 under the supervision of Dr. Bhavnesh Jaint (Department of Electrical Engineering, Delhi Technological University, New Delhi-110042, India) and Dr. Rajesh Kumar (Department of Electrical Engineering, NIT Kurukshetra, Kurukshetra-136119, India). The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Shobana.R
2K20/PhD/EE/509

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE BY THE SUPERVISORS

Certified that **Shobana.R (2k20/PhD/EE/509)** has carried out her research work presented in this thesis entitled "**Intelligent Identification and Adaptive Control of Nonlinear Dynamical Systems"** for the award of **Doctor of Philosophy** from Department of Electrical Engineering, Delhi Technological University, Delhi, under our supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Dr. Bhavnesh Jaint
Assistant Professor,
Department of Electrical Engineering
DTU, Delhi-110042
(Supervisor)

Dr. Rajesh Kumar
Assistant Professor [Grade-I],
Department of Electrical Engineering,
NIT Kurukshetra,
Kurukshetra -136119
(Co-Supervisor)

The Ph.D. viva-voce examination of Ms.Shobana. R has been held on _____.

(Supervisor) (Co-Supervisor) (Head, Dept. of Electrical Engg.)

ACKNOWLEDGMENTS

I extend my deepest gratitude to God for blessing me with the strength and determination to undertake and complete this research successfully. The achievement would not have been achieved without his blessings. I would like to express my sincere gratitude to my supervisor, **Dr. Bhavnesh Jaint**, Assistant Professor, Department of Electrical Engineering, Delhi Technological University, New Delhi, and my co-supervisor **Dr. Rajesh Kumar**, Assistant Professor, Department of Electrical Engineering, NIT Kurukshetra, Haryana for their constant support and encouragement throughout my research period. Their knowledge and guidance have been essential to the success of the work. I deeply appreciate the time and effort they extended beyond their working hours. It has been an honor to work under their mentorship. I would also like to thank Prof. Dr. Rachana Garg, Head of the Department, Department of Electrical Engineering, and all DRC members who has been a constant support to the concept of the research work. I would also like to thank all the faculty, and staff of the Department of Electrical Engineering for their assistance throughout the journey.

Lastly, my heartfelt thanks to my family, my husband, and my daughter for their patience, cooperation, and sustained encouragement from inception to conclusion.

Shobana.R
2K20/PhD/EE/509

ABSTRACT

Most of the physical systems in nature are highly nonlinear. Such systems are characterized by time-varying and complex behavior making it challenging to derive accurate models for understanding their behavior. Further developing a controller for such systems becomes more complex, rendering a conventional controller inadequate. Soft computing has become a viable alternative method for system dynamics modeling. Artificial Neural Networks (ANNs), among other methods, are notable for their exceptional capacity to approximate complex nonlinear functions and dynamically adjust to the changing behavior of the system. This thesis explores, the application of ANN-based methodologies for the identification and adaptive control of nonlinear dynamic systems. In this thesis, we have proposed several modifications to the existing ANN structures to improve the capability of ANN to handle dynamic systems. In particular, we have modified the structure of Recurrent Neural Networks (RNNs) and trained them using the back-propagation (BP) algorithm. The proposed design considered in this thesis is independent of the order of the system. Building on the novel identification structure, this work extends its application to design an adaptive controller for nonlinear dynamic systems. The controller is developed to operate online, enabling simultaneous identification and control. This ability makes the controller design robust enough to adapt in real time and capture the changing dynamics of the plant effectively. The performance of an ANN is influenced by the learning algorithm and structure. In this thesis, we have developed a constructive algorithm with an adaptive learning rate to dynamically optimize the structures of Feed Forward Neural Network (FFNN) and RNN. This approach enables efficient modeling and learning by facilitating dynamic growth of the network architecture. Furthermore, to optimize the parameters of the ANN and enhance the performance of the BP learning algorithm, we have developed an adaptive Particle Swarm Optimization (PSO)-based BP algorithm. The parameters of PSO such as inertia weight (w) and hyperparameters (c_1 and c_2) are also dynamically updated to improve the optimization capability of PSO. The proposed approaches are tested on various nonlinear benchmark systems such as liquid-level systems, Mackey glass series prediction prob-

lems, and various degrees of nonlinear plant equations. The stability and convergence of the update weight equations are derived in the sense of Lyapunov stability principles. The proposed approaches are evaluated in terms of Mean Square Error (MSE), Mean Absolute Error (MAE), and Relative Mean Absolute Error (RMSE) against state-of-the-art neural structures in the literature. The robustness of the proposed approaches is validated by using parameter variations and disturbance. Detailed simulation analysis has been also carried out in the thesis to evaluate the performance of the proposed approaches. The results demonstrate the effectiveness of these approaches in accurately learning the dynamics of nonlinear systems.

Contents

List of Figures

List of Tables

Abbreviations

PID	Proportional-Integral-Derivative
NARX	Non-linear Auto-Regressive network with EXogenous inputs
ANN	Artificial Neural Networks
GA	Genetic Algorithms
MLP	Multi-Layer Perceptron
RBFN	Radial Basis Function Network
FFNN	Feed-Forward Neural Network
TDL	Tapped-Delayed Lines
RNN	Recurrent Neural Network
LRNN	Local Recurrent Neural Network
FRNN	Fully Recurrent Neural Network
DNN	Dynamic neural network
EKF	Extended Kalman filter
PSO	Particle swarm optimisation
GSA	Gravitational search algorithm
TCN	Temporal convolutional network
ANFIS	Adaptive neuro-fuzzy inference system
LSTM	Long short-term memory
HNN	Hopfield neural network

IAC	Indirect adaptive control
GRU	Gated recurrent unit
TDE	Time-delayed estimator
SMC	Sliding mode control
ALR	Adaptive learning rate
BP	Back-Propagation
ENN	Elman neural network
JNN	Jordan neural network
HFRNN	Hybrid compound recurrent feed-forward neural network
JNC	Jordan network-based controller
LRNC	Local recurrent network based controller
CFFNN	Constructive feed-forward neural networks
CLRNN	Constructive Local recurrent neural networks
MSE	Mean square error
FLR	Fixed Learning rate
APSOBP	Adaptive Particle swarm optimisation-Back Propagation algorithm
MISO	Multi-input single output
MAE	Mean absolute error
RMSE	Root mean squared error
AMSE	Average mean square error
AMAE	Average mean absolute error
RMSE	Root mean square error
BIBO	Bounded input bounded output
MRAC	Model Reference Adaptive Control
SGD	Stochastic Gradient Descend

Chapter 1

Introduction

Modeling and control of complex nonlinear dynamic processes has been a long-standing problem for many engineering applications, particularly in automatic control applications. The dynamic process often exhibits time-varying and complex behavior like chaos, bifurcation, and limit cycles [1]. Due to their intrinsic complexity, developing model structures for robust control and dynamic modeling or identification is both hard and challenging [2]. The dynamic models can be derived using the direct approach, which involves studying the system's interactions for a while and applying physical laws but this method suffers from several drawbacks. The resulting models are often highly complex and require detailed knowledge of the system. Such models can also be very expensive and time-consuming. They take a lot of simulation time, making them unsuitable for online applications [3]. Such models can also be incomplete or inaccurate, deciding which effects to consider and which to neglect. Consequently, the resulting models are not the exact representation of the system. In contrast to the direct approach, the system identification approach is an alternate way of developing dynamic models. This approach derives models based on the observed input-output experimental data [4]. It uses expert knowledge about the system to build a dynamic model that is well-suited for online applications and adaptive control, which are widely used in process control. While system identification still requires human expertise to satisfactorily construct models, they are much better than the models derived using a direct approach [5]. The system identification approach involves the following steps: firstly, the process involves choosing a model suitable for the application. Secondly, input signals are designed, the identification is carried out and the resulting model is validated. These steps allow the development of models that effectively capture nonlinear relationships between inputs and outputs, leading to accurate representations of complex

systems [6]. The development of dynamic models using linear theories has been the primary focus of most researchers. These have been successfully applied in many engineering applications and controls [7]. Despite their relative simplicity, the dynamic models created with linear theories often require some assumptions and approximations to be considered during development [8]. Real-life systems mostly show nonlinear dynamic behavior, and linear models often fail to work efficiently. This is because (i) they assume the system operates within a narrow range around the equilibrium point, and when a significant disturbance occurs, the model's accuracy declines (ii) In addition, the use of linear models necessitates the linearisation of nonlinear relationships by discarding higher-order interactions [9]. As a result, the linear model's capacity to accurately represent and regulate the dynamics of nonlinear processes in real-time is limited. Therefore, this has led to the development of nonlinear system identification methods [10]. The system identification approaches are often classified based on the amount of prior information available. White box models require adequate information regarding the system interactions. Black-box models use no prior knowledge regarding the system and depend only on the observed input-output data. Grey box models combine partial prior knowledge with the experimental data [11]. Among these, black-box models are commonly used for nonlinear system identification due to the challenges associated with obtaining prior knowledge about the system interactions. Non-linear system identification can be represented using the state-space approach or the difference equation (input-output representation) [12]. Unlike the state-space approach, which uses internal state variables for identification, the input-output representation focuses on the direct relationship between system inputs and outputs over time. For a general non-linear system, the input-output relationship is given as follows:

$$y_p(k) = f[y_p(k-1), .., y_p(k-n), r(k), r(k-1), .., r(k-m)] \tag{1.1}$$

In Eq.(1.1), let $r(k)$ denote the external input of the plant and $y_p(k)$ denote the present output of the plant. The present output of the plant depends on both the present as well as past inputs and past outputs of the plant. Here, n and m are the orders of the plant and to ensure causality $m \leq n$. Black box identification methods like block-structured models, Volterra series, and Non-linear Auto-Regressive network with Exogenous inputs (NARX) models are useful for non-linear identification. However, they suffer from some limitations, such as computational complexity, and sensitivity to noise [13]. ANNs, fuzzy logic methods, and Genetic Algorithms (GA) have emerged as useful tools for nonlinear system identification and control.

1.1 Artificial neural networks for modeling and control of nonlinear systems

The data-driven nature of ANNs and their capacity to approximate extremely complex, nonlinear functions have made them important tools. ANNs are ideal for black-box modeling techniques since they don't require explicit prior knowledge about the system. They have been successfully used to solve many engineering and control challenges and are capable of accurately representing nonlinear mappings [14], [15]. ANNs are non-parametric methods that infer the characteristics of biological neurons. They consist of interconnected layers of neurons, including hidden units, with associated weights. They quickly adapt to learn input-output behavior through training and hence are widely used for nonlinear identification and control strategies. ANNs possess many characteristics such as fault tolerance, robustness, and learning adaptivity to uncertainties and noisy data [16]. They also possess more advantages over fuzzy systems such as their ability to learn large and complex datasets by continuous training, which makes them efficient in handling complex applications. Fuzzy logic uses rule-based reasoning to model processes and might miss complex patterns [17]. These characteristics make ANNs used widely to solve a variety of applications including image recognition, medical diagnosis, control and identification, forecasting, and speech recognition [18], [16]. Among the various ANN structures in literature, two major variants of ANN: FFNNs and RNNs are widely used by researchers. FFNNs are structures that propagate information in a forward direction (input to output layer). Multi-layer perceptron (MLP) and Radial basis function network (RBFN) are two major types of FFNN. MLP consists of multiple layers of neurons, including one or more layers of hidden units between the input and the output layers. The MLPs are generally trained using a BP algorithm to globally optimize the weights of the networks [19]. Another alternative to MLP is the RBFN. RBFN is a type of FFNN whose activation function is radial basis function. The activation function is determined by calculating the Euclidean distance between the input and neuron centres [20]. In dynamic systems, the current output depends on the present and past values of input and output. FFNN structures being static connections, cannot hold the information of past observations within them. The FFNN can be made dynamic by supplying temporal dynamics into its structure either by knowing the order of the system in advance or by using Tapped-Delayed Lines (TDL) [21]. This makes the FFNN structure dynamic to handle nonlinear problems and when the order of the system is known, FFNN

can use all the past inputs and outputs fed into them using TDL. For example, if the system's input order is 'n', then the inputs of FFNN will be $x(k-1), x(k-2), \ldots, x(k-n)$. However, practically the order of the system is not always known. During such scenarios, FFNN employing TDL structures possess limitations such as increased complexity, difficulty in capturing long-term dependencies, and inefficiency in handling large sequences of data [22]. This has led to the development of ANN with memory structures within them. RNNs are alternately widely used ANN types to model and control nonlinear dynamical systems. The availability of feed-forward and feedback connections creates a memory in the recurrent neural networks by nature [23], [24], [25]. The RNN structure gives ANN a dynamic structure by linking the current output state to a combination of network input and the previous state of the network. Based on the feedback, the RNN is further classified into two broad categories Locally Connected Recurrent Neural Network (LRNN) and Fully Connected Recurrent Neural Network (FCRNN). FCRNN structure is one where all the neurons except the input layer are connected with all others through a trainable weight. Hopfield neural network (HNN) [26] is a form of an FCRNN where every output is connected to the input. The network is designed symmetric and has no target to achieve as in supervised training, hence they suffer from memory limitation and inefficiency in learning new patterns. Elman neural network (ENN) [27] and Jordan neural network (JNN) [28] form the other types of FCRNN. To store the past outputs of the hidden or output layer, a context layer is added to their structure. In ENN, the context layer is used to feed the delayed outputs of the hidden layer as inputs, but in JNN, the context layer is used to feed the delayed outputs of the output layer as inputs to the hidden layer. Traditional ENN and JNN suffer limitations, with JNN structures becoming very large and experiencing slow convergence when more outputs are involved, while ENN networks, though efficient due to the addition of an extra input layer, are not suitable for online identification [29]. The LCRNN [30], on the other hand, shares the same fundamental structure as the FFNN model but also includes self-feedback, which plays an important role in retaining past information. These structures have dynamic neurons unlike static neurons present in FFNN [31]. To improve RNN models, many novel hybrid and improved variants of these networks continue to be the point of research, to improve the network performance in dynamic environments [32]. Once trained, the network becomes a mathematical representation of the system, enabling it to predict the system's dynamics accurately.

1.1.1 Adaptive control using ANN

In control applications, ANNs play a crucial role in handling nonlinear dynamics, where traditional control techniques such as Proportional-Integral-Derivative (PID) controllers, state-space control, and gain scheduling may fail [33]. These applications require an intelligent control system that depends on continuous feedback from the system. Adaptive control is one such control technique that adjusts to the changing environment in real time [34]. ANNs are very effective in managing nonlinear dynamics and therefore a perfect fit for adaptive control of nonlinear systems. In [35], the author gave a foundation for an adaptive method with the development of a Model Reference Adaptive Control (MRAC) scheme. In this, the controller parameters were developed based on the reference model. However as the system exhibits complex behavior, the traditional approaches were not able to model non-linearities, highlighting the need for flexible techniques. With the development of ANN, it became evident that ANN can handle non-linear relationships without a need for any explicit models [36]. Further, an indirect adaptive control methods were developed and it was showed how the controller can be identified and updated in real-time to adapt to changing environments in [37]. This also enhanced the controller's ability to capture dynamics and uncertainties. The adaptive control techniques are combined with many other optimization techniques such as GA and PSO to improve the convergence in adaptive control systems [38]. The RNNs were used for adaptive control of nonlinear systems, where the time-dependent dynamics in a system where temporal depends is critical was modeled. Deep Neural Networks (DNN) are types of ANN that consist of more than two hidden layers, allowing to modeling of complex relationships [39]. With further development of DNN, adaptive control is revolutionized to handle high dimensional complex and nonlinear systems. This has led to even more robust solutions even in real-time environments, though challenges remain in computational complexity and the need for large training datasets [40]. These advancements highlight the transformative role of neural networks in adaptive control, offering enhanced performance, robustness, and flexibility in managing nonlinear and uncertain systems [41].

Adaptive control generates control inputs by taking the system's slowly varying dynamics into account and driving it to follow the desired control law , [42]. As a result, even in the presence of disturbance, the controller performance is faster and more reliable. Generally, the adaptive control approaches are of two types: direct and the indirect adaptive control scheme. In the

direct control scheme the parameters of the controller are updated based on the controller error, whereas in the indirect adaptive scheme, the controller parameters are updated online using the identification error. The plant is identified parallel to update the controller parameters based on the changing dynamics of the system. By updating the model continuously, this approach shows improved stability and reliability in handling time-varying systems even in the presence of disturbance [43].

1.2 Identification and control scheme using ANN

The effectiveness of indirect adaptive control depends on the accuracy of the system model. Any inaccuracy in the identification of the model could affect the performance of the controller. The knowledge about the dynamics of the plant under observation must be known at all times. In scenarios where the knowledge about the system is limited or unavailable, system identification becomes essential [44]. The process of identification involves choosing the model structure and setting up the structure for ANN to approximate the unknown dynamic plant or system. The nonlinear plant may belong to the Model 1 as given in Eq.(1.1) or can belong to any of three models as below [45]:

Model 2:

$$y_p(k+1) = \sum_{i=0}^{n-1} a_i y_p(k-i) + g[r(k), r(k-1), \ldots, r(k-m+1)] \tag{1.2}$$

Model 3:

$$y_p(k+1) = f\left[y_p(k), y_p(k-1), \ldots, y_p(k-n+1)\right] + \sum_{i=0}^{m-1} b_i r(k-i) \tag{1.3}$$

Model 4:

$$y_p(k+1) = f\left[y_p(k), y_p(k-1), \ldots, y_p(k-n+1)\right] + g[r(k), r(k-1), \ldots, r(k-m+1)] \tag{1.4}$$

Model 1 described earlier in Eq.(1.1) is the general equation that is commonly used to describe a dynamical system. Model 1, integrates both past outputs and inputs into a unified model, providing a comprehensive approach to system identification. In Model 2, the relationship between past outputs is linear and that of past and present inputs is nonlinear. While, in Model 3, the relationship between the past outputs is nonlinear and that of present and past outputs is linear. In Model 4, two separate nonlinear functions for past outputs and present and past inputs, offer more flexibility in modeling the system dynamics is considered. The

identification scheme for these models can be implemented in two modes: series-parallel and parallel-parallel mode [46], [47].

1. **Series-parallel identification scheme**: In this mode, the output of the plant is fed to the identification model for computing ANN output. Figure 1.1 shows the series-parallel identification scheme for ANN.

2. **Parallel-parallel identification scheme**: In this mode, ANN uses its past values of input to compute the output of ANN. Figure 1.2 shows the series-parallel identification scheme for ANN.

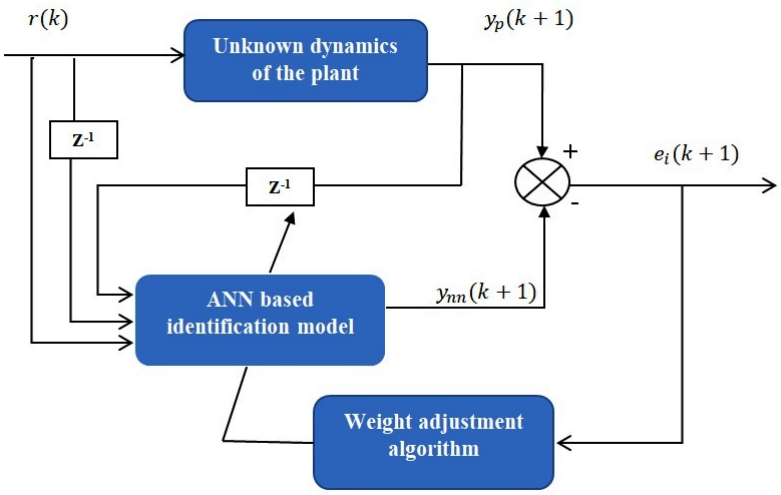


Figure 1.1: Series-parallel identification scheme

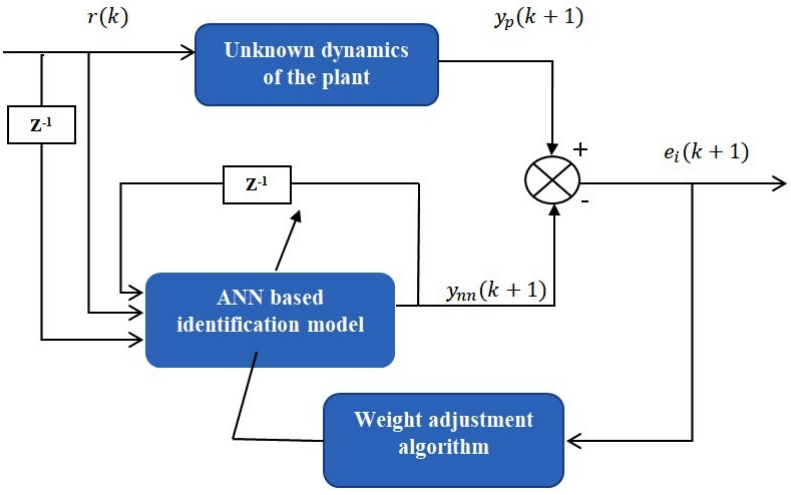


Figure 1.2: Parallel-parallel identification scheme

The parallel-parallel mode, while useful, does not guarantee stability, as the system stability depends on the input stability. The series-parallel identification is found to perform with better convergence and stability as compared to parallel-parallel-based identification [48]. Hence series-parallel based identification is used for all models mentioned above. In this thesis, Model 1 is used for the identification and control of nonlinear dynamic systems.

1.3 Different learning approaches to optimize parameters of ANN

Optimization algorithms play a major role in improving the performance of ANN for system identification and control. These algorithms are used to fine-tune the ANN parameters such as weights, and architecture to achieve better modeling and control of non-linear systems [49]. Gradient descent-based optimization is one of the most widely used optimization techniques for training ANNs. It works by computing the gradient of the error with respect to the network parameters to minimize the error [50]. In addition to gradient descent-based methods, evolutionary algorithms, such as GA have also been successfully applied to optimize the ANN architecture and weights. These population-based methods do not require gradient information and can efficiently search large, complex solution spaces. They are particularly useful when dealing with highly nonlinear and non-differentiable problems [51]. The next class of optimization techniques that have been widely integrated with ANN are swarm intelligence algorithms such as PSO. These algorithms are inspired by natural processes, such as the social behavior of birds to find the optimal solution. They are particularly effective in optimizing ANN parameters in high-dimensional and dynamic environments, offering advantages in terms of speed, robustness, and avoiding local minima [52]. The limitations of individual optimization techniques often motivate the development of hybrid methods, where two or more algorithms are combined to optimize the parameters of ANN. Such optimization techniques have improved the efficiency and performance of the training process. For instance, a combination of gradient descent BP algorithm and GA is a commonly used hybrid approach. By combining the fast convergence of gradient-based BP methods with the global search capabilities of GA, hybrid algorithms can efficiently navigate the parameter space of an ANN to find better solutions that would be difficult for each method to achieve alone [53]. Another popular approach uses PSO and when combined with BP or GA, PSO can help ANN optimization algorithms more

effectively explore large and dynamic spaces, leading to faster convergence and better solutions. PSO is often preferred over GA due to simpler implementation, faster convergence,and faster exploring of search space, especially on high dimensional space. While GA can struggle with complex high dimensional space due to tuning of mutation and crossover operations [54]. Thus, hybrid optimization approaches not only enhance the performance of ANNs in system identification and control but also offer greater flexibility and robustness in dealing with complex and time-varying systems.

1.4 Literature Review

Research works on the identification-based control and modeling of nonlinear dynamic systems are increasingly been carried out in past decades. Neural networks are being increasingly used in the literature for the identification and control of nonlinear dynamic systems that cannot be done using the conventional linear structures [55], [56]. The effectiveness of neural networks for the identification of nonlinear dynamic systems is cited in various related works [57], [58], [7]. In [31], [30], a Dynamic Recurrent Neural Network (DRNN) structure is proposed. The structure resembles NARX and MLP with recurrent self-weighted hidden neurons. The results show that DRNN performs better in terms of robustness and parameter variations due to memory's presence. FFNN models like MLP, RBFN, and functional link networks are some of the universal architectures that are capable of identifying complex nonlinear systems as suggested in the literature. In [59], another novel form of RNN, context layered local recurrent neural network is proposed. An extra context layer is included in the existing RNN structure and is trained with an adaptive learning rate. It has been found to retain past information and improve the performance of the structure better than other RNN networks in the literature. This structure is found to have rectified the drawback of FFNN making the network more dynamic and stable for identifying complex dynamics. In [60], four structures based on adaptive time-delay neural networks are proposed for the identification of different classes of nonlinear systems. The proposed structures are found to use fewer adjustable parameters and less prior information about the system over FFNN models. Four ANN structures namely ENN, modified ENN, time-delayed ANN, and internal time-delayed RNN are proposed in [61]. All four selected networks are trained using GA. The results show that the RNN structures such as ENN, modified ENN, and the internal time-delayed RNN have better identification precision than

static time-delayed ANN. In [62], the authors have evaluated the superiority of the GA over the BP algorithm to train the modified ENN and JNN network against the standard ENN and JNN structure by the addition of self-feedback connections for the context units with weights fixed between 0 and 1. Though GA does not get trapped in local minima like the BP algorithm yet updates weights on the entire population of a network. BP generally does for one layer at a time instant. In [63], a novel RNN structure is proposed and implemented using a series-parallel identification scheme. It is found to provide good mapping capabilities for training and robustness for parameter variations of complex non-linear systems. In [64], an Extended Kalman Filter (EKF) is used to update the weight equations. Though EKF generates faster convergence than BP, but gets trapped in instability caused by initial conditions during linearization. In [65], a multivalued connection weight depending on inputs involved for a better performance of modified recurrent structure over others is proposed. The RNN structures are widely used for different applications for modeling and identification of non-linear dynamical systems such as fuel cells, DC motors, chemical processes, tank systems, and fault detection [66]. In [67], a PID-based controller for automatic load frequency control of the power system is developed. The combination of PSO and Gravitational Search Algorithm (GSA) based recurrent HNN is used for the identification of the model and tuning of the parameters of the controller. The weight update equations are derived and checked for stability using the Lyapunov-based stability analysis. In [68], the author has proposed a Temporal Convolutional Network (TCN) for the identification of dynamic systems. TCN is verified against MLP and Long Short-Term Memory (LSTM). TCN and LSTM were found to give better results for large data sets and non-white noise. In [69], the authors have proposed a hybrid combination of JNN-ENN structure for a single input single output system. Online training and control of the continuously stirred tank reactor plant is carried out and EKF is used as an optimization algorithm. The performance of the proposed RNN is found to be better than FFNN. In this paper [70], a modified ENN-JNN structure with GA as an optimization algorithm is used. Optimization algorithms such as EKF, and GA though perform better than BP, yet suffer long training times due to their confined search space. With the addition of white noise into the system, the optimization method fails to predict accurate models. In [71], a Hybrid ANN based on global clustering and local learning is proposed. The clustering algorithm is used for updating model weights. Though this structure performs well, a good number of input densities is required for cluster pairing and faster convergence. In [72], the authors have

proposed two new architectures introducing a hidden layer of morphological neurons instead of a perceptron layer. Both the proposed architectures are trained using a stochastic gradient descent algorithm. The result shows that morphological neural network structure requires fewer learning parameters than perceptron structure. In [73], the authors have proposed a cascaded FFNN with ENN for disease prediction The results were verified on six different data sets and hybrid models are found to outperform single models efficiently. In [74], the author has used FFNN, RBFN, Runge-kutta neural networks, and Adaptive Neuro-Fuzzy Inference System (ANFIS) models for the identification of nonlinear systems. Runge-Kutta ANN has shown better performance than feed-forward structures and ANFIS. In this work [75], the authors have proposed a novel hybrid deep learning model for 1 hour-ahead solar forecasting. A hybrid RNN model is designed and the proposed method is found to give better forecasting results compared to standard MLP, and RNN models. The authors in [76], have proposed a cascaded FFNN with ENN to predict six categories of diseases. The results demonstrate the higher accuracy of the proposed method over other standard ENN models. In [77], ENN with NARX is designed for system identification. The results once again show better performance with hybrid models over single models. The twin-rotor multi-input multi-output system is identified on modified MLP and ENN structure in [78]. In [79], a time delay recursive neural network is used to develop an online-based direct adaptive controller. The proposed control method can be easily generalized to the actual systems, which exhibit hysteresis behavior, in contrast to those approaches established under the generic Lipschitz condition. A HNN-based identifier is proposed to identify the system for the controller. It is found that the suggested controller eliminates the performance effects caused by the direct adaptive controller's inaccuracy. In [80], RNN is used as a model to develop a model reference control method for the rotary inverted pendulum. It incorporates paraconsistent annotated logic with two-value annotations as an activation function for hidden layer neurons. The rotating inverted pendulum is a perfect tool for applying and testing the RNN due to its non-linearity, two-degree-of-freedom motion, and under-actuated system. Three RNN neural models are used to develop a model reference controller: two of them are used to represent the arm and pendulum angles, and the third one is used to operate the system while following a reference trajectory. The proposed controller is compared with classical control methods. The proposed controller is found to give better trajectory tracking capability and complexities. In [81], the authors developed a stock price prediction model through the neural network to enhance the stock

price prediction effect based on the enhanced PSO algorithm. To improve the global search ability of the algorithm in the early stage of evolution and the local search ability in the later stage of evolution, the adaptive adjustment of inertial weight is proposed, and the algorithm is improved by combining it with a neural network. This approach is based on the idea of avoiding particles falling into the same local solution as much as possible and always keeping the particles with a certain diversity. In [82], the authors created a reduced-order RNN model for distributed model predictive control of nonlinear processes utilizing feature selection techniques. A subset of input characteristics that significantly affect the prediction of system output is initially chosen using a filter, wrapper, and embedded feature selection approach. The creation of reduced-order RNN models utilizing only the chosen input features after integrating the feature selection techniques to capture the system dynamics. To stabilize the nonlinear system at steady-state, the reduced-order RNN models are then included in sequential and iterative distributed model predictive controls. In [83], a Lyapunov-based economic model predictive control technique with RNNs is used for managing switched nonlinear systems. The initial RNN structure is trained offline using operational data from the past and later updated to online learning to enhance prediction accuracy. The proposed approach is applied to systems with predetermined switching schedules. The results show its effectiveness in managing complex dynamics. In [84], an iterative learning model predictive control for complicated nonlinear systems based on a fuzzy neural network is proposed. A data-driven model is first created using a dynamic linearization technique that solely uses input and output data. An FFNN is utilized to analyze the disturbance in the established model since it has an unidentified disturbance term that could affect the controller performance. The developed controller is found to be capable of ensuring the stability of the closed-loop system while gradually reducing both modeling errors and tracking errors over time. Finally, the experimental findings support the superiority and efficacy of the developed controller. In [85], the data-driven robust optimum control approach is suggested for the control of nonlinear complex systems. The proposed technique has three advantages: To capture the relationship between the approximation errors and the control variables, a data-driven assessment technique is first developed. After that, the nonlinear system's control performance indices can be determined inside uncertain disturbances. Second, a co-evolution technique is used to construct a multi-objective resilient optimization algorithm. The control performance can then be enhanced by obtaining reliable optimal control laws. Third, a theoretical discussion of data-driven robust optimum control's robust boundedness

is presented. Analytical assurance of the control systems' stability is then possible. Last but not least, two multiple input multiple output second-order nonlinear systems are used to demonstrate the efficacy of data-driven robust optimum control. In [86], to prevent accuracy loss due to premature convergence without adding to the computational load, the authors carefully examined the evolution of a PSO algorithm and used it to incorporate more dynamic information into it. This creatively led to the development of a novel position-transitional PSO algorithm. ANNs are suitable for modeling and control of any nonlinear system as they can approximate any nonlinear function [87], [88]. Additionally, they have fast processing capabilities, allowing them for real-time control. A self-recurrent wavelet neural network-based control is proposed in [89] for the control of nonlinear dynamical systems. The modified structure is trained using the gradient descent algorithm. Adaptive learning rates are used for faster convergence. The simulation result shows the effectiveness of the suggested structure. In [90], an indirect adaptive switch controller that switches online between PID and indirect adaptive controller is proposed. The adaptation laws are developed using the gradient descent-based method and the stability is analyzed using the Lyapunov function. The results show that the proposed controller performs better than an indirect adaptive controller, and adaptive switch controller. In [91], an adaptive control approach combining both the direct and indirect adaptive scheme is proposed. The proposed controller convergence is proved by using the Lyapunov function and the proposed controller is found to improve the tracking accuracy of the controller. In [92], adaptive neural control is proposed for the control of dynamic systems. Input-output linearization technique is used to convert the object's model into a multi-integrator system and for online training, the state observation technique is used. Further, in [93], a modified adaptive controller is developed. The controller used is the backstepping control and an adaptive algorithm was developed to find the states and deal with non-symmetrical input. In [94], online adaptive control is developed with a policy iteration algorithm. The optimal control laws are developed using the Ricatti equation. In [95], a gated RNN structure such as a Gated Recurrent Unit (GRU) and LSTM for control of a stable nonlinear system is proposed. Identification-based control was carried out and the stability was proven by using incremental input and output state stability. The results show that the proposed controller has better convergence over other controllers. In [96], the authors have designed a neural controller with a dynamic surface control scheme to overcome the effect of uncertainties on the system. A control law is proposed and is approximated using an RBFN. The stability is

also analyzed using the Lyapunov function. In [97], a recurrent wavelet neural structure is used to identify and control nonlinear systems. The parameters are updated using the back-propagation algorithm and adaptive learning rates are designed to ensure fast convergence. In [30], an LRNN-based adaptive control is proposed for identification-based control of nonlinear systems, and the Lyapunov function is used for training the algorithm. The results show better response than FFNNs. In [98], a comparative analysis of various FFNN and RNN-based controllers for adaptive control of nonlinear dynamic systems is carried out. Dynamic back propagation is used for updating the weights of the controller and the convergence of the structure is also analyzed. Simulation results prove that the RBFN-based controller performs better than NARX and FFNN-based controllers even in the presence of uncertainties. In [99], the authors have designed an optimal controller using RNN, and learning of the structure is done using reinforcement training. The controller is designed for both continuous and discrete-time systems. The performance is analyzed for robustness. Though the results look promising, the optimal controllers require the entire knowledge of the system to be known to compute the Jacobian matrix. In [100], an adaptive back-stepping-based controller for a higher-order nonlinear system is proposed. An RBFN-based time-delayed estimator architecture is designed and controlled using a back-stepping controller. The controller model is designed off-line and tested for efficiency by comparing the model with PID based controller and the original back-stepping controller model-free approach. In [101], the authors have designed an RBFN-based back-stepping controller for solving the tracking problem for quad-rotor systems. Though RBFN is a memory-less structure in combination with the fuzzy controller, it is found to increase the accuracy by taking into account the unmodelled dynamics. The performance is compared with an MLP-based back-stepping controller. It is found to provide better tracking performance. In another work [102], the author has analyzed the dynamic behavior of fractional order chaotic systems with RBFN. The parameters of the system are computed with a Runge-Kutta solver and designed with RBFN for various initial conditions of the Lorentz system. Average mutual information techniques are used to evaluate the time delay pattern and the performance of the proposed RBFN is found to be exceptionally high. In [103], an RBFN model is developed based on the fuzzy responses and predictions. A squared error-based method is used to estimate the fuzzy parameters and tuning constants. The performance of the proposed Fuzzy RBFN is found to be effective in comparison to conventional fuzzy controllers. A fuzzy RBFN-based SMC is used for control of nonlinear systems of varying degrees in this work [104].

The proposed method does not require prior information about the system as the controller is simulated online. The method was found to improve the final threshold stability and yielded better performance than RBFN-based SMC. Further, the based adaptive controllers have found numerous applications in real-time. In [105], the characteristics and progress of NN-based adaptive control are tested on Nuclear power plants. The performance is compared with ANN-based PID, self-tuning PID, and tested for efficiency with respect to disturbance. A car driving-based simulator model based on adaptive and robust control with online learning is designed in [106]. This RBFN structure-based adaptive control does not require system information. In another work [107], an iterative learning control of a nonlinear system based on ANN is designed. The controller update rule is updated by iterative training and the updated rules were incorporated in the next trial process to achieve a zero error at convergence even in terms of uncertainties. The update rules are further tested in servomechanism and magnetic levitation applications. In [108], the author has studied the Lyapunov-kravoskii theorem for global asymptotic and local exponential stability for a nonlinear delay-free system. Point-wise dissipation rates are used. The proposed theorem is found to replace the use of the Lipschitz property of the function describing the system, thus maintaining local and global asymptotic stability. In [109], the author has studied a Lyapunov-kravovskii function for semi-globally practical fixed-time stable systems. L'Hopital's rule is used to determine the boundness of the Lyapunov-kravoskii function and the signals are bounded within the integral function range. This makes the tracking error converge into a small region around zero. Further, it is extended to adaptive control of higher-order stochastic time-delay nonlinear systems. The author in [110], has studied and proposed a kravoskii-based stability theorem for a model-based reinforcement training framework. The changing dynamics of the system are captured using feature extraction methods. A prime dual approach based Kravoskii theorem is used to derive the Lyapunov function. The proposed approach is found to have applications in online control of safety-critical systems such as robotic manipulators and locomotive tasks. Further in [111], a novel l_∞ input to state stability for infinite networks is proposed. Feedback for each agent is established and an ISS-based Lyapunov function is developed for infinite networks. The authors in [112] have discussed the various techniques that could effectively optimize the gain in the Lyapunov function. The Hamilton energy as a suitable Lyapunov function is proved against Lorentz and Chua oscillation in a chaotic state and the paper claims that the control of energy flow when done completely can control chaos in a nonlinear system and improve

stability.

1.5 Motivation

Based on the literature survey, many novel architectural modifications in artificial neural networks, particularly RNN are developed for specific applications. Despite these developments, how the temporal nature of RNN influences and modifies the network behavior remains an area of exploration. This gap in research motivates the need to examine how architectural changes can improve RNN's temporal learning capability and performance. For that, we have proposed new modifications in the RNN structure and tested their efficiency over the state of art networks. The controller in many works is developed online without estimating the dynamics of the plant online. This gap motivates the need to design a controller scheme that simultaneously identifies and controls a nonlinear dynamic system. The Generalisation capacity of ANN depends on the size of training data, epochs, architecture of the network, no of hidden units, and learning rate. Achieving the balance between the network size and generalization capacity remains an open area of exploration, as large networks exhibit fault tolerance but may overfit, while small networks have good generalization capacity but may require a lot of effort due to their few processing elements. Both small and large networks possess their advantages and disadvantages. An optimal architecture is large enough to learn the problem but small enough to generalize well. Approaches such as constructive algorithms, which involve the addition of hidden neurons or hidden layers to optimize model size still need to be explored. The third motivation is to design a constructive approach to optimize the network structure. The learning rate is again another important hyperparameter that determines the performance of ANN. Too small learning rate leads to slow convergence while too large leads to fluctuation around the minimum or even diverge resulting in an unstable process. The role and impact of adaptive learning rate still need exploration. The fourth motivation is to develop an adaptive learning rate that would help in attaining faster convergence and better accuracy than existing state-of-the-art methods. The major limitations of gradient descent algorithms and their variations are slow convergence and getting stuck in local minima. There are many evolutionary optimization algorithms in the literature for optimizing the parameters of ANN. However, these algorithms do not ensure stability at all times. They also exhibit more computation time and complexities in optimizing parameters. This has motivated us to explore and develop a novel hybrid learning

algorithm that is simple, robust, and ensures the overall stability of the system. Further, in most of the works in the literature, the robustness analysis for the proposed structure is not carried out. In our work, we have considered the proposed approaches for robustness analysis and tested them on various benchmark problems such as various degrees of nonlinear plant equation, Mackey glass series prediction, liquid level prediction, etc.

1.6 Contributions of the thesis

The main contributions of the thesis are as follows:

- Novel architectural modifications for ANNs, particularly on RNN topologies, have been developed to improve the network's ability to capture temporal patterns and complex dependencies.
- The modified architecture is used to design a simultaneous approach for online system identification and control, enabling real-time system adjustments and improved system adaptability.
- Lyapunov's stability principles are applied to prove the stability of the weight update equations and ensure network convergence.
- An adaptive learning rate (ALR) is developed to optimize algorithm performance, resulting in more efficient training processes.
- The structure optimization of ANN is explored. A constructive algorithm is developed for dynamically designing a feed-forward and recurrent architecture, aiming to improve the network's performance and effectiveness.
- The study also focuses on optimizing the parameters of ANN. An adaptive PSO-BP hybrid algorithm is developed to optimize the weights in feed-forward networks, aiming to improve the performance of the network.
- A comprehensive comparative analysis of the proposed models is conducted with other structures of ANN on various nonlinear problems to evaluate their relative effectiveness
- An elaborate robustness analysis to validate the ability of the proposed methods under varying conditions is also performed.

1.7 Organization of the thesis

The thesis is organized into seven parts; a brief description of the chapters is shown below,

- **Chapter 2**: The introduction was discussed in chapter 1. In this chapter, a novel modified Hybrid Elman-Jordan Neural Network (HEJNN) structure is proposed for the identification of unknown dynamics of nonlinear systems. The proposed recurrent structure consists of internal feedback layers of adjustable weights which impart necessary memory properties to the structure and improve its ability to handle the dynamical systems. The BP algorithm is used to derive the weight update equations of the proposed model. The convergence of the proposed approach is proven in the sense of Lyapunov-stability analysis. The results obtained from HEJNN are compared with other state-of-the-art neural network models such as FFNN, ENN, JNN, and LRNN. Robustness analysis is also performed to validate the efficiency of the proposed structure.

- **Chapter 3**: In this chapter, a novel hybrid Compound Recurrent Feed-Forward Neural Network (CFRNN) based on the combination of FFNN and LRNN is proposed for the identification of nonlinear dynamical systems. BP algorithm is used to derive the weight update equations and the stability of the proposed structure is also analyzed using Lyapunov-stability principles. The proposed model performance is evaluated and compared with state-of-the-art neural models such as ENN, JNN, LRNN, and FFNN. The results of the simulation demonstrate that, in comparison to other neural models, the suggested structure has provided greater prediction accuracy, better performance in the scenario of disturbance signals, and better response in the case of parameter variation.

- **Chapter 4**: In this chapter, an online simultaneous identification and indirect adaptive control framework for the nonlinear dynamical systems is developed. Both the identifier and adaptive controller utilizes the CFRNN structure developed in the previous chapter. The indirect adaptive control scheme is represented as HFRNN. To derive the weights update equations, we have applied the BP algorithm, and the stability of the proposed learning strategy is proven using the Lyapunov stability principles. We also compared the proposed method's results with those of the Jordan Network-based Controller (JNC) and the Local Recurrent Network-based Controller (LRNC) in the simulation examples. The results demonstrate that our approach performs satisfactorily, even in the presence

of disturbance signals.

- **Chapter 5**: In this chapter, a novel hybrid constructive algorithm for FFNN and RNN is developed and applied for identification of non-linear dynamical systems. The constructive algorithm developed for FFNN is denoted as CFFNN and for LRNN as CLRNN. The algorithm constructs starting from a minimum network with no hidden node and adds a hidden node when the network fails to converge properly. The hidden nodes depend on the effect of MSE on the validation dataset. To enhance the learning algorithm's performance, a novel Lyapunov's stability-based ALR is also developed. BP with ALR is used as the learning algorithm for growing both the networks. The proposed algorithm's effectiveness is demonstrated using two nonlinear systems examples. The results of CLRNN and CFFNN with ALR are compared to those of standard fixed FFNN and LRNN structures with Fixed Learning Rates (FLR), as well as CLRNN and CFFNN structures with FLR. The experimental results show that the CLRNN and CFFNN with ALR outperform the other selected neural models.

- **Chapter 6**: In previous chapters, a standard BP algorithm with and without ALR was used to train ANNs. In this chapter, a novel hybrid APSOBP algorithm for training ANN is proposed and is applied to FFNN for identification of nonlinear dynamical systems. The proposed training algorithm begins by using PSO to optimize the network weights of FFNN. Following this, BP is used to fine-tune the optimized weights, hence improving the overall solution quality. To avoid early convergence, we dynamically adjust PSO parameters such as inertia weight (w) and hyperparameters (c_1, c_2) based on a performance index e_i mechanism. The performance index is calculated as the difference between the fitness value of the global best solution of consecutive iterations. The proposed approach is evaluated against three benchmark nonlinear problems to ensure its effectiveness. The experimental results of training FFNN with the APSOBP algorithm are compared to traditional FFNN-PSO and FFNN-BP The results show APSOBP algorithm outperforms the selected methods in terms of convergence, accuracy, and robustness.

- **Chapter 7: Major conclusions and future directions** This chapter presents the conclusions of the major work carried out in the thesis and suggests some future directions for the study.

Chapter 2

Design of a Novel Robust Recurrent Neural Network for the Identification of Complex Nonlinear Dynamical Systems

2.1 Introduction

Dynamic models are ones whose output behavior depends on over time. Identification of dynamic models is a fundamental step in designing an effective controller. However, most practical problems exhibit non-linear characteristics in nature making the selection of a nonlinear model and computation of parameters difficult especially for tasks where no prior knowledge about the system under consideration is available. Unlike linear approaches, soft computing approaches offer a significant advantage in handling nonlinear systems. They can approximate complex relations without knowing about the system. ANNs, which perform well in a range of tasks, are commonly employed to approximate complex mapping functions. Because they can capture temporal dependencies, RNNs are more effective than FFNNs at modeling dynamic systems. The ENNs and JNNs are two of the most widely utilized RNN types in the field of system identification and control of nonlinear systems. The ENNs are very efficient in capturing state dependencies with the addition of a recurrent input layer. However, they are not suitable for online identification as they cannot adapt in real-time [29]. In JNN, the hidden layer receives the network's delayed outputs as inputs. The output layer's size determines the

size of the context layer. In the presence of many outputs, the JNN grows quite massive and exhibits sluggish convergence. To address the limitations, many works on combining ENN and JNN structures are being proposed. These hybrid models utilize the advantages of both networks to improve performance. This has motivated us to propose a novel modified hybrid Elman-Jordan neural network in this thesis. The proposed structure is referred to as a Hybrid Elman-Jordan Neural Network (HEJNN). It introduces an optimized additional layer between input and output layer. This novel modification not only enhances the network's adaptability for identification but also improves the convergence speed. The proposed structure is validated for its effectiveness by applying various degrees of nonlinear plant equations and disturbance ability.

2.2 Mathematical structure of HEJNN

The proposed HEJNN structure is an enhancement of the hybrid Elman-Jordan neural network with few modifications.

1. An additional trainable link has been introduced between the input layer and the output layer, forming a robust fully recurrent neural structure.

2. The delayed state of the output layer also serves as one of the inputs to the output layer. This is also made trainable.

The proposed HEJNN is shown in Figure 2.1. The HEJNN is implemented as a series-parallel identification structure as shown in Figure 2.2. The function of each layer is as below:

1. **Input layer:** This layer 1 consists of the input signals and it distributes input signals to the hidden layer neurons. The input vector $X(k) = [x_1(k), x_2(k), \ldots, x_n(k)]$ consists of 'n' number of input signals. In this work, $y_p(k-1)$ and $r(k-1)$ are used as inputs for the HEJNN structure. The proposed structure uses a minimum number of inputs to identify nonlinear dynamic systems.

2. **Hidden layer:** The input signals are further propagated to the hidden layer. To calculate its output, each hidden layer neuron multiplies its signals of input vector $X(k)$, context layer 1 vector $P(k) = [p_1(k), p_2(k), \ldots, p_n(k)]$, and context layer 2 vector $d(k) = [d_1(k), d_2(k), \ldots, d_o(k)]$ by their respective weights. A tangent hyperbolic nonlinear activation function is used. $W_x(k) = [w_{x1}(k), w_{x2}(k), \ldots, w_{xn}(k)]$ represents the input

layer's weight vector, $W_p(k) = [w_{p1}(k), w_{p2}(k), \ldots, w_{pm}(k)]$ represents the weight vector of the context layer 1, and $W_o(k) = [w_{o1}(k), w_{o2}(k), \ldots, w_{om}(k)]$ represents the weight vector of the output layer.

3. **Context layer 1:** This is the additional input layer between input and the 1^{st} hidden layer. The size of the context layer 1 is similar to that of the hidden layer. Each neuron stores the delay states of the corresponding self-connected hidden neurons. The feed-forward connections are fixed, but the recurrent connections of this layer are trainable. The feedback connections are multiplied by the context weight vector, $W_p(k)$, before being sent as input to the hidden layer.

4. **Output layer:** The output of the network is calculated by multiplying its hidden layer vector $S(k) = [s_1(k), s_2(k), \ldots, s_n(k)]$, context layer 2 vector $d(k)$, and additional links signals by their respective weight vector $W_\alpha(k) = [w_{\alpha 1}(k), w_{\alpha 2}(k), \ldots, w_{\alpha p}(k)]$. $W_d(k)$ represents the recurrent weight vector of the context layer 2. Each neuron of the output layer is computed as a function of the linear activation function. Purelin is used as a linear activation function.

5. **Context layer 2:** This layer has a local feedback. The context layer 2 is the same size as the output layer. The delayed states of the output are stored in this layer. The forward connections are fixed, but the recurrent connections are trainable. Before reaching the hidden and output layers as input, the feedback connections are compounded by multiplying with context layer 2 weight vector $W_d(k)$.

6. **An Additional weighted link:** This study introduces an additional trainable layer between the input and output layers. This new link transforms the structure into a fully recurrent neural structure. To make the layer trainable, input signals are multiplied with weight vector $W_\alpha(k)$ before reaching the output layer. The additional link layer has 'α' number of inputs to be mapped with the output layer.

The hidden layer output $S_n(k)$ is given as follows:

$$S_n(k) = g_1\left(\sum_{i=1}^{m} X(k-i)W_{xi}(k) + b_x(k)W_b(k) + P_i(k)W_{pi}(k) + d_i(k)W_{di}(k)\right) \tag{2.1}$$

where the input bias vector is $b_x(k)$, and the weight vectors for the input bias and context layer 1 are $W_b(k)$ and $W_p(k)$. The tangent hyperbolic activation function is indicated by g_1. The

context layer 1 vector is denoted by $P(k)$, the context layer 2 vector by $d(k)$, and the context layer 2 weight vector by $W_d(k)$. $W_{xi}(k)$ is the weight vector of input layer. The following is an expression for the network's output at time step k:

$$y_{hej}(k) = g_2\left(\sum_{i=1}^{n} S_n(k)W_{oi}(k) + b_o(k)W_{bo}(k) + d_i(k)W_{di}(k) + W_{\alpha i}(k)X(k-i)\right) \tag{2.2}$$

Where $b_o(k)$ is the output bias vector with the corresponding weight vector denoted as $W_{bo}(k)$. g_2 denotes the purelin activation function, $W_\alpha(k)$ is the weight vector corresponding to the additional link between input and output, and $X(k-i)$ denotes the input vector.

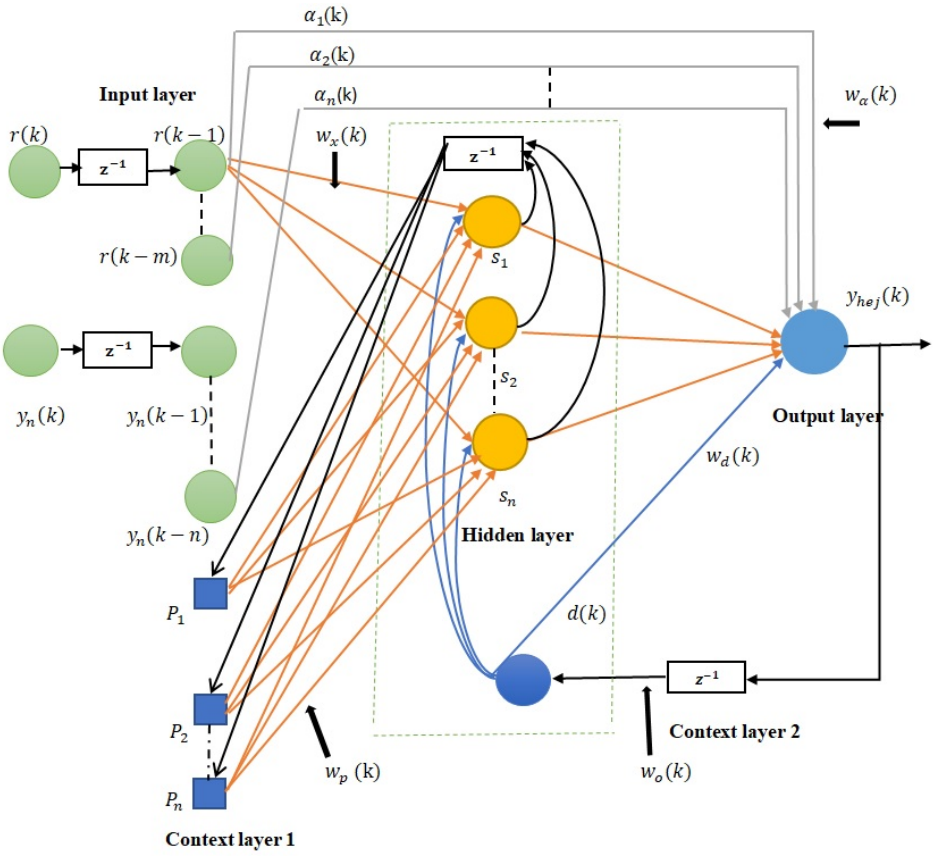


Figure 2.1: Proposed HEJNN structure

2.3 Problem statement

Let us consider a non-linear plant with desired outputs such as $y_p(k-1), y_p(k-2), \ldots, y_p(k-m)$ and desired inputs such as $r(k), r(k-1), r(k-2), \ldots, r(k-n)$. f is the non-linear mapping

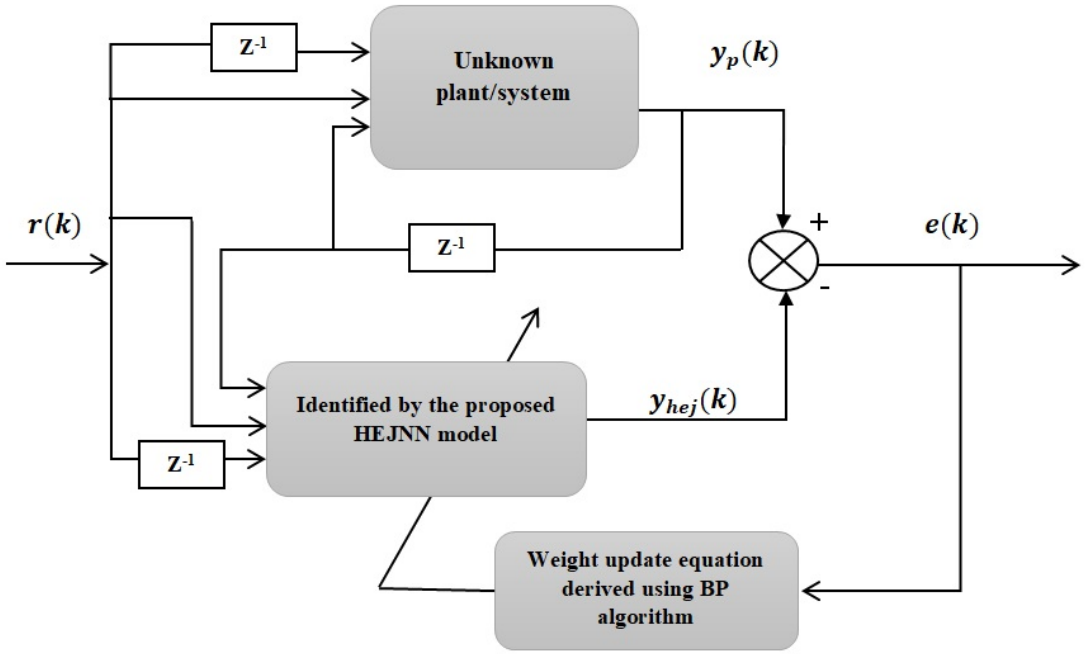


Figure 2.2: Series-parallel identification model

function between them. The identification structure of the plant is :

$$y_p(k) = f[y_p(k-1), .., y_p(k-n), r(k-1), .., r(k-m)] \tag{2.3}$$

Here, $f(.)$ can be a neural network, wavelet, or sigmoid function. In this study, a neural network is considered for non-linear approximation. n and m are the orders of the plant. Now, if HEJNN is selected as an identifier, with $\hat{f}$ being the unknown non-linear and differentiable function, the identification structure will be:

$$y_{hej}(k) = \hat{f}[y_p(k-1), r(k)] \tag{2.4}$$

where, $y_{hej}(k)$ denotes the output of HEJNN. The main objective is to approximate the non-linear function $\hat{f} \simeq f$ by keeping the error as low as possible and achieving the desired response using ANNs such as HEJNN. By tuning the parameters of the identification model, the error is reduced, i.e.:

$$\lim_{k\to\infty} |y_p(k) - y_{hej}(k)| \leq \epsilon \tag{2.5}$$

where $\epsilon \to 0$. To reach the requirement as given in Eq.(2.5), the trainable network weights are continuously updated using the standard BP algorithm

2.4 Learning algorithm

To update the tunable parameters of HEJNN, a gradient descent-based BP algorithm is used. All the network weights $W_i(k) = [W_x(k), W_o(k), W_p(k), W_d(k), W_\alpha(k)]$ are updated at every epoch by applying the chain rule. To attain this, a cost function is defined in the first instance. MSE is chosen as the cost function in this work. MSE is the average squared difference between the predicted values and the actual values. It is expressed as:

$$E(k) = \frac{1}{2}[e(k)]^2 \tag{2.6}$$

and

$$e(k) = y_p(k) - y_{hej}(k) \tag{2.7}$$

where $e(k)$ denotes the identification error. To update the weights of the output layer, the gradient of error with respect to output weight is computed using the chain rule:

$$\frac{\partial E(k)}{\partial W_o(k)} = \frac{\partial E(k)}{\partial y_p(k)} \times \frac{\partial y_p(k)}{\partial V(k)} \times \frac{\partial V(k)}{\partial W_o(k)} \tag{2.8}$$

On simplification,

$$\frac{\partial E(k)}{\partial W_o(k)} = -e(k) \times S(k) \tag{2.9}$$

where $S(k)$ indicates the output of the induced field and derived using Eq.(2.1). A linear activation function such as purelin is considered in the output layer. Further, the output weight $W_o(k)$ is updated using the Stochastic Gradient Descent (SGD) formula as below:

$$W_o(k+1) = W_o(k) - \eta \frac{\partial E(k)}{\partial W_o(k)} \tag{2.10}$$

i.e.

$$W_o(k+1) = W_o(k) + \eta e(k) S(k) \tag{2.11}$$

where η is the learning rate and its range is considered between 0 to 1. To update the weights of the input layer, the gradient of the error with respect to the input layer weight is computed using the chain rule:

$$\frac{\partial E(k)}{\partial W_x(k)} = \frac{\partial E(k)}{\partial y_p(k)} \times \frac{\partial y_p(k)}{\partial Z(k)} \times \frac{\partial Z(k)}{\partial S(k)} \times \frac{\partial S(k)}{\partial V(k)} \times \frac{\partial V(k)}{\partial W_x(k)} \tag{2.12}$$

where Z(k) and V(k) denote the induced field of the hidden layer and the induced field of the output layer respectively. On simplification, we get:

$$\frac{\partial E(k)}{\partial W_x(k)} = -e(k)W_o(k)(I - S(k)^2)X_i(k) \tag{2.13}$$

Further using the SGD rule, the weight is updated as follows:

$$W_x(k+1) = W_x(k) + \eta \frac{\partial E(k)}{\partial W_x(k)} \tag{2.14}$$

Similarly, the other weights such as $W_p(k)$, $W_d(k)$, and $W_\alpha(k)$ are updated as below:

$$W_p(k+1) = W_p(k) + \eta e(k)(I - S(k)^2)W_o(k)P(k) \tag{2.15}$$

$$W_\alpha(k+1) = W_\alpha(k) + \eta e(k)X_i(k) \tag{2.16}$$

$$W_d(k+1) = W_d(k) + \eta e(k)d(k)W_o(k)(I - S(k)^2) \tag{2.17}$$

The various iterative steps followed in the execution of the HEJNN algorithm are as shown in the flowchart in Figure 2.3. During the training, the network weights are updated every epoch, and at each epoch, the MSE is calculated. This training continues until the cost function converges to the minimum value. Once the terminating condition is met, additional performance indices such as MAE and RMSE are used to calculate the model's performance.

2.5 Lyapunov stability analysis

Stability is the main characteristic of a system's behavior. According to the Lyapunov stability criteria, if there is any energy measure in the system, then the rate of change of error derives the stability of the system. This study involves declaring the weight update equations of the structure and checking whether stability is achieved or not. The system is found to have achieved stability when the Lyapunov-based error function is minimum and positive. The goal

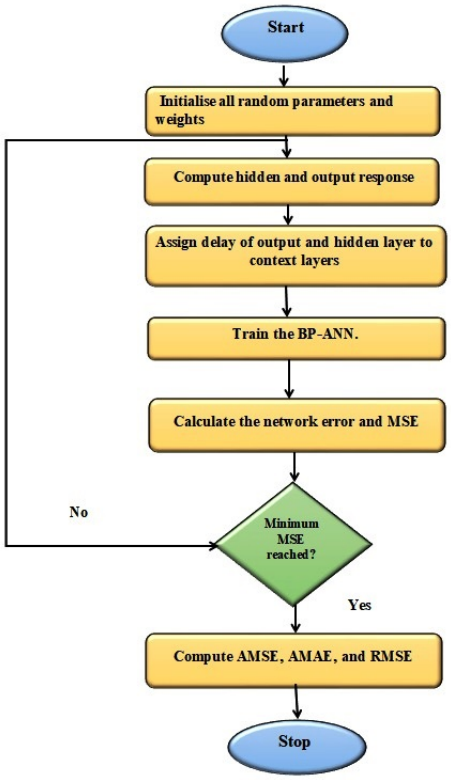


Figure 2.3: Flowchart indicating HEJNN structure training

is to minimize the Lyapunov function $V(W)$:

$$J = min(V(W)) \tag{2.18}$$

where W denotes the weights of the network.

$$V(W) > 0 \quad \text{for} \quad W > 0 \qquad V(W) = 0 \quad \text{for} \quad W = 0 \qquad \Delta V(k) \leq 0 \tag{2.19}$$

where V(W) is the Lyapunov function. For the discrete-time, the change in the Lyapunov function is:

$$\Delta V(k) = V(k) - V(k-1) \tag{2.20}$$

To update weights using the SGD formula, it is given as follows:

$$W_i(k+1) = W_i(k) - \eta \frac{\partial E(k)}{\partial W_i(k)} \tag{2.21}$$

where η is the learning rate and $E(k)$ is the error function. Assuming that the Lyapunov function $V(k)$ measures the error at each iteration, we can express it as:

$$V(k) = E(k) \tag{2.22}$$

The change in the Lyapunov function becomes:

$$\Delta V(k) = E(k) - E(k-1) \tag{2.23}$$

Substituting the weight update from Eq.(2.21) :

$$E(k) = E(k-1) - \eta \frac{\partial E(k-1)}{\partial W_i(k-1)} \tag{2.24}$$

Now, substituting this back into the expression for $\Delta V(k)$:

$$\Delta V(k) = E(k-1) - \eta \frac{\partial E(k-1)}{\partial W_i(k-1)} - E(k) \tag{2.25}$$

$$\Delta V(k) = -\eta \frac{\partial E(k-1)}{\partial W_i(k-1)} \tag{2.26}$$

Conditions for stability: For the system to be stable, $\Delta V(k) < 0$:

$$-\eta \frac{\partial E(k-1)}{\partial W_i(k-1)} < 0 \tag{2.27}$$

This condition is met when η is positive and small, and $\frac{\partial E(k-1)}{\partial W_i(k-1)} > 0$. This indicates that the Lyapunov function decreases as the error E(k) decreases, yet the system stays stable.

2.6 Simulation experiments

A total of three examples of Multi Input Single Output (MISO) nonlinear plant equations are considered to illustrate the efficiency of the HEJNN identifier. The maximum number of hidden neurons considered is 5 with a fixed learning rate of 0.001 for HEJNN, ENN, JNN, and LRNN. FFNN takes 6 hidden neurons with a fixed learning rate of 0.001 to track the desired model of the plant.

2.6.1 Example-1

Consider a non-linear dynamic plant with a difference equation as given below [36]:

$$y_p(k) = \frac{y_p(k-1)}{1+y_p^2(k-2)} + r^3(k-1) \tag{2.28}$$

The output of the plant, $y_p(k)$, depends on both its output and its prior input. The plant takes the following identification structure :

$$y_p(k) = f[y_p(k-1), y_p(k-2), r(k-1)] \tag{2.29}$$

A variable input $r(k)$ is fed to the plant as follows:

$$r(k) = \begin{cases} \sin\left(\frac{\pi k}{45}\right), & \text{for } 0 < k \leq 250 \\ 0.1\sin\left(\frac{\pi k}{45}\right) - 0.1\cos\left(\frac{\pi k}{40}\right), & \text{for } 250 < k \leq 500 \\ -\sin\left(\frac{\pi k}{20}\right), & \text{for } 500 < k \leq 900 \end{cases} \tag{2.30}$$

The performance of HEJNN is compared with other network structures such as ENN, JNN, LRNN, and FFNN in terms of performance criteria such as MSE, MAE, and RMSE.
The HEJNN is supplied with 2 inputs and the identification structure of HEJNN is as follows:

$$y_{hej}(k) = \hat{f}[y_p(k-1), r(k-1)] \tag{2.31}$$

Similarly, the ENN, JNN, LRNN, and FFNN identifiers are supplied with the following inputs below:

$$y_{ENN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1)] \tag{2.32}$$

$$y_{JNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1)] \tag{2.33}$$

$$y_{LRNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1)] \tag{2.34}$$

$$y_{FFNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k), r(k-1)] \tag{2.35}$$

The identification process has been carried over 700 time steps. Figure 2.6 shows the comparison of the response of the HEJNN identifier with other selected identifiers. Figure 2.4 shows

the comparison of MSE curves obtained for selected identifiers. Figure 2.5 shows the comparison of MAE curves obtained for selected identifiers. It is observed from these results that the proposed identifier requires fewer inputs to give a better prediction accuracy over other selected identifier structures. Table 2.1 shows the comparison of the response of all selected structures. Hence, the performance comparison can be written as follows: HEJNN > ENN > JNN > LRNN > FFNN.

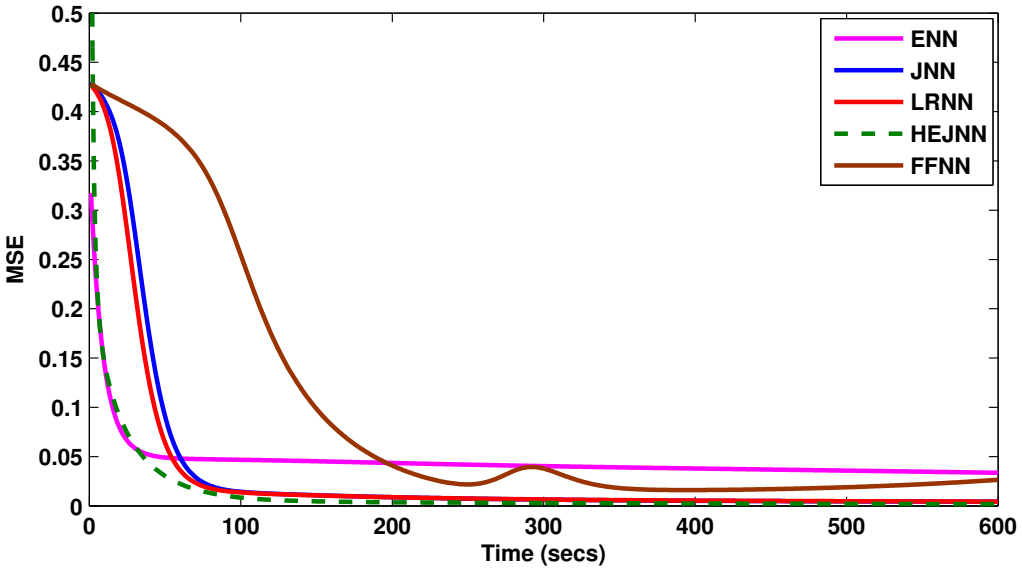


Figure 2.4: Comparison of MSE curves obtained for selected identifiers [Example-1]

Table 2.1: Comparison of performance of selected identifiers [Example-1]

S.No	Structure	No of Epochs	No of hidden neurons	No of samples in each epoch	MSE	MAE	RMSE
1	HEJNN	600	05	500	$\mathbf{1.20 \times 10^{-5}}$	**0.0021**	**0.0037**
2	ENN	600	05	500	0.0336	0.1848	0.1832
3	JNN	600	05	500	0.0043	0.0753	0.0656
4	LRNN	600	05	500	0.0042	0.0759	0.0646
5	FFNN	600	06	500	0.0143	0.1772	0.1195

2.6.2 Disturbance rejection test [Example-1]

To validate the performance of the HEJNN identifier, a sine wave as a disturbance signal, dis(k) is

$$dis(k) = \sin\left(\frac{2\pi k}{15}\right), \quad \text{for } 250 < k \leq 450 \tag{2.36}$$

is added in the above interval to the output of the network. Figure 2.7 illustrates the identifier's response to the disturbance signal. It can be seen that the identifier's response first deviated

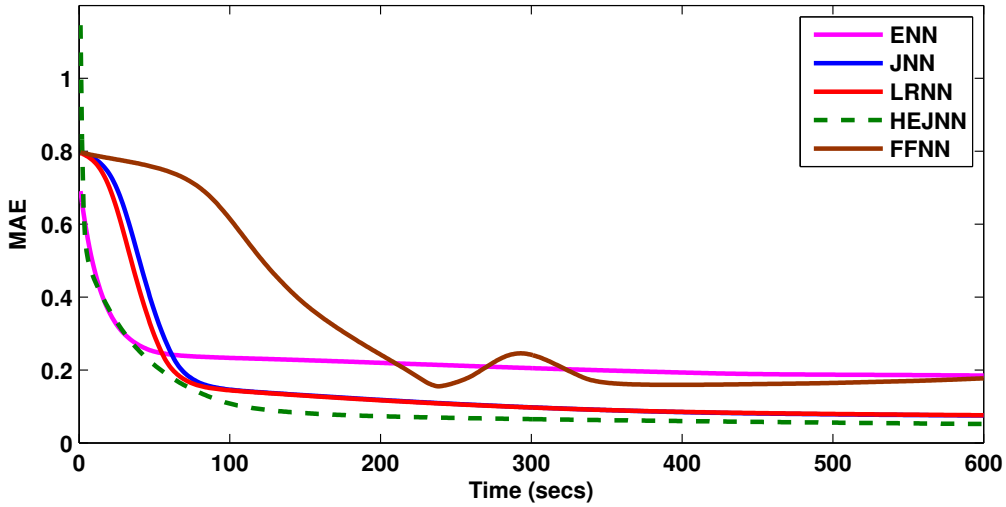


Figure 2.5: Comparison of MAE curves obtained for selected identifiers [Example-1]

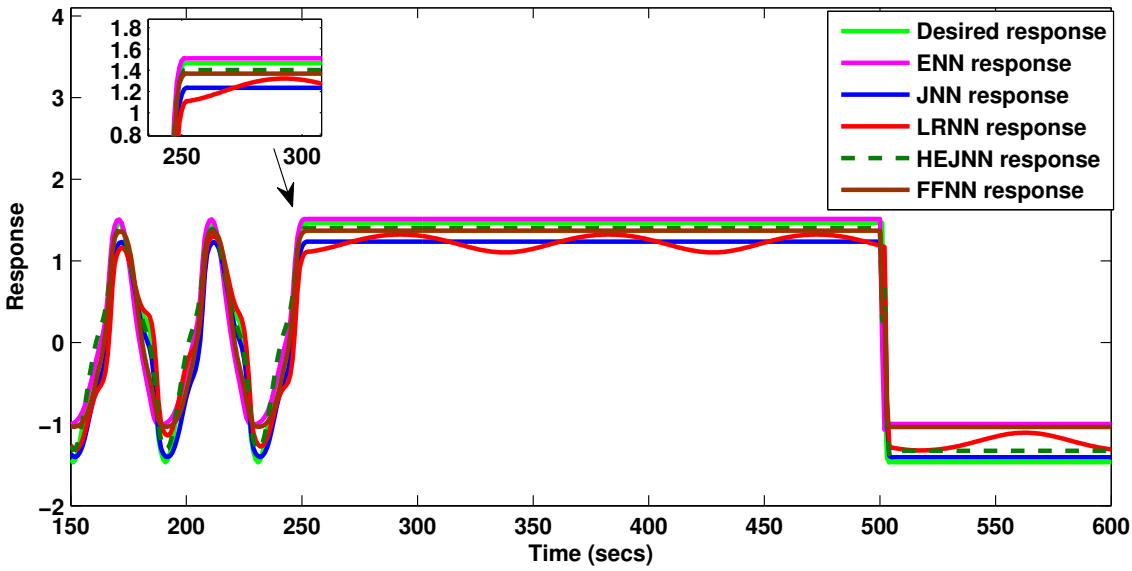


Figure 2.6: Comparison of the identifier response at the end of the training [Example-1]

from the desired response with an increase in the MSE value. However, after training, it is found that the proposed identifier response once more starts to track the intended performance and minimizes MSE to a minimum value. The comparison of the disturbance rejection ability of HEJNN with other selected neural identifiers is also shown in Figure 2.8. HEJNN is found to show better disturbance rejection ability among others.

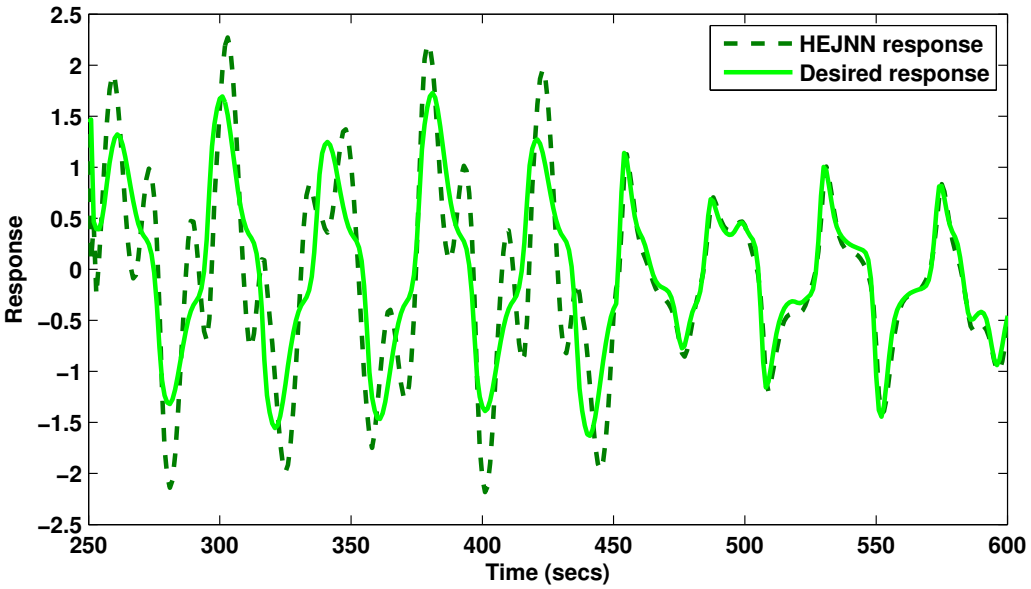


Figure 2.7: Recovering ability of the HEJNN on disturbance signal [Example-1]

2.6.3 Example-2

A difference equation of non-linear dynamic plant of degree 2 as given in [36] is considered:

$$y_p(k) = 1.8398y_p(k-1) - 0.86070y_p(k-2) + 0.010688r(k-1) + 0.0101r(k-2) \quad (2.37)$$

The identification structure of the plant is:

$$y_p(k) = f[y_p(k-1), y_p(k-2), r(k-1), r(k-2)] \quad (2.38)$$

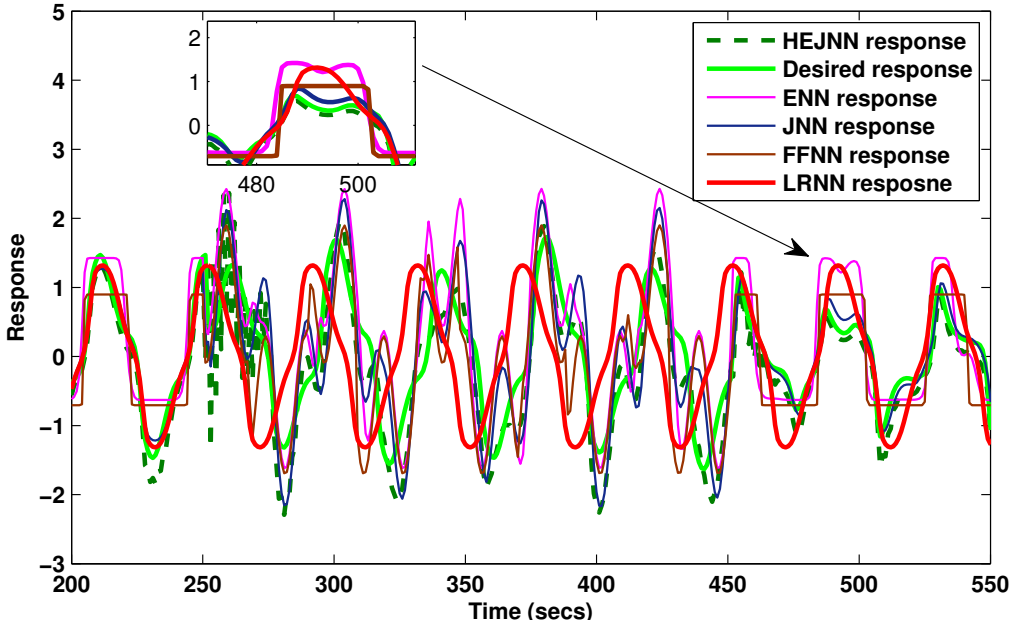


Figure 2.8: Comparison of recovering ability of selected identifier structures [Example-1]

A variable input $r(k)$ is supplied to the plant as below:

$$r(k) = \begin{cases} \sin\left(\frac{\pi k}{45}\right), & \text{for } 0 < k \leq 250 \\ 0.1\sin\left(\frac{\pi k}{45}\right) - 0.1\cos\left(\frac{\pi k}{40}\right), & \text{for } 250 < k \leq 500 \\ -\sin\left(\frac{\pi k}{20}\right), & \text{for } 500 < k \leq 900 \end{cases} \tag{2.39}$$

when HEJNN is selected as an identifier, the identification structure is as below:

$$y_{hej}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1)] \tag{2.40}$$

when ENN, JNN, LRNN, and FFNN are selected as an identifier, they take the following inputs as below:

$$y_{ENN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1), r(k-2)] \tag{2.41}$$

$$y_{JNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1), r(k-2)] \tag{2.42}$$

$$y_{LRNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1), r(k-2)] \tag{2.43}$$

$$y_{FFNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k), r(k-1), r(k-2)] \tag{2.44}$$

The identification process has been carried over 700 time steps. Figure 2.9 shows the comparison of MSE curves obtained for selected identifiers. The comparison of the MAE curves

obtained for particular identifiers is shown in Figure 2.10. The comparison of the HEJNN identifier's response with that of other chosen identifiers is displayed in Figure 2.11. The performance comparison of all the chosen identifiers is displayed in Table 2.2. The results show that the proposed identifier has better prediction accuracy over other selected structures with fewer inputs.

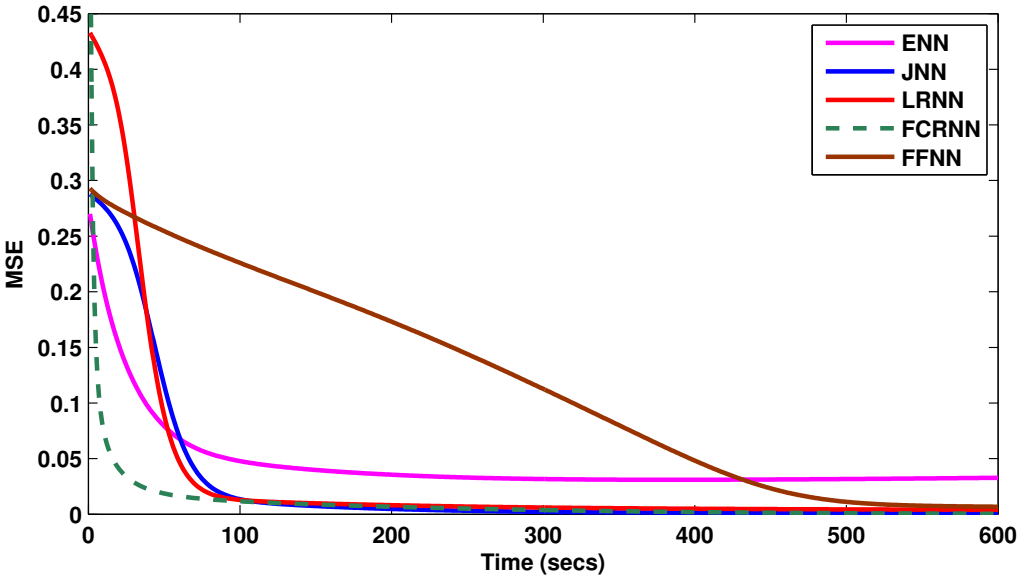


Figure 2.9: Comparison of MSE curves obtained for selected identifiers [Example-2]

Table 2.2: Comparison of performance of selected identifiers [Example-2]

S.No	Structure	No of Epochs	No of hidden neurons	No of samples in each epoch	MSE	MAE	RMSE
1	HEJNN	600	05	500	$\mathbf{1.83 \times 10^{-5}}$	**0.0024**	**0.0043**
2	ENN	600	05	500	0.0388	0.2266	0.1969
3	JNN	600	05	500	0.0011	0.0429	0.0339
4	LRNN	600	05	500	0.0012	0.0431	0.0340
5	FFNN	600	06	500	0.0105	0.1185	0.1026

2.6.4 Disturbance rejection test [Example-2]

A disturbance signal, $dis(k)$ in the form of a sine wave is added at the output as a range of time instants between $250 < k \leq 450$ to evaluate the robustness of the proposed HEJNN model.

$$dis(k) = \sin\left(\frac{2\pi k}{15}\right), \quad \text{for } 250 < k \leq 450 \tag{2.45}$$

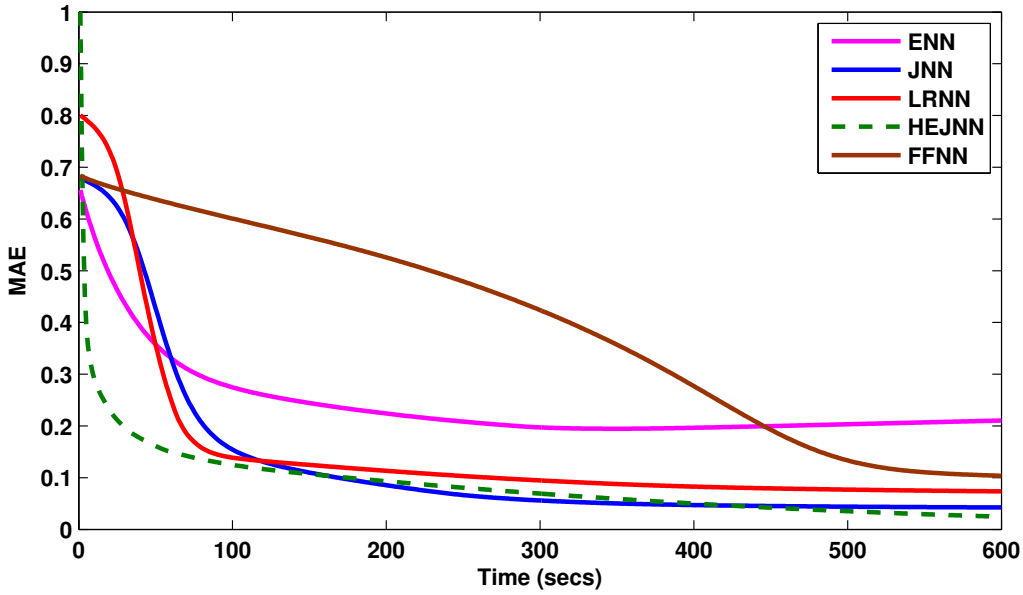


Figure 2.10: Comparison of MAE curves obtained for selected identifiers [Example-2]

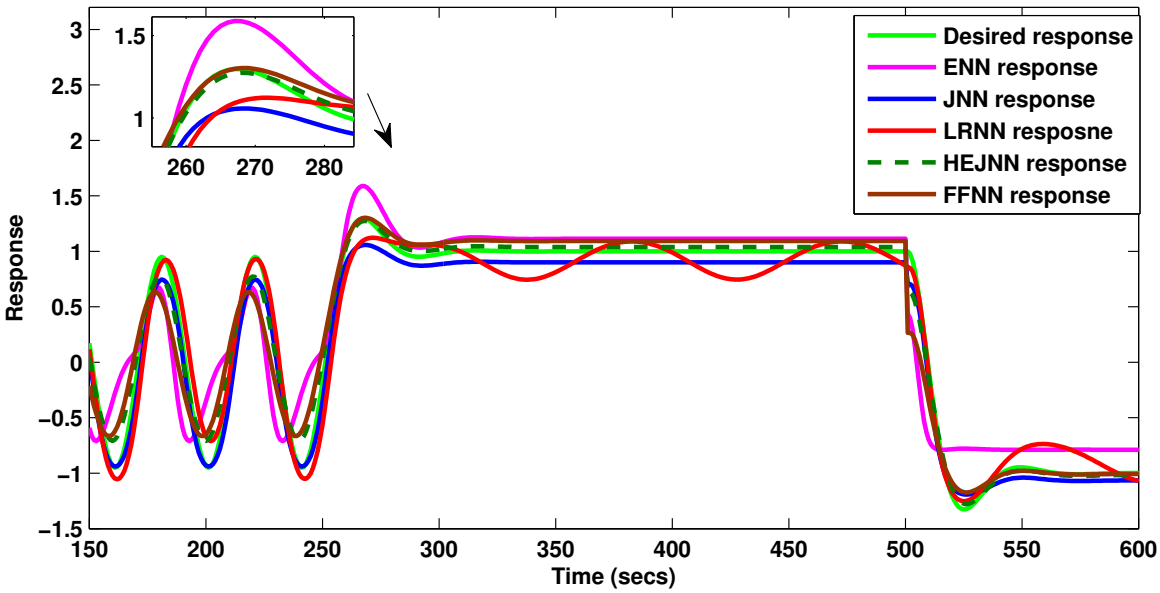


Figure 2.11: Comparison of identifier response at the end of the training [Example-2]

Table 2.3: Comparison of Performance of selected identifiers [Example-3]

S.No	Structure	No of Epochs	No of Hidden Neurons	No of Samples in Each Epoch	MSE	MAE	RMSE
1	HEJNN	600	05	500	$\mathbf{8.69 \times 10^{-6}}$	**0.0024**	**0.0029**
2	ENN	600	05	500	0.0127	0.1435	0.1128
3	JNN	600	05	500	0.0020	0.0569	0.0442
4	LRNN	600	05	500	0.0019	0.0558	0.0435
5	FFNN	600	06	500	0.0080	0.1112	0.0896

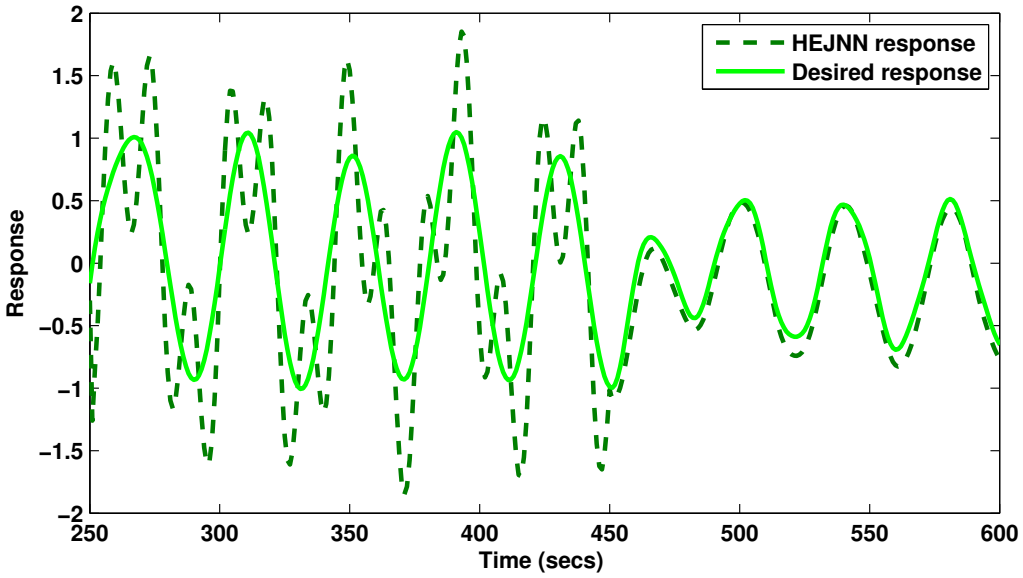


Figure 2.12: Recovering ability of the HEJNN on disturbance signal [Example-2]

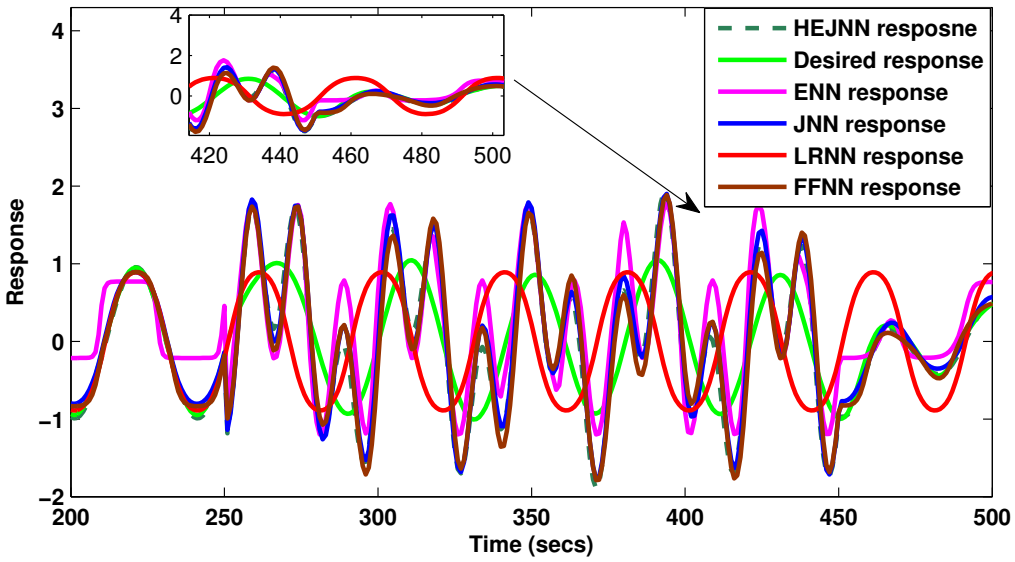


Figure 2.13: Comparison of recovering ability of selected identifier structures [Example-2]

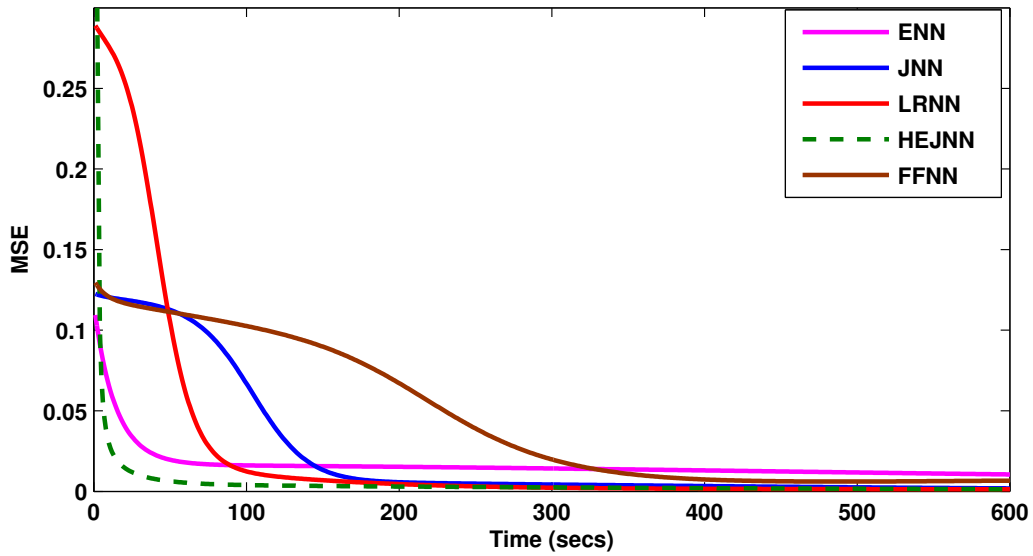


Figure 2.14: Comparison of MSE curves obtained for selected identifiers [Example-3]

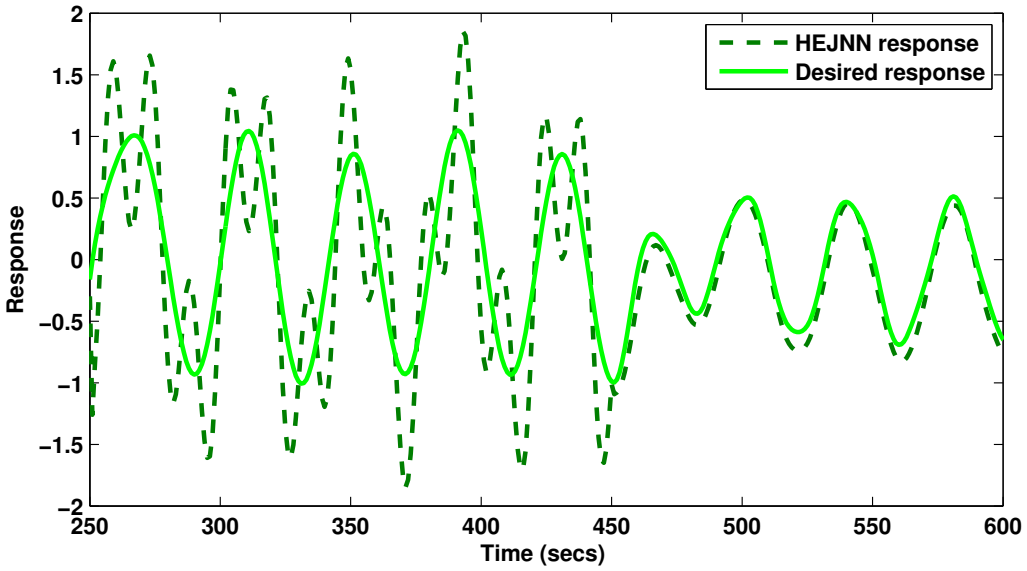


Figure 2.15: Comparison of MAE curves obtained for selected identifiers [Example-3]

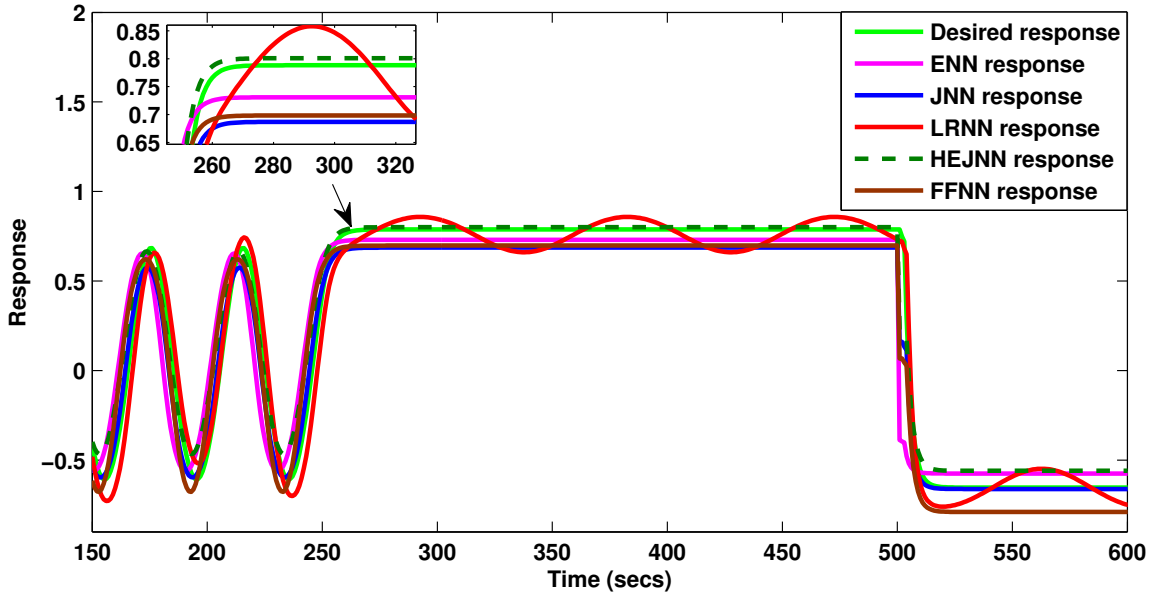


Figure 2.16: Comparison of identifier response at the end of the training [Example-3]

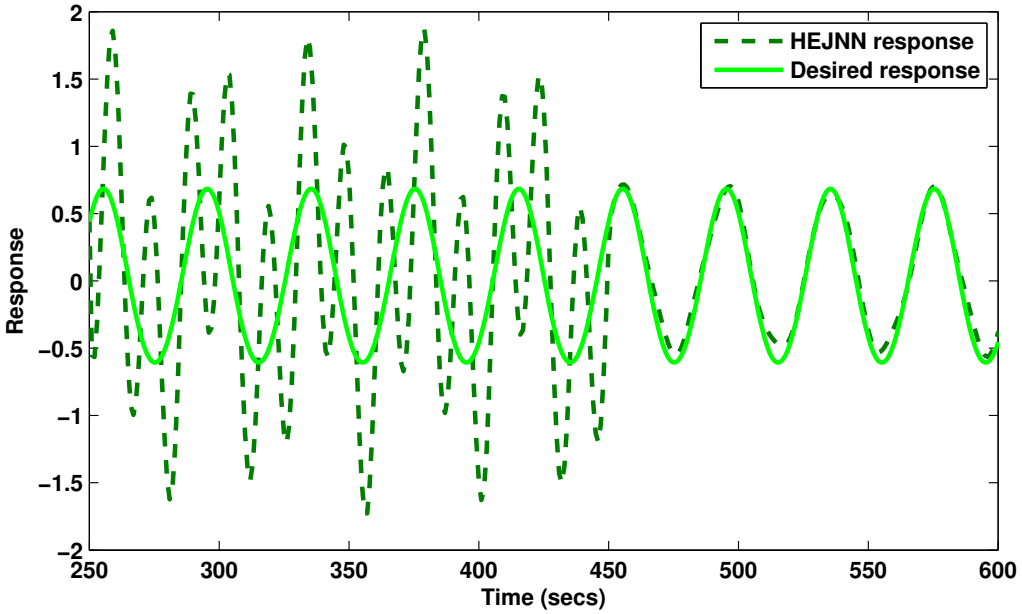


Figure 2.17: Recovering ability of the HEJNN on disturbance signal [Example-3]

The corresponding response is shown in Figure 2.12. The proposed identifier is robust enough to bring back the increased MSE to zero matching the desired plant response. The comparison of the disturbance rejection ability of HEJNN with other selected neural identifiers is also shown in Figure 2.13. The HEJNN shows better disturbance rejection ability among others.

2.6.5 Example-3

Consider the non-linear plant described by the difference equation as given in [36]:

$$y_p(k) = 0.72y_p(k-1) + 0.025y_p(k-2)r(k-1) + 0.001r^2(k-2) + 0.2r(k-3) \quad (2.46)$$

The identification structure of the plant will be:

$$y_p(k) = f[y_p(k-1), y_p(k-2), r(k-1), r(k-2), r(k-3)] \quad (2.47)$$

The above equation is supplied a variable input $r(k)$, where

$$r(k) = \begin{cases} \sin\left(\frac{\pi k}{45}\right), & \text{for } 0 < k \leq 250 \\ 0.1\sin\left(\frac{\pi k}{45}\right) - 0.1\cos\left(\frac{\pi k}{40}\right), & \text{for } 250 < k \leq 500 \\ -\sin\left(\frac{\pi k}{20}\right), & \text{for } 500 < k \leq 900 \end{cases} \quad (2.48)$$

Again, the HEJNN identifier will take the following identification structure as below:

$$y_{hej}(k) = \hat{f}[y_p(k-1), r(k-1)] \quad (2.49)$$

where $r(k-1)$ and $y_p(k-1)$ are two inputs used by the proposed identifier. The ENN, JNN, LRNN, and FFNN take the following identification structure:

$$y_{ENN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1), r(k-2)] \quad (2.50)$$

$$y_{JNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1), r(k-2)] \quad (2.51)$$

$$y_{LRNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1), r(k-2)] \quad (2.52)$$

$$y_{FFNN}(k) = \hat{f}[y_p(k-1), y_p(k-2), r(k-1), r(k-2), r(k-3)] \quad (2.53)$$

From the results, the proposed HEJNN model shows better performance and the performance can be given in the following order: HEJNN > ENN > JNN > LRNN > FFNN. Figure 2.14 shows the comparison of MSE curves obtained for selected identifiers for this example. Figure 2.15 shows the comparison of MAE curves obtained for selected identifiers for this example. Figure 2.16 shows the response obtained for the proposed identifier among others. Table 2.3 gives the comparison of the performance of HEJNN with other selected structures. It can be seen from the results that the proposed identifier gives a better prediction accuracy as compared to other considered structures with fewer inputs.

2.6.6 Disturbance rejection test [Example-3]

The HEJNN structure is validated for robustness by introducing a sine wave as a sudden disturbance, $dis(k)$ to the output of the network as

$$dis(k) = \sin\left(\frac{2\pi k}{15}\right), \quad \text{for } 250 < k \leq 450 \tag{2.54}$$

The response of HEJNN for the introduced disturbance is as shown in Figure 2.17. The proposed identifier though initially fluctuates, but is found to respond quickly and bring down the increased MSE to zero, hence tracking the plant's desired response. The comparison of the performance of the disturbance rejection ability of HEJNN with other selected identifiers is shown in Figure 2.18. HEJNN shows better disturbance rejection ability among others.

2.7 Conclusion

In this work, a novel recurrent neural network, known as the HEJNN model, is proposed for the identification of complex nonlinear dynamical systems. The weights of the proposed model are updated using the BP method. The convergence of the learning algorithm is proved using the Lyapunov-stability analysis. A total of 3 simulation examples are considered in the experimental study for testing the identification ability of the proposed model. The comparison is done in terms of MSE, MAE, RMSE, the number of hidden neurons, the number of input parameters, and the recovering ability of the identifier. From the results and simulation, it can be seen that the proposed HEJNN performed better than the other selected structures for all the considered examples. The ability of HEJNN to recover from any external disturbance is

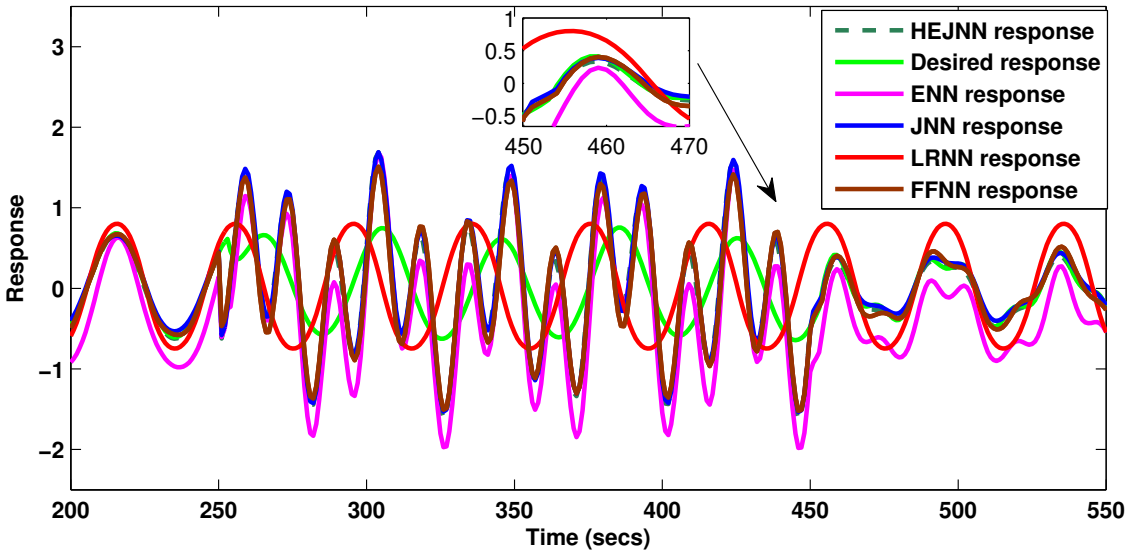


Figure 2.18: Comparison of recovering ability of selected identifier structures [Example-3]

also found to be faster than ENN, JNN, LRNN, and FFNN structures.

Chapter 3

A recurrent neural network-based identification of complex nonlinear dynamical systems: a novel structure,stability analysis and a comparative study

3.1 Introduction

A modified hybrid Elman-Jordan neural network (HEJNN) has been proposed in the previous chapter for the effective identification of nonlinear dynamical systems. Based on simulations and findings, the hybrid structure model is found to outperform other traditional neural network topologies in terms of performance and noise recovery. However, in real-time scenarios when simplicity is necessary, its complexity presents challenges. The HEJNN structure is found to have a greater number of tunable weights and parameters. The computation time taken by HEJNN is also large and has more layers before producing the output. To address this issue, this chapter introduces another novel architecture simpler than HEJNN that retains effective identification dynamics for nonlinear dynamic systems. We have proposed a hybrid Compound Recurrent Feed-forward Neural network (CRFNN) structure. The proposed CRFNN structure is a combination of FFNN and LRNN. FFNN is known to have a simpler architecture with good static mapping ability and LRNN is known to have good temporal dynamic capturing ability.

By combining these two structures, the proposed hybrid model combines the advantages of both FFNN and LRNN. BP algorithm is used to derive the weight update equations and the Lyapunov-stability principles are applied to test the stability of the proposed framework. The proposed model is evaluated against the existing neural models, including the ENN and JNN neural networks. The result shows that the performance of the proposed CRFNN outperforms other neural structures with lesser inputs and complexity.

3.2 Mathematical structure of Hybrid CRFNN model

Figure 3.1 shows the structure of the proposed CRFNN. The network consists of an input layer, a hybrid hidden layer that combines both FFNN and LRNN and an output layer. The functions of the proposed CRFNN structure are as follows:

1. **Input layer:** The input layer generates the input signals and sends them to the hidden layer. In this work, delayed external input $r(k-1)$ and output of the plant $y_p(k-1)$ are considered as inputs. The input vector is denoted as $X(k) = [r(k-1), y_p(k-1)]$. The input signals are passed to the FFNN structure in the hidden layer through the feed-forward weight connections $W_{if}(k)$ and to the LRNN portion of the hidden layer through the recurrent weight connections $W_{ir}(k)$.

2. **Hidden layer:** The output of the hidden layer is computed through two separate paths. In the FFNN path, the output of the induced field is calculated from the received input signals from the input layer, and in the LRNN path, the output of the induced field is calculated from the received input signals from the input layer and the local recurrent connections from the hidden neurons. Before being transferred to the output layer, the two outputs are added together and sent through the non-linear activation function.

3. **Output layer:** The combined hidden layer output is sent to the output layer and is sent through a linear activation function.

The induced field of the hidden layer is computed below:

$$F_j(k) = f_1\left(\sum_{j=1}^{n} X(k-j)W_{if}(k) + b_{fx}(k)W_b(k)\right) \tag{3.1}$$

$$R_j(k) = g_1\left(\sum_{j=1}^{m} X(k-j)W_{ir}(k) + L(k)W_{lr}(k) + b_{fx}(k)W_b(k)\right) \tag{3.2}$$

where f_1 and g_1 indicate the non-linear tangent hyperbolic activation function at the hidden layer. The output of hybrid CRFNN is given by:

$$y_{crfnn}(k) = f\left(\sum_{j=1}^{m} F_j(k)W_{of}(k) + b_{oh}(k)W_{bh}(k) + R_j(k)W_{or}(k)\right) \tag{3.3}$$

Where $W_{of}(k)$ and $W_{or}(k)$ are the output weight vectors of FFNN and RNN respectively. b_{oh} denotes the output bias vector and $W_{bh}(k)$ is the corresponding weight vector. f is the linear activation function used in the output layer.

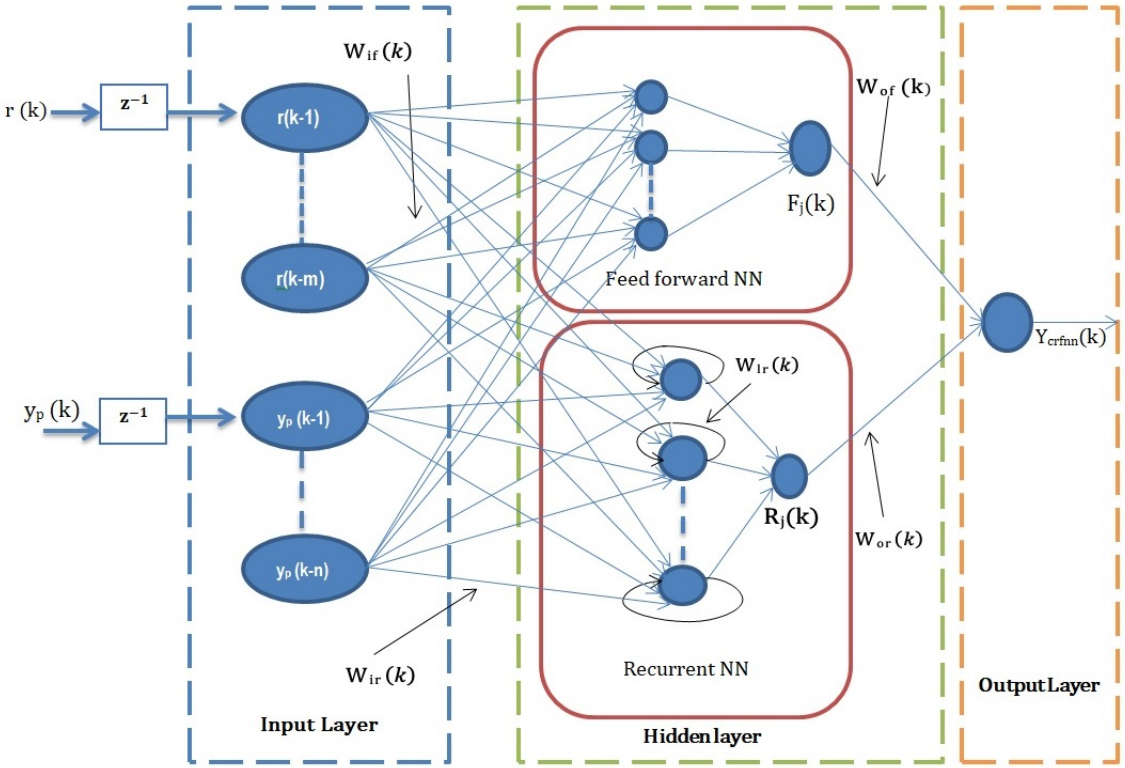


Figure 3.1: Proposed CRFNN structure

3.3 Learning algorithm

A gradient-based backpropagation algorithm is used to train the CRFNN network. Now, the first step will be to define a cost function. MSE is taken as the cost function. The MSE is defined as:

$$E(k) = \frac{1}{2}[e(k)]^2 \tag{3.4}$$

where, $e(k) = y_p(k) - y_{crfnn}(k)$ is identification error. The MSE is computed until the identification error reaches a minimum value or zero.

3.3.1 Update weights between output layer and hidden layer

To update the output layer weights, the error is back-propagated from the output layer $y_{crfnn}(k)$ to the hidden layer of FFNN denoted as $F_j(k)$ and RNN as $R_j(k)$. The gradient of error with respect to the weight of the FFNN $w_{of}(k)$ and that of RNN $W_{or}(k)$ are calculated as below:

$$\frac{\partial E(k)}{\partial W_{of}(k)} = \frac{\partial E(k)}{\partial y_{crfnn}(k)} \times \frac{\partial y_{crfnn}(k)}{\partial W_{of}(k)} \tag{3.5}$$

$$\frac{\partial E(k)}{\partial W_{or}(k)} = \frac{\partial E(k)}{\partial y_{crfnn}(k)} \times \frac{\partial y_{crfnn}(k)}{\partial W_{or}(k)} \tag{3.6}$$

On further simplification:

$$\frac{\partial E(k)}{\partial W_{of}(k)} = -e(k) \times F_j(k) \tag{3.7}$$

$$\frac{\partial E(k)}{\partial W_{or}(k)} = -e(k) \times R_j(k) \tag{3.8}$$

where $F_j(k)$ and $R_j(k)$ indicates the induced fields of CRFNN.

3.3.2 Update weights between the hidden layer and input layer

To calculate and update weights of the input and hidden layer, the error is back propagated further from the hidden to the input layer. The change of gradient error with respect to hidden layer local weights $W_{lr}(k)$, and input layer weights $W_{if}(k)$ are calculated as below:

$$\frac{\partial E(k)}{\partial W_{lr}(k)} = \frac{\partial E(k)}{\partial y_{crfnn}(k)} \times \frac{\partial y_{crfnn}(k)}{\partial R_j(k)} \times \frac{\partial R_j(k)}{\partial W_{lr}(k)} \tag{3.9}$$

$$\frac{\partial E(k)}{\partial W_{if}(k)} = \frac{\partial E(k)}{\partial y_{crfnn}(k)} \times \frac{\partial y_{crfnn}(k)}{\partial F_j(k)} \times \frac{\partial F_j(k)}{\partial W_{if}(k)} \tag{3.10}$$

Further, the calculated gradients from Eq.(3.7) - Eq.(3.10) are used to update the weights using the SGD formula. The weights $W_{lr}(k), W_{if}(k), W_{of}(k), W_{or}(k)$ are updated as follows:

$$W_{lr}(k)(new) = W_{lr}(k)(old) - \eta e(k) \frac{\partial E(k)}{\partial W_{lr}(k)} \tag{3.11}$$

$$W_{if}(k)(new) = W_{if}(k)(old) - \eta e(k) \frac{\partial E(k)}{\partial w_{if}(k)} \tag{3.12}$$

$$W_{of}(k)(new) = W_{of}(k)(old) - \eta e(k) \frac{\partial E(k)}{\partial w_{of}(k)} \tag{3.13}$$

$$W_{or}(k)(new) = W_{or}(k)(old) - \eta e(k)\frac{\partial E(k)}{\partial w_{or}(k)} \tag{3.14}$$

Here η denotes the learning rate and the range considered is between 0 to 1.

3.4 Lyapunov stability analysis

This study involves declaring the weight update equations of the proposed structure and checking whether stability is achieved or not. The system is found to have achieved stability when the Lyapunov-based error function is minimum and positive. The goal is to minimize the Lyapunov function $V(W)$:

$$J = min(V(W)) \tag{3.15}$$

where W denotes the weights of the network.

$$V(W) > 0 \quad \text{for} \quad W > 0 \qquad V(W) = 0 \quad \text{for} \quad W = 0 \qquad \Delta V(k) \leq 0 \tag{3.16}$$

where V(W) is the Lyapunov function. For the discrete-time, the change in the Lyapunov function is:

$$\Delta V(k) = V(k) - V(k-1) \tag{3.17}$$

To update weights using the stochastic gradient descent formula:

$$W_i(k+1) = W_i(k) - \eta\frac{\partial E(k)}{\partial W_i(k)} \tag{3.18}$$

where η is the learning rate and $E(k)$ is the error function. Assuming that the Lyapunov function $V(k)$ measures the error at each iteration, we can express it as:

$$V(k) = E(k) \tag{3.19}$$

The change in the Lyapunov function becomes:

$$\Delta V(k) = E(k) - E(k-1) \tag{3.20}$$

Substituting the weight Update :

$$E(k) = E(k-1) - \eta \frac{\partial E(k-1)}{\partial W_i(k-1)} \tag{3.21}$$

Now, substituting this back into the expression for $\Delta V(k)$:

$$\Delta V(k) = E(k-1) - \eta \frac{\partial E(k-1)}{\partial W_i(k-1)} - E(k) \tag{3.22}$$

$$\Delta V(k) = -\eta \frac{\partial E(k-1)}{\partial W_i(k-1)} \tag{3.23}$$

Conditions for stability: For the system to be stable, $\Delta V(k) < 0$:

$$-\eta \frac{\partial E(k-1)}{\partial W_i(k-1)} < 0 \tag{3.24}$$

This condition is met when η is positive and small, and $\frac{\partial E(k-1)}{\partial W_i(k-1)} > 0$. This indicates that the Lyapunov function decreases as the error E(k) decreases, yet the system stays stable.

3.5 Simulation experiments

In this section, the performance of CRFNN is evaluated. The proposed structure is compared against neural models such as ENN, JNN, LRNN, and FFNN. Performance indices such as AMSE, AMAE, and RMSE are selected as evaluation indexes to measure the efficiency of the proposed structure. A fixed learning rate of 0.0001 is considered for all the simulation examples.

3.5.1 Example-1: A nonlinear plant with degree 3

A real-time nonlinear plant equation as in [36] is considered:

$$y_p(k) = \frac{y_p(k-1)}{1 + y_p^2(k-2)} + r^3(k-2) \tag{3.25}$$

where $y_p(k)$ is the plant equation. The plant's output depends on both the present and past input-output values. The plant takes the following identification structure:

$$y_p(k) = f[y_p(k-1), y_p(k-2), r(k-2)] \tag{3.26}$$

Here, f is the nonlinear function that maps the inputs and outputs. The proposed structure is applied with an external input $r(k) = sin(\frac{2\pi k}{100})$. When CRFNN, FFNN, LRNN, ENN, and JNN are chosen as identifiers, the identification structure is as below:

$$y_{crfnn}(k) = \hat{f}_1[y_p(k-1), r(k-1)] \tag{3.27}$$

$$y_{lrnn}(k) = \hat{f}_2[y_p(k-1), y_p(k-2), r(k-1)] \tag{3.28}$$

$$y_{ffnn}(k) = \hat{f}_3[y_p(k-1), y_p(k-2), r(k), r(k-1)] \tag{3.29}$$

$$y_{ENN}(k) = \hat{f}_4[y_p(k-1), y_p(k-2), r(k-1)] \tag{3.30}$$

$$y_{JNN}(k) = \hat{f}_5[y_p(k-1), y_p(k-2), r(k-1)] \tag{3.31}$$

Where $\hat{f}_1, \hat{f}_2, \hat{f}_3, \hat{f}_4, \hat{f}_5$ are the nonlinear functions of the respective identifier. When $\hat{f}_1, \hat{f}_2, \hat{f}_3, \hat{f}_4, \hat{f}_5 \simeq f$, the identified model is said to behave like a plant model. About 900 samples are used for training and 400 samples are used for validation. The training is conducted in offline mode for about 900 time-epochs.

3.5.2 Parameter variation in testing phase [Example-1]

To validate the proposed structure, a new input $r(k)$ is supplied as given below:

$$r(k) = \begin{cases} \frac{sin(\pi k)}{40}, & \text{for } 0 < k <= 250 \\ \frac{0.09sin(\pi k)}{45} - & \\ \frac{cos(2\pi k)}{40}, & \text{for } 250 < k <= 450 \\ \frac{0.3sin(2\pi k)}{15} + & \\ \frac{0.1sin(2\pi k)}{320} + & \\ \frac{0.6sin(2\pi k}{40}, & \text{for } 450 < k <= 900 \end{cases} \tag{3.32}$$

Figure 3.2 shows the response obtained from all the selected identifier structures. Figure 3.3 shows the MSE curve obtained from all the selected identifier structures. Figure 3.4 shows the MAE curve obtained from all the selected identifier structures. The proposed CRFNN identifier is found to perform superior than other selected neural models. The error is also found to decrease to a very minimum value with very little computational time. The performance

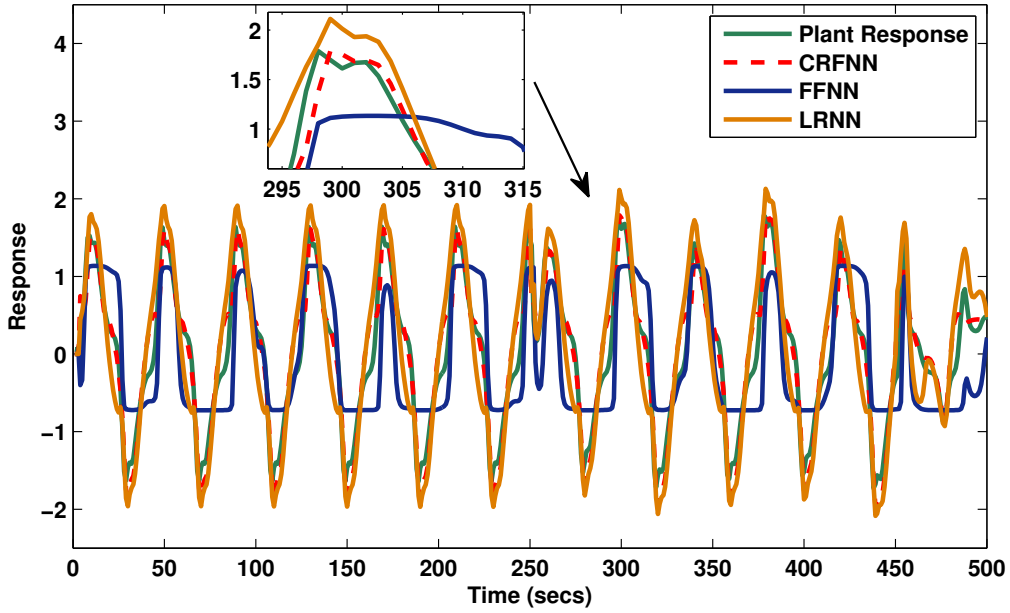


Figure 3.2: Comparison of identifier response obtained for selected identifiers [Example-1]

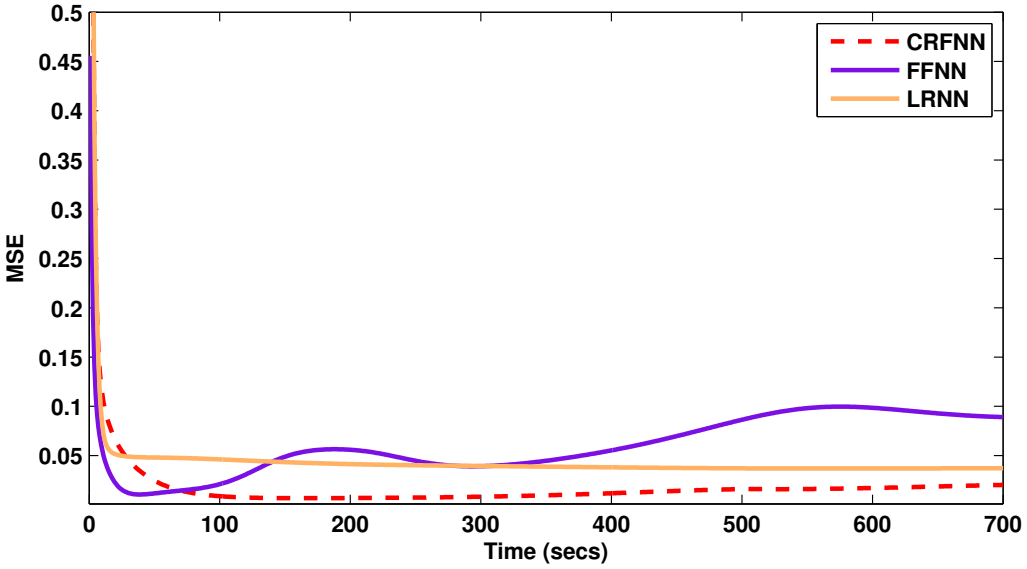


Figure 3.3: Comparison of MSE curves obtained for selected identifiers [Example-1]

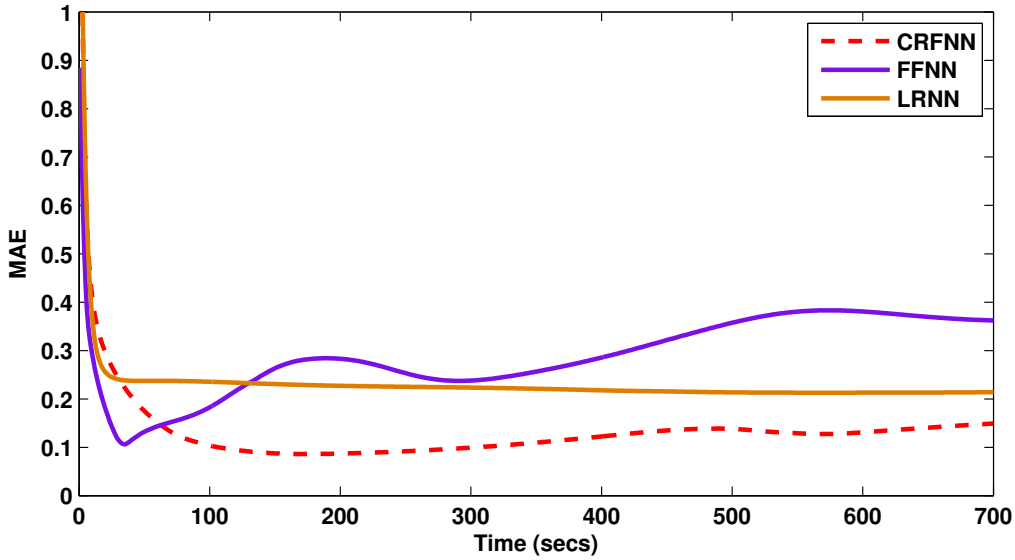


Figure 3.4: Comparison of MAE curves obtained for selected identifiers [Example-1]

of the identifiers is evaluated against major performance parameters such as AMSE, AMAE, and RMSE. From the simulation results, it can be seen that the proposed CRFNN model is capable of extracting the dynamics of the system.

3.5.3 Random sine wave noise injection test [Example-1]

Again to validate the efficiency of the proposed identifier, a random sine wave $dis(k) = sin(\frac{2\pi k}{15})$ was added to the network as a disturbance signal between the time interval $250 < k < 450$. Initially, the network fluctuated during this interval and deviated from the plant response. With continuous training, the network managed to track back the plant response in a very short time. The proposed network is found efficient in learning the input-output patterns and predicting the similar unknown patterns supplied to them. The response of the proposed CRFNN under the effect of random noise is shown in Figure 3.5. Table 3.1 gives the comparison of proposed CRFNN with ENN, JNN, FFNN, and LRNN models. From the table, the proposed structure is found to have better performance indices.

3.5.4 Example-2: A nonlinear plant with order 3

The proposed structure is further tested for identification ability by applying another nonlinear plant equation of order 3. The plant equation is given below as in [36]:

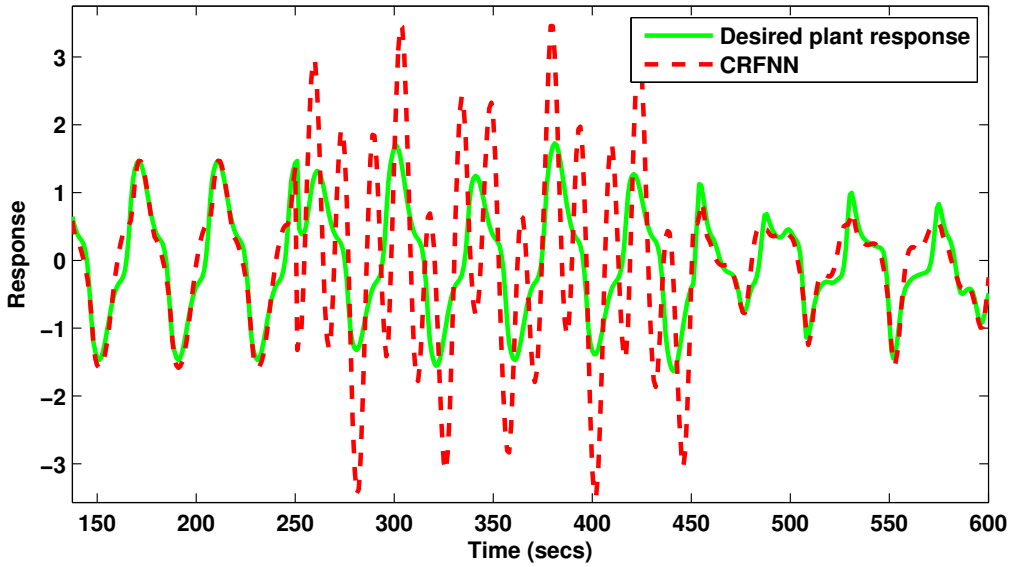


Figure 3.5: Effect of random sine noise [Example-1]

$$y_p(k) = 0.72y_p(k-1) + 0.025y_p(k-2)r(k-1) + 0.001r^2(k-2) + 0.2r(k-3) \tag{3.33}$$

The proposed structure is applied with an external input $r(k) = sin(\frac{2\pi k}{100})$. The plant equation takes the identification structure as below:

$$y_p(k) = g[y_p(k-1), y_p(k-2), r(k-1), r(k-2), r(k-3)] \tag{3.34}$$

Where g is the nonlinear function. When CRFNN, LRNN, FFNN, ENN, and JNN are selected as identifiers, they take the following identification model as below:

$$y_{cfrnn}(k) = \hat{g}_1[y_p(k-2), r(k)] \tag{3.35}$$

$$y_{lrnn}(k) = \hat{g}_2[y_p(k-2), r(k-1), r(k)] \tag{3.36}$$

$$y_{ffnn}(k) = \hat{g}_3[y_p(k-2), y_p(k-1), r(k-2), r(k-3)] \tag{3.37}$$

$$y_{ENN}(k) = \hat{g}_4[y_p(k-2), r(k-1), r(k)] \tag{3.38}$$

$$y_{JNN}(k) = \hat{g}_5[y_p(k-2), r(k-1), r(k)] \tag{3.39}$$

where $\hat{g}_1, \hat{g}_2, \hat{g}_3, \hat{g}_4, \hat{g}_5$ are the nonlinear functions to be identified. The models are tested by supplying about 900 samples in batch mode of identification. From the results, it can be seen that the proposed method shows the best efficiency compared to other selected neural models.

3.5.5 Parameter variation in testing phase [Example-2]

During validation, a new input $r(k)$ is supplied to the network as below:

$$r(k) = \begin{cases} \frac{sin(\pi k)}{40}, & \text{for } 0 < k <= 250 \\ \frac{0.09sin(\pi k)}{45} - & \\ \frac{cos(2\pi k)}{40}, & \text{for } 250 < k <= 450 \\ \frac{0.3sin(2\pi k)}{15} + & \\ \frac{0.1sin(2\pi k)}{320} + & \\ \frac{0.6sin(2\pi k}{40}, & \text{for } 450 < k <= 900 \end{cases} \tag{3.40}$$

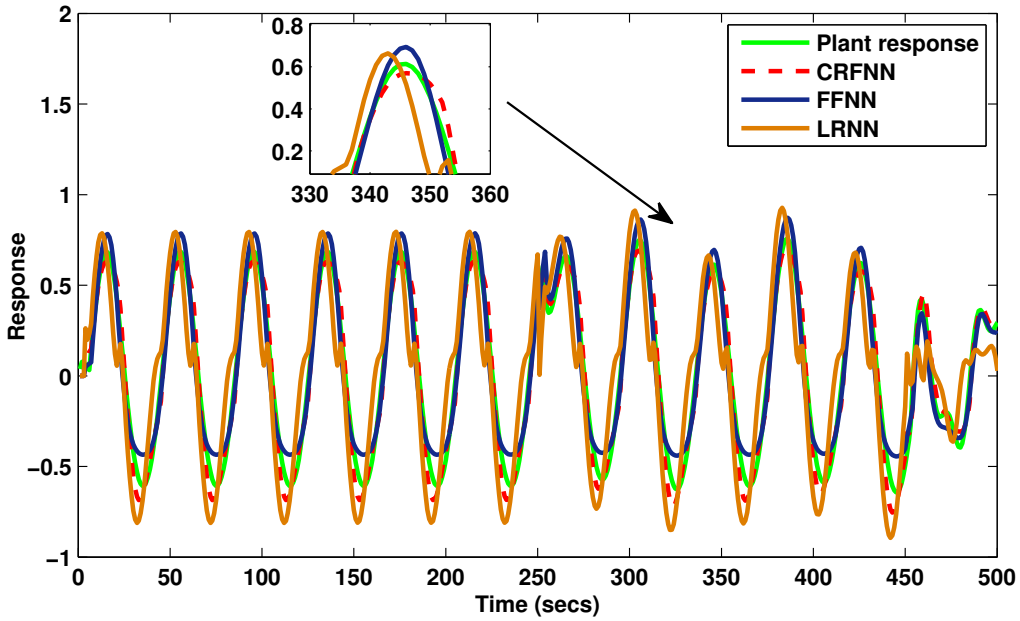


Figure 3.6: Comparison of identifier response obtained for selected identifiers [Example-2]

Figure 3.6 shows the response obtained from all the selected neural structures. The error is found to decrease with time to a very minimum value. Figure 3.7 shows the MSE curve obtained from selected neural structures. Figure 3.8 shows the MAE curve obtained from selected neural structures. From above simulation results, it can be concluded that the proposed CRFNN model identifies better dynamics of the system.

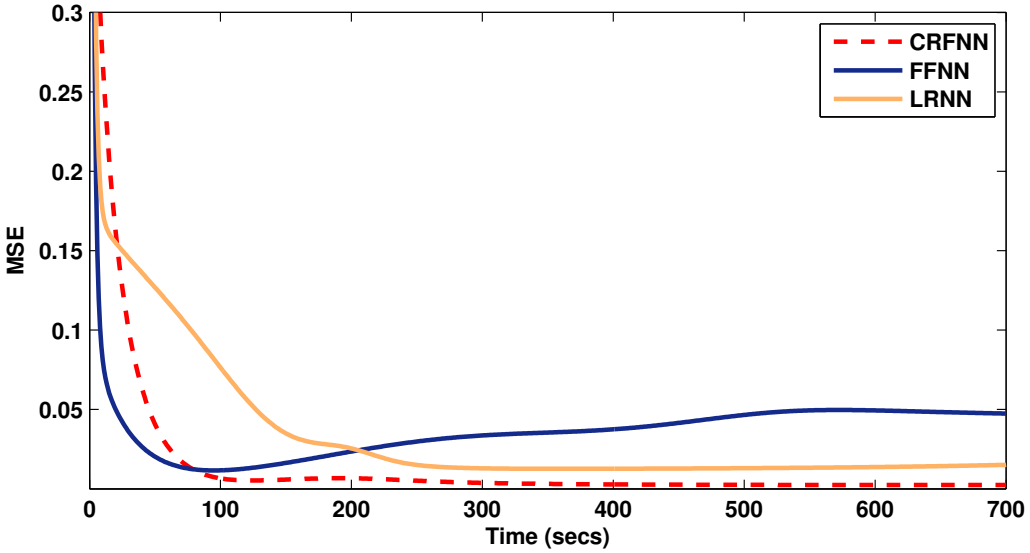


Figure 3.7: Comparison of MSE curves obtained for selected identifiers [Example-2]

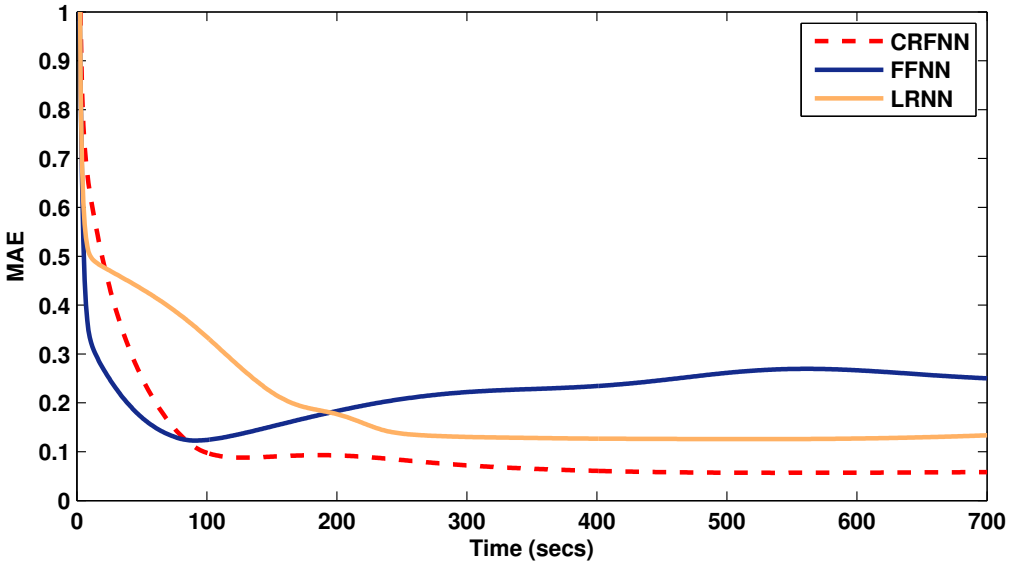


Figure 3.8: Comparison of MAE curves obtained for selected identifiers [Example-2]

3.5.6 Random sine wave noise injection test [Example-2]

Again a random sine wave $dis(k) = sin(\frac{2\pi k}{15})$ as disturbance is supplied to the network between $250 < k < 450$ time interval. The structure was initially found to fluctuate and deviate from the plant's desired trajectory, yet managed to track the desired plant trajectory after more training. Figure 3.9 shows the effect of random noise on the network. Table 3.2 gives the comparison of proposed CRFNN with other selected identifiers. The proposed model is found to give better performance indices over ENN, JNN, LRNN, and FFNN.

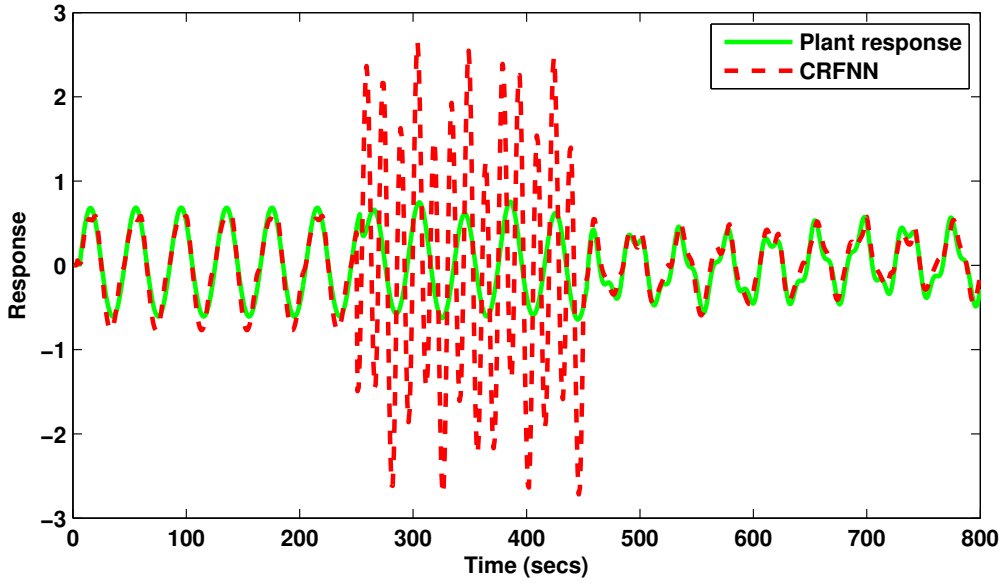


Figure 3.9: Effect of random sine noise [Example-2]

Table 3.1: Performance comparison of CRFNN with other selected identifiers [Example-1]

S.no	Parameters	CRFNN	FFNN	LRNN	ENN	JNN
1	No of input parameters	02	04	03	03	03
2	No of hidden neurons	04	06	04	04	04
3	No of tunable weights (excluding bias weights)	10	4	6	7	7
4	Computation time	30 sec	76 sec	50 sec	32 sec	26 sec
5	AMSE	**0.0139**	0.0336	0.0366	0.0230	0.0389
6	AMAE	**0.0940**	0.1994	0.1867	0.1831	0.1641
7	RMSE	**0.0107**	0.2930	0.1203	0.0915	0.0239

3.5.7 Example-3: Mackey-Glass time series identification

Further, the proposed identifier is tested on the benchmark nonlinear times-series prediction problem. The well-known Mackey-Glass identification problem differential equation is given

Table 3.2: Performance comparison of CRFNN with other selected identifiers [Example-2]

S.no	Parameters	CRFNN	FFNN	LRNN	ENN	JNN
1	No of input parameters	02	04	03	03	03
2	No of hidden neurons	04	06	04	04	04
3	No of tunable weights (excluding bias weights)	10	4	6	7	7
4	Computation time	41 sec	67 sec	43 sec	46 sec	37 sec
5	AMSE	**0.0137**	0.0459	0.0324	0.0174	0.0297
6	AMAE	**0.1096**	0.2045	0.1968	0.1268	0.1727
7	RMSE	**0.0446**	0.0851	0.1221	0.999	0.1112

below [30]:

$$\frac{dy_p(t)}{dt} = -\beta \times y_p(t) + \frac{\alpha \times y_p(t-\tau)}{1 + y_p^{10}(t-\tau)} \tag{3.41}$$

The series is applied with parameter values such as $\alpha = 0.2$ and $\beta = 0.1$. Symbol t denotes the time-series sequence of the prediction. When $\tau \geq$17, the time-series prediction is found to have a chaotic behavior. Hence the value of the sampling rate is selected as, τ=17. The equivalent difference equation is given below:

$$y_p(k) = -\beta \times y_p(k) + \frac{\alpha \times y_p(k-\tau)}{1 + y_p^{10}(k-\tau)} \tag{3.42}$$

Out of the 900 samples considered, 500 values were taken for training, and the remaining 400 values were taken for validation. The proposed identifier takes the series-parallel model form as $y_{crfnn}(k) = [y_p(k), y_p(k-17)]$. The problem is applied to all the selected neural structures. Figure 3.10 shows the response obtained from all the selected identifier models. Figure 3.11 shows the MSE curve obtained from all the selected identifier models. Figure 3.12 shows the MAE curves obtained from all the selected identifier models.

3.5.8 Random sine wave noise injection test [Example-3]

The problem was also introduced to a sudden disturbance of sine wave, $dis(k) = sin(\frac{2\pi k}{15})$ between the time interval $250 < k < 450$ to test its recovering ability. Figure 3.13 shows the effect of random noise on the network. The network is found to work superior and more efficiently compared to other selected neural structures. The same can be concluded from the results of Table 3.3. The overall performance parameters are also achieved minimum for the proposed CRFNN structure.

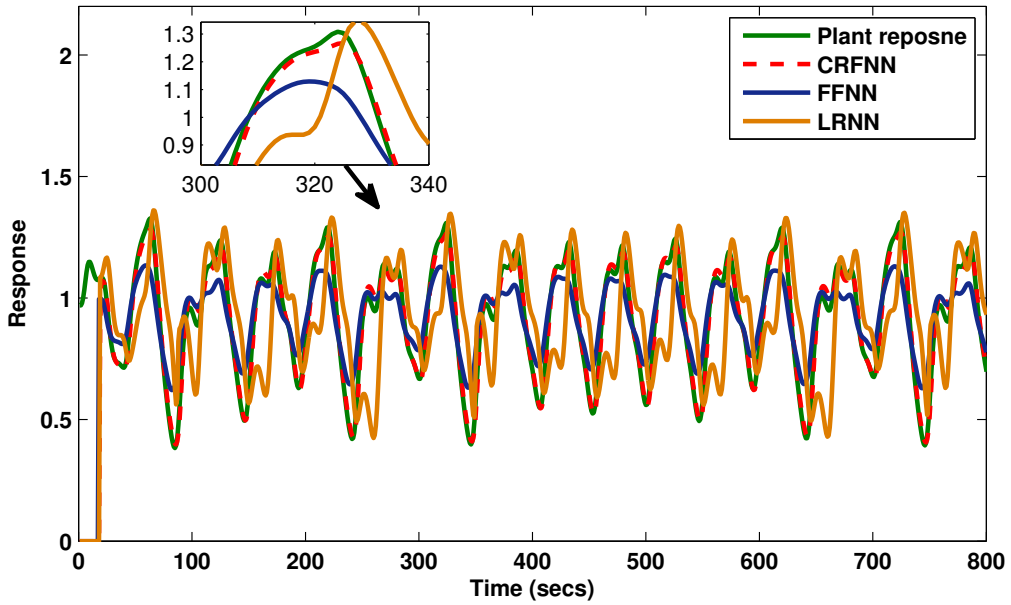


Figure 3.10: Comparison of identifier response at the end of training [Example-3]

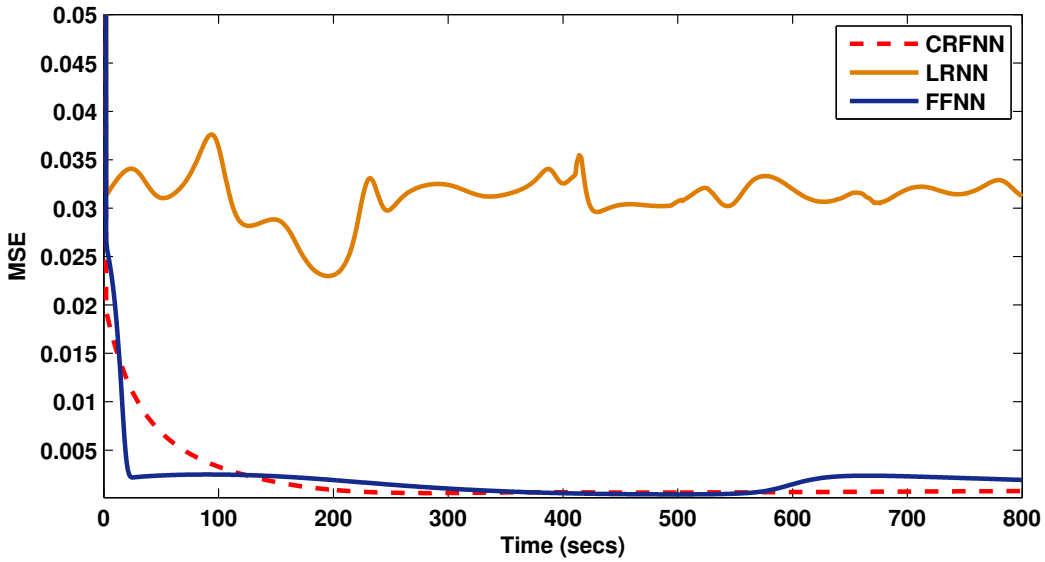


Figure 3.11: Comparison of MSE curves obtained for selected identifiers [Example-3]

Table 3.3: Performance comparison of CRFNN with other selected identifiers [Example-3]

S.no	Parameters	CRFNN	FFNN	LRNN	ENN	JNN
1	No of input parameters	02	04	03	03	03
2	No of hidden neurons	04	06	04	04	04
3	No of tunable weights (excluding bias weights)	10	4	6	7	7
4	Computation time	47 sec	67 sec	43 sec	49 sec	41 sec
5	AMSE	**0.0065**	0.0324	0.0281	0.0072	0.0156
6	AMAE	**0.0755**	0.0852	0.1271	0.0852	0.1271
7	RMSE	**0.0525**	0.1778	0.1821	0.0653	0.0396

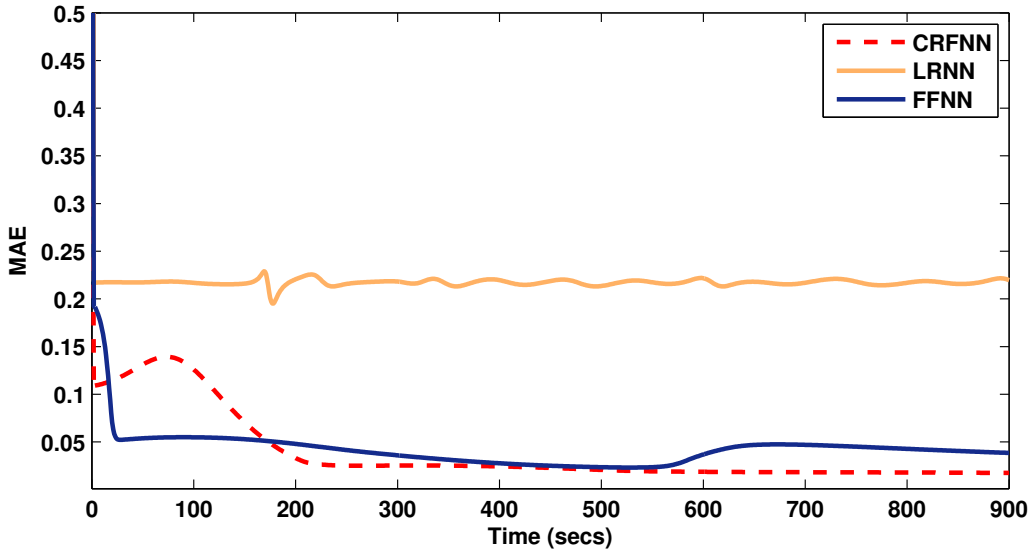


Figure 3.12: Comparison of MAE curves obtained for selected identifiers [Example-3]

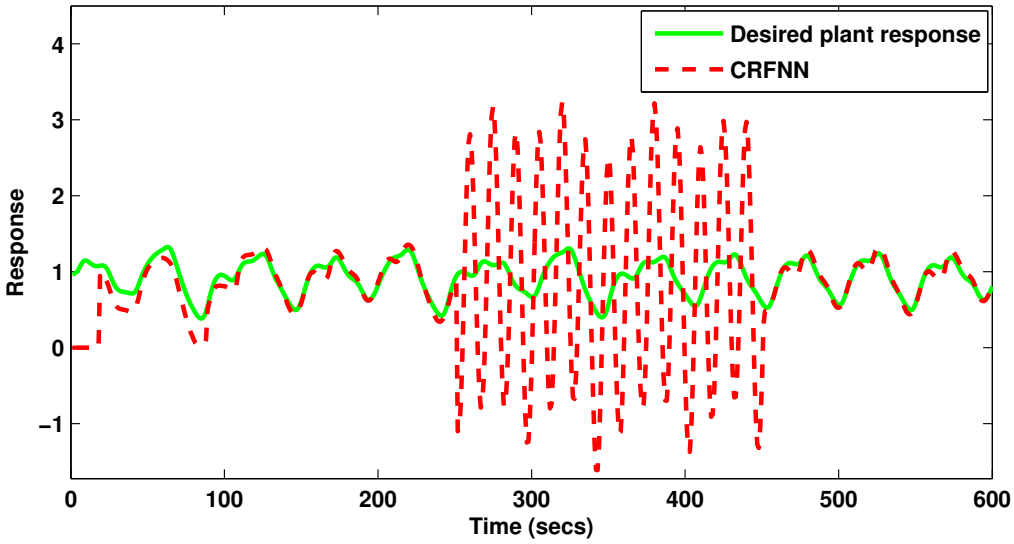


Figure 3.13: Effect of random sine noise [Example-3]

3.6 Conclusion

In this paper, CRFFNN is proposed for the identification of complex nonlinear dynamical systems. The proposed structure is the hybridization of the LRNN and a single-layer FFNN model that are combined to develop the proposed model. With the help of three benchmark non-linear problems, the structure's performance is evaluated and compared with other well-known neural models. The AMSE and AMAE values obtained are far less than other selected neural structures considered in this work. The structure also required only a lesser number of inputs and tunable parameters as compared to other structures. This is also an indication that the proposed structure is independent of the order of the plant. From the results of a random noise injection test, it is evident that the proposed structure can recover quickly and adjust itself to the changing dynamics of the system. These characteristics make the structure an efficient one for nonlinear identification. The structure can also be expanded to develop an adaptive control of nonlinear dynamic systems. Furthermore, it is found that the structure is resistant to perturbations and parameter modifications.

Chapter 4

Simultaneous adaptive control and modeling based on hybrid compound recurrent feed-forward neural network: Simulation and stability analysis

4.1 Introduction

In the previous chapter, we explored the efficiency of the proposed CRFNN for the identification of nonlinear dynamic systems. Through the simulation study, we found that CRFNN captures the temporal dynamics of a nonlinear system with exceptional accuracy. They are found to have numerous advantages, including a simpler structure, an excellent adaptive ability to handle input noise, and independence of the system's order. The present work focuses on the development of simultaneous intelligent control and modeling strategies utilizing the CRFNN architecture. Nonlinear control theory has gained popularity in recent times for designing nonlinear control systems. While linear control theory approaches, such as PID controllers, state space control approaches, have been used to identify and handle non-linear dynamic systems since the late 1930s, they have many limitations [113], [114]. This resulted in the development of nonlinear control theory methods including sliding mode control (SMC) [115], robust control [116], adaptive control [117], and fuzzy-based control [118]. Adaptive controllers, among these, have received a lot of interest from researchers. Unlike other control approaches, adaptive control generates control inputs by taking the system's slowly varying dynamics into

account and driving it to follow the desired control law [41], [42]. As a result, even in the presence of disturbance, the controller performance is faster and better. Among the types of adaptive controller, indirect adaptive control scheme is widely used due to their adaptability to their changing environment. For designing an effective controller using the indirect adaptive control scheme, the knowledge regarding the plant must be known. When the knowledge regarding the plant is unknown, identification is required. This has motivated us to develop an effective control scheme for identification-based adaptive control of nonlinear dynamic systems. Both the identifier and controller use the hybrid CRFNN model. The hybrid neuro architecture takes advantage of both the feed-forward and recurrent structure (fast processing capacity and the presence of memory neurons). The structure is trained online, therefore the structure does not require prior information about the plant. The gradient-descent-based BP is applied to dual up the weight equations and the convergence is proved using the Lyapunov-based stability principles. The efficiency of the based controller is tested on two benchmark plant equations of varying degrees and is also analyzed for their disturbance ability over selected neural-based controllers such as LRNC and JNC.

4.2 HFRNN based indirect adaptive control scheme for nonlinear dynamical systems

Figure 4.1 shows the block diagram of the proposed indirect adaptive control scheme. The proposed controller scheme consists of two major blocks: HFRNN-based identifier and HFRNN-based controller. The controller scheme is trained online in real time by simultaneously identifying the plant model through an identifier. The HFRNN-based identifier is used to identify the unknown plant and the HFRNN-based controller to produce the control input $u_c(k)$. Both the parameters are trained using the backpropagation algorithm. The HFRNN-based controller takes the external input $r(k)$, previous control input $u_c(k-1)$, and output of the plant $y_p(k-1)$ as inputs. The weights of the controller are updated by detecting the error between the plant and the reference model and are updated continuously until the control error is minimized. The plant remains generally unknown, hence the information about the plant is crucial for properly tuning the controller. The HFRNN-based identifier will estimate the plant sensitivity for the controller. The HFRNN-based identifier takes the present control input $u_c(k)$ and previous plant output $y_p(k-1)$ as inputs. The identifier is trained using the error generated

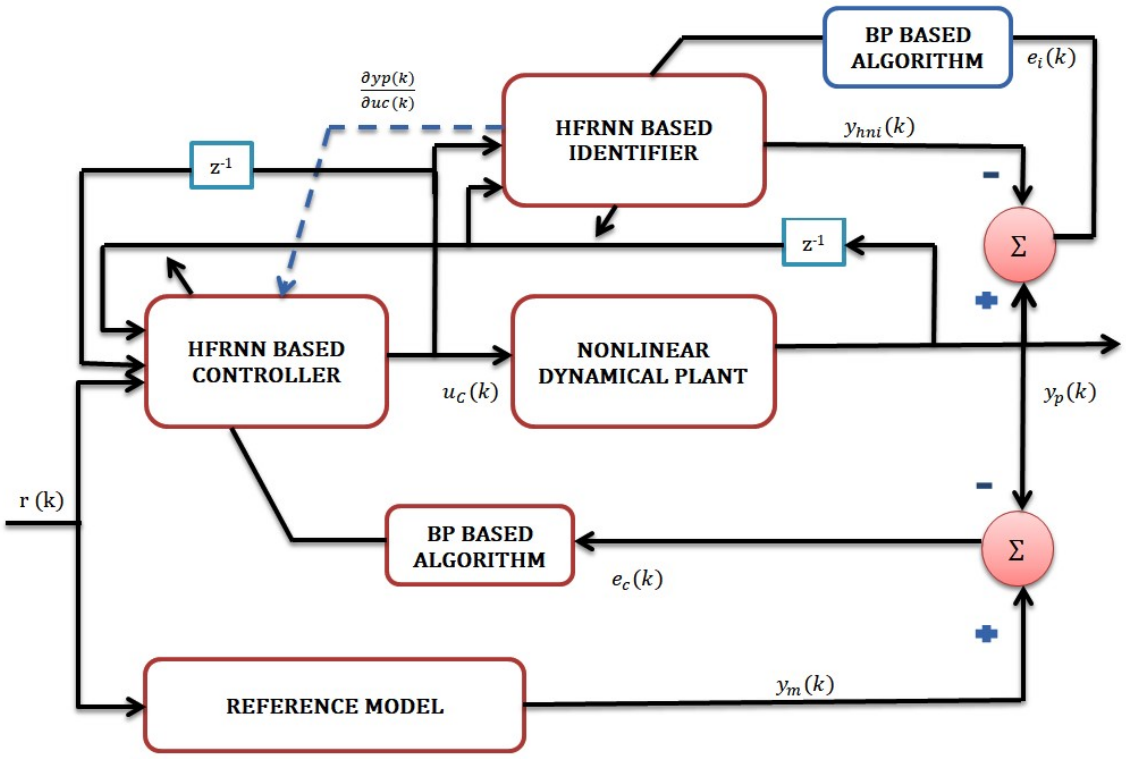


Figure 4.1: Block diagram of the proposed HFRNN scheme

between the output of the plant and the neural identifier and is updated continuously until the identification error is minimized. The training continues until the control error is minimized and the plant tracks the reference model.

4.3 Problem statement

Let's consider a plant $y_p(k)$ represented by the following difference equation as follows:

$$y_p(k+1) = f[y_p(k), y_p(k-1), .., y_p(k-n), u_c(k), u_c(k-1), .., u_c(k-m)] \tag{4.1}$$

where $y_p(k+1)$ denotes the one-step ahead output of plant model, n refers to the order of the plant and m refers to the order of the controller input $u_c(k)$, f refers to the unknown differentiable nonlinear function. Let's consider a reference model with the following difference equation :

$$y_m(k+1) = g[y_m(k), y_m(k-1), .., y_m(k-n), r(k), r(k-1), .., r(k-m)] \tag{4.2}$$

Where $y_m(k+1)$ refers to the one-step ahead output of the reference model, n denotes the order of the reference model and m denotes the order of external input $r(k)$, g denotes the unknown nonlinear function that is assumed differentiable. The main objective of our work is as follows: 1. To identify the unknown plant model by minimizing the error between the plant and the identified neural model. This is done by updating the identifier weight equations based on the error between the plant and the identified neural model such that

$$\lim_{k\to\infty} |y_p(k+1) - y_{hni}(k+1)| \leq \epsilon \tag{4.3}$$

where $\epsilon \to 0$. 2. To make the plant follow the reference model by determining the control input $u_c(k)$. This is achieved by continuously training the ANN-based controller update equations to generate a control input $u_c(k)$ such that

$$\lim_{k\to\infty} |y_m(k+1) - y_p(k+1)| \leq \epsilon \tag{4.4}$$

where $\epsilon \to 0$. The error between the plant and the reference model updates the controller equations.

4.4 Lyapunov stability analysis

The convergence of the proposed structure is studied using the Lyapunov notion of stability. According to the Lyapunov stability criteria, if there is any energy measure in the system, then the rate of change of error derives the stability of the system. This study involves declaring the weight update equations of the structure and checking whether stability is achieved or not. The system is found to have achieved stability when the Lyapunov-based error function is minimum and positive. The goal is to minimize the Lyapunov function $V(W)$:

$$J = min(V(W)) \tag{4.5}$$

$$V(W) > 0 \quad \text{for} \quad W > 0 \qquad V(W) = 0 \quad \text{for} \quad W = 0 \qquad \Delta V(k) \leq 0 \tag{4.6}$$

where V(W) is the Lyapunov function and W denotes the weight of the network. For the discrete-time, the change in the Lyapunov function is:

$$\Delta V(k) = V(k) - V(k-1) \tag{4.7}$$

To update weights using the stochastic gradient descent formula:

$$W_i(k+1) = W_i(k) - \eta \frac{\partial E(k)}{\partial W_i(k)} \tag{4.8}$$

where η is the learning rate and $E(k)$ is the error function. Assuming that the Lyapunov function $V(k)$ measures the error at each iteration, we can express it as:

$$V(k) = E(k) \tag{4.9}$$

The change in the Lyapunov function becomes:

$$\Delta V(k) = E(k) - E(k-1) \tag{4.10}$$

Substituting the weight update :

$$E(k) = E(k-1) - \eta \frac{\partial E(k-1)}{\partial W_i(k-1)} \tag{4.11}$$

Now, substituting this back into the expression for $\Delta V(k)$:

$$\Delta V(k) = E(k-1) - \eta \frac{\partial E(k-1)}{\partial W_i(k-1)} - E(k) \tag{4.12}$$

$$\Delta V(k) = -\eta \frac{\partial E(k-1)}{\partial W_i(k-1)} \tag{4.13}$$

Conditions for stability: For the system to be stable, $\Delta V(k) < 0$:

$$-\eta \frac{\partial E(k-1)}{\partial W_i(k-1)} < 0 \tag{4.14}$$

This condition is met when η is positive and small, and $\frac{\partial E(k-1)}{\partial W_i(k-1)} > 0$. This indicates that the Lyapunov function decreases as the error E(k) decreases, yet the system stays stable.

4.5 Learning algorithm

The gradient descent-based BP algorithm is used for updating the controller and identifier weights. The main objective is to minimize the control error $E_c(k)$. It is the instantaneous error calculated between the reference model output and the plant output. $E_c(k)$ is given by:

$$E_c(k) = \frac{1}{2}[e_c(k)]^2 \tag{4.15}$$

where, $e_c(k) = y_m(k) - y_p(k)$. Here, $y_m(k)$ and $y_p(k)$ denote the reference model output and the plant output respectively. To update the controller weights in the output layer, the gradient chain rule is applied by calculating the change of gradient of $E_c(k)$ with respect to recurrent output weight $W_{orc}(k)$ as below:

$$\frac{\partial E_c(k)}{\partial W_{orc}(k)} = \frac{\partial E_c(k)}{\partial y_p(k)} \times \frac{\partial y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W_{orc}(k)} \tag{4.16}$$

On simplification:

$$\frac{\partial E_c(k)}{\partial W_{orc}(k)} = -e(k) \times R_j(k) \times J(k) \tag{4.17}$$

Further to update the weight, the SGD formula is applied as:

$$W_{orc}(k+1) = W_{orc}(k) + \eta\left(- \frac{\partial E_c(k)}{\partial W_{orc}(k)}\right) \tag{4.18}$$

Similarly to update the controller's output weight $W_{ofc}(k)$, the gradient change of $E_c(k)$ with respect to $W_{ofc}(k)$ is calculated as follows:

$$\frac{\partial E_c(k)}{\partial W_{ofc}(k)} = \frac{\partial E_c(k)}{\partial y_p(k)} \times \frac{\partial y_p(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W_{ofc}(k)} \tag{4.19}$$

where $\frac{\partial E_c(k)}{\partial y_p(k)} = -e(k)$, $\frac{\partial y_p(k)}{\partial u_c(k)} = F_j(k)$ and $\frac{\partial u_c(k)}{\partial W_{ofc}(k)} = J(k)$, so above equation can be written as follows:

$$\frac{\partial E_c(k)}{\partial W_{ofc}(k)} = -e(k) \times F_j(k) \times J(k) \tag{4.20}$$

Again, the weight update equation of $W_{ofc}(k)$ is given by :

$$W_{ofc}(k+1) = W_{ofc}(k) + \eta\left(- \frac{\partial E_c(k)}{\partial W_{ofc}(k)}\right) \tag{4.21}$$

Similarly, the hidden layer local weights of the controller $W_{hc}(k)$ is updated as follows:

$$\frac{\partial E_c(k)}{\partial W_{hc}(k)} = \frac{\partial E_c(k)}{\partial y_p(k)} \times \frac{\partial y_p(k)}{\partial y_{rc}(k)} \times \frac{\partial y_{rc}(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W_{hc}(k)} \tag{4.22}$$

On simplification:

$$\frac{\partial E_c(k)}{\partial W_{hc}(k)} = -e(k) \times (I - R_j^2(k)) \times W_{orc}(k) \times T(k) \tag{4.23}$$

where $T(k)$ denotes the delayed states of the self-recurrent hidden nodes in the LRNN network. Every calculation of the controller requires knowledge of the changing dynamics of the plant, and $J(k)$ computes the sensitivity of the controller. If the plant is completely unknown, it becomes very necessary to design an online identifier in parallel to the controller to automatically adapt to the changing dynamics. The proposed structure is trained online, hence requiring the weights and errors of the identifier to be updated simultaneously with the controller parameters. $E_i(k)$ is the instantaneous error calculated between the plant, $y_p(k)$ and the neural identifier, $y_{hni}(k)$. The instantaneous error is calculated by:

$$E_i(k) = \frac{1}{2}[e_i(k)]^2 \tag{4.24}$$

where, $e_i(k) = y_p(k) - y_{hni}(k)$. As the structure is trained online, the output of the controller $u_c(k)$ becomes one of the inputs of the identifier. This results in calculation of $\frac{\partial y_p(k)}{\partial u_c(k)}$. This indicates the Jacobian matrix, J(k). This computation updates the FFNN identifier weights $W_{ifc}(k)$ with respect to $E_c(k)$. For this, the gradient of $E_c(k)$ is calculated with respect to $W_{ifc}(k)$

$$\frac{\partial E_c(k)}{\partial W_{ifc}(k)} = \frac{\partial E_c(k)}{\partial y_p(k)} \times \frac{\partial y_p(k)}{\partial y_{fc}(k)} \times \frac{\partial y_{fc}(k)}{\partial u_c(k)} \times \frac{\partial u_c(k)}{\partial W_{ifc}(k)} \tag{4.25}$$

On substituting the values:

$$\frac{\partial E_c(k)}{\partial W_{ifc}(k)} = -e(k) \times w_{ofc}(k) \times (I - F_j^2(k)) \times X(k) \tag{4.26}$$

Further, the weights $w_{ifc}(k)$ are updated using the weight update rule applied as below:

$$w_{ifc}(k+1) = w_{ifc}(k) + \eta\left(- \frac{\partial_c(k)}{\partial w_{ifc}(k)} \right) \tag{4.27}$$

Here, η denotes the static learning rate and is taken between 0 and 1. As the training progresses, the neural network starts following the plant model i.e $\frac{\partial y_{hni}(k)}{\partial u_c(k)} \approx \frac{\partial y_p(k)}{\partial u_c(k)}$. Once this is achieved, the identified model can accurately represent the changing dynamics of the plant. With the unknown plant identified, the HFRNN-based controller starts to learn the dynamics and hence the plant model starts to follow the reference model i.e. $\frac{\partial y_p(k)}{\partial u_c(k)} \approx \frac{\partial y_m(k)}{\partial u_c(k)}$. Figure 4.2 illustrates the various iterative steps followed in training the controller scheme.

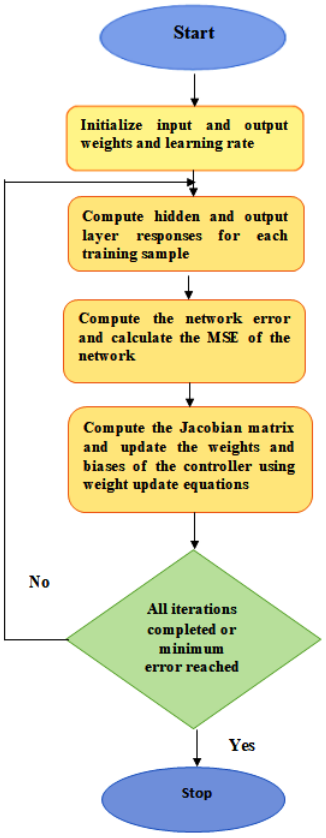


Figure 4.2: Steps followed in training HFRNN-based controller structure

4.6 Simulation examples

To evaluate the efficiency of the proposed HFRNN-based controller, the controller is compared with two other neural-based controllers such as the Jordan-based neural network controller (JNC), and the local recurrent-based neural network controller (LRNC) on two different non-linear plant equations of varying degrees. All the controllers are designed with a single hidden layer with 20 neurons and a static learning rate of 0.001. The initial weights and biases are taken randomly in the interval (0,1). The controller is trained online for about 15000 seconds.

4.6.1 Example 1: A nonlinear dynamic plant equation

In this example, a plant with the following difference equation is considered as in [45].

$$y_p(k+1) = \frac{5y_p(k)y_p(k-1)}{(1+y_p(k)^2+y_p(k-1)^2+y_p(k-2)^2)} + u_c(k) + 0.8u_c(k-1) + v(k) \tag{4.28}$$

The nonlinear plant takes the identification structure as below:

$$y_p(k+1) = f[y_p(k), y_p(k-1), y_p(k-2), u_c(k), u_c(k-1)] \tag{4.29}$$

Consider a reference model with the following difference equation:

$$y_m(k+1) = 0.72y_m(k) + 0.64y_m(k-1) - 0.5y_m(k-2) + r(k) \tag{4.30}$$

where f is the nonlinear function, $r(k)$ denotes the external reference input, $u_c(k)$ is the control input generated by the controller and $v(k)$ denotes the disturbance signal. The system is applied with an external reference input $r(k) = sin(\frac{2\pi k}{25})$. The disturbance applied to the system initially is $v(k) = 0$. The main objective is to obtain the control input $u_c(k)$ and make the plant follow the reference model. The inputs of the controller are $[y_p(k), y_p(k-1), y_p(k-2), r(k), u_c(k-1), u_c(k-2)]$ and that of identifier are $[u_c(k), y_p(k-1), y_p(k)]$. Figure 4.3 shows the open-loop response of the plant. It can be seen that, without controller action, the plant does not track the reference model. Figure 4.4 and Figure 4.5 show the response of the plant during the initial and final stages of training. It is observed initially, that the plant does not follow the reference model. But, as training progresses, the plant starts to follow the reference model and after 15000 epochs, the error minimizes and converges to a minimum value. The instantaneous MSE obtained is also shown in Figure 4.6. It is evident from Figure 4.6 and Table 4.1 that the proposed controller gives good control performance even with a minimum number of parameters. On the other hand, both JNC and LRNC take time to converge to a minimum MSE.

Table 4.1: Comparison of controller parameters [Example-1]

S.No	Structure	Instantaneous MSE	Simulation time (secs)
1	HFRNN based controller	**0.0010**	1.39
2	JNC	0.0244	1.66
3	LRNC	0.0095	2.35

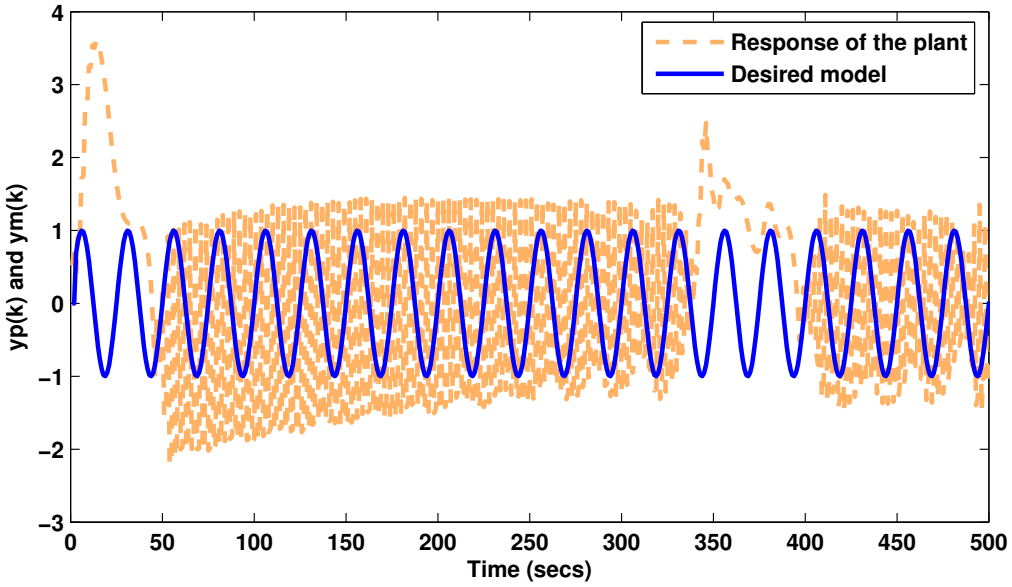


Figure 4.3: Plant's response without controller action [Example-1]

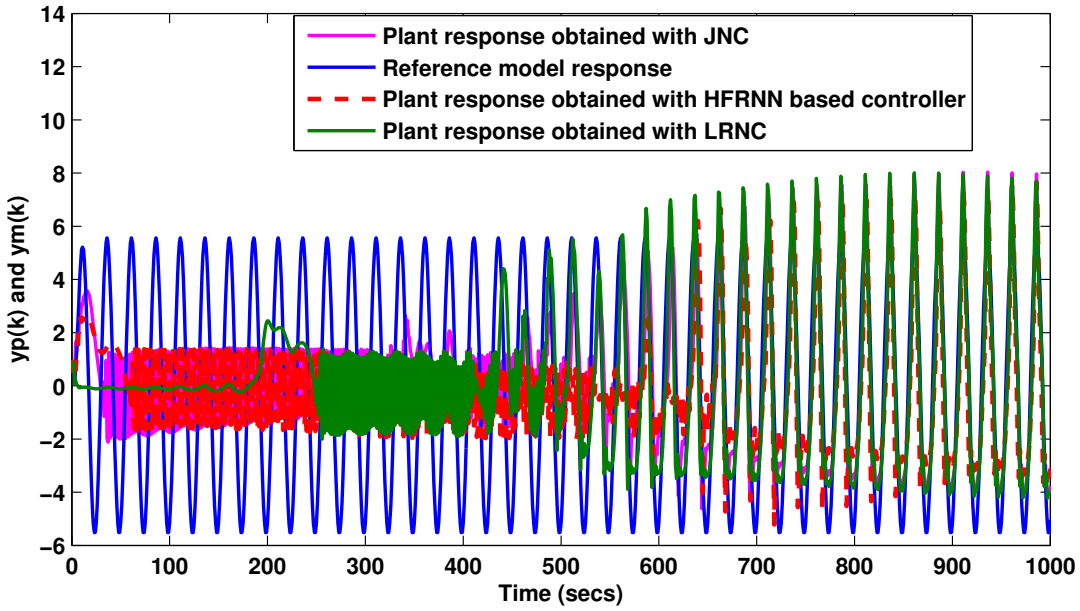


Figure 4.4: Response obtained from the plant during the initial training phase [Example-1]

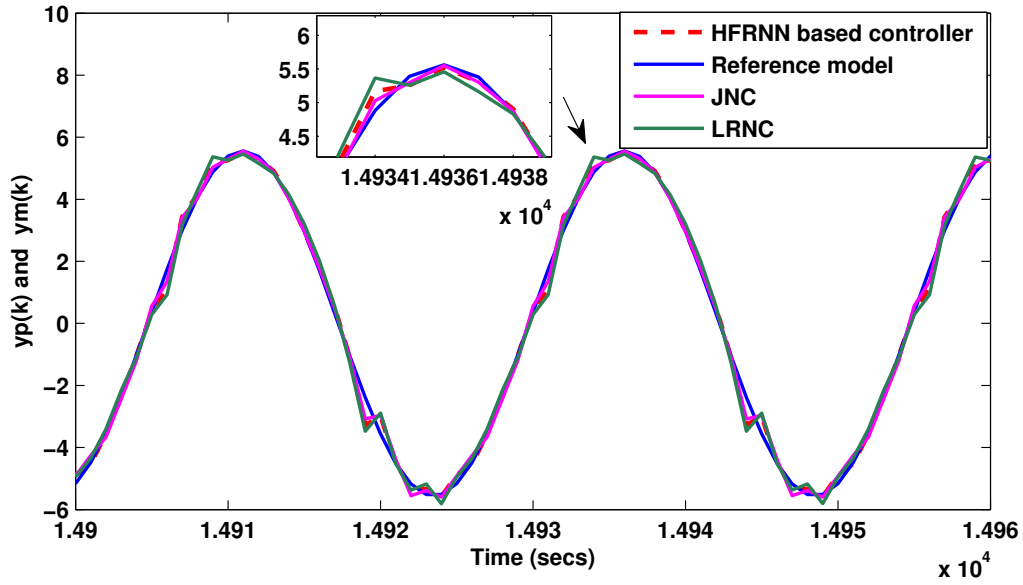


Figure 4.5: Response of controllers during the final phase of training [Example-1]

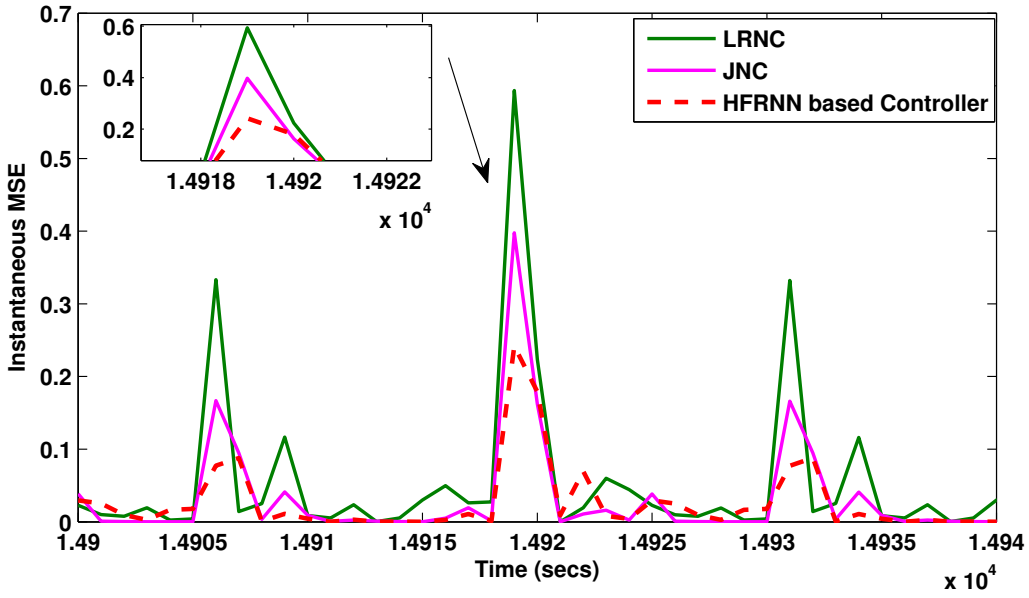


Figure 4.6: Comparison of Instantaneous MSE curves [Example-1]

4.6.2 Disturbance testing

To validate the recovering ability of the proposed controller, a sudden disturbance signal $dis(k) = sin(\frac{2\pi k}{40})$ is introduced between the time interval $9000 < k \leq 11000$. A static learning rate of 0.001 is considered, and the effect of external noise on the HFRNN-based controller, JNC, and LRNC was analyzed. Initially, the system deviated from the reference model due to the disturbance signal. But with training, the control parameters were adjusted to the changing environment and the system recovered. However, it took a little longer time. The instantaneous MSE during the disturbance signal of an HFRNN-based controller is found to be very small as compared to JNC and LRNC. The MSE obtained for the based controller was around 0.0653. Refer to Figure 4.7 for the comparison of the effect of disturbance signal over HFRNN-based controller, JNC, and LRNC.

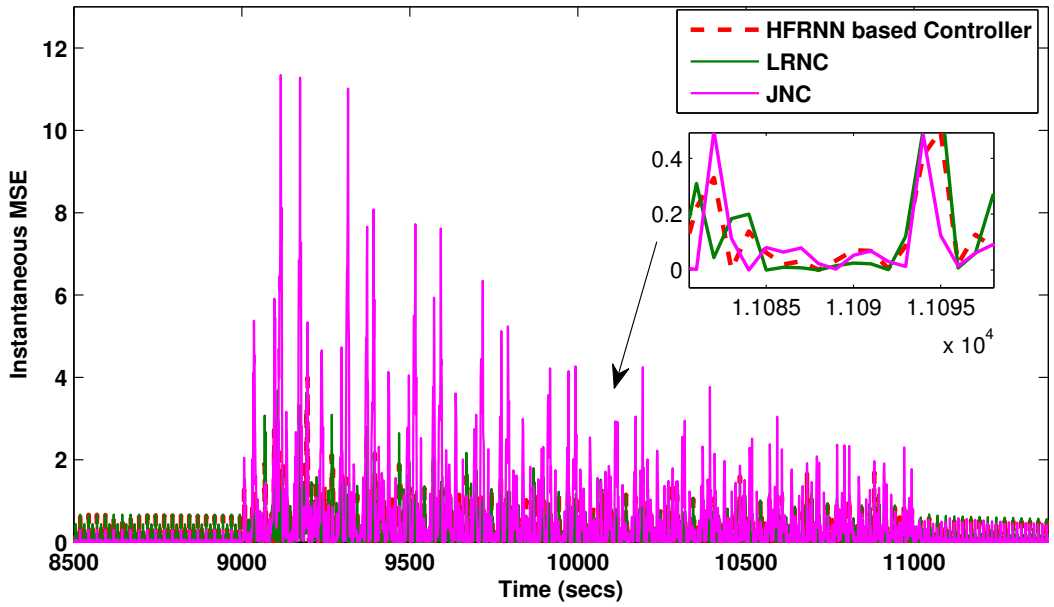


Figure 4.7: Comparison of the effect of external noise on controllers [Example-1]

4.6.3 Recovering ability of the HFRNN based controller

The recovering ability of the proposed controller was also tested by changing the reference input. For this, a new reference input $r(k)$ was considered as below between the interval $5000 < k \leq 15000$:

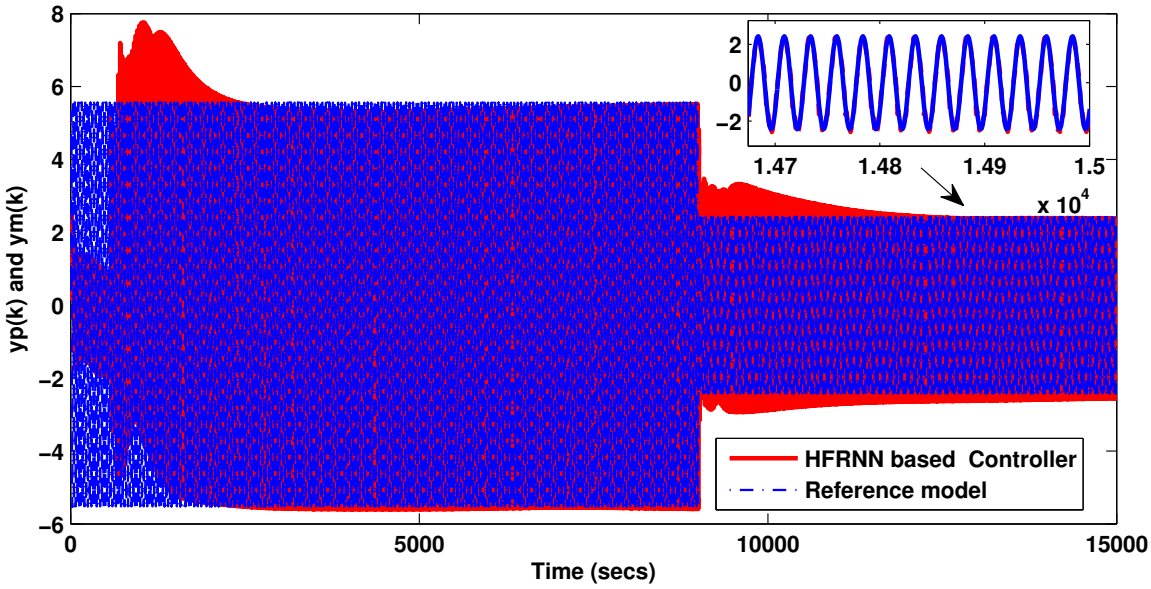


Figure 4.8: Recovering ability of HFRNN based controller to change in reference input [Example-1]

$$r(k) = \begin{cases} \sin\left(\frac{2\pi k}{25}\right), & \text{for } 0 < k \leq 5000 \\ \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right), & \text{for } 5000 < k \leq 15000 \end{cases} \tag{4.31}$$

When the reference input was changed after 5000 training epochs, the network was able to capture and adapt to the new reference input. The controller was found to show good recovering ability. Figure 4.8 shows the recovering ability of the based controller to the new reference input.

Table 4.2: Comparison of performance of controller parameters [Example-2]

S.No	Structure	Instantaneous MSE	Simulation time (secs)
1	HFRNN based controller	$\mathbf{3.160 \times 10^{-4}}$	1.52
2	JNC	0.0012	1.43
3	LRNC	0.0111	2.43

Testing of models with Square wave as a reference signal

The controller was further supplied with a square waveform as a reference signal to test the recovery ability. The network was found to capture and adapt to a new reference input. For this, the new reference input $r(k) = square(\frac{\pi k}{40})$ was considered. The controller showed good adaptability, with the plant slowly following the desired model. During the end of the training, the plant response resembled the desired model. The instantaneous MSE of the proposed

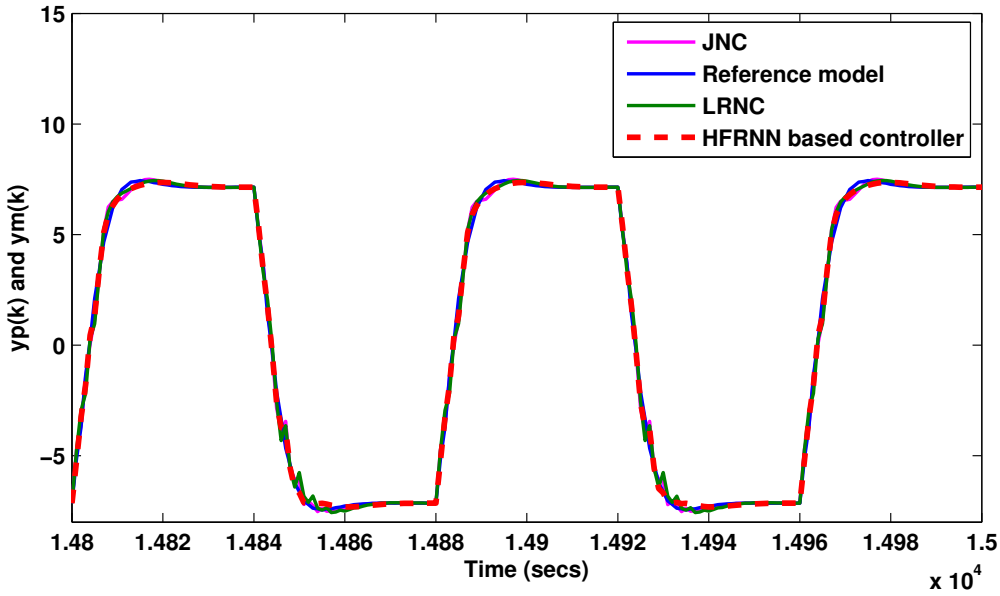


Figure 4.9: Response of controllers with a square waveform as a reference signal [Example-1]

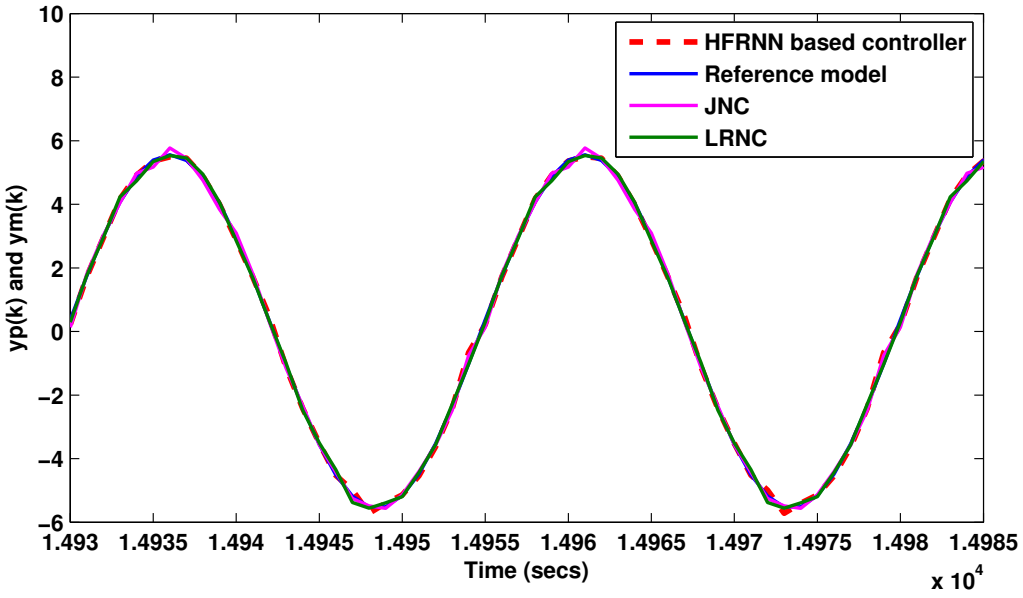


Figure 4.10: Response of controllers for different initial conditions [Example-1]

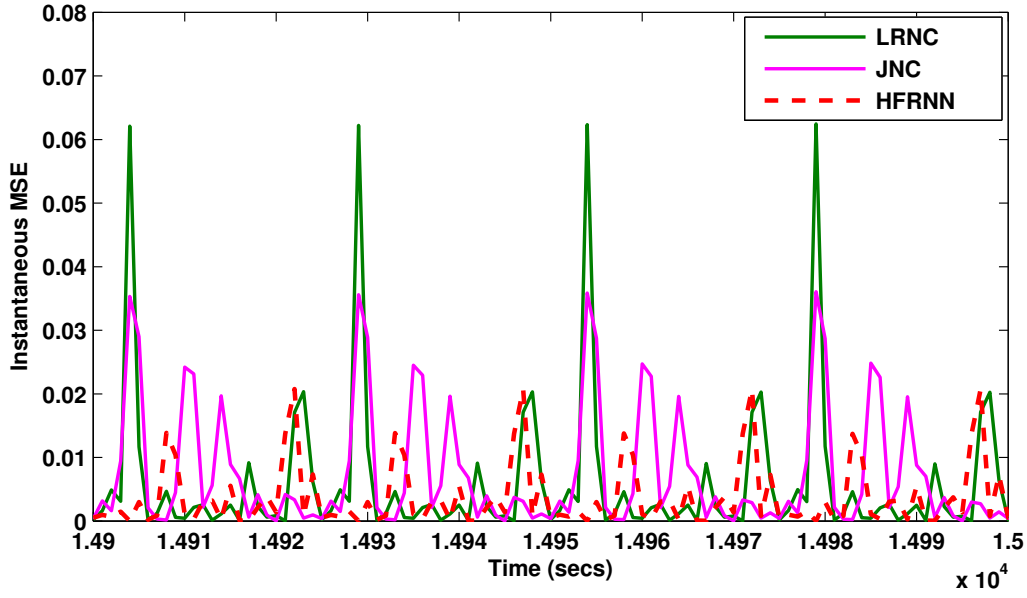


Figure 4.11: Instantaneous MSE of controllers for different initial conditions [Example-1]

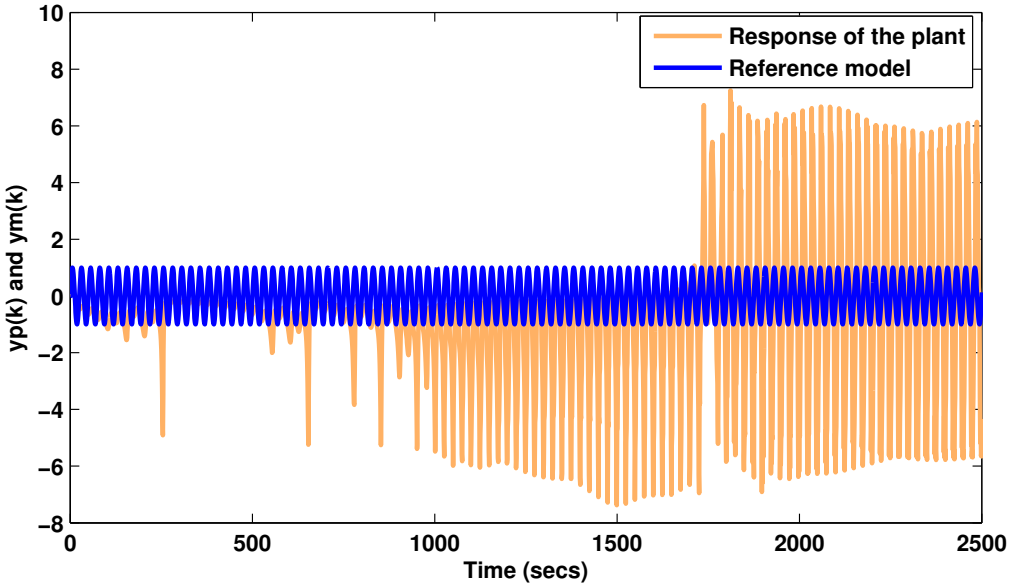


Figure 4.12: Plant's response without controller action [Example-2]

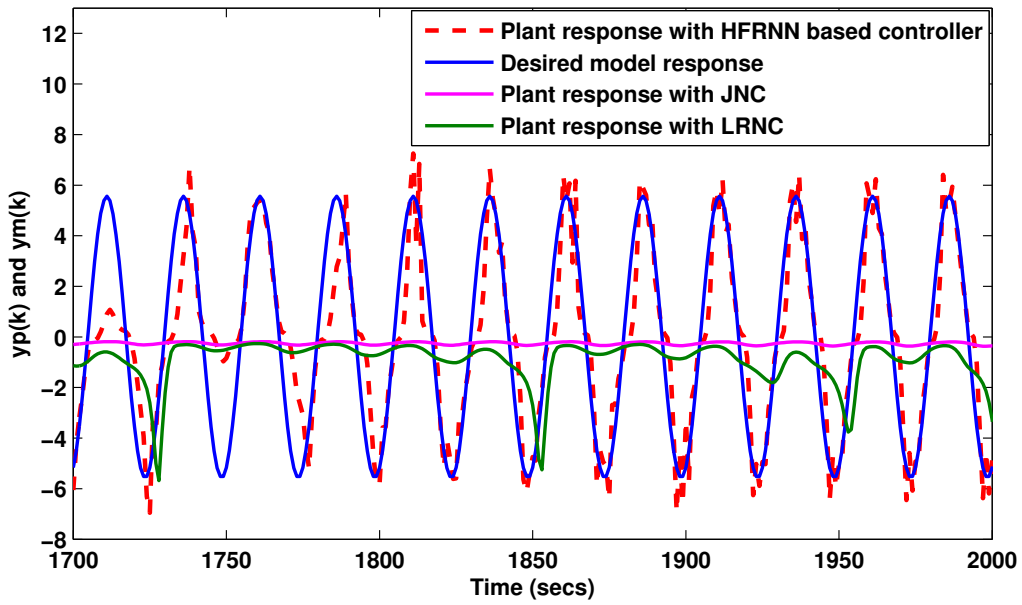


Figure 4.13: Response obtained from the plant during the initial training phase [Example-2]

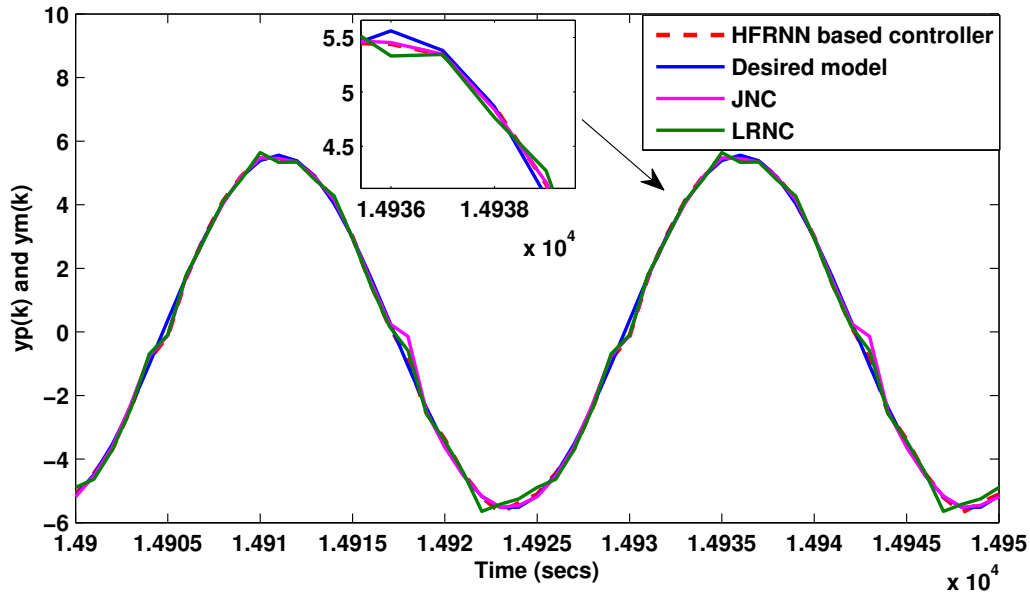


Figure 4.14: Response of controllers during the final phase of training [Example-2]

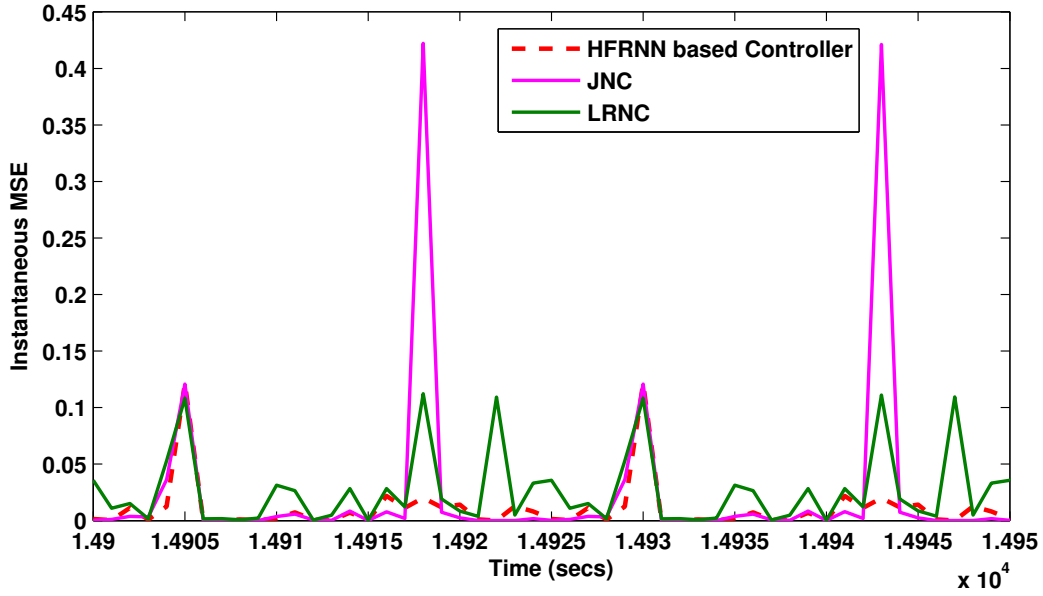


Figure 4.15: Comparison of Instantaneous MSE curves [Example-2]

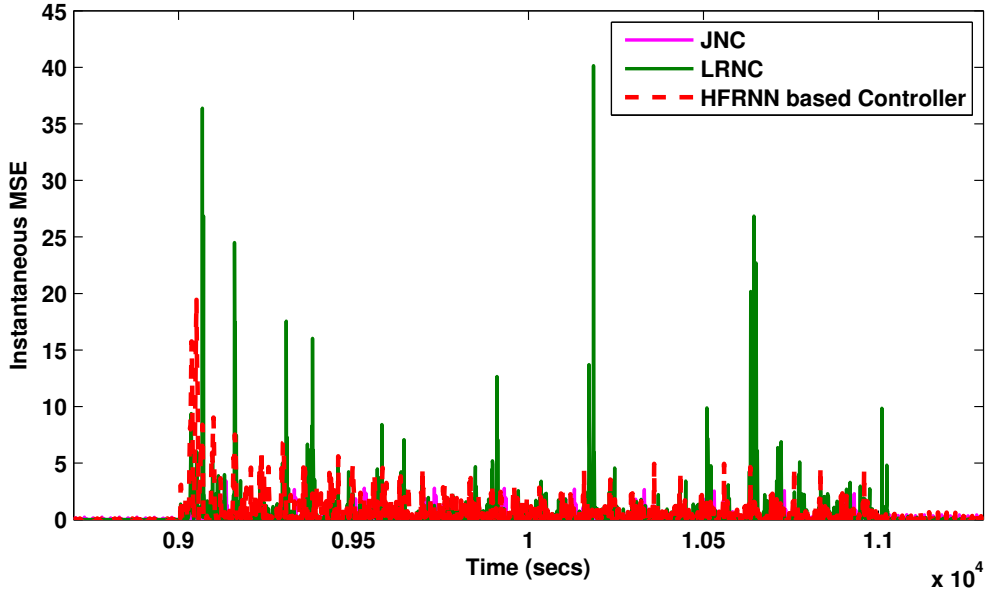


Figure 4.16: Comparison of the effect of external noise on controllers [Example-2]

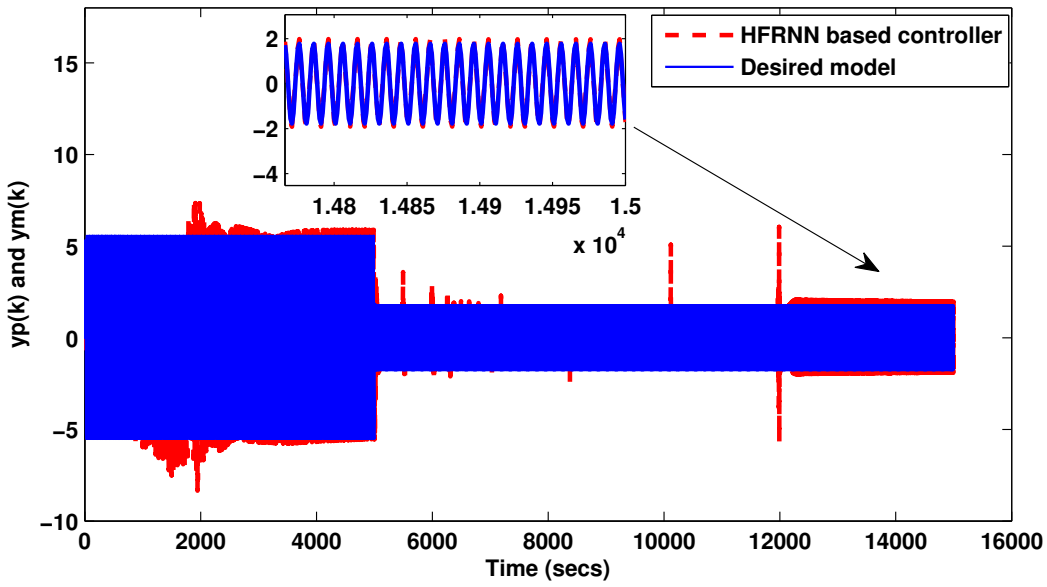


Figure 4.17: Recovering ability of HFRNN based controller [Example-2]

HFRNN controller is found to be 4.7×10^{-7} and that of JNC and LRNC are found to be 4.88×10^{-8} and 6.17×10^{-7} respectively. Figure 4.9 shows the responses of the controller during the final stages of training with a square waveform as a reference signal.

Testing of models with different initial conditions

Additionally, the controller was also tested by assigning a different initial condition of interval (-1,1) for weights and biases. Initially, the plant failed to track the reference model, but with training and updates to network weights through the BP algorithm, the controller started to learn the dynamics of the plant. Towards the end of training, it is found to resemble the desired response of the plant. Figure 4.10 shows the response of the controller during the final stages of training. The instantaneous MSE is also shown in Figure 4.11. The proposed HFRNN controller is seen to provide better MSE compared to other selected JNC and LRNC controllers.

4.6.4 Example 2: A non-linear dynamic plant

To test the efficiency of the proposed controller, another BIBO plant in [45] is considered:

$$y_p(k+1) = \frac{y_p(k)}{(1+y_p(k-1)^2) + u_c(k)^3} + v(k) \tag{4.32}$$

The nonlinear plant takes the identification structure as below: $y_p(k+1) = f[y_p(k), y_p(k-1), u_c(k)]$ The reference model is given by the following difference equation:

$$y_m(k+1) = 0.72y_m(k) + 0.64y_m(k-1) - 0.5y_m(k-2) + r(k) \tag{4.33}$$

where f is the nonlinear function, $r(k)$ denotes the external reference input, $u_c(k)$ is the control input generated by the controller and $v(k)$ denotes the disturbance signal. The reference signal applied to the system is, $r(k) = sin(\frac{2\pi k}{25})$. The inputs of the controller are $[y_p(k), y_p(k-1), y_p(k-2), r(k), u_c(k-1), u_c(k-2)]$ and that of identifier are $[u_c(k), y_p(k-1), y_p(k)]$. The response of the plant without controller action is shown in Figure 4.12. The plant does not track the desired model response without controller action. With the application of the controller, the plant slowly starts to follow the desired model and after 15000 epochs the plant resembles the desired model response. Figure 4.13 shows the responses of the controller during the initial training phase and Figure 4.14 shows the responses of the controller during the final stages of training. The plant is trained online and the instantaneous MSE is also shown in Figure 4.15. The MSE is found to increase to the maximum during the initial stages of training and has reached a minimum value after 15000 epochs. The Figures 4.13 and 4.14 and the Table 4.2 show that the HFRNN-based controller performs better than the Jordan and the LRNN-based controller scheme.

4.6.5 Disturbance testing

To examine the recovering ability of the proposed controller, a sudden disturbance signal, $dis(k) = sin(\frac{2\pi k}{40})$ was added between the time interval $9000 < k \leq 11000$. A static learning rate of 0.001 was considered and the effect of external noise on HRC, JNC, and LRNC was observed. The MSE of all controllers is found to increase with the addition of external noise. This is due to the model deviating from the reference model. After a few epochs of training, the system is seen to adapt to its new changes with MSE decreasing. The instantaneous MSE of the proposed controller is found to be very small around 0.0080 as compared to JNC and LRNC. Figure 4.16 shows the effect of disturbance on the proposed controller, JNC and LRNC.

4.6.6 Recovering ability of the HFRNN based controller

The recovering ability of the proposed controller was also tested by changing the reference input. When the reference input was changed from the previous one to the new one, the network was able to capture and adapt to the new reference input. For this, the new reference input $r(k)$ is considered as below between the interval $5000 < k \leq 15000$:

$$r(k) = \begin{cases} \sin\left(\frac{2\pi k}{25}\right), & \text{for } 0 < k \leq 5000 \\ \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right), & \text{for } 5000 < k \leq 15000 \end{cases} \tag{4.34}$$

Figure 4.17 shows the online adaptability of the controller.

Testing of models with Square wave as a reference signal

The controller was also supplied with a square waveform as a reference signal to test the recovery ability. The network was found to capture and adapt to a new reference input. For this, the new reference input $r(k) = square(\frac{\pi k}{40})$ was considered. The controller was found to show good adaptability. The plant slowly starts to follow the desired model and after 13000 epochs the plant resembles the desired model response. The instantaneous MSE of the proposed HFRNN controller is found to be 1.660×10^{-7} and that of JNC and LRNC is found to be 9.880×10^{-7} and 6.000×10^{-7} respectively. Figure 4.18 shows the responses of the controller during the final stages of training with a square waveform as a reference signal.

Testing of models with different initial conditions

The controller was also tested by assigning a different initial condition of interval (-1,1) for weights and biases. Initially, the plant does not track the reference model, but with training and an update of network weights through the BP algorithm, the controller starts to learn the dynamics of the plant, and slowly towards the end of training it is found to resemble the desired response of the plant. Figure 4.19 shows the response of the controller during the initial and final stages of training. The instantaneous MSE is also shown in Figure 4.20. The proposed HFRNN controller is seen to provide a better MSE as compared to other selected JNC and LRNC controllers. From the results, it is very evident that the structure could capture the changing dynamics of the plant very efficiently. The proposed approach is order-independent and the structure is also simple as compared to other recurrent structures in the literature. All

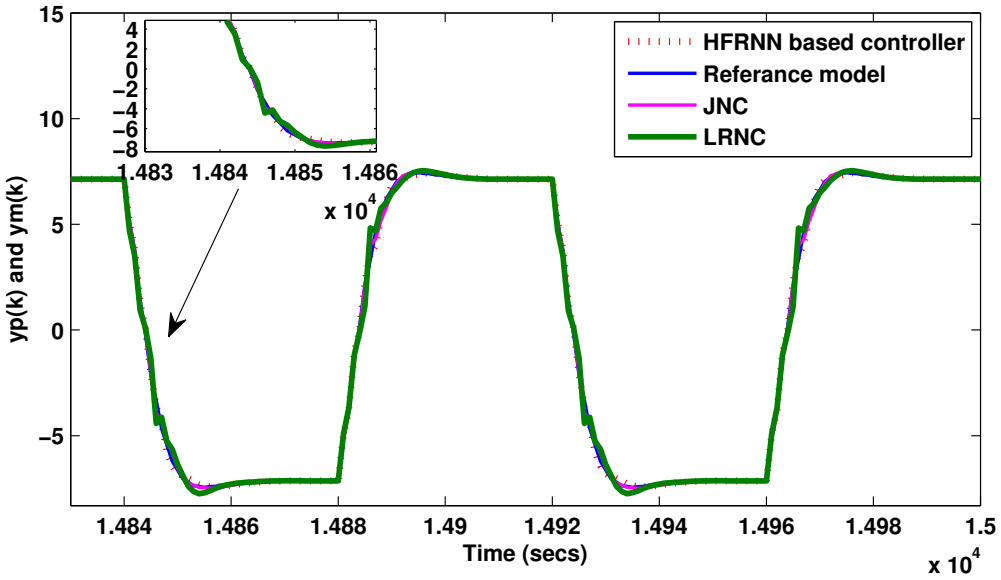


Figure 4.18: Response of controllers with a square waveform as a reference signal[Example-2]

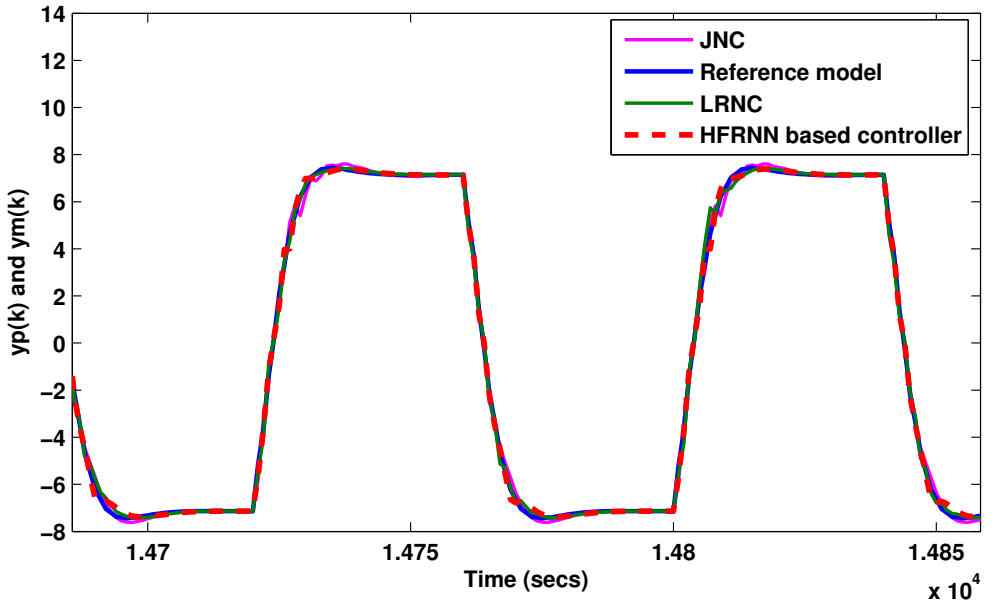


Figure 4.19: Response of controllers for different initial conditions [Example-2]

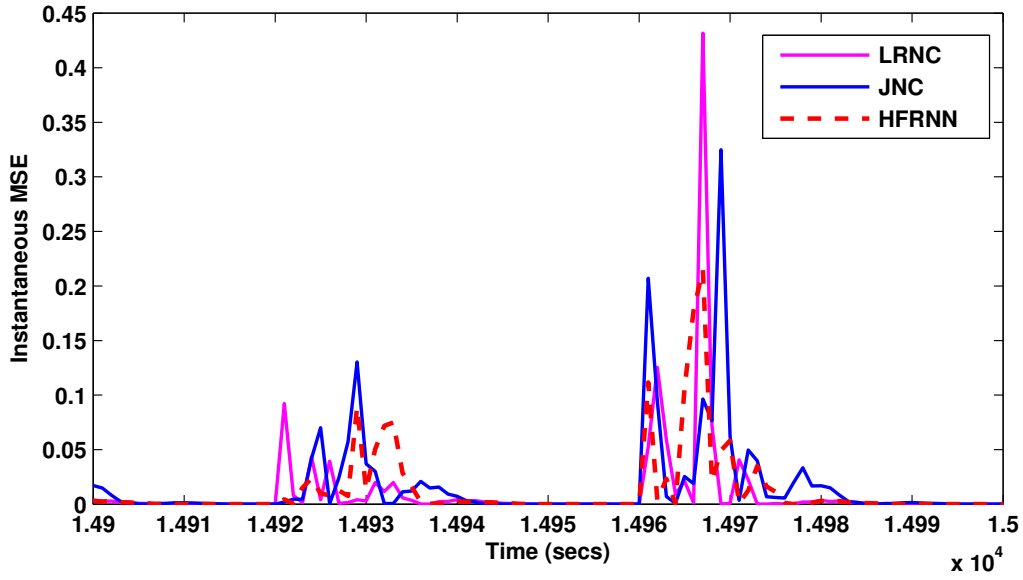


Figure 4.20: Instantaneous MSE of controllers for different initial conditions [Example-2]

the selected controller structures were provided with the same inputs and hidden neurons. The proposed HFRNN though initially took time to adapt to changes, but with iterative learning, it was able to track the reference model. The model is more efficient than JNC and LRNC, as shown by the lower MSE for HFRNN.

4.7 Conclusion

In this paper, an HFRNN-based indirect adaptive control scheme for the identification-based control of nonlinear dynamic systems. The controller scheme has both identifier and controller blocks. The controller's performance is compared with Jordan neural controller (JNC) and Local recurrent neural Controller (LRNC) models. The proposed model gives better MSE than JNC and LRNC for any degree of BIBO system. The proposed model can be used to control the unknown plant applications as the sensitivity of the plant is identified online through the HFRNN-based identifier block. The stability and convergence of the updated equations are guaranteed by Lyapunov stability principles. The proposed model is also tested for its recovering ability and its ability to recover from disturbance. The results suggest that the proposed model can be used for real-time nonlinear applications in the future.

Chapter 5

A hybrid constructive training of FFNN and RNN using adaptive learning rate for nonlinear system identification

5.1 Introduction

The identification and control of nonlinear dynamic systems were explored in the previous chapter using the RNN structures with a fixed structure and fixed learning rate. These approaches provided a reliable framework for modeling and real-time control of complex dynamical systems but they are limited by the static nature of the network and the learning rate, which may not fully utilize the system's ability to expand as the problem becomes more complicated. The performance of any type of ANN depends on the architecture, size of the network, number of hidden neurons, number of hidden layers, and training parameters [119]. The architecture of ANN (FFNN, RNN, and so on) is mostly selected through trial and error based on the user experience. These fixed networks are trained using either standard gradient descent-based algorithms such as BP or evolutionary algorithms. The resulting networks are either large or small. Bigger complex networks tend to overfit the data and result in poor generalization. Smaller networks lead to poor convergence and irregular pattern learning [120]. Likewise, the value of the learning rate also influences the speed and convergence of the algorithm. A smaller learning rate leads to slow convergence and a higher learning rate leads to faster convergence

and instability. To address these limitations, this chapter introduces a novel methodology that employs a growing neural network with ALR. A novel hybrid constructive algorithm for FFNN and LRNN for identifying non-linear dynamical systems is proposed. The constructive algorithm for FFNN is denoted as CFFNN and the constructive algorithm for LRNN is denoted as CLRNN. This approach allows the network to dynamically grow its structure, optimizing complexity and improving generalization. The integration of ALR into a novel constructive algorithm enhances the training process by adjusting the learning rate leading to faster convergence and stability of the network. The learning rate conditions are derived to generate system stability using the Lyapunov stability theory. The performance of the proposed CFFNN and CLRNN with ALR is evaluated under various conditions, demonstrating their advantages over fixed-neural structures.

5.2 Mathematical structure of FFNN and LRNN

5.2.1 Feed forward neural network

In this section, the structure of FFNN is discussed as shown in Figure 5.1.

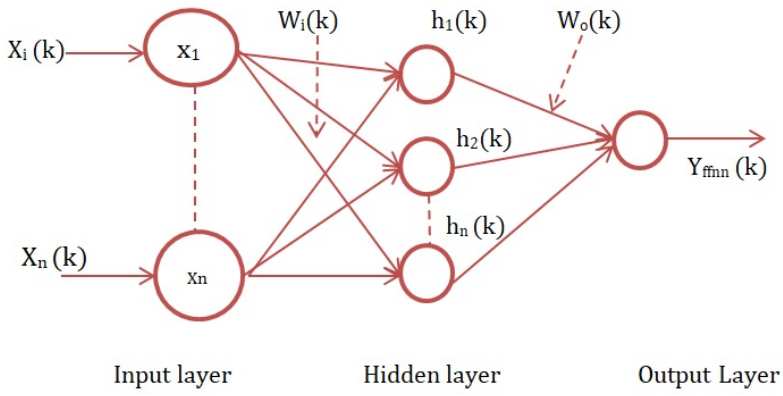


Figure 5.1: FFNN structure

1. **Input layer :** The input layer has input neurons and the input vector is denoted as $X(k) = [X_1(k), X_2(k), ..., X_n(k)]$. The input weight vector is given as $W_i(k)$. In our problem, three inputs $y(k), y(k-1), r(k)$ are considered for constructing the FFNN network.

2. **Hidden layer:** In this layer, each neuron processes the information from the input layer to the output layer. It is acted upon by either a linear or non-linear activation function. The output of the k^{th} hidden layer is

$$h_n(k) = f_1\left(\sum_{i=1}^{n} X(k-i)W_i(k) + b_i(k)W_{bi}(k)\right) \tag{5.1}$$

Here, $b_i(k)$ denotes the input bias vector, $W_{bi}(k)$ denotes the input bias weight vector and f_1 denotes the non-linear activation function of the hidden layer.

3. **Output layer:** The output layer computes the final output values as

$$y_{ffnn}(k) = f_2\left(\sum_{i=1}^{m} h_n(k)W_o(k) + b_o(k)W_{bo}(k)\right) \tag{5.2}$$

Here, $y_{ffnn}(k)$ is one step ahead predicted output of the network, $b_o(k)$ denotes the output bias vector, $W_{bo}(k)$ denotes the output bias weight and f_2 denotes the linear activation function of the output layer.

5.2.2 Local recurrent neural network

In this section, the structure of LRNN is discussed as shown in Figure 5.2.

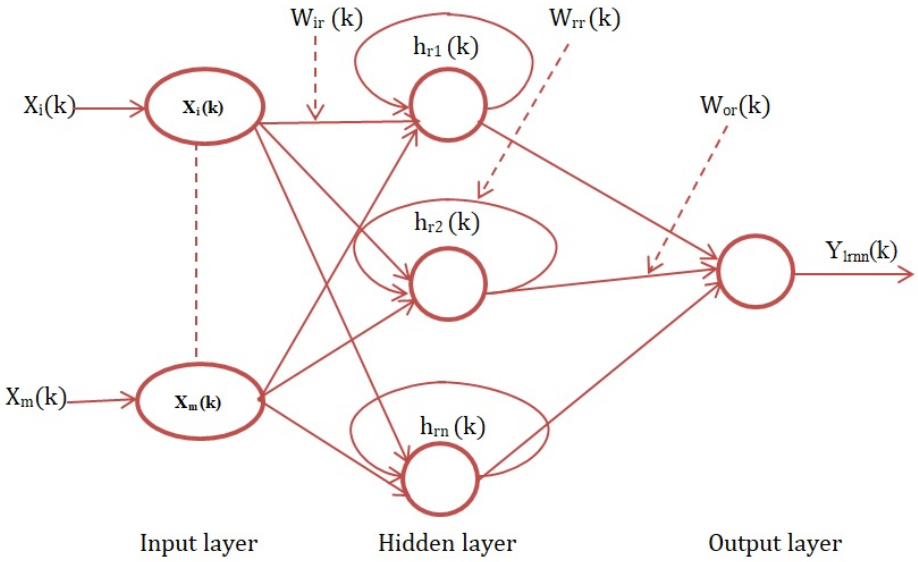


Figure 5.2: LRNN structure

1. **Input layer :** The input layer has 'm' input neurons and the input vector is denoted as $X = [X_1(k), X_2(k), ..., X_m(k)]$. The input weight vector is given as $W_{ir}(k)$. In our problem, two inputs $y(k-1), r(k)$ are considered for constructing the LRNN network. The RNN takes fewer inputs as compared to FFNN due to the presence of memory connections in their architecture.

2. **Hidden layer:** Each neuron in this layer processes the information received from the input layer and its self-recurrent layer to the output layer. It is acted upon by either a linear or non-linear activation function. The output of the k^{th} hidden layer is

$$h_{rn}(k) = g_1\left(\sum_{i=1}^{m} X(k-i)W_{ir}(k) + b_r(k)W_{br}(k) + P(k)w_{rr}(k)\right) \tag{5.3}$$

   Here, $b_r(k)$ denotes the input bias vector, $W_{ir}(k)$ denotes the input bias weight, $W_{br}(k)$ denotes the bias weight, $P(k)$ denotes the self-recurrent hidden neurons and g_1 denotes the non-linear activation function of the hidden layer.

3. **Output layer:** The output layer computes the final output values as

$$y_{lrnn}(k) = g_2\left(\sum_{i=1}^{m} H_{rn}(k)W_{or}(k) + b_o(k)W_{bo}(k)\right) \tag{5.4}$$

   Here, $y_{lrnn}(k)$ is one step ahead predicted output of the network, $b_o(k)$ denotes the output bias vector, $W_{bo}(k)$ denotes the output bias weight, and g_2 denotes the linear activation function of the output layer.

5.3 Learning algorithm

To update the training parameters of CLRNN and CFFNN, a gradient descent-based BP algorithm with ALR is used. The weights to be updated for the proposed algorithm include $[W_i(k), W_o(k), W_{ir}(k), W_{rr}(k), W_{or}(k)]$. The computation of gradient is performed layer by layer in both the cases of FFNN and LRNN. We have considered one output for this problem. y_{ffnn} denotes output of FFNN and y_{lrnn} denotes the output of LRNN. As with another gradient-based algorithm, the gradient is obtained with respect to the learning error. The weights are trained every epoch to reduce the cost function. MSE is considered as the cost function in this

work and is calculated as :

$$E(k) = \frac{1}{2}e(k)^2 \tag{5.5}$$

The errors are backpropagated from the output through the hidden layer to update the output layer weights of both FFNN and LRNN. Now, to update the input layer weights of both structures, the errors are back-propagated from the output layer through the hidden layer to the input layer. For FFNN, the input weights $W_i(k)$ and output weights $W_o(k)$ are updated by calculating the change of gradient of the error with respect to weights as:

$$\frac{\partial E(k)}{\partial W_o(k)} = -e(k) \times \frac{\partial y_{ffnn}(k)}{\partial W_o(k)} \tag{5.6}$$

$$\frac{\partial E(k)}{\partial W_i(k)} = -e(k) \times \frac{\partial y_{ffnn}(k)}{\partial h_n(k)} \times \frac{\partial h_n(k)}{\partial W_i(k)} \tag{5.7}$$

Where $e(k) = y_p(k) - y_{ffnn}(k)$ is the learning error of the feed-forward network.
Similarly, for LRNN the input weights $W_{ir}(k)$, self-recurrent hidden weights $W_{rr}(k)$, and output weights $W_{or}(k)$ are updated by calculating the change of gradient of the error with respect to weights as:

$$\frac{\partial E(k)}{\partial W_{or}(k)} = -e(k) \times \frac{\partial y_{lrnn}(k)}{\partial W_{or}(k)} \tag{5.8}$$

$$\frac{\partial E(k)}{\partial W_{ir}(k)} = -e(k) \times \frac{\partial y_{lrnn}(k)}{\partial h_{rn}(k)} \times \frac{\partial h_{rn}(k)}{\partial W_{ir}(k)} \tag{5.9}$$

$$\frac{\partial E(k)}{\partial w_{rr}(k)} = -e(k) \times \frac{\partial y_{Lrnn}(k)}{\partial h_{rn}(k)} \times \frac{\partial h_{rn}(k)}{\partial w_{rr}(k)} \tag{5.10}$$

The new weights are calculated as per the stochastic gradient weight update rule as

$$W(k+1) = W(k) - \eta e(k) \frac{\partial E(k)}{\partial W(k)} \tag{5.11}$$

Where $W(k)$ is the weight matrix given as $W = [W_i(k), W_{ir}(k), W_{rr}(k), W_{or}(k), W_o(k)]$. η is the adaptive learning rate. For the initial iteration, η is chosen as 0.001. Further, the learning rate is adjusted based on the fractional error $e_f(k)$, given by

$$e_f(k) = \frac{mse(k) - mse(k-1)}{mse(k)} \tag{5.12}$$

where $mse(k)$ is the cost function. The learning rate is increased or decreased based on the value of validation MSE.

5.4 Lyapunov stability analysis for adaptive learning rate

The learning rate is derived using the Lyapunov stability principles. The stability and convergence of the proposed algorithm are also studied using Lyapunov stability principles. According to Lyapunov stability, the system achieves stability when the Lyapunov-based function is minimum and positive. To derive the conditions, firstly a Lyapunov-based function $V(k)$ is chosen such as :

$$V(k) = \frac{1}{2}e^2(k) \tag{5.13}$$

$V(k) > 0$ for $e(k) \neq 0$ and $V(k) = 0$ for $e(k) = 0$.
The time derivative of the Lyapunov function is

$$\dot{V(k)} = V(k+1) - V(k) \tag{5.14}$$

(or)

$$\dot{V(k)} = \frac{1}{2}e^2(k+1) - \frac{1}{2}e^2(k) \tag{5.15}$$

Further expressing $e(k+1)$ in terms of $e(k)$ as

$$e(k+1) = e(k) + \Delta e(k) \tag{5.16}$$

where $\Delta e(k)$ is the change in error defined as $\frac{\delta e(k)}{\delta w(k)}\Delta w(k)$. During learning rate adjustment, the change of error $\Delta e(k)$ becomes

$$\Delta e(k) = -\alpha \times \eta_k \times e(k) \tag{5.17}$$

Substituting Eq.(5.17) into Eq.(5.16),

$$e(k+1) = e(k) - \alpha \times \eta_k \times e(k) \tag{5.18}$$

(or)

$$e(k+1) = e(k)[1 - \alpha \times \eta_k] \tag{5.19}$$

Substituting Eq.(5.19) into Eq.(5.14),

$$\dot{V(k)} = \frac{1}{2}e^2(k)[1 - \alpha\eta_k]^2 - \frac{1}{2}e^2(k) \tag{5.20}$$

(or)

$$\dot{V(k)} = \frac{1}{2}e^2(k)([1 - \alpha\eta_k]^2 - 1) \tag{5.21}$$

(or)

$$\dot{V(k)} = \frac{1}{2}e^2(k)([1 - 2\alpha\eta_k + \alpha^2\eta_k^2 - 1) \tag{5.22}$$

(or)

$$\dot{V(k)} = \frac{1}{2}e^2(k)([-2\alpha\eta_k + \alpha^2\eta_k^2) \tag{5.23}$$

The condition for stability is $\dot{V(k)} \leq 0$. Therefore, from Eq.(5.23),

$$-2\alpha\eta_k + \alpha^2\eta_k^2 \leq 0 \tag{5.24}$$

(or)

$$\eta_k(-2\alpha + \alpha^2\eta_k) \leq 0 \tag{5.25}$$

(or)

$$\eta_k\alpha(-2 + \alpha^2\eta_k) \leq 0 \tag{5.26}$$

where learning rate is η_k and α is the constant. From above Eq.(5.26), the value of learning rate is $0 \leq \eta_k \leq \frac{2}{\alpha^2}$. The system remains bounded and does not diverge, as long as the learning rate is within this range and $\dot{V(k)}$ remains negative and converges to 0 as $\lim_{k\to\infty} V(k) = 0$

5.5 A constructive approach for growing feed-forward and recurrent networks

Figure 5.3 shows the main blocks of the constructive algorithm. The initial network is empty, with only connections between the input and output layers. The algorithm enters the constructive loop when the validation error is found more than the threshold efficiency. The hidden layer forms the 1^{st} connection with the input and output layer. This also results in the addition of one new hidden neuron. The network's performance is checked after every epoch until the

stopping criteria are reached. The detailed steps followed in each block are discussed below:

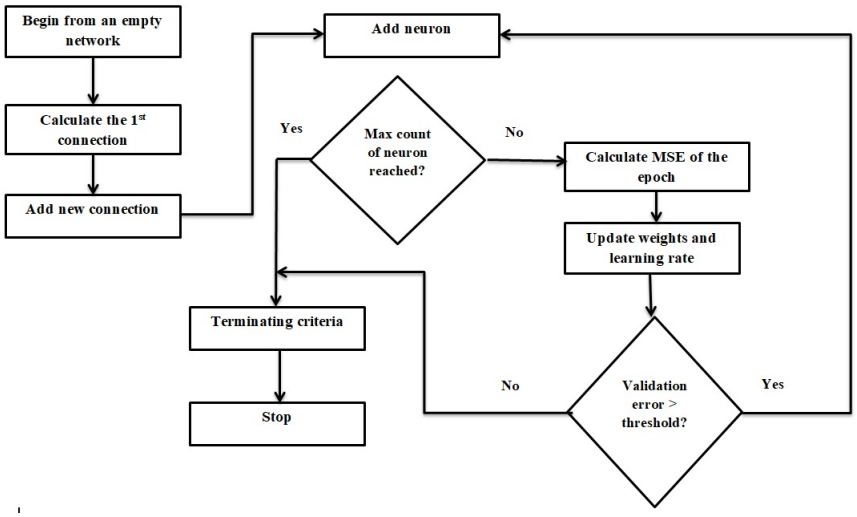
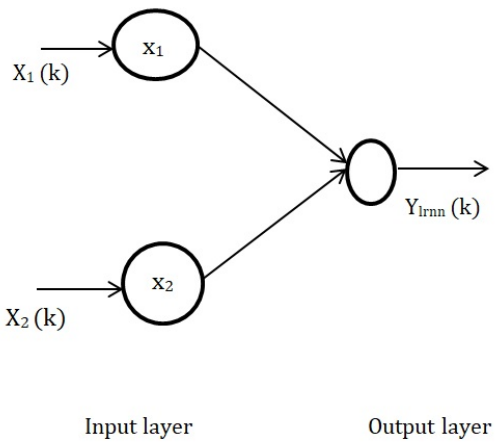


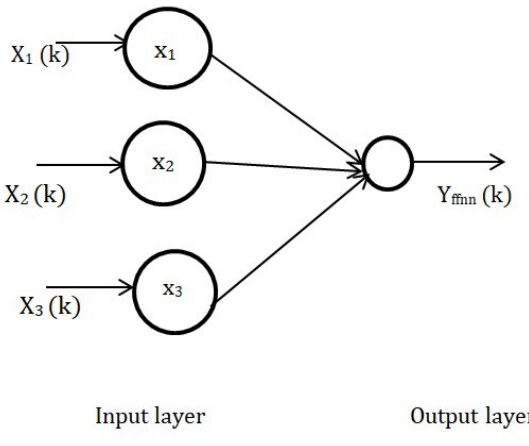
Figure 5.3: A constructive algorithm procedure for feed-forward and recurrent networks

5.5.1 Building from an empty network

The network's starting point consists of only the input and output layers and associated neurons. The number of inputs in the prediction problem X_p equals the number of observations O and the number of outputs in the prediction problem Y_o equals the number of predictions P (i.e) $X_p = O, Y_o = P$. In our analysis, two inputs and one output are considered for LRNN, and three inputs and one output are considered for FFNN structure. After the execution of the first step, the LRNN and FFNN network structure is as in Figure 5.4 respectively.



(a) Initial CLRNN network



(b) Initial CFFNN network

Figure 5.4: Initial network

5.5.2 Adding Connections

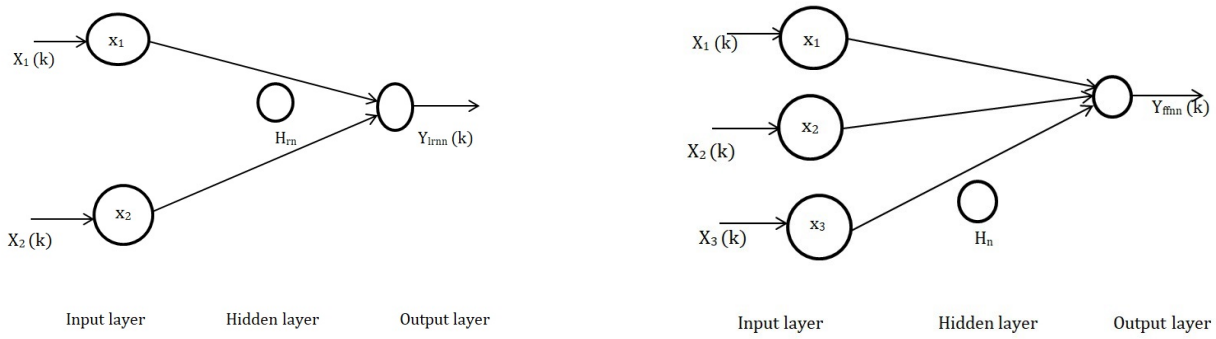
This is the first step of building the constructive algorithm. The network is calculated for the validation MSE. The validation MSE (MSE_{val}) calculates the average of the squared difference between the predicted and the actual values from the unseen validation data set. If the $MSE_{val}(k)$ is found promising below the threshold efficiency (ϵ), then no new connections are added as the network is found performing satisfactorily :

$$MSE_{val}(k) \leq \epsilon_{threshold}, \text{no new neurons are added} \tag{5.27}$$

However, if there is no improvement in performance, then 1^{st} hidden connection is added successfully:

$$MSE_{val}(k) > \epsilon_{threshold}, \text{one new neurons is added} \tag{5.28}$$

The newly established neuron obtains each of the possible connections from input, output, and self-recurrent neurons (in the case of LRNN). Figure 5.5 shows the addition of new connections of CLRNN and CFFNN.



(a) Adding connection in CLRNN **(b)** Adding connection in CFFNN

Figure 5.5: Adding new connections

5.5.3 Adding neurons

The addition of neurons to the hidden layer of the network is done as follows:

1. If no hidden neurons are present, then the newly added neuron $h_1(k)$ becomes the first hidden neuron in the network, forming the first hidden layer (shown with black lines under the hidden layer in Figure 5.6).

2. If hidden neurons $(h_1(k), h_2(k), ..., h_k(k))$ are already present, then the newly added neuron $h(k+1)$ will become part of the existing hidden layer (shown with red dotted lines under hidden layer in Figure 5.6).

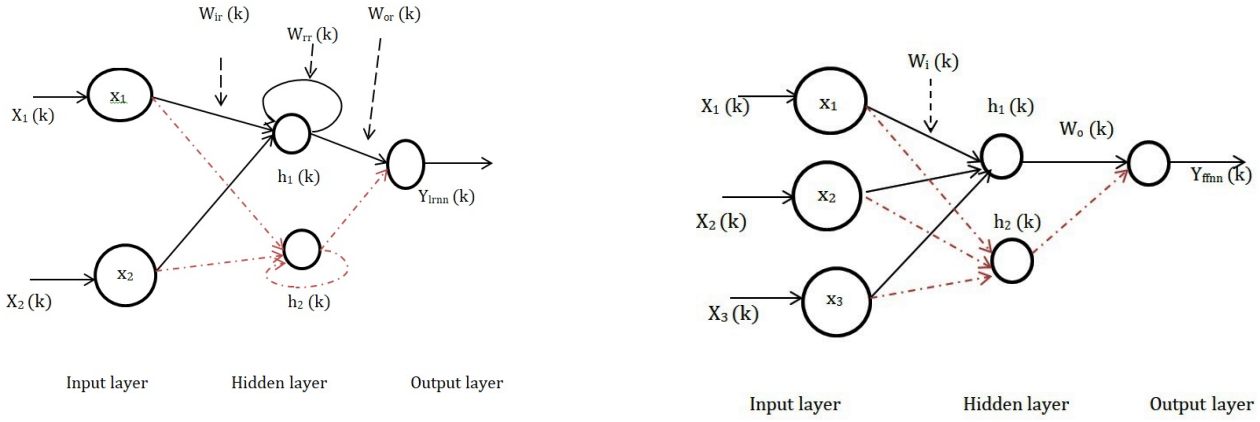
After the addition of each neuron, the network enters the constructive loop and performs the following steps:

1. **Forward pass**: The forward pass is executed, processes the input data, and calculates the output by using Eq.(5.1)-Eq.(5.4).

2. **Calculation of Error and new learning rate**: The error of the network is further calculated and at the end of each epoch, the $MSE_{val}(k)$ is found and compared with a desired threshold. If found higher, the adaptive learning rate is updated based on the fractional error of the network.

3. **Back propagation of error**: The error is back propagated to update the weights for the current epoch using Eq.(5.6)-Eq.(5.11).

4. **Addition of new neurons**: The performance of the network is checked for successful decrement or increment of a validation error. The new neurons are added when the the validation MSE is found to decrease at every epoch thus improving the performance. The algorithm quits the constructive loop and enters the stopping criteria block. The performance criteria such as RMSE and AMSE are also calculated to evaluate the performance of the network.



(a) case 1: First neuron **(b)** case 2: subsequent neurons

Figure 5.6: Adding new neurons

5.5.4 Terminating criterion

The terminating criterion of the algorithm is executed when the constructive loop faces failure in more than 3 subsequent MSE epochs. The algorithm monitors the validation error at the end of all epochs before the addition of a new novel neuron and it executes stopping criteria in either of two cases:

1. **Validation MSE is increasing for 3 cycles**: The algorithm waits for three consecutive cycles for the MSE_{val} to decrease. If the condition is not met, no further neurons are added:

   If $\text{MSE}_{\text{val}}(k) > \text{MSE}_{\text{val}}(k-1) > \text{MSE}_{\text{val}}(k-2)$, no new neurons are added.

2. **Validation MSE is decreased or remains the same for 3 cycles**: If the validation MSE has either remained the same or decreased constant for 3 consecutive training cycles, then no new neurons are added further. In such case, the network is performing satisfactorily and the addition of new neurons will lead to increased complexity (i.e)

   If $\text{MSE}_{\text{val}}(k) \leq \text{MSE}_{\text{val}}(k-1) \leq \text{MSE}_{\text{val}}(k-2)$, no new neurons are added.

3. **Maximum neurons reached**: The algorithm also exits the constructive loop, if the maximum neurons are reached

The final network through a constructive approach is obtained when no further neurons are added to the hidden layer.

5.5.5 Adaptive learning rate

To improve the speed and performance of the network, a novel ALR is proposed with this algorithm. When the learning rate remains fixed, the network fails to converge quickly and gets stuck in local minima. Hence, it is required to adjust the step sizes based on the gradients. When the gradients are large and the error remains small, a high learning rate is required and at times when the gradients are small with large error, smaller learning rates are required. Hence, ALR based on fractional error $e_f(k)$ is proposed in this work. For this, a fractional error $e_f(k)$ is defined by calculating the relative change in MSE between consecutive iterations as:

$$e_f(k) = \frac{mse(k) - mse(k-1)}{mse(k)} \tag{5.29}$$

where $mse(k)$ is the cost function at iteration k. The ALR condition adapts as below: Based on the fractional error $e_f(k)$, the learning rate is made adaptive as follows:

1. If the present value of MSE is less than the previous MSE, then the learning rate is increased i.e.

   if $e < e(k-1)$ **then**
   $\eta(k+1) = \eta(k)(1 + \alpha e_f(k))$
   end if

2. If the present value of MSE is greater than the previous MSE, then the learning rate is decreased i.e.

   if $e \geq e(k-1)$ **then**
   $\eta(k+1) = \eta(k)(1 - \alpha e_f(k))$
   end if

Where the learning rate $\eta(k)$ ranges from $\left(0 < \eta < \frac{2}{\alpha^2}\right)$ and α is the fixed scaling constant to control the sensitivity of the change in error and to maintain stability throughout the procedure. The value of α is taken as 0.001 for all examples.

5.6 Simulation experiments

In this section, the performance of CFFNN and CLRNN is tested over one example of nonlinear plant equations of varying degrees and one nonlinear benchmark problem for simplicity. The Mackey glass series is considered as the nonlinear benchmark problem. The initial parameters are considered the same for both FFNN and LRNN structures as $\eta = 0.001, \alpha = 0.01$. $r(k)$ denotes the external input of the plant, $y_p(k)$ denotes the output of the plant. The performance of CLRNN and CFFNN is measured using RMSE and AMSE. The RMSE and AMSE are defined as follows

$$\text{RMSE}(k) = \sqrt{\frac{\sum_{i=0}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{5.30}$$

$$\text{AMSE}(k) = \frac{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}}{k} \tag{5.31}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, N indicates the number of samples and k is the number of iterations

5.6.1 Example-1: A nonlinear plant equation

In this example, the performance of the algorithm is tested on the following II-order nonlinear difference equation [36]

$$y_p(k) = \frac{y_p(k-1)}{1+y_p^2(k-2)} + r^3(k-1) \tag{5.32}$$

The above equation depends on both past input and output values of the plant. The identification structure of the plant can be written as

$$y_p(k) = f[y_p(k-1), y_p(k-2), r(k-1)] \tag{5.33}$$

The following external input $r(k)$ is supplied to the plant :

$$r(k) = sin\left(\frac{2\pi k}{40}\right) \tag{5.34}$$

The entire dataset is divided into training and validation values. A total of 1000 data samples is considered with 500 values for training (t=1 to 500) and 500 values for validation (t=501 to 1000). The example is tested on both CFFNN and CLRNN algorithms. The maximum number of neurons for all three structures is kept at 15.

Performance of CLRNN on example-1

The initial structure of the network is selected as 2-0-1 i.e. 2 input neurons as $r(k)$ and $y(k-1)$, no hidden layer, and 1 output layer with 1 output neuron. The network takes the following identification structure:

$$y_{lrnn}(k) = \hat{f}[y_p(k-1), r(k)] \tag{5.35}$$

The proposed CLRNN is compared with CLRNN with a fixed learning rate and fixed LRNN structures. The initial MSE of CLRNN at the start of the training was found to be 0.016. Figure 5.7 shows the predicted response of the CLRNN over selected LRNN structures. Figure 5.8 shows the effect of MSE on CLRNN over the increase of hidden neurons. From the results, it is evident that the proposed algorithm is found to adjust and adapt the learning rate with the training progress. The performance is found to be effective over time series prediction problems even with fewer inputs and hidden neurons. At the final stages of training, RMSE is found to be 0.2223 and AMSE is found to be 0.005. Table 5.1 shows the comparison of performance

obtained for CLRNN over other selected structures. The number of neurons utilized by fixed LRNN is 15 and that utilized by CLRNN is 10. After the final stages of training, CLRNN with ALR results in 2-10-1 architecture with a fixed LRNN resulting in 2-15-1.

Performance of CFFNN on example-1

Generally, FFNN networks do not have memory neurons in their structure and so they require more inputs as compared to RNN for effective prediction [121]. For training the FFNN network, we have kept the initial structure of the network as 3-0-1 i.e. 3 input neurons as $r(k)$, $r(k-1)$, and $y(k-1)$, no hidden layer, and 1 output layer with 1 output neuron. The network takes the following identification structure:

$$y_{ffnn}(k) = \hat{f}[y_p(k-1), r(k), r(k-1)] \tag{5.36}$$

The initial MSE at the start of the training was found to be 0.0089. The performance of CFFNN is compared with CFFNN with a fixed learning rate and fixed FFNN structures. Figure 5.9 shows the predicted response of CFFNN over selected FFNN structures. Figure 5.10 shows the effect of MSE over the increase of hidden neurons on CFFNN with ALR. From the results, it is evident that the proposed algorithm is found to adjust and adapt the learning rate with the training progress. The performance is found to be effective over time series prediction problems. RMSE and AMSE of CFFNN are found to be 0.0167 and 0.0003 respectively at the final stages of training. The number of neurons utilized by fixed FFNN is 15 and that utilized by CFFNN is 12. After the final stages of training, CFFNN with ALR results in 3-12-1 architecture with a fixed FFNN resulting in 3-15-1. Table 5.2 shows the comparison of performance obtained for CFFNN over other selected structures.

5.6.2 Example-2: Mackey-Glass Time series problem

Further, the performance of the algorithm is tested on the following Mackey-glass series prediction problem given by [122]

$$y_p(k+1) = \frac{(1-\alpha)y_p(k) + b * y_p(k-\tau)}{1 + y_p^{10}(k-\tau)} \tag{5.37}$$

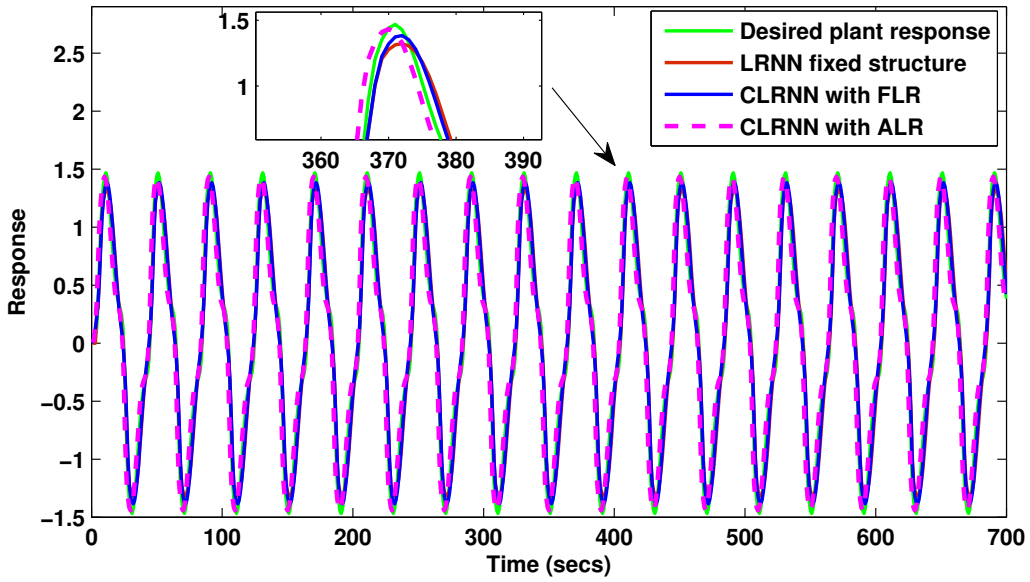


Figure 5.7: Response of CLRNN over selected LRNN structures [Example-1]

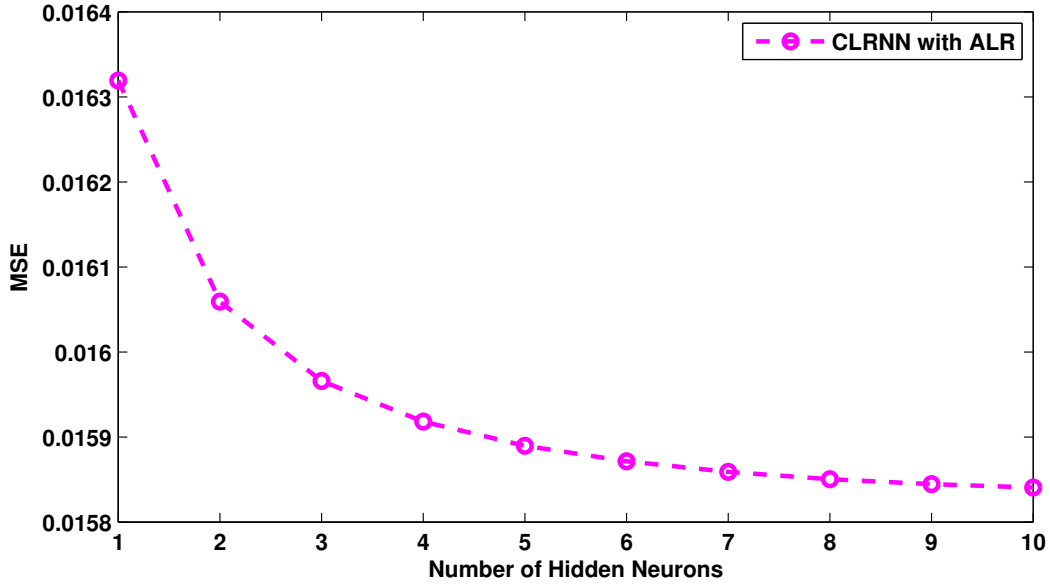


Figure 5.8: MSE curve over the addition of hidden neurons on CLRNN [Example-1]

Table 5.1: Tabulation of performance of CLRNN over selected structures [Example-1]

S.No	Algorithm	Network type	No of samples in each epoch	MSE	AMSE	RMSE
1	Traditional Fixed Learning rate	LRNN	1000	0.0086	0.1173	0.0929
2	Hybrid growing algorithm with ALR	CLRNN	1000	**0.0002**	**0.0005**	**0.0223**
3	Hybrid growing algorithm with FLR	CLRNN	1000	0.1274	0.0162	0.1272

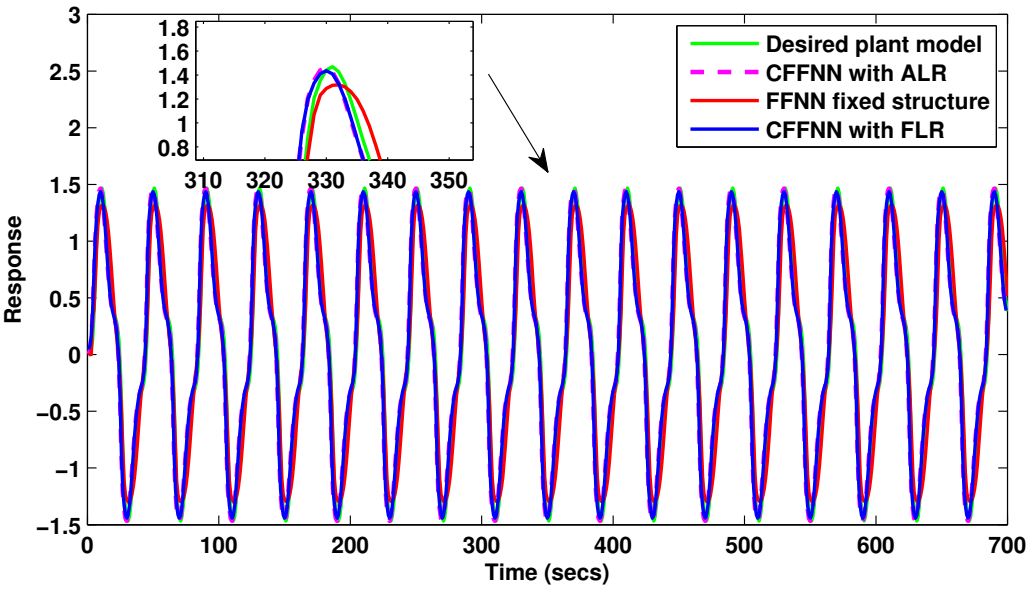


Figure 5.9: Response of CFFNN over selected FFNN structures [Example-1]

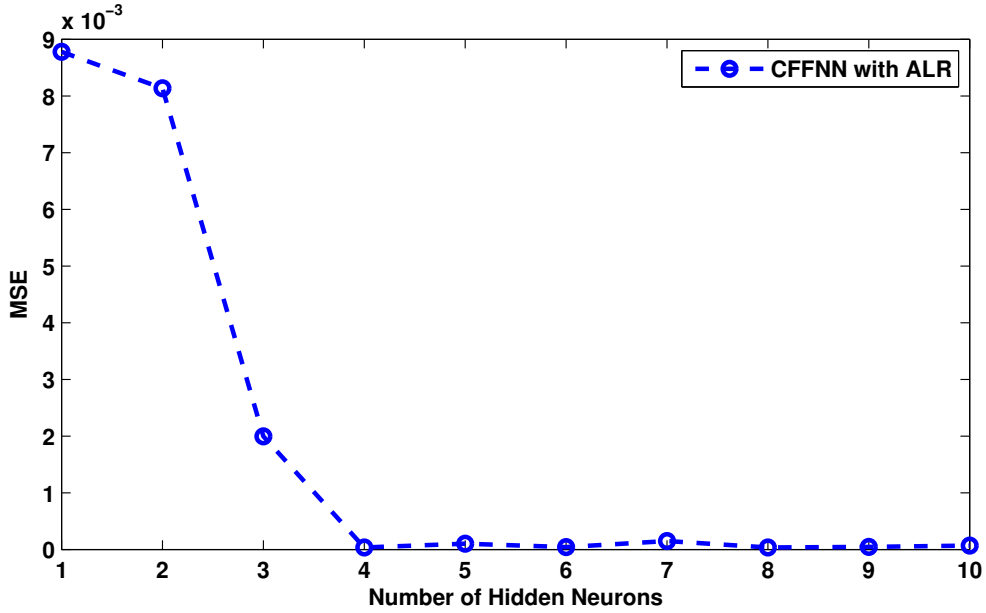


Figure 5.10: MSE curve over the addition of hidden neurons on CFFNN [Example-1]

The above problem takes the identification structure as below:

$$y_p(k+1) = g[y_p(k), y_p(k-17)] \tag{5.38}$$

The time-series problem is applied with values of $\alpha = 0.2$, $\tau = 17$ and $\beta = 0.1$. When $\tau \geq 17$, the problem is said to exhibit chaotic behavior. Thus $\tau = 17$ is considered in this example. Out of 1000 data samples, 500 samples are taken for training, and the rest 500 samples are considered for validation. The maximum number of neurons for all three structures is kept at 20.

Performance of CLRNN on example-2

The initial structure of the network is selected as 2-0-1 i.e. 2 input neurons as $y_p(k)$ and $y_p(k-\tau)$, no hidden layer, and 1 output layer with 1 output neuron. The network takes the following identification structure:

$$y_{lrnn}(k) = \hat{f}[y_p(k), y_p(k-\tau)] \tag{5.39}$$

The proposed CLRNN is compared with CLRNN with a fixed learning rate and fixed LRNN structures. Figure 5.11 shows the predicted response of the CLRNN over selected LRNN structures. Figure 5.12 shows the effect of MSE on CLRNN over the increase of hidden neurons.

From the results, it can be seen that the performance of CLRNN is found to be effective over time series prediction problems even with fewer inputs and hidden neurons. At the final stages of training, RMSE is found to be 0.0027 and AMSE is found to be 0.0046. Table 5.3 shows the comparison of performance obtained for CLRNN over other selected structures. The number of neurons utilized by fixed FFNN is 15 and that utilized by CLRNN is 10. After the final stages of training, CLRNN with ALR results in 2-10-1 architecture with a fixed LRNN resulting in 2-15-1.

Performance of CFFNN on example-2

For training the CFFNN network, we have kept the initial structure of the network as 2-0-1 i.e. 2 input neurons as $y_p(k)$,$y_p(k-17)$, no hidden layer, and 1 output layer with 1 output neuron. The network takes the following identification structure:

$$y_{ffnn}(k) = \hat{f}[y_p(k), y_p(k-17)] \tag{5.40}$$

The initial MSE at the start of the training was found to be 0.8381. The performance of CFFNN is compared with CFFNN with a fixed learning rate and fixed FFNN structures. Figure 5.13 shows the predicted response of CFFNN over selected FFNN structures. Figure 5.14 shows the effect of MSE over the increase of hidden neurons on CFFNN with ALR. The proposed algorithm is seen to adjust and adapt the learning rate with the training progress. From the results, it is evident that the proposed algorithm is found to adjust and adapt the learning rate with the training progress. The performance is found to be effective over time series prediction problems. RMSE and AMSE of CFFNN are found to be 0.0218 and 0.0005 respectively at the final stages of training. The number of neurons utilized by both fixed FFNN and CFFNN is 15. Table 5.4 shows the comparison of performance obtained for CLRNN over other selected structures. After the final stages of training, CFFNN with ALR and fixed FFNN results in 2-15-1 architecture.

5.7 Conclusion

The study represents a hybrid constructive algorithm for FFNN and LRNN known as CFFNN and CLRNN, for identifying a nonlinear dynamic system. The main contribution of the work is as follows: (i) Hidden neurons are added to the network based on their validation MSE

Table 5.2: Tabulation of performance of CFFNN over selected structures [Example-1]

S.No	Algorithm	Network type	No of samples in each epoch	MSE	AMSE	RMSE
1	Traditional Fixed Learning rate	FFNN	1000	0.0319	0.2184	0.1786
2	Hybrid growing algorithm with ALR	CFFNN	1000	**0.0001**	**0.0003**	**0.0167**
3	Hybrid growing algorithm with FLR	CFFNN	1000	0.0013	0.0005	0.0020

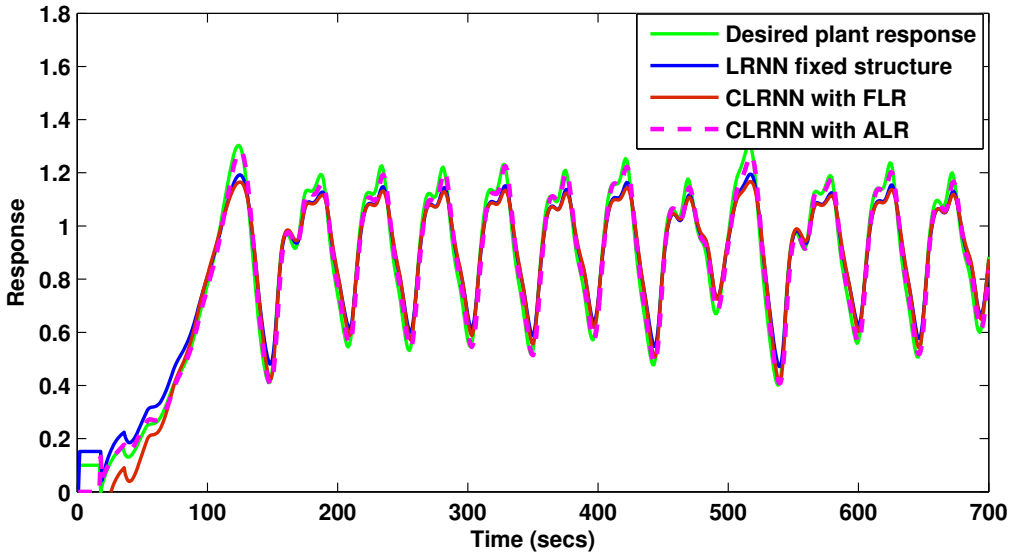


Figure 5.11: Response of CLRNN over selected LRNN structures [Example-2]

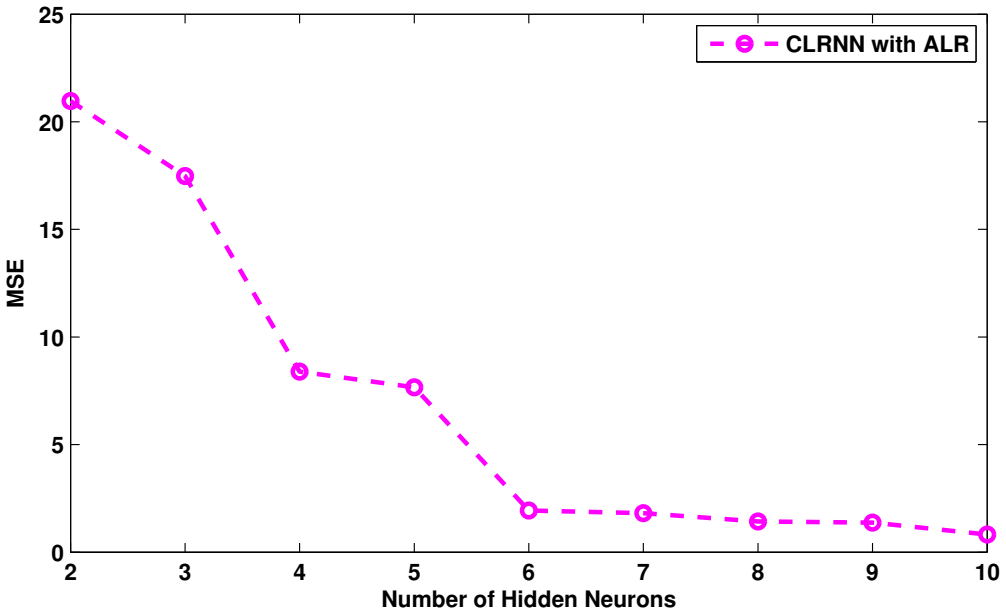


Figure 5.12: MSE curve over the addition of hidden neurons on CLRNN [Example-2]

Table 5.3: Tabulation of performance of CLRNN over selected structures [Example-2]

S.No	Algorithm	Network type	No of samples in each epoch	MSE	AMSE	RMSE
1	Traditional Fixed Learning rate	LRNN	1000	0.0087	0.2184	0.1786
2	Hybrid growing algorithm with ALR	CLRNN	1000	**0.0001**	**0.0774**	**0.060**
3	Hybrid growing algorithm with FLR	CLRNN	1000	0.0874	0.0624	0.0039

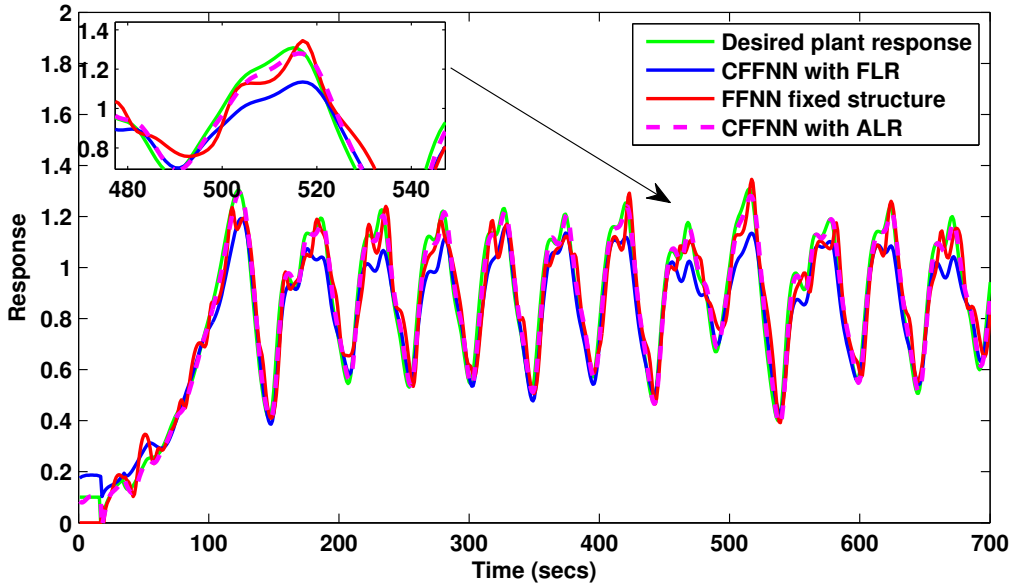


Figure 5.13: Response of CFFNN over selected FFNN structures [Example-2]

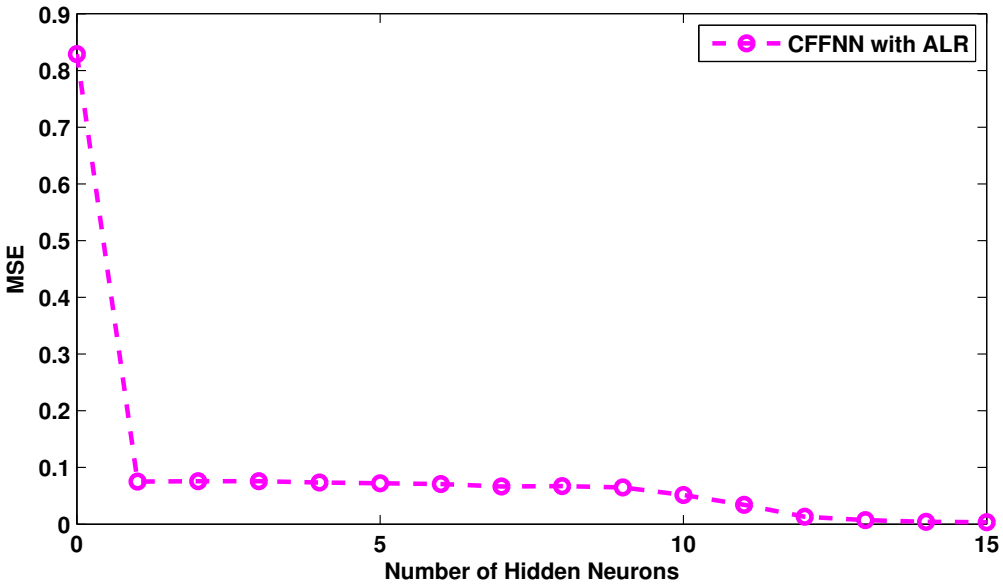


Figure 5.14: MSE curve over the addition of hidden neurons on CFFNN [Example-2]

Table 5.4: Tabulation of performance of CFFNN over selected structures [Example-2]

S.No	**Algorithm**	**Network type**	**No of samples in each epoch**	**MSE**	**AMSE**	**RMSE**
1	Traditional Fixed Learning rate	FFNN	1000	0.0012	0.0400	0.0346
2	Hybrid growing algorithm with ALR	CFFNN	1000	**0.001**	**0.0046**	**0.0027**
3	Hybrid growing algorithm with FLR	CFFNN	1000	0.0810	0.0066	0.0046

(ii) A novel adaptive learning rate based on fractional error is proposed. (iii) The weights are updated at every epoch, resulting in faster convergence. The resulting network is small enough, minimizing the computational complexity. The stability and convergence are also studied theoretically. Tables and simulation results show that CLRNN and CFFNN outperform fixed neural structures. Low RMSE and AMSE results indicate that the proposed algorithm may produce the best architecture for nonlinear system identification.

Chapter 6

Feedback-based optimization of FFNNs for modeling complex dynamic systems with APSOBP algorithm

6.1 Introduction

ANN has a wide range of applications in time series prediction, classification, regression problems, and so on [18], [123], [124]. Training of ANN for identification and control requires robust and efficient optimization methods. ANN structures are mostly trained using BP [125], and PSO [126]. BP is the simplest gradient-descent-based algorithm that updates weights in a neural network using the gradient of error. BP algorithms are simple, having fewer tunable parameters and a fast convergence. However, they have a few disadvantages, including being trapped in local optima for highly nonlinear non-differentiable problems. PSO is an optimization algorithm that replicates the behavior of a flock of birds or a school of fish to solve optimization problems. PSO algorithm is found to have many advantages over BP and evolutionary algorithms. PSO updates only two components of the particle (personal best and global best). In comparison to BP, PSO is more robust to noisy data and avoids local minima as it can handle multi-objective optimization effectively. Despite all these, traditional PSO algorithms may experience premature convergence, parameter sensitivity, and reduced exploration if the parameters are not properly set [127]. This has led to the development of many variations of the PSO algorithm in literature, all to improve the convergence and performance of the algorithm. Combining PSO with BP combines the strengths of both algorithms such

as the global and local search capabilities of PSO and BP while avoiding local minima. This hybrid technique is further improved by the adaptive PSO-BP algorithm, which dynamically modifies the PSO parameters and learning rate throughout the training phase. The adaptive PSO-BP algorithm further enhances this hybrid approach by dynamically adjusting the PSO parameters and learning rate during the training process. These adaptive mechanisms allow the algorithm to respond to varying problem complexities, ensuring faster convergence, improved stability, and better generalization performance. This chapter evaluates the adaptive PSO-BP methodology on diverse nonlinear dynamic system identification tasks, providing a comprehensive comparison with standard BP and static PSO-BP approaches. The results demonstrate the advantages of the adaptive PSO-BP algorithm in terms of convergence speed, accuracy, and robustness, establishing it as a powerful tool for training ANNs in complex, nonlinear domains.

6.2 Mathematical structure of FFNN

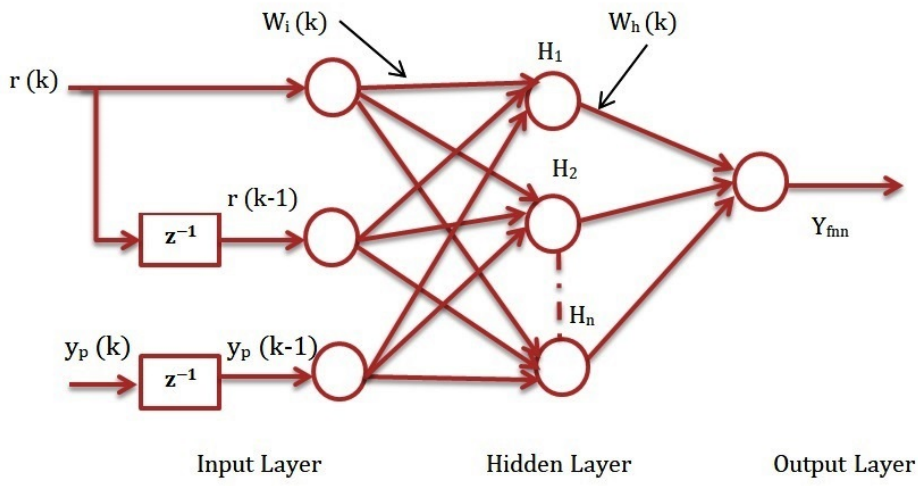


Figure 6.1: Feed Forward Neural Structure

This section describes the structure of FFNN and its learning using the BP algorithm. The structure of FFNN is as shown in Figure 6.1. FFNN is a forward propagating structure with information flowing in a single direction. In this work, three inputs are considered: $y_n(k-1), x(k), x(k-1)$. Let the input vector $U(k) = [y_n(k-1), x(k), x(k-1)]$. The input signals are multiplied with the input weight vector $W_i(k) = [w_{i1}(k), w_{i2}(k), ..., w_{id}(k)]$ before being sent to the hidden layer. The hidden layer receives these weighted input signals and is acted by a sigmoidal activation function. Let $H_n(k)$ denote the hidden layer vector with $H_n(k) =$

$[h_{n1}(k), h_n2(k), ..., h_{np}(k)]$ and hidden layer weight vector $W_h(k) = [w_{h1}(k), w_{h2}(k), ..., w_{hp}(k)]$. Further, the computed information is carried towards the output layer. A linear activation function is acted upon on the output layer. Let $y_{fnn}(k)$ denote the output of the network at k^{th} instant. The output of the FFNN at k^{th} instant is:

$$y_{fnn}(k) = f1\left(\sum_{l=1}^{p} H_l(k)W_{hl}(k) + b_{hl}(k)W_{ohl}(k)\right) \tag{6.1}$$

where $b_{hl}(k)$ represents the bias vector applied at the output and $W_{ohl}(k)$ denotes the corresponding output bias weight vector. f_1 indicates the linear activation function. $W_{hl}(k)$ represents the corresponding hidden weight vector. The output of the induced field at k^{th} instant is:

$$H_n(k) = g1\left(\sum_{n=1}^{d} U(k-n)W_{in}(k) + b_{nn}(k)W_{xnn}(k)\right) \tag{6.2}$$

where $b_{nn}(k)$ denotes input bias vector and $W_{xnn}(k)$ represents the corresponding input bias weight vector. g_1 denotes the sigmoidal activation function.

6.3 Training using BP algorithm

The training of FFNN is usually carried out using the BP algorithm. BP is a popular gradient-based algorithm. This method updates the neural network's weights using error gradients. The first step of training using the BP algorithm is to define a cost function. We have considered MSE as the objective function which is defined as the sum of squares of difference between actual and predicted output of the plant under observation. The objective function at k^{th} instant is given as:

$$E_i(k) = \frac{1}{2}[y_n(k) - y_{fnn}(k)]^2 \tag{6.3}$$

The modeling error is calculated at the output layer, and the gradients of weights from the output layer to the hidden layer are computed and propagated back till the input layer is reached.

6.3.1 Update equations for hidden layer weights

To update the hidden layer weight, calculate the change in gradient of error $E_i(k)$ with respect to hidden layer weight $W_h(k)$. It is calculated at k^{th} instant as:

$$\frac{\partial E_i(k)}{\partial W_h(k)} = \frac{\partial E_i(k)}{\partial y_{fnn}(k)} \times \frac{\partial y_{fnn}(k)}{\partial W_h(k)} \tag{6.4}$$

(or)

$$\frac{\partial E_i(k)}{\partial W_h(k)} = -e(k) \times H_n(k) \tag{6.5}$$

6.3.2 Update equations for input layer weights

To update the input layer weight, calculate the change in gradient of error $E_i(k)$ with respect to input layer weight $W_i(k)$ at as:

$$\frac{\partial E_i(k)}{\partial W_i(k)} = \frac{\partial E_i(k)}{\partial y_{fnn}(k)} \times \frac{\partial y_{fnn}(k)}{\partial H_n(k)} \times \frac{\partial H_n(k)}{\partial W_i(k)} \tag{6.6}$$

(or)

$$\frac{\partial E_i(k)}{\partial W_i(k)} = -e(k) \times W_h(k) \times (1 - H_n^2)(k) \times U(k) \tag{6.7}$$

To find new weights, the SGD approach is applied and is calculated as follows:

$$W_h(k)(new) = W_h(old) - \eta e(k) \frac{\partial E_i(k)}{\partial W_h(k)} \tag{6.8}$$

$$W_i(k)(new) = W_i(old) - \eta e(k) \frac{\partial E_i(k)}{\partial W_i(k)} \tag{6.9}$$

The learning rate is denoted by η, with a range of 0 to 1. In this work, a fixed learning rate is used to train the FFNN-APSOBP model.

6.4 Overview of PSO algorithm

PSP is a bio-inspired approach that is inspired by the behavior of a flock of birds or a school of fish as they search for food in groups. They are found to exchange their hunting experience and help other members of the group to find the best hunt. The algorithm works by randomly selecting a group of birds in search space. Every bird is referred to as a 'particle'. A population

or swarm can contain up to 'i' particles. Each particle in the swarm searches the N-dimensional space for the optimal solution. With every iteration, the particles adjust their velocity and position. In this way, the entire swarm is guided toward the location of the desired optimum solution in the search space. Consider a swarm of 'i' particles which has a current position p_i and current velocity v_i. They fly in N-dimensional search space until they find the global optimum solution. They adjust their velocity based on their best position is known as personal best (p_{best}) as well as the best solution of the entire swarm known as global best (g_{best}). This updates their position and moves them forward in N-dimensional space. The velocity and position update equation of classical PSO is:

$$v_i(k+1) = v_i(k) + c_1 r_1(k)(p_{best}(k) - p_i(k)) + c_2 r_2(k)(g_{best}(k) - p_i(k)) \tag{6.10}$$

The position is updated using the new calculated velocity in Eq.(6.10) as :

$$p_i(k+1) = p_i(k) + v_i(k+1) \tag{6.11}$$

Here c_1 and c_2 are the cognitive and social components respectively, and r_1 and r_2 are random numbers in the interval (0,1). When the problems are highly nonlinear, the classical PSO converges too early giving sub-optimal solutions. This has led to modification in the classical PSO algorithm. Shi and Eberhart introduced the APSO algorithm [128]. The Eq.(6.10) and Eq.(6.11) were modified with the addition of inertia weight to improve the convergence and performance of the optimization problems. The velocity and position equations of APSO are as below :

$$v_i(k+1) = w v_i(k) + c_1 r_1(k)(p_{best}(k) - p_i(k)) + c_2 r_2(k)(g_{best}(k) - p_i(k)) \tag{6.12}$$

$$p_i(k+1) = p_i(k) + v_i(k+1) \tag{6.13}$$

where $w = w_{max} - k\left(\frac{w_{max} - w_{min}}{k}\right)$ denotes the inertia weight. As per the APSO algorithm, w should begin with a large value at the initial stage and should be reduced gradually as the training progresses to attain a good convergence.

6.5 Problem statement

Let the input vector $X(k) = [x(k), x(k-1), ..., x(k-b)]$ and output vector $Y(k) = [y_n(k), y_n(k-1), ..., y_n(k-a)]$ of a nonlinear plant. The differential equation of the plant at k^{th} instant is :

$$y_n(k) = h[X(k), Y(k), k] \tag{6.14}$$

Where h is the nonlinear mapping function relating input to output. The current output $y_n(k)$ relies on both the current and previous values of the plant as well as external input. a, b denotes the number of past outputs and the number of past input values of the plant respectively. This work aims to identify the nonlinear function that satisfies $\hat{h} \simeq h$. This ensures that the identified model appropriately represents the plant. When the identifier is taken as FFNN, the structure is given by:

$$y_{fnn}(k) = \hat{h}[X(k), Y(k)] \tag{6.15}$$

A hybrid adaptive PSO-BP algorithm is used for training FFNN using MSE as the evaluation function. The evaluation function at k^{th} instant is given as:

$$MSE(k) = \frac{1}{2}[y_n(k) - y_{fnn}(k)]^2 \tag{6.16}$$

Here, $y_n(k) - y_{fnn}(k)$ is the modelling error. The fitness function is evaluated for every particle. PSO aims to minimize the fitness value by adjusting the weights. BP further fine-tunes the search process. The training continues until the identification error decreases to a minimum value or zero.

$$\lim_{k \to \infty} |y_n(k) - y_{fnn}(k)| \leq \epsilon \tag{6.17}$$

where $\epsilon \to 0$.

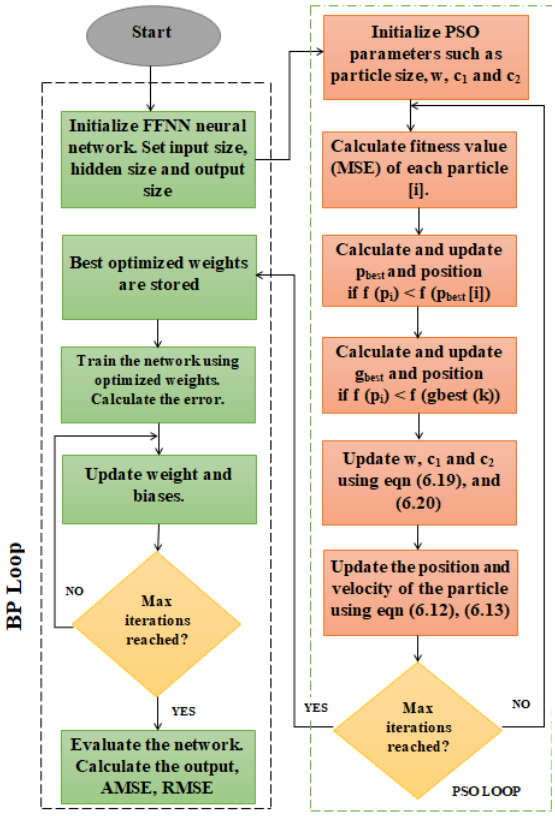


Figure 6.2: Steps to train FFNN network using hybrid adaptive PSO-BP algorithm

6.6 Optimising FFNN using proposed adaptive PSO-BP (APSOBP) algorithm

This section discusses how to train FFNN using the proposed APSOBP algorithm. The proposed algorithm combines the advantages of both PSO (global search) and BP (local search). In this paper, we present a novel inertia weight approach that decreases dynamically throughout the optimization process. Inertia weight is crucial for improving the performance and convergence of PSO The performance status of every particle is a key to adjusting inertia weight and hyperparameters of PSO. If a particle's best position is near its prior position, it is getting close to the global optimum. In this case, an increase in inertia weight is required to keep the particle near the global optimum. Alternatively, if the particle's best position is significantly different from its prior best position, it is moving away from the global optimum. This necessitates a decrease in inertia weight to encourage more exploration of new paths rather than staying with a less effective solution. Based on this concept, we developed a method for calculating inertia weight using feedback from the global best position of the swarm. A performance index e_i is calculated by finding the difference between global best positions over consecutive iterations.

The e_i is calculated as

$$e_i(k) = (g_{best}(k) - g_{best}(k-1)) \tag{6.18}$$

The performance-index e_i is used to derive inertia weight as follows:

$$w_k(k) = w_{min} + \frac{e_i(k)}{1 + |e_i(k)|} \tag{6.19}$$

Here, g_{best} denotes the global best position at iteration k, and w_{min} denotes the minimum inertia weight. Further, the hyperparameters are made dynamic to adapt to changes in the optimization landscape of the problem. c_1 and c_2 are dynamically adjusted based on the error metric e_i in comparison to a fixed threshold. The adaptive method is as follows:

$$\begin{cases} c_1 = c_1 + \alpha \\ c_2 = c_2 + \alpha \end{cases} \quad \text{if } e_i < \text{threshold}$$
$$\begin{cases} c_1 = c_1 - \alpha \\ c_2 = c_2 - \alpha \end{cases} \quad \text{if } e_i > \text{threshold} \tag{6.20}$$

Where α is a constant that determines the rate of adjustment. The value of α is kept constant at 0.1. By dynamically adjusting the inertia weight and hyperparameters based on particle performance, the APSOBP algorithm effectively balances exploration and exploitation, leading to improved convergence and performance in training the FFNN. The flowchart of FFNN training using APSOBP is shown in Figure 6.2. The detailed steps are described below:

1. The algorithm starts with the initializing of FFNN architecture. For simplicity, the 3-10-1 FFNN structure is considered in this work.

2. The next step is to set the PSO parameters such as swarm size, inertia coefficient w, hyperparameters c_1, and c_2, velocity, and position.

3. The algorithm enters the PSO loop (shown inside the green dotted box in Figure 6.2). After every iteration, the fitness value of every particle $f(p_i)$ is calculated. MSE is considered as the fitness function. The particle moves forward in the search space in search of the optimum solution and updates its personal best p_{best} and position p_{best_i} if the current fitness value $f(p_i)$ is less than the particle's fitness value $f(p_{best})$. Further, the global best g_{best} and its position $g_{best}(k)$ of the swarm is updated if the current fitness

value $f(p_i)$ is less than the swarm's $f(g_{best}(k))$.

4. The inertia weights, cognitive component c_1, and social component c_2 are dynamically changed using the inertia weight method in Eq.(6.18) and Eq.(6.19). Furthermore, the hyperparameters c_1 and c_2 are dynamically modified based on Eq.(6.20).

5. The velocity and position get updated using the new inertia weight and hyperparameters.

6. The process of PSO continues until the termination criteria are met. The stopping criteria for PSO is considered as the reach of maximum iterations in our work.

7. Further the optimized weights from PSO training are sent to the BP algorithm for further local tuning of the search space. Many times BP faces the problem of producing sub-optimal solutions for highly non-convex nonlinear problems. One of the reasons is due to improper initialization of the weights at the start of BP. This problem is solved in this work, as PSO provides a good set of initial weights, and BP has a better chance to converge to a high global solution.

8. Further BP, fine-tunes the search locally around the global optimum. The algorithm now enters the BP loop (shown inside the black dotted box in Figure 6.2).

9. The network searches locally using the BP algorithm. A constant learning rate β is considered. The value of β is taken as 0.001. The identification error is calculated at every epoch and the weights and biases are updated at every epoch until the termination condition is met.

10. Finally, the network is validated and the performance of FFNN-APSOBP is evaluated in terms of AMSE and RMSE. The BP algorithm is usually stopped with the termination criteria. In this work, the reach of maximum iterations is considered as a stopping criterion.

The use of Eq.(6.18), Eq.(6.19), and Eq.(6.20) is intended to provide better optimum solutions than classical PSO and BP algorithms.

6.7 Simulation examples

To illustrate the efficiency of the FFNN-APSOBP approach, three examples of nonlinear benchmark problems are used. The obtained results are compared to the traditional FFNN-BP and

FFNN-PSO structures.

6.7.1 Selection of parameter

To study the effectiveness of the proposed approach, 10 neurons in the hidden layer are considered for FFNN-APSOBP, FFNN-PSO, and FFNN-BP structures. A constant learning rate of 0.001 is considered to simulate the FFNN-BP and FFNN-ABPPSO structure. For the FFNN-PSO and FFNN-APSOBP structures, the particle size is set to 30 and is randomly created. The maximum number of iterations to train PSO-BP is 700 for all three structures. The inertia weight, c_1 and c_2 are made dynamic using Eq.(6.19) and Eq.(6.20).

6.7.2 Case-1: Modelling of nonlinear plant equation with degree 3

Consider a non-linear plant described by the differential equation as given in [36]:

$$y_n(k) = \frac{y_n(k-1)}{1+y_n^2(k-2)} + x^3(k-2) \tag{6.21}$$

The dynamic behavior of the plant relies on both current and previous inputs and outputs. The identification model of the plant in Eq.(6.21) is:

$$y_n(k) = a[y_n(k-1), y_n(k-2), x(k-2)] \tag{6.22}$$

The plant is supplied with the following input signal as $x(k) = sin(\frac{2\pi k}{40})$. Now, with FFNN as the identifier, the structure becomes :

$$y_{fnn}(k) = \hat{a}[y_n(k-1), u(k), u(k-1)] \tag{6.23}$$

A total of 700 samples are taken for training and another 700 samples are considered for validation. Figure 6.5 shows the variation of inertia weights over several epochs. From Figure 6.5, it can be seen that w varies from 0.7 to 0.4. The system assures stability if the value of w does not exceed 1. The comparison of the response of the plant using FFNN-APSOBP, FFNN-PSO, and FFNN-BP approach is shown in Figure 6.3. Figure 6.4 compares the MSE curves obtained using the FFNN-APSOBP, FFNN-PSO, and FFNN-BP approaches across epochs. The efficiency of the FFNN-APSOBP approach is evaluated using performance indexes such as AMSE, and RMSE. From the results of Figure 6.3, 6.4 and Table 6.1, it is observed that

the performance of the training is as follows: FFNN-APSOBP > FFNN-PSO > FFNN-BP.

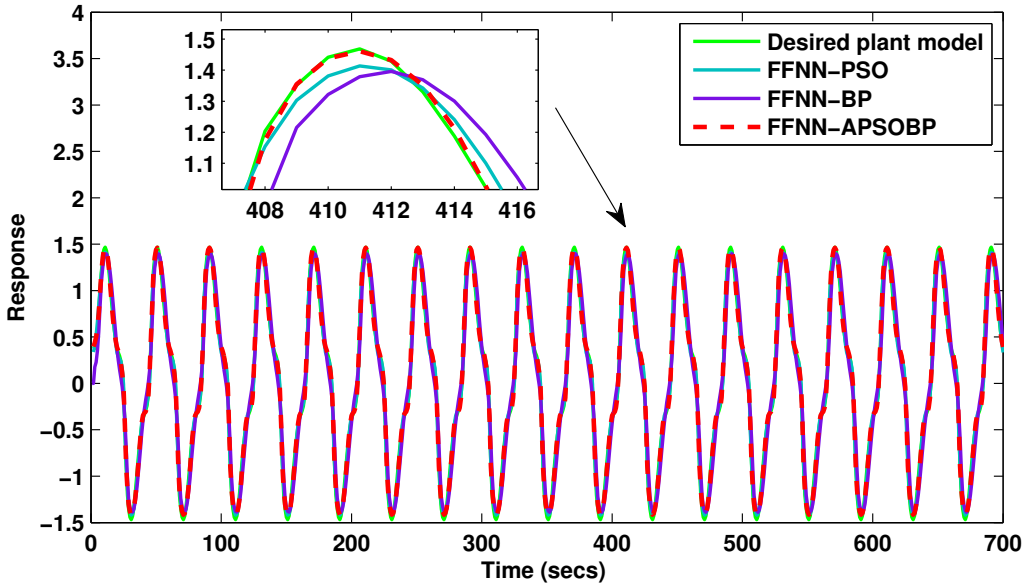


Figure 6.3: Comparison of the response of training FFNN with different algorithms [Case-1]

Table 6.1: Comparison of performance of training FFNN with different algorithms [Case-1]

S.No	Algorithm	Network type	Samples per epoch	MSE	AMSE	RMSE
1	BP	FFNN	700	0.0619	0.3052	0.2487
2	APSOBP	FFNN	700	**0.0004**	**0.0082**	**0.0213**
3	PSO	FFNN	700	0.0258	0.0268	0.1463

6.7.3 Noise tolerance test [Case-1]

To investigate the effectiveness and adaptability of the novel FFNN-APSOBP approach, a sudden disturbance $dis(k)$ is added to the plant in the period $250 < k < 450$. A disturbance in the form of a sine wave $dis(k) = sin(\frac{2\pi k}{15})$ is added in the interval $250 < k < 450$. Figure 6.6 shows the response of the FFNN-APSOBP approach over the disturbance. It can be seen from the figure that FFNN-APSOBP initially diverted from the desired response, thus increasing MSE values. With training the MSE has become zero thus tracking the desired performance.

Analysis: Apart from the above observations, Figure 6.5 illustrates that the algorithm starts with encouraging exploration and once the promising areas are explored, w reduces to a lower value (0.7 to 0.4), allowing the particles to use the best-known solutions. Additionally, a slight increase in inertia coefficients after a few iterations signifies the algorithm's ability to escape early convergence, allowing for further exploration in search space. The maintenance

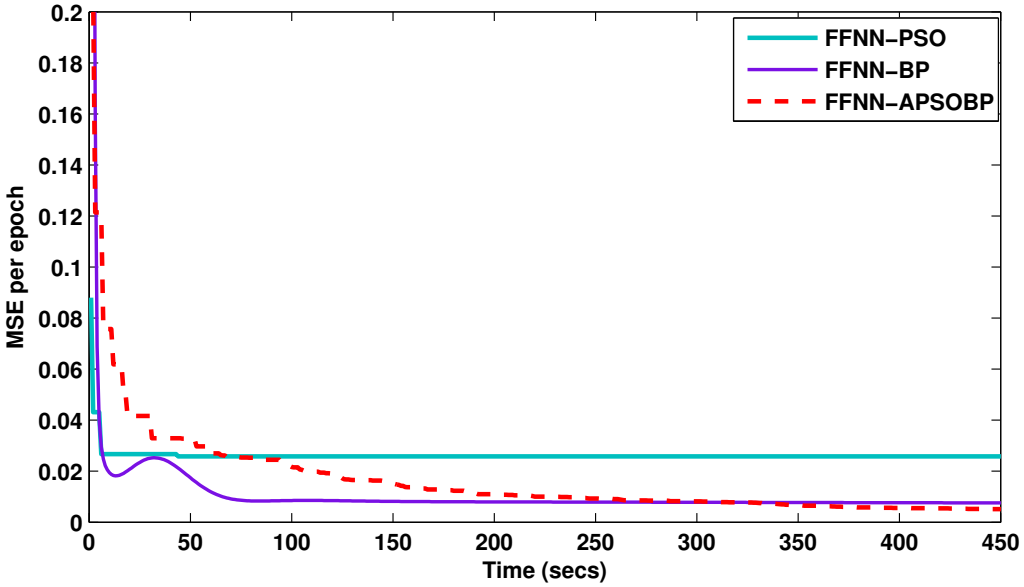


Figure 6.4: Comparison of the effect of MSE on training FFNN with different algorithms [Case-1]

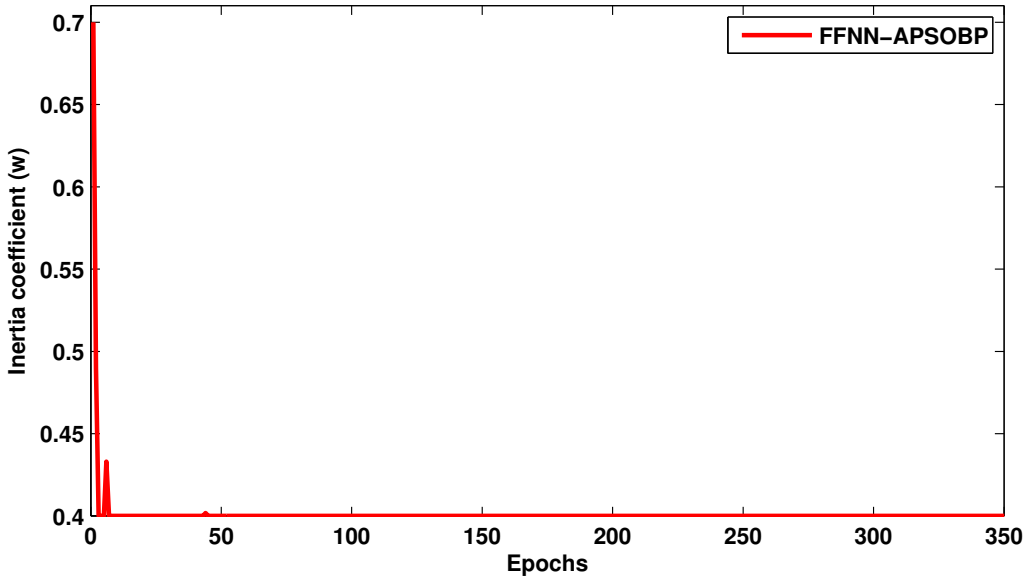


Figure 6.5: Variation of Inertia weight over epochs on FFNN-APSOBP model [Case-1]

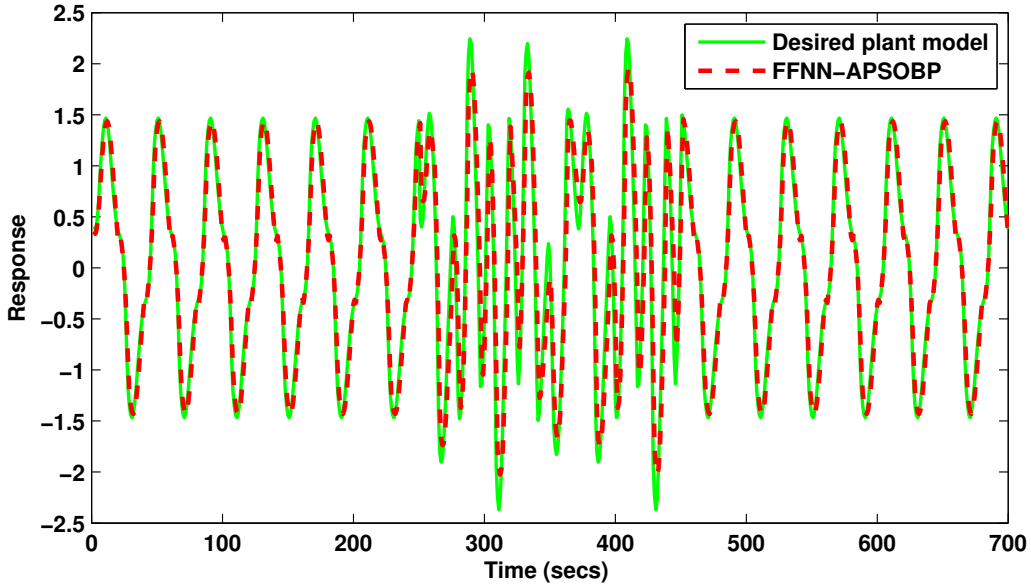


Figure 6.6: Effect of disturbance on FFNN-APSOBP structure [Case-1]

of this lower inertia weight after a few iterations indicates the algorithm's convergence for this problem.

6.7.4 Case-2 : Identification of liquid level system

A model of the liquid level system is considered for identification as given in [129]. The system comprises a DC water pump. The pump feeds a conical flask which then feeds a square tank, resulting in second-order dynamics. The input voltage of the motor is the controlled variable and the level of the water in the conical flask is taken as the output variable. The differential equation of the liquid level system to be identified is given by:

$$\begin{aligned} y_n(k+1) = {} & 0.9722y_n(k) + 0.3578x(k) - 0.1295x(k-1) - 0.3103y_n(k)x(k) - 0.04228y_n^2(k-1) \\ & +0.1663y_n(k-1)x(k-1) - 0.03259y_n^2(k)y_n(k-1) - 0.3513y_n^2(k)x(k-1) \\ & +0.3084y_n(k)y_n(k-1)x(k-1) + 0.1087y_n(k-1)x(k)x(k-1) + 0.2573y_n^2(k-1)e(k) \\ & +0.2939y_n^2(k-1)e(k) + 0.4770y_n(k-1)x(k)e(k) \end{aligned} \tag{6.24}$$

where $e(k)$ is the white noise in the range [-0.025,0.025] and $x(k)$ denotes the input applied to the plant and is taken as $x(k) = 0.4sin\left(\frac{2\pi k}{250}\right) + 0.6cos\left(\frac{2\pi k}{25}\right)$.

The plant takes the following identification structure:

$$y_n(k+1) = h[y_n(k), y_n(k-1), x(k), x(k-1)] \tag{6.25}$$

When FFNN is considered an identifying model, then it takes the following structure:

$$y_{fnn}(k+1) = \hat{h}[y_n(k), x(k), x(k-1)] \tag{6.26}$$

A total of 700 samples are taken for training and another 700 samples are considered for validation. Figure 6.9 shows the variation of inertia weights over several epochs. From the result, it can be seen that w varies from 0.9 to 0.4. The system assures stability if the value of w does not exceed 1. The comparison of the response of the plant using FFNN-APSOBP, FFNN-PSO, and FFNN-BP approach is shown in Figure 6.7. Figure 6.8 shows the comparison of the MSE curve of the plant using FFNN-APSOBP, FFNN-PSO, and FFNN-BP approach. The efficiency of the FFNN-APSOBP approach is evaluated using indexes such as AMSE, and RMSE. From the simulation results of Figure 6.7, 6.8 and Table 6.2, it is observed that the performance of the training is as follows: FFNN-APSOBP > FFNN-PSO > FFNN-BP.

Table 6.2: Comparison of performance of training FFNN with different algorithms [Case-2]

S.No	Algorithm	Network type	Samples per epoch	MSE	AMSE	RMSE
1	BP	FFNN	700	0.2793	0.1794	0.4124
2	APSOBP	FFNN	700	**0.0014**	**0.0032**	**0.0039**
3	PSO	FFNN	700	0.0481	0.0481	0.1612

6.7.5 Noise tolerance test [Case-2]

To investigate the effectiveness and adaptability of the novel FFNN-APSOBP model, a sudden disturbance $dis(k)$ is added to the plant in the period $250 < k < 450$. A disturbance in the form of a sine wave $dis(k) = sin(\frac{2\pi k}{15})$ is added in the interval $250 < k < 450$. Figure 6.10 shows the response of FFNN-APSOBP over the disturbance. It can be seen from the figure that FFNN-APSOBP initially diverted from the desired response, thus increasing MSE values. With training the MSE has become zero thus tracking the desired performance.

Analysis: Apart from the above observations, Figure 6.9 illustrates that the algorithm starts with encouraging exploration in systems with many local minima (increasing w from 0.7 to 0.9). Once the promising areas are explored, w reduces to a lower value (0.7 to 0.4), allowing the

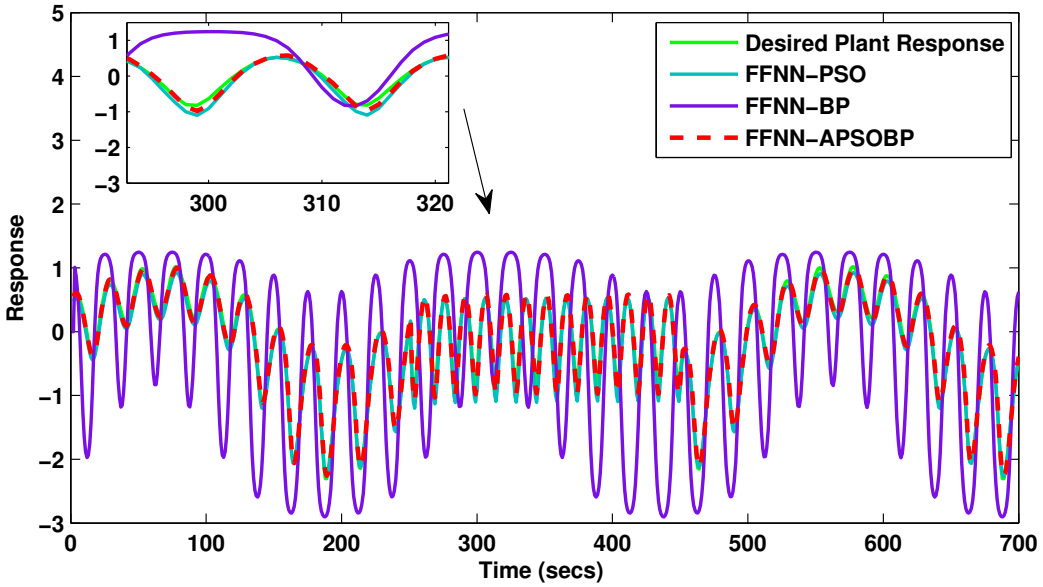


Figure 6.7: Comparison of the response of training FFNN with different algorithms [Case-2]

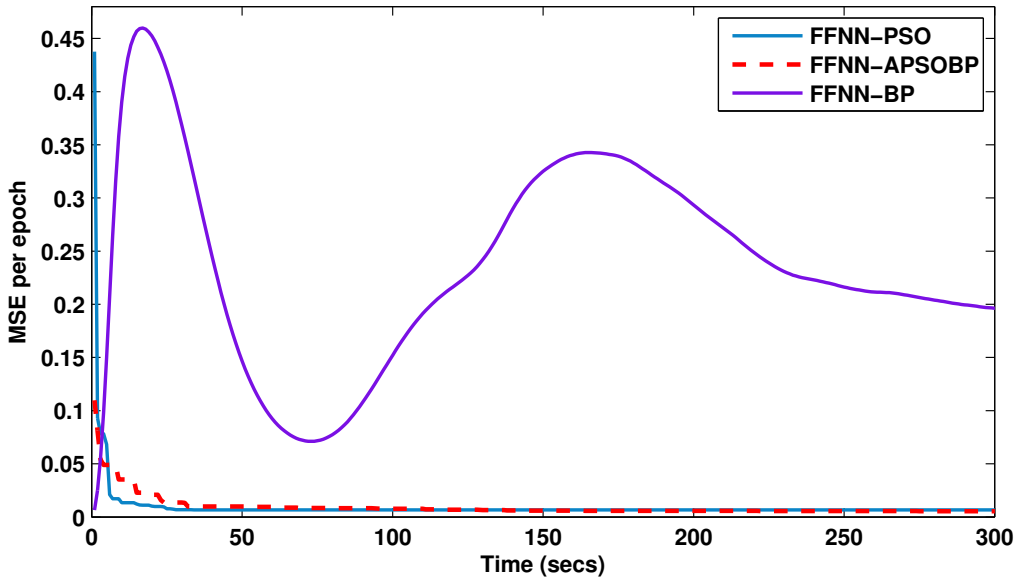


Figure 6.8: Comparison of the effect of MSE of training FFNN with different algorithms [Case-2]

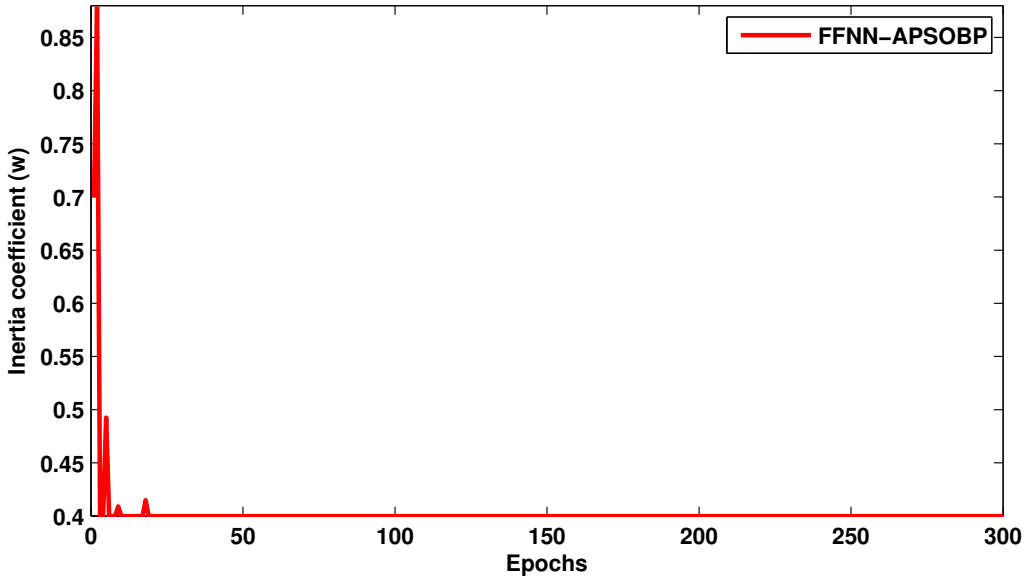


Figure 6.9: Variation of inertia weight over epochs on FFNN-APSOBP model [Case-2]

particles to use the best-known solutions. Additionally, a slight increase in inertia coefficients after a few iterations signifies the algorithm's ability to escape early convergence, allowing for further exploration in search space. The maintenance of this lower inertia weight after a few iterations indicates the algorithm's convergence for this problem.

6.7.6 Case-3 : Modelling of Mackey glass series

The suggested approach was tested on a nonlinear Mackey glass time series prediction problem as described in [130]. The prediction of the Mackey glass series is given by:

$$\frac{dy_n(t)}{dt} = -\gamma \times y_n(t) + \frac{\beta \times y_n(t-\tau)}{1 + y_n^{10}(t-\tau)} \tag{6.27}$$

The problem is solved using the parameter values $\beta = 0.2$ and $\gamma = 0.1$. The problem exhibits chaotic behavior when $\tau \geq 17$. Hence the time delay is chosen as $\tau = 17$. The corresponding differential equation of Eq.(6.27) is as follows:

$$y_n(k) = -\gamma \times y_n(k) + \frac{\beta \times y_n(k-\tau)}{1 + y_n^{10}(k-\tau)} \tag{6.28}$$

The structure of identification for the plant is:

$$y_n(k) = h[y_n(k), y_n(k-\tau))] \tag{6.29}$$

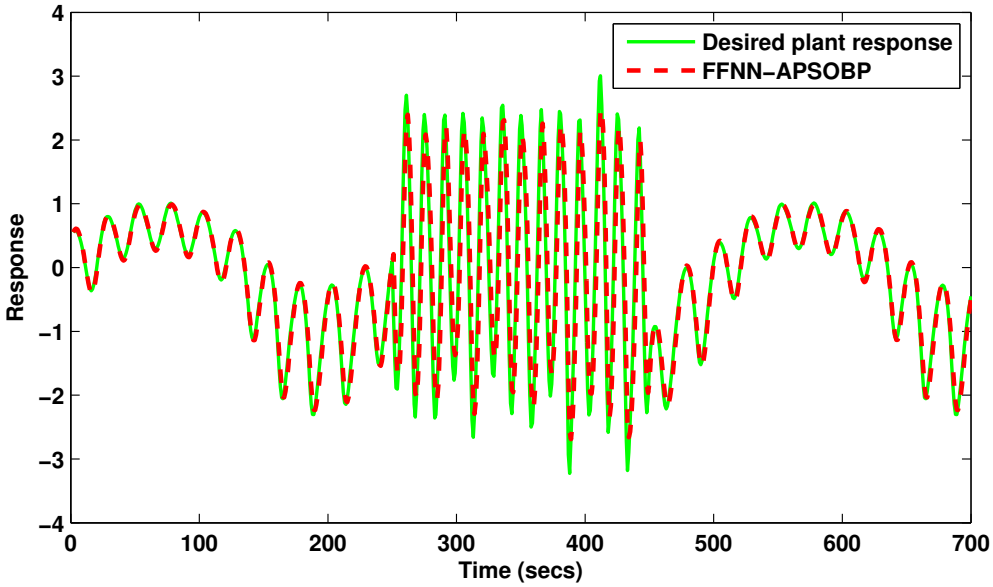


Figure 6.10: Effect of disturbance on FFNN-APSOBP structure [Case-2]

When FFNN becomes the identifier, the identification structure is:

$$y_{fnn}(k+1) = \hat{h}[y_n(k), y_n(k-\tau)] \tag{6.30}$$

A total of 700 samples are taken for training and another 700 samples are considered for validation. Figure 6.13 shows the variation of inertia weights over several epochs. From the figure, it can be seen that w varies from 0.7 to 0.4. The system assures stability if the value of w does not exceed 1. Figure 6.11 compares the plant's response using the FFNN-APSOBP, FFNN-PSO, and FFNN-BP approaches. Figure 6.12 compares the MSE curve obtained for the plant using FFNN-APSOBP, FFNN-PSO, and FFNN-BP approach. The efficiency of the FFNN-APSOBP is evaluated using evaluation indexes such as AMSE, and RMSE. From the results of Figure 6.11, 6.12 and Table 6.3, it is observed that the performance of the training is as follows: FFNN-APSOBP > FFNN-PSO > FFNN-BP.

Table 6.3: Comparison of training of performance of FFNN with different algorithms [Case-3]

S.No]	Algorithm	Network type	Samples per epoch	MSE	AMSE	RMSE
1	BP	FFNN	700	0.0087	0.1146	0.0930
2	APSOBP	FFNN	700	**0.0017**	**0.0018**	**0.033**
3	PSO	FFNN	700	0.0230	0.0231	0.0677

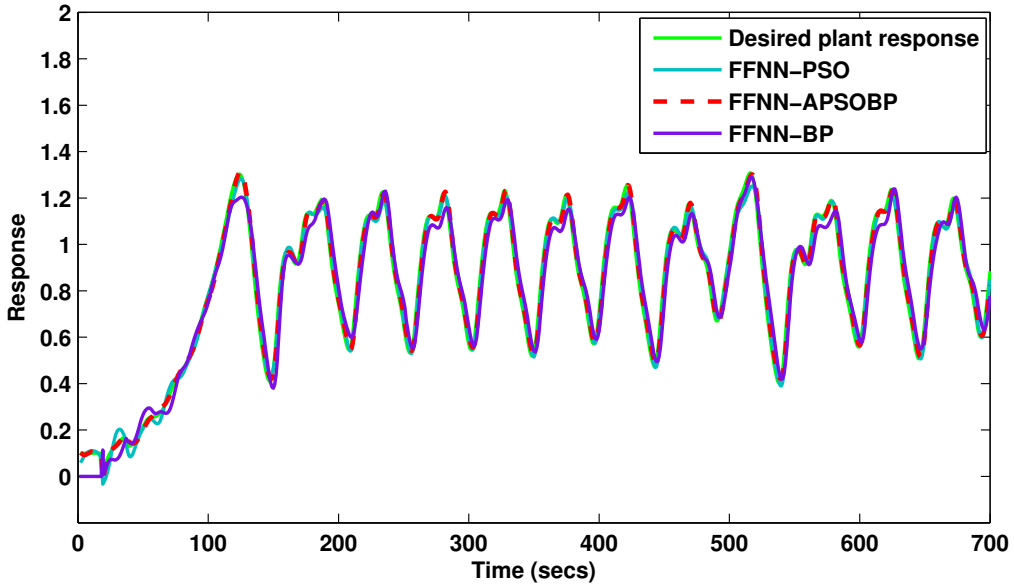


Figure 6.11: Comparison of the response of training FFNN with different algorithms [Case-3]

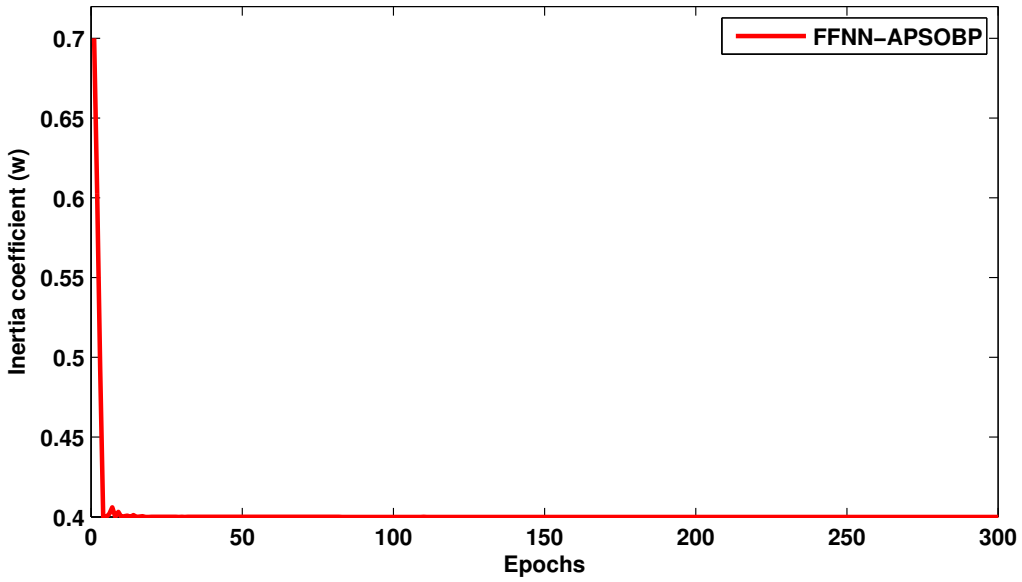


Figure 6.12: Comparison of the effect of MSE on training FFNN with different algorithms [Case-3]

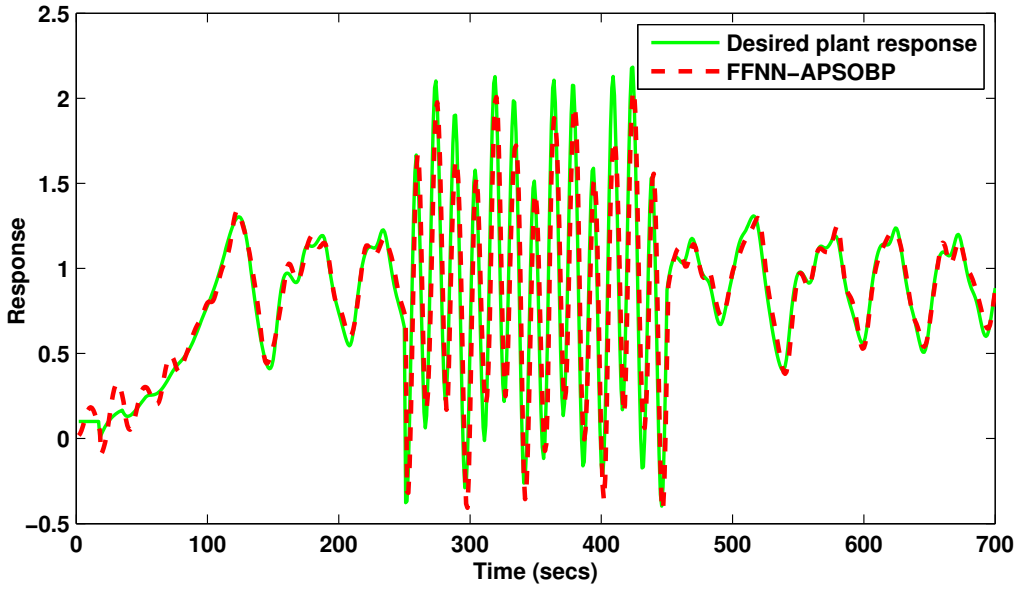


Figure 6.13: Variation of Inertia weight over epochs [Case-3]

6.7.7 Noise tolerance test [Case-3]

To investigate the effectiveness and adaptability of the novel FFNN-APSOBP model, a sudden disturbance $dis(k)$ is added to the plant in the period $250 < k < 450$. A sine wave as disturbance $dis(k) = sin(\frac{2\pi k}{15})$ is added in the interval $250 < k < 450$. Figure 6.14 shows the response of FFNN-APSOBP over the disturbance. It can be seen from the figure that FFNN-APSOBP initially diverted from the desired response, thus increasing MSE values. With training the MSE has become zero thus tracking the desired performance.

Analysis: Apart from the above observations, Figure 6.13 illustrates that the algorithm starts with encouraging exploration and once the promising areas are explored, w reduces to a lower value (0.7 to 0.4), allowing the particles to use the best-known solutions. Additionally, a slight increase in inertia coefficients after a few iterations signifies the algorithm's ability to escape early convergence, allowing for further exploration in search space. The maintenance of this lower inertia weight after a few iterations indicates the algorithm's convergence for this problem.

6.8 Conclusion

In this work, a novel APSOBP algorithm for training FFNN to identify complex nonlinear systems is proposed. The algorithm incorporates the abilities of both particle swarm optimization

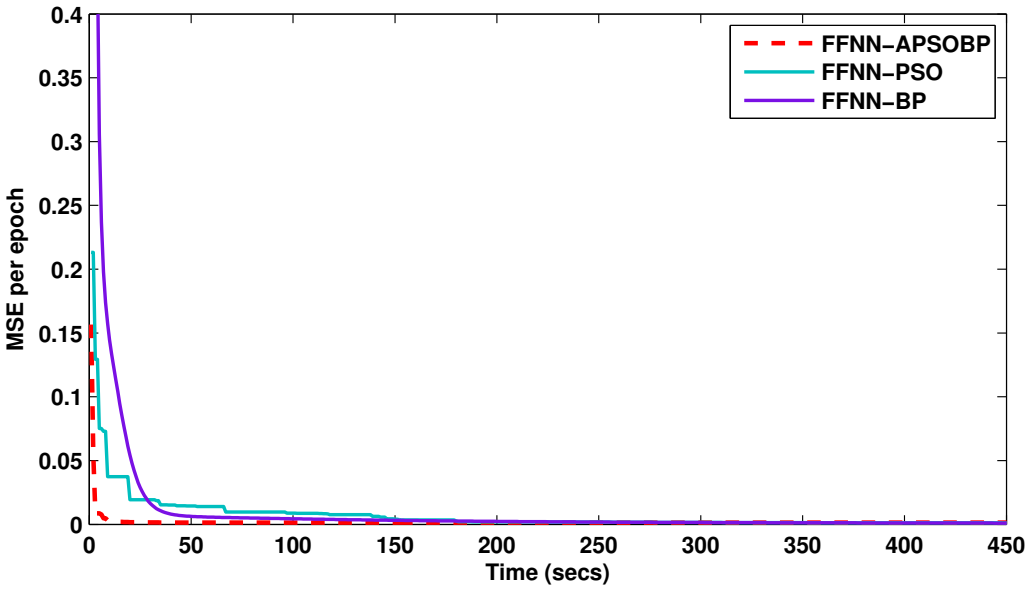


Figure 6.14: Effect of disturbance on FFNN-APSOBP structure [Case-3]

and backpropagation algorithm. The weights of the FFNN are trained using the APSOBP algorithm. PSO optimizes the weights at the initial stage, which are further fine-tuned by BP around the optimum solution. Furthermore, to avoid the PSO algorithm's early convergence, we made the inertia coefficient and hyperparameters dynamic using the feedback mechanism that computes the difference between the fitness value of global best in consecutive iterations. The proposed inertia weight is observed to decrease nonlinearly with epochs ensuring adequate exploration and exploitation of the PSO. The proposed method also ensures network stability as the w ranges between 0 to 1 for all the selected examples. The structure's performance is evaluated using three examples of complex nonlinear problems and a sudden disturbance. The results show that training FFNN using the APSOBP algorithm outperforms FFNN-BP and FFNN-PSO approaches in terms of efficiency and robustness.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

The main aim of this thesis is to develop novel approaches for the identification and adaptive control of nonlinear dynamic systems, with a focus on effectively capturing their temporal dynamics. To achieve this, we have proposed two novel modifications to the existing RNN structures. Both the proposed RNN structures are applied to identify the nonlinear system. They are independent of the plant order and involve memory neurons to handle the complexity of the nonlinear dynamics. The developed architecture is further utilized to design adaptive controllers for nonlinear systems. The BP algorithm, being the simplest learning algorithm for ANN, is used to derive weight update equations. While ANN demonstrates robust performance, their effectiveness depends on factors such as the network architecture and the learning algorithm. To optimize the structure of ANNs, we have developed a constructive algorithm for FFNN and RNN, allowing them to dynamically grow based on their performance on the validation data. Despite their simplicity, the BP algorithm faces limitations such as getting trapped in local minima for highly complex spaces. To address this, we proposed an adaptive learning rate method to accelerate BP and developed a novel adaptive PSO-BP algorithm to enhance ANN performance. These methods are validated for handling system uncertainties, including parameter variations and disturbance signals and are validated against state-of-the-art neural network structures to demonstrate their effectiveness.

In Chapter 2, a novel HEJNN structure is developed and applied for the identification of nonlinear dynamic systems. BP algorithm is used to derive the weight update equations of the proposed model. The stability of the proposed approach is proven in the sense of Lyapunov-

stability analysis. The performance of HEJNN structure is evaluated by comparing it with the results obtained from other ANNs such as ENN, JNN, FFNN,and LRNN. Experimental results show that the HEJNN model outperforms the other neural models in terms of identification accuracy and robustness.

In chapter 3, an another novel modification of RNN referred as CRFNN is proposed. This architecture combines the strengths of FFNN and LRNN. The BP algorithm is used to derive the weight update equations and the Lyapunov-stability principles are applied to test the stability of the proposed framework. MISO systems (various degrees of plant equation) and Mackey glass series benchmark problem are considered and the performance of the proposed structure is evaluated by comparing it with the results obtained from other popular neural network models such as FFNN, ENN, JNN, and LRNN. The results of the simulation demonstrate that CRFNN shows greater prediction accuracy, and better performance even in the scenario of disturbance signals and parameter variation.

In Chapter 4, a simultaneous online identification and adaptive control framework for simultaneous identification and adaptive control of nonlinear dynamical systems is proposed. The scheme is referred as HFRNN and a gradient-descent-based BP algorithm is used to derive weight update equation. The stability of the proposed learning strategy is proven using the Lyapunov stability principles. The results of HFRNN are compared with the JNC and LRNC. The results demonstrate that HFRNN performs satisfactorily, even in the presence of disturbance signals.

In Chapter 5, we have developed a novel hybrid constructive algorithm to dynamically grow FFNN and RNN structures. The proposed algorithm is applied to identify non-linear dynamical systems. The constructive algorithm for FFNN is denoted as CFFNN and for RNN is CLRNN. The hidden nodes are grown depending on the effect of mean square error (MSE) on the validation dataset. To enhance the learning algorithm's performance, a novel Lyapunov's stability-based ALR is also developed. The experimental results show that the CLRNN and CFFNN with ALR outperform the other selected neural models.

In chapter 6, a novel APSOBP for training ANN is developed and is applied to FFNN to identify nonlinear dynamical systems. This method integrates the strengths of PSO mainly its good global search ability with the high convergence speed of BP. To avoid early convergence, the PSO parameters such as inertia weight (w) and hyperparameters (c_1, c_2) are also dynamically adapted. The proposed algorithm is evaluated against three benchmark nonlinear

problems to ensure its effectiveness. The results show APSOBP algorithm outperforms the selected methods in terms of convergence, accuracy, and robustness.

7.2 Future scope

Future studies will explore the integration of reinforcement learning to enhance control strategies and learning of ANN. The other studies would include exploring techniques to improve the computational efficiency of adaptive control algorithms, enabling faster processing in real-time applications. Adaptive learning rate and PSO-BP algorithms developed in this thesis can also be further enhanced by incorporating unsupervised learning approaches or another advanced evolutionary algorithm to develop more versatile learning approaches.

List of Publications

Journal Publications [Published/Communicated]:

1. Shobana, R., Bhavnesh Jaint, and Rajesh Kumar. "Design of a novel robust recurrent neural network for the identification of complex nonlinear dynamical systems." **Soft Computing** 28, no. 3 (2024): 2737-2751. https://doi.org/10.1007/s00500-023-09187-5. Published.

2. Shobana, R., Rajesh Kumar, and Bhavnesh Jaint. "A recurrent neural network-based identification of complex nonlinear dynamical systems: a novel structure, stability analysis and a comparative study." **Soft Computing** (2023): 1-17.https://doi.org/10.1007/s00500-023-09390-4. Published.

3. Shobana, R., Rajesh Kumar, and Bhavnesh Jaint. "Nonlinear dynamical system approximation and adaptive control based on hybrid-feed-forward recurrent neural network: Simulation and stability analysis." **Expert Systems** (2024): e13619. https://doi.org/10.1111/exsy.13619. Published.

4. Shobana, R, Rajesh Kumar and Bhavnesh Jaint. "A hybrid constructive training of FFNN and DRNN using adaptive learning rate for nonlinear system identification". Communicated to SCI journal [under review].

5. Shobana, R, Rajesh Kumar and Bhavnesh Jaint. "Feedback-based optimization of FFNNs for modeling complex dynamic systems with APSOBP algorithm". Communicated to SCI journal [under review].

Conference Publications [Published]:

1. R. Shobana., R. Kumar and B. Jaint, "A novel locally connected recurrent neural network for identification of nonlinear dynamical system," 2023 10th International Conference on

Signal Processing and Integrated Networks (SPIN), Noida, India, 2023, pp. 828-833, doi: 10.1109/SPIN57001.2023.10117315.

2. R. Shobana, R. Kumar and B. Jaint, "An Input Context Layered Local Recurrent Neural Network for Nonlinear System Identification," 2024 IEEE Third International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), Delhi, India, 2024, pp. 430-435, doi: 10.1109/ICPEICES62430.2024.10719117.

References

[1] G. Chen and J. L. Moiola, "An overview of bifurcation, chaos and nonlinear dynamics in control systems," *Journal of the Franklin Institute*, vol. 331, no. 6, pp. 819–858, 1994.

[2] R. C. Hilborn, *Chaos and nonlinear dynamics: an introduction for scientists and engineers*. Oxford university press, 2000.

[3] L. Ljung, "Perspectives on system identification," *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

[4] S. Haykin, *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

[5] A. K. Tangirala, *Principles of system identification: theory and practice*. CRC press, 2018.

[6] D. P. Moeller, "Parameter identification of dynamic systems," in *Mathematical and Computational Modeling and Simulation*. Springer, 2004, pp. 257–310.

[7] A. Kroll and H. Schulte, "Benchmark problems for nonlinear system identification and control using soft computing methods: Need and overview," *Applied Soft Computing*, vol. 25, pp. 496–513, 2014.

[8] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm," *Computers & structures*, vol. 169, pp. 1–12, 2016.

[9] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Systems Magazine*, vol. 39, no. 6, pp. 28–99, 2019.

[10] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön, "Deep learning and system identification," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1175–1181, 2020.

[11] Y. Li, Z. O'Neill, L. Zhang, J. Chen, P. Im, and J. DeGraw, "Grey-box modeling and application for building energy simulations-a critical review," *Renewable and Sustainable Energy Reviews*, vol. 146, p. 111174, 2021.

[12] I. J. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear systems part i: deterministic non-linear systems," *International journal of control*, vol. 41, no. 2, pp. 303–328, 1985.

[13] O. Calin, *Deep learning architectures*. Springer, 2020.

[14] M. A. Mohammed, B. Al-Khateeb, A. N. Rashid, D. A. Ibrahim, M. K. Abd Ghani, and S. A. Mostafa, "Neural network and multi-fractal dimension features for breast cancer classification from ultrasound images," *Computers & Electrical Engineering*, vol. 70, pp. 871–882, 2018.

[15] S. A. Mostafa, A. Mustapha, M. A. Mohammed, R. I. Hamed, N. Arunkumar, M. K. Abd Ghani, M. M. Jaber, and S. H. Khaleefah, "Examining multiple feature evaluation and classification methods for improving the diagnosis of parkinson's disease," *Cognitive Systems Research*, vol. 54, pp. 90–99, 2019.

[16] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[17] Z. Pezeshki and S. M. Mazinani, "Comparison of artificial neural networks, fuzzy logic and neuro fuzzy for predicting optimization of building thermal consumption: a survey," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 495–525, 2019.

[18] G. Quaranta, W. Lacarbonara, and S. F. Masri, "A review on computational intelligence for identification of nonlinear dynamical systems," *Nonlinear Dynamics*, vol. 99, no. 2, pp. 1709–1761, 2020.

[19] R. Kruse, S. Mostaghim, C. Borgelt, C. Braune, and M. Steinbrecher, "Multi-layer perceptrons," in *Computational intelligence: a methodological introduction*. Springer, 2022, pp. 53–124.

[20] Q. Yu, Z. Hou, X. Bu, and Q. Yu, "Rbfnn-based data-driven predictive iterative learning control for nonaffine nonlinear systems," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 4, pp. 1170–1182, 2019.

[21] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modelling and identification," *International journal of control*, vol. 56, no. 2, pp. 319–346, 1992.

[22] R. Kumar, "Memory recurrent elman neural network-based identification of time-delayed nonlinear dynamical system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 2, pp. 753–762, 2022.

[23] P. Sastry, G. Santharam, and K. Unnikrishnan, "Memory neuron networks for identification and control of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 306–319, 1994.

[24] A. Savran, "Multifeedback-layer neural network," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 373–384, 2007.

[25] H.-W. Ge, W.-L. Du, F. Qian, and Y.-C. Liang, "Identification and control of nonlinear systems by a time-delay recurrent neural network," *Neurocomputing*, vol. 72, no. 13-15, pp. 2857–2864, 2009.

[26] H. Lin, C. Wang, F. Yu, J. Sun, S. Du, Z. Deng, and Q. Deng, "A review of chaotic systems based on memristive hopfield neural networks," *Mathematics*, vol. 11, no. 6, p. 1369, 2023.

[27] X. Gao, X.-M. Gao, and S. Ovaska, "A modified elman neural network model with application to dynamical systems identification," in *1996 IEEE international conference on systems, man and cybernetics. Information intelligence and systems (Cat. No. 96CH35929)*, vol. 2. IEEE, 1996, pp. 1376–1381.

[28] M. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Eighth Annual Conference of the Cognitive Science Society, 1986*, 1986, pp. 513–546.

[29] C.-C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE transactions on neural networks*, vol. 6, no. 1, pp. 144–156, 1995.

[30] R. Kumar, S. Srivastava, and J. Gupta, "Diagonal recurrent neural network based adaptive control of nonlinear dynamical systems using lyapunov stability criterion," *ISA transactions*, vol. 67, pp. 407–427, 2017.

[31] R. Kumar, S. Srivastava, J. Gupta, and A. Mohindru, "Comparative study of neural networks for dynamic nonlinear systems identification," *Soft Computing*, vol. 23, no. 1, pp. 101–114, 2019.

[32] Z. S. Mohsen and M. J. Mohamed, "Fopid neural network controller design for nonlinear cstr system." *International Journal of Intelligent Engineering & Systems*, vol. 16, no. 6, 2023.

[33] R. P. Borase, D. Maghade, S. Sondkar, and S. Pawar, "A review of pid control, tuning methods and applications," *International Journal of Dynamics and Control*, vol. 9, pp. 818–827, 2021.

[34] J. Hozefa, S. Shadab, G. Revati, S. Wagh, and N. M. Singh, "Adaptive control of nonlinear systems: Parametric and non-parametric approach," in *2021 29th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2021, pp. 1007–1012.

[35] S. Sastry and M. Bodson, "Adaptive control: stability, convergence, and robustness," 1989.

[36] S. N. Kumpati, P. Kannan *et al.*, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990.

[37] K. S. Narendra and A. M. Annaswamy, *Stable adaptive systems*. Courier Corporation, 2012.

[38] L. dos Santos Coelho and A. A. R. Coelho, "Model-free adaptive control optimization using a chaotic particle swarm approach," *Chaos, Solitons & Fractals*, vol. 41, no. 4, pp. 2001–2009, 2009.

[39] T. Hossen, S. J. Plathottam, R. K. Angamuthu, P. Ranganathan, and H. Salehfar, "Short-term load forecasting using deep neural networks (dnn)," in *2017 North American Power Symposium (NAPS)*. IEEE, 2017, pp. 1–6.

[40] X. Jin, J. Shao, X. Zhang, W. An, and R. Malekian, "Modeling of nonlinear system based on deep learning framework," *Nonlinear Dynamics*, vol. 84, pp. 1327–1340, 2016.

[41] D.-D. Zheng, Y. Pan, K. Guo, and H. Yu, "Identification and control of nonlinear systems using neural networks: A singularity-free approach," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2696–2706, 2019.

[42] D.-P. Li, Y.-J. Liu, S. Tong, C. P. Chen, and D.-J. Li, "Neural networks-based adaptive control for nonlinear state constrained systems with input delay," *IEEE transactions on cybernetics*, vol. 49, no. 4, pp. 1249–1258, 2018.

[43] K. S. Narendra and L. S. Valavani, "Direct and indirect model reference adaptive control," *Automatica*, vol. 15, no. 6, pp. 653–664, 1979.

[44] O. Nelles and O. Nelles, *Nonlinear dynamic system identification.* Springer, 2020.

[45] K. S. Narendra and K. Parthasarathy, "Neural networks and dynamical systems," *International Journal of Approximate Reasoning*, vol. 6, no. 2, pp. 109–131, 1992.

[46] F. Abdollahi, H. Talebi, and R. Patel, "A stable neural network-based identification scheme for nonlinear systems," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 4. IEEE, 2003, pp. 3590–3595.

[47] A. Yazdizadeh and K. Khorasani, "Identification of a class of nonlinear systems using dynamic neural network structures," in *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 1. IEEE, 1997, pp. 194–198.

[48] A. H. Ribeiro and L. A. Aguirre, ""parallel training considered harmful?": Comparing series-parallel and parallel feedforward network training," *Neurocomputing*, vol. 316, pp. 222–231, 2018.

[49] T. K. Gupta and K. Raza, "Optimization of ann architecture: a review on nature-inspired techniques," *Machine learning in bio-signal analysis and diagnostic imaging*, pp. 159–182, 2019.

[50] J. Zhang, "Gradient descent based optimization algorithms for deep learning models training," *arXiv preprint arXiv:1903.03614*, 2019.

[51] S. Mirjalili, "Evolutionary algorithms and neural networks," *Studies in computational intelligence*, vol. 780, pp. 43–53, 2019.

[52] N. Kayarvizhy, S. Kanmani, and R. Uthariaraj, "Ann models optimized using swarm intelligence algorithms," *WSEAS Transactions on Computers*, vol. 13, no. 45, pp. 501–519, 2014.

[53] Y. Kassa, J. Zhang, D. Zheng, and D. Wei, "A ga-bp hybrid algorithm based ann model for wind power prediction," in *2016 IEEE Smart Energy Grid Engineering (SEGE)*. IEEE, 2016, pp. 158–163.

[54] J. Li, X. Dong, S. Ruan, and L. Shi, "A parallel integrated learning technique of improved particle swarm optimization and bp neural network and its application," *Scientific Reports*, vol. 12, no. 1, p. 19325, 2022.

[55] C. C. Aggarwal *et al.*, "Neural networks and deep learning," *Springer*, vol. 10, no. 978, p. 3, 2018.

[56] M. J. Willis, G. A. Montague, C. Di Massimo, M. T. Tham, and A. J. Morris, "Artificial neural networks in process estimation and control," *Automatica*, vol. 28, no. 6, pp. 1181–1187, 1992.

[57] J.-P. Noël and G. Kerschen, "Nonlinear system identification in structural dynamics: 10 more years of progress," *Mechanical Systems and Signal Processing*, vol. 83, pp. 2–35, 2017.

[58] D. Yu, Y. Wang, H. Liu, K. Jermsittiparsert, and N. Razmjooy, "System identification of pem fuel cells using an improved elman neural network and a new hybrid optimization algorithm," *Energy Reports*, vol. 5, pp. 1365–1374, 2019.

[59] R. Coban, "A context layered locally recurrent neural network for dynamic system identification," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 241–250, 2013.

[60] A. Yazdizadeh and K. Khorasani, "Adaptive time delay neural network structures for nonlinear system identification," *Neurocomputing*, vol. 47, no. 1, pp. 207–240, 2002.

[61] X. Li, Y. Bai, and C. Huang, "Nonlinear system identification using dynamic neural networks based on genetic algorithm," in *2008 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, vol. 1. IEEE, 2008, pp. 213–217.

[62] D. T. Pham and D. Karaboga, "Training elman and jordan networks for system identification using genetic algorithms," *Artificial Intelligence in Engineering*, vol. 13, no. 2, pp. 107–117, 1999.

[63] J. Deng, "Dynamic neural networks with hybrid structures for nonlinear system identification," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 281–292, 2013.

[64] S. Li, "Comparative analysis of backpropagation and extended kalman filter in pattern and batch forms for training neural networks," in *IJCNN'01. international joint conference on neural networks. proceedings (Cat. No. 01CH37222)*, vol. 1. IEEE, 2001, pp. 144–149.

[65] A. Thammano and P. Ruxpakawong, "Dynamic system identification using recurrent neural network with multi-valued connection weight," in *2009 IEEE International Conference on Fuzzy Systems*. IEEE, 2009, pp. 2077–2082.

[66] N. Bhat and T. J. McAvoy, "Use of neural nets for dynamic modeling and control of chemical process systems," *Computers & Chemical Engineering*, vol. 14, no. 4-5, pp. 573–582, 1990.

[67] V. Veerasamy, N. I. A. Wahab, R. Ramachandran, M. L. Othman, H. Hizam, J. S. Kumar, and A. X. R. Irudayaraj, "Design of single-and multi-loop self-adaptive pid controller using heuristic based recurrent neural network for alfc of hybrid power system," *Expert Systems with Applications*, vol. 192, p. 116402, 2022.

[68] A. Perrusquía and W. Yu, "Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview," *Neurocomputing*, vol. 438, pp. 145–154, 2021.

[69] D. C. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE Journal*, vol. 38, no. 10, pp. 1499–1511, 1992.

[70] G. D. Şen, G. Ö. Günel, and M. Güzelkaya, "Extended kalman filter based modified elman-jordan neural network for control and identification of nonlinear systems," in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 2020, pp. 1–6.

[71] N. Mohajerin and S. L. Waslander, "Multistep prediction of dynamic systems with recurrent neural networks," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3370–3383, 2019.

[72] G. Hernández, E. Zamora, H. Sossa, G. Téllez, and F. Furlán, "Hybrid neural networks for big data classification," *Neurocomputing*, vol. 390, pp. 327–340, 2020.

[73] M. S. Alkhasawneh, "Hybrid cascade forward neural network with elman neural network for disease prediction," *Arabian Journal for Science and Engineering*, vol. 44, no. 11, pp. 9209–9220, 2019.

[74] Y.-J. Wang and C.-T. Lin, "Runge-kutta neural network for identification of dynamical systems in high accuracy," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 294–307, 1998.

[75] X. Huang, Q. Li, Y. Tai, Z. Chen, J. Zhang, J. Shi, B. Gao, and W. Liu, "Hybrid deep neural model for hourly solar irradiance forecasting," *Renewable Energy*, vol. 171, pp. 1041–1060, 2021.

[76] M. S. Alkhasawneh and L. T. Tay, "A hybrid intelligent system integrating the cascade forward neural network with elman neural network," *Arabian Journal for Science and Engineering*, vol. 43, pp. 6737–6749, 2018.

[77] A. Kalinli and S. Sagiroglu, "Elman network with embedded memory for system identification," *Journal of Information Science and Engineering*, vol. 22, no. 6, pp. 1555–1568, 2006.

[78] S. F. Toha and M. O. Tokhi, "Mlp and elman recurrent neural network modelling for the trms," in *2008 7th IEEE international conference on cybernetic intelligent systems*. IEEE, 2008, pp. 1–6.

[79] Y. Wang, M. Zhou, C. Shen, W. Cao, and X. Huang, "Time delay recursive neural network-based direct adaptive control for a piezo-actuated stage," *Science China Technological Sciences*, vol. 66, no. 5, pp. 1397–1407, 2023.

[80] A. de Carvalho Junior, B. A. Angelico, J. F. Justo, A. M. de Oliveira, and J. I. da Silva Filho, "Model reference control by recurrent neural network built with paraconsistent neurons for trajectory tracking of a rotary inverted pendulum," *Applied Soft Computing*, vol. 133, p. 109927, 2023.

[81] F. Yang, J. Chen, and Y. Liu, "Improved and optimized recurrent neural network based on pso and its application in stock price prediction," *Soft computing*, vol. 27, no. 6, pp. 3461–3476, 2023.

[82] T. Zhao, Y. Zheng, and Z. Wu, "Feature selection-based machine learning modeling for distributed model predictive control of nonlinear processes," *Computers & Chemical Engineering*, vol. 169, p. 108074, 2023.

[83] C. Hu, S. Chen, and Z. Wu, "Economic model predictive control of nonlinear systems using online learning of neural networks," *Processes*, vol. 11, no. 2, p. 342, 2023.

[84] H.-G. Han, C.-Y. Wang, H.-Y. Sun, H.-Y. Yang, and J.-F. Qiao, "Iterative learning model predictive control with fuzzy neural network for nonlinear systems," *IEEE Transactions on Fuzzy Systems*, 2023.

[85] H. Han, J. Zhang, H. Yang, Y. Hou, and J. Qiao, "Data-driven robust optimal control for nonlinear system with uncertain disturbances," *Information Sciences*, vol. 621, pp. 248–264, 2023.

[86] X. Luo, Y. Yuan, S. Chen, N. Zeng, and Z. Wang, "Position-transitional particle swarm optimization-incorporated latent factor analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3958–3970, 2020.

[87] M. M. Polycarpou and P. A. Ioannou, *Identification and control of nonlinear systems using neural network models: Design and stability analysis.* Citeseer, 1991.

[88] V. A. Akpan and G. D. Hassapis, "Nonlinear model identification and adaptive model predictive control using neural networks," *ISA transactions*, vol. 50, no. 2, pp. 177–194, 2011.

[89] S. J. Yoo, J. B. Park, and Y. H. Choi, "Indirect adaptive control of nonlinear dynamic systems using self recurrent wavelet neural networks via adaptive learning rates," *Information Sciences*, vol. 177, no. 15, pp. 3074–3098, 2007.

[90] S. Slama, A. Errachdi, and M. Benrejeb, "Neural adaptive pid and neural indirect adaptive control switch controller for nonlinear mimo systems," *Mathematical Problems in Engineering*, vol. 2019, 2019.

[91] M. K. Ciliz, "Combined direct and indirect adaptive control for a class of nonlinear systems," *IET Control Theory & Applications*, vol. 3, no. 1, pp. 151–159, 2009.

[92] L. Cheng, Z. Wang, F. Jiang, and J. Li, "Adaptive neural network control of nonlinear systems with unknown dynamics," *Advances in Space Research*, vol. 67, no. 3, pp. 1114–1123, 2021.

[93] Z. Chen, Z. Li, and C. P. Chen, "Adaptive neural control of uncertain mimo nonlinear systems with state and input constraints," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 6, pp. 1318–1330, 2016.

[94] S. He, H. Fang, M. Zhang, F. Liu, and Z. Ding, "Adaptive optimal control for a class of nonlinear systems: The online policy iteration approach," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 2, pp. 549–558, 2019.

[95] F. Bonassi and R. Scattolini, "Recurrent neural network-based internal model control design for stable nonlinear systems," *European Journal of Control*, vol. 65, p. 100632, 2022.

[96] Z. Zhou, D. Tong, Q. Chen, W. Zhou, and Y. Xu, "Adaptive nn control for nonlinear systems with uncertainty based on dynamic surface control," *Neurocomputing*, vol. 421, pp. 161–172, 2021.

[97] R. Kumar, S. Srivastava, J. Gupta, and A. Mohindru, "Self-recurrent wavelet neural network–based identification and adaptive predictive control of nonlinear dynamical systems," *International Journal of Adaptive Control and Signal Processing*, vol. 32, no. 9, pp. 1326–1358, 2018.

[98] R. Kumar, S. Srivastava, and J. Gupta, "Comparative study of neural networks for control of nonlinear dynamical systems with lyapunov stability-based adaptive learning rates," *Arabian Journal for Science and Engineering*, vol. 43, pp. 2971–2993, 2018.

[99] A. Perrusquía and W. Yu, "Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview," *Neurocomputing*, vol. 438, pp. 145–154, 2021.

[100] H. V. A. Truong, M. H. Nguyen, D. T. Tran, and K. K. Ahn, "A novel adaptive neural network-based time-delayed estimation control for nonlinear systems subject to disturbances and unknown dynamics," *ISA transactions*, vol. 142, pp. 214–227, 2023.

[101] A. Kapnopoulos, C. Kazakidis, and A. Alexandridis, "Quadrotor trajectory tracking based on backstepping control and radial basis function neural networks," *Results in Control and Optimization*, vol. 14, p. 100335, 2024.

[102] A. H. Bukhari, M. Shoaib, A. K. Kiani, N. I. Chaudhary, M. A. Z. Raja, and C.-M. Shu, "Dynamical analysis of nonlinear fractional order lorenz system with a novel design of intelligent solution predictive radial base networks," *Mathematics and Computers in Simulation*, 2023.

[103] G. Hesamian, A. Johannssen, and N. Chukhrova, "Fuzzy nonlinear regression modeling with radial basis function networks," *IEEE Transactions on Fuzzy Systems*, pp. 1–10, 2023.

[104] J. Tavoosi, "Sliding mode control of a class of nonlinear systems based on recurrent type-2 fuzzy rbfn," *International Journal of Mechatronics and Automation*, vol. 7, no. 2, pp. 72–80, 2020.

[105] G. Zhou and D. Tan, "Review of nuclear power plant control research: Neural network-based methods," *Annals of Nuclear Energy*, vol. 181, p. 109513, 2023.

[106] C. N. Manh, T. N. Manh, D. H. Thi Kim, Q. N. Van, and T. L. Nguyen, "An adaptive neural network-based controller for car driving simulators," *International Journal of Control*, vol. 96, no. 1, pp. 82–93, 2023.

[107] K. Patan and M. Patan, "Neural-network-based iterative learning control of nonlinear systems," *ISA transactions*, vol. 98, pp. 445–453, 2020.

[108] M. Di Ferdinando, P. Pepe, and S. Di Gennaro, "A converse lyapunov–krasovskii theorem for the global asymptotic local exponential stability of nonlinear time–delay systems," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 7–12, 2020.

[109] K. Xu, H. Wang, and P. X. Liu, "Adaptive fixed-time control for high-order stochastic nonlinear time-delay systems: An improved lyapunov–krasovskii function," *IEEE Transactions on Cybernetics*, 2023.

[110] S. Lale, Y. Shi, G. Qu, K. Azizzadenesheli, A. Wierman, and A. Anandkumar, "Kcrl: Krasovskii-constrained reinforcement learning with guaranteed stability in nonlinear dynamical systems," *arXiv preprint arXiv:2206.01704*, 2022.

[111] S. Dashkovskiy and S. Pavlichkov, "Stability conditions for infinite networks of nonlinear systems and their application for stabilization," *Automatica*, vol. 112, p. 108643, 2020.

[112] P. Zhou, X. Hu, Z. Zhu, and J. Ma, "What is the most suitable lyapunov function?" *Chaos, Solitons & Fractals*, vol. 150, p. 111154, 2021.

[113] O. J. Oaks and G. Cook, "Piecewise linear control of nonlinear systems," *IEEE Transactions on Industrial Electronics and Control Instrumentation*, vol. IECI-23, no. 1, pp. 56–63, 1976.

[114] C. Zhao and L. Guo, "Control of nonlinear uncertain systems by extended pid," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3840–3847, 2021.

[115] A. Y. Alanis, "Adaptive neural sensor and actuator fault-tolerant control for discrete-time unknown nonlinear systems," *Franklin Open*, vol. 1, pp. 9–16, 2022.

[116] C. Qin, J. Wang, H. Zhu, J. Zhang, S. Hu, and D. Zhang, "Neural network-based safe optimal robust control for affine nonlinear systems with unmatched disturbances," *Neurocomputing*, vol. 506, pp. 228–239, 2022.

[117] P. Swarnkar, S. K. Jain, and R. Nema, "Adaptive control schemes for improving the control system dynamics: a review," *IETE Technical Review*, vol. 31, no. 1, pp. 17–33, 2014.

[118] J. Wang, Y. Gao, J. Qiu, and C. Ki Ahn, "Sliding mode control for non-linear systems by takagi–sugeno fuzzy model and delta operator approaches," *IET Control Theory & Applications*, vol. 11, no. 8, pp. 1205–1213, 2017.

[119] R. Zemouri, D. Racoceanu, and N. Zerhouni, "Recurrent radial basis function network for time-series prediction," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 5-6, pp. 453–463, 2003.

[120] R. Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[121] R. V. Kulkarni and G. K. Venayagamoorthy, "Generalized neuron: Feedforward and recurrent architectures," *Neural networks*, vol. 22, no. 7, pp. 1011–1017, 2009.

[122] R. Shobana, R. Kumar, and B. Jaint, "A recurrent neural network-based identification of complex nonlinear dynamical systems: a novel structure, stability analysis and a comparative study," *Soft Computing*, pp. 1–17, 2023.

[123] J. Cai, G. Zhou, M. Dong, X. Hu, G. Liu, and W. Ni, "Real-time arrhythmia classification algorithm using time-domain ecg feature based on ffnn and cnn," *Mathematical problems in engineering*, vol. 2021, no. 1, p. 6648432, 2021.

[124] B. Subudhi and D. Jena, "A differential evolution based neural network approach to nonlinear system identification," *Applied Soft Computing*, vol. 11, no. 1, pp. 861–871, 2011.

[125] A. Levin and K. Narendra, "Chapter 6 - identification of nonlinear dynamical systems using neural networks," in *Neural Systems for Control*, O. Omidvar and D. L. Elliott, Eds. San Diego: Academic Press, 1997, pp. 129–160. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780125264303500076

[126] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. ieee, 1995, pp. 1942–1948.

[127] E. Kaya, "A comprehensive comparison of the performance of metaheuristic algorithms in neural network training for nonlinear system identification," *Mathematics*, vol. 10, no. 9, p. 1611, 2022.

[128] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. IEEE, 1998, pp. 69–73.

[129] M. Ahmed and M. F. Anjum, "Neural-net-based direct self-tuning control of nonlinear plants," *International Journal of Control*, vol. 66, no. 1, pp. 85–104, 1997.

[130] R. Shobana, B. Jaint, and R. Kumar, "Design of a novel robust recurrent neural network for the identification of complex nonlinear dynamical systems," *Soft Computing*, vol. 28, no. 3, pp. 2737–2751, 2024.