Evolutionary Neuro-Computing Approaches to System Identification

Debashisha Jena

Department of Electrical Engineering
National Institute of Technology, Rourkela
Rourkela-769 008, Orissa, India

Evolutionary Neuro-Computing Approaches to System Identification

Thesis submitted in partial fulfillment
of the requirement for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Debashisha Jena
(Roll: 50702002)

under the guidance of

Dr. Bidyadhar Subudhi

Department of Electrical Engineering
National Institute of Technology, Rourkela
Rourkela-769 008, Orissa, India
April 2010

Department of Electrical Engineering
National Institute of Technology, Rourkela
Rourkela-769 008, Orissa, India.

Certificate

This is to certify that the work in the thesis entitled *Evolutionary Neuro-Computing Approaches to System Identification* by *Debashisha Jena* is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Electrical Engineering during the session 2006–2010 in the Department of Electrical Engineering, National Institute of Technology (NIT), Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr. Bidyadhar Subudhi
Professor
Electrical Engineering Department
NIT, Rourkela

Place: NIT, Rourkela
Date:

Acknowledgment

I would like to gratefully acknowledge the enthusiastic supervision and guidance of Prof. Bidyadhar Subudhi during this work. He is my source of inspiration. As my supervisor, his insight, observations and suggestions helped me to establish the overall direction of the research and contributed immensely to the success of the work. I would like to thank Professor P. C. Panda, Professor S. Das, Professor D. R. K. Parhi and Professor S. Ghosh for their detailed comments and suggestions during the final phases of the preparation of this thesis.

I would also like to convey my sincere thanks to Professor M. M. Gupta of The University of Saskatchewan, Saskatoon, Canada for his discussion and help on neural networks theories and applications.

I thank to all my friends for being there whenever I needed them.

I must acknowledge the academic resource that I have got from NIT, Rourkela giving me a comfortable and active environment for pursuing my research.

Finally, I would like to thank my parents and my wife for their endless encoragement for pursuing my research.

Debashisha Jena

Abstract

System models are essentially required for analysis, controller design and future prediction. System identification is concerned with developing models of physical system. Although linear system identification got enriched with several useful classical methods, nonlinear system identification always remained active area of research due to the reason that most of the real world systems are nonlinear in nature and moreover, having non-unique models. Among the several conventional system identification techniques, the Volterra series, Hammerstein-Wiener and polynomial model identification involve considerable computational complexities. The other techniques based on regression models such as nonlinear autoregressive exogenous (NARX) and nonlinear autoregressive moving average exogenous (NARMAX), also suffer from difficulty in choosing regressors.

To overcome the above difficulties, nonlinear system identification using neural networks (NNs) have been given considerable attention over last three decades. This is because NNs has the capability of approximating almost every nonlinear function. However, it requires appropriate training to optimally tune the wights of the NN. For this, conventional methods such as back-propagation (BP), Levenberg-Marquardt (LM) etc are usually applied with the objective that a cost function e.g. mean squared error (MSE) between the actual and estimated output gets minimized. However, these conventional techniques suffer from the problem of local minima and are much sensitive to initial values of weights. Hence, to overcome the problem of initilization and local minima, evolutionary algorithms (EAs) have been paid importance as attractive approach for NN training. Moreover, the same EA can be used to train different NNs such as feed-forward, recurrent and higher order NNs, to save a lot of computational effort. The objective of this thesis is to exploit the global optimization properties of evolutionary computation (EC) approaches to train

NNs so that one can acheive successful system identification strategies using NNs.

First this thesis considers devloping sequential hybridization (SH) algorithms for nonlinear system identification by combining differential evolution (DE) with the local search algorithm i.e. LM algorithm in sequential manner. The efficiency of this hybrid training increases by combining the DE's global search ability with LM's local search ability to fine tune the search space. Initially DE will locate a point i.e. a set of initial weights for the local search LM, in the basin of attraction of the global minimum. LM starts its search with these initial weights so that it will be easy to obtain global optimal weights. By pursuing a number of simulation studies, the effectiveness of the proposed SH algorithm used as system identifier has been accessed. Studies on the effectiveness of this proposed DE+LM+NN identifier has been made together with the convergence analysis of this approach.

The problem of SH algortihm lies on deciding when to stop one algorithm and start the next one. So, the thesis proposes other type of hybrid algorithm known as memtic algorithm (MA) where the local search BP algorithm is used as an operator like crossover and mutation operator for genetic algorithm (GA), particle swarm optimization (PSO) and DE. A detailed MSE analysis for updating the weights of a NN has been made. From this, it is observed that the proposed differential evolution back-propagation (DEBP) memetic algorithm trained NN approach to system identification is an efficient method that offers better optimal solutions compared to GA and PSO based identification schemes. Both the SH and MAs have been successfully applied to a highly nonlinear twin rotor multi-input-multi-output (TRMS) identification problem.

Following the above development of identifiers, the thesis next describes how the DE can be extended to obtain better identification performance. This ex-

tension has led to a different variation of DE algorithm called an opposition based differential evolution (ODE) algorithm using opposition based learning approach to train a feed-forward neural network (FNN) that is found to be effective for identification of nonlinear systems. Simulation results obtained envisage that the system identification using ODE is faster and the identification error is less compared to the case of identification of nonlinear systems using the DE. A further development of the identification scheme has been proposed exploiting a new variant of the DE known as opposition based mutation differential evolution (OMDE). This approach has provided a further improvement in optimization compared to the DE. The proposed OMDE algorithm used as a parameter estimator. A comparative analysis of parameter estimation of a three phase induction motor using the DE and OMDE has been made which shows a significant reduction in computational overhead as well as a substantial improvement in estimation accuracy.

The efficacies of the developed system identification strategies have been demonstrated by their application to model a number of nonlinear systems such as Box-Jenkin's gas furnance system, TRMS, induction motor and two bench mark problems.

The work described in the thesis contributes towards development of number of neuro-evolutionary system identification approaches which are useful for achieving successful nonlinear system identification.

Contents

List of Figures

List of Tables

Acronyms

- BFO: Bacteria Foraging Optimization
- BP: Back-propagation
- BSNN: B-spline Neural Network
- DE: Differential Evolution
- DEBP: Differential Evolution Back-propagation
- DNN: Dynamic Neural Network
- DOF: Degrees of Freedom
- EA: Evolutionary Algorithm
- EANN: Evolutionary Neural Networks
- EC: Evolutionary Computation
- ES: Evolutionary Strategy
- FNN: Feeed Forward Neural Network
- GA: Genetic Algortihm
- GABP: Genetic Algorithm Back-propagation
- GD: Gradient Descent
- IM: Induction Motor
- LM: Levenberg Marquardt
- LMS: Least Mean Square
- LSR: Least Square Regression
- MA: Memetic Algorithm

- MIMO: Multi Input Multi Output
- MLP: Multi Layer Perceptron
- MSE: Mean Squared Error
- NARX: Nonlinear Autoregressive Exogenous
- NARMAX: Nonlinear Autoregressive Moving Average Exogenous
- NN: Neural Network
- OBL: Opposition based Learning
- ODE: Opposition based Differential Evolution
- OMDE: Opposition based Mutation Differential Evolution
- PSO: Particle Swarm Optimization
- PSOBP: Particle Swarm Optimization Back-propagation
- RBFNN: Radial Basis Function Neural Network
- RFNN: Recurrent Fuzzy Neural Network
- RLS: Recurssive Leat Square
- RMSE: Root Mean Squared Error
- SGA: Simple Genetic Algorithm
- SH: Sequential Hybridization
- SI: System Identification
- SISO: Single Input Single Output
- SSE: Sum Squared Error
- SVM: Support Vector Machines

- TRMS: Twin Rotor Multivariable System
- WNN: Wavelet Neural Network

Chapter 1

Introduction

1.1 Introduction

System identification (SI) is an important research area primarily devoted to developing models of physical systems based on observed input output data. During the past three decades a lot of research has been directed towards developing efficient system identification algorithms with a view to obtain models that closely match to the real physical systems. Motivated by the nice property of function approximation of Neural Networks (NNs) many research work use these networks for identification of nonlinear dynamic systems. However, selection of appropriate neural network topology, fast and efficient training are of important concerns for achieving successful system identification. Training in NNs is usually guided by the minimization of an error function, such as the mean square error (MSE) or sum squared error (SSE) or root mean square error (RMSE) between actual output and estimated output averaged over all samples, by iteratively adjusting connection weights. Most training algorithms, such as back-propagation (BP) and conjugate gradient algorithms are based on gradient descent principles that often get trapped in a local minimum of the error function. Hence, these algorithms have the inability of finding a global minimum if the error function is multimodal and/or non-differentiable.

Recently, there is an increasing interest in exploiting evolutionary neural networks for different applications where evolution can be introduced into NNs at

different levels such as in inter layer connection weights training and architecture design etc. Evolutionary approaches are considered as global approaches to connection weight training of NNs, especially when gradient information of the error function is difficult to obtain. Gradient-based training algorithms often have to be run multiple times in order to avoid the problem of being trapped in a poor local optimum. Motivated by the global optimizing feature of the Evolutionary Computation (EC), recently, a lot of research works consider the use of evolutionary computing techniques such as Genetic Algorithm (GA), Evolutionary Algorithm (EA), Particle Swarm Optimization (PSO) and Differential Evolution (DE) etc. for efficient training of NNs. Identification problem can be conceived as an optimization problem in which the error between the actual physical measured response of a system and the identified response of a model is minimized. Therefore, interest in system identification lies in minimizing the error norm of the outputs. This thesis considers the identification of nonlinear systems using a number of neuro-evolutionary approaches. It has been demonstrated in this work that the success of the combined use of local and global search methods for training of the neural network yields efficient nonlinear system identification strategy.

1.2 Background

1.2.1 System Identification

The first step in designing the controller is to model the plant. System identification is the process of building models of dynamic process from input-output signals. The aim of system identification [1] can be identified as to find a model with adjustable parameters and then to adjust them so that the predicted output matches the measured output. Two important points on system identification are:

- Which model parameterization is to be used?
- How to know if the fitted model is good?

Most of system identification techniques have their roots in statistical methods like Least squares fitting, maximum likelihood estimation etc. Apart from parametric methods of system identification there are non-parametric techniques for system identification such as spectral analysis, correlation analysis and transient analysis. The various steps involved in system identification are experiment setup and data collection, data preprocessing, model structure selection, parametric estimation and validation. In nonlinear system identification [2] one approach is the black box model that uses various selected model structures and the model that gives optimum fit for the test data is the identified model.

Linear system identification

A linear system obeys two properties namely superposition and scaling. Hence, if f is a linear operator given by

$$y_1(t) = f\left(u_1(t)\right) \tag{1.1}$$

$$y_2(t) = f\left(u_2(t)\right) \tag{1.2}$$

where, u_1, u_2, y_1 and y_2 are the inputs and outputs of the system, then according to the definition of linearity, we have

$$f\left(k_1u_1(t) + k_2u_2(t)\right) = k_1y_1(t) + k_2y_2(t) \tag{1.3}$$

where k_1 and k_2 are constants.

Parametric representations

A parametric model consists of a set of differential or difference equations which describe the system dynamics. Such equations usually contain a small number of parameters, which can be varied to alter the behavior of the equations. The identification of an unknown system comprises two stages. First, the structure of the parametric model is chosen, and then the parameters themselves are

estimated by using an optimization algorithm.

Linear Difference Equations

We can write the relationship between the input, output, and noise as a linear difference equation given by

$$y(t) + a_1 y(t-1) + ... + a_{n_x} y(t-n_x) = b_1 u(t-1) + b_2 u(t-2) + ...b_{n_y} u(t-n_y) + e(t) + c_1 e(t-1) + ... + c_{n_z} e(t-n_z) \tag{1.4}$$

which can be written more compactly as

$$A(q)y(t) = B(q)u(t) + C(q)e(t) \tag{1.5}$$

where

$$A(q) = 1 + a_1 q^{-1} + ... + a_{n_x} q^{-n_x}$$

$$B(q) = b_1 + b_2 q^{-1} + ... + b_{n_y} q^{-n_y+1}$$

$$C(q) = 1 + c_1 q^{-1} + ... + c_{n_z} q^{-n_z}$$

q^{-1} is the backward shift operator. This is the auto-regressive, moving average exogenous (ARMAX) model. The current output $y(t)$ depends on an exogenous input $u(t)$, an innovations process $e(t)$ and the past values of the output. The polynomials ($A(q), B(q)$) known as deterministic model, whereas ($A(q), C(q)$) represent the stochastic system model. This model has several special cases, the first of which is the autoregressive (AR) model:

$$A(q)y(t) = e(t) \tag{1.6}$$

in which the output depends on the current disturbance, as well as the previous values of the output. Another special case is the moving average (MA) model:

$$y(t) = C(q)e(t) \tag{1.7}$$

in which the output depends on the previous values of the disturbance, Combining these two, we get the autoregressive moving average (ARMA) model:

$$A(q)y(t) = C(q)e(t) \tag{1.8}$$

If we add an accessible input, $u(t)$, to the AR model, the result is an autoregressive exogenous input (ARX) model:

$$A(q)y(t) = B(q)u(t) + e(t) \tag{1.9}$$

A special case of the ARX structure, in which there is no disturbance input, is the finite impulse response (FIR) model:

$$y(t) = B(q)u(t) \tag{1.10}$$

In this case, the output depends solely on the previous values of the exogenous input. This structure forms the basis of a number of so-called non-parametric identification schemes. Once a candidate model structure and order have been chosen, the model representation can be reduced to a parameter vector, $\theta = [A(q)\, B(q)\, C(q)]$

State Space Models

Another parametric system representation is the state space model. In this case, we consider a set of equations of the form:

$$\mathrm{x}(t+1) = A\mathrm{x}(t) + B\mathrm{u}(t) \tag{1.11}$$

$$y(t+1) = C\mathrm{x}(t) + D\mathrm{u}(t) \tag{1.12}$$

where the sequences $u(t)$, $y(t)$ and $x(t)$ represent the system's input, output and state respectively. The impulse response (Markov parameters) of the system is first identified from input-output data, and then used to compute the system matrices A, B, C and D.

Nonparametric representations

A linear system can be represented by its impulse response. In continuous time, we can compute the output via the convolution integral: [20]

$$y(t) = \int_0^T h(\tau)u(t-\tau)d\tau \tag{1.13}$$

where T is the memory length of the system, and $h(\tau)$ is the impulse response. In this case, as the lower bound of the integration is 0, the system is causal. Given that the analysis will be performed using sampled data on a digital computer, we will require a discrete time formulation. One benefit gained by restricting ourselves to discrete time is that it avoids the mathematical difficulties associated with a continuous-time white-noise signal. In continuous time, a white noise signal has infinite bandwidth and hence infinite power. In discrete time, however, it is simply a sequence of independently distributed random variables. In discrete time, the convolution integral becomes the summation:

$$y(t) = \Delta t. \sum_{\tau=0}^{T-1} h(\tau)u(t-\tau) \tag{1.14}$$

Here, the memory length, T, and the lag τ, are integers. If the system is non causal, then the lower limit of the summation will be negative. The sampling increment is Δt; which is assumed to be 1 here so that it can be dropped. If the input process is white, it can be shown that the impulse response can be recovered from the input/output cross-correlation function. Given N data points, a biased estimate of the cross-correlation can be obtained as:

$$\hat{\Phi}_{uy}(\tau) = \frac{1}{N} \sum_{t=\tau+1}^{N} u(t-\tau)y(t) \tag{1.15}$$

Substituting the value $y(t)$ of from (1.14) in (1.15) we have

$$\begin{aligned}\hat{\Phi}_{uy}(\tau) &= \tfrac{1}{N}\sum_{t=\tau+1}^{N} u(t-\tau)\sum_{j=0}^{T-1} h(j)u(t-j) \\ &= \sum_{j=0}^{T-1} h(j)\left\{\tfrac{1}{N}\sum_{t-\tau+1}^{N} u(t-\tau)u(t-j)\right\} \\ &= \sum_{j=0}^{T-1} h(j)\hat{\Phi}_{xx}(\tau-j)\end{aligned} \tag{1.16}$$

Hence, from equation (1.16) the input-output cross-correlation is equal to the convolution of the impulse response with the input auto-correlation function. If the input is white, the auto-correlation function is an impulse, and the cross-correlation and impulse response are equal. If the input is non-white, the input auto-correlation function must be deconvolved, from the cross-correlation estimate. This problem was approached by modeling the observed input as a white noise process filtered by an autoregressive filter. This filter can be estimated, and its inverse (a moving average filter) applied to both the input and output signals. The cross-correlation between the filtered input and filtered output is then estimated. Since the filtered input signal is effectively white, the cross-correlation estimate provides an estimate of the impulse response [20]. The input auto-correlation is estimated, and the convolution between the input auto-correlation and the impulse response can be written in matrix form as:

$$\begin{bmatrix} \hat{\phi}_{uy}(0) \\ \hat{\phi}_{uy}(1) \\ \vdots \\ \hat{\phi}_{uy}(T-1) \end{bmatrix} = \begin{bmatrix} \hat{\phi}_{uu}(0) & \hat{\phi}_{uu}(1) & \cdots & \hat{\phi}_{uu}(T-1) \\ \hat{\phi}_{uu}(1) & \hat{\phi}_{uu}(0) & \cdots & \hat{\phi}_{uu}(T-2) \\ \vdots & \vdots & \vdots & \\ \hat{\phi}_{uu}(T-1) & \hat{\phi}_{uu}(T-2) & \cdots & \hat{\phi}_{uu}(0) \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(T-1) \end{bmatrix} \tag{1.17}$$

This equation can be solved efficiently, using Levinson's algorithm [3], since $\hat{\phi}_{uu}$, is the matrix derived from the input auto-correlation function, has a symmetric Toeplitz structure.

Nonlinear system identification

Nonlinear system identification is the task of determining or estimating a systems input-output relationship F, based on (possibly noisy) output measurements.

$$y(t) = F\left[x(t)\right] + e(t) \tag{1.18}$$

where $e(t)$ is the noise, disturbance or another source of error in the process of measurement. Nonlinear systems can be modeled into nonparametric and parametric forms. In parametric modeling the input-output relationship are defined by finite number of parameters. Nonparametric nonlinear system identification includes the Volterra and Wiener series models which are based on Taylor series expansion of time invariant nonlinear systems. The Volterra model expresses the input-output relationship of a nonlinear system in terms of Volterra kernels. The output $y(t)$ in response to the input $x(t)$ can be expressed as

$$y(t) = h_0 + \sum_{n=1}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n\left(\tau_1, \cdots, \tau_n\right) x\left(t-\tau_1\right) \cdots x\left(t-\tau_n\right)\, d\tau_1 \cdots d\tau_n \tag{1.19}$$

where h_0 is a constant and $h_j\left(\tau_1 \cdots \cdots \tau_j\right),\ \ 1 \leq j \leq \infty$ is the j^{th} order Volterra kernel coefficients defined for $\tau_i = -\infty$ to $+\infty,\ i = 1, 2, \cdots, n$. We assume $h_j\left(\tau_1, \cdots, \tau_j\right) = 0$ if any $\tau_i < 0,\ 1 \leq i \leq j$ which implies causality. In parametric nonlinear system models, the input-output relation can be expressed by a mathematical function determined by a finite number of parameters. Parametric models can be viewed as special case of nonparametric models. Truncated N-th order Volterra series can be taken as parametric nonlinear system model described as

$$y(t) = h_0 + \sum_{n=1}^{N} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n\left(\tau_1 \cdots \cdots \tau_n\right) x\left(t-\tau_1\right) \cdots x\left(t-\tau_n\right)\, d\tau_1 \cdots d\tau_n \tag{1.20}$$

Linear-in-parameter models

Parametric representations of nonlinear systems typically contain a small number of coefficients that can be varied to alter the behavior of the equation and may be linked to the underlying system. Leontaritis and Billings [4] have proposed the NARMAX structure as a general parametric form for modeling nonlinear systems. This structure is suitable for modeling both the stochastic and deterministic components of a system and capable of describing a wide variety of nonlinear systems [5, 6, 7, 8]. NARMAX models have been successfully demonstrated for modeling the input output behavior of many complex systems such as adaptive polynomial filters, and offer a promising framework for describing nonlinear behavior such as aircraft dynamics. Often, this formulation yields compact model descriptions that may be readily identified and afford greater interpretability. This system representation, however, can yield a large number of possible terms required to represent the dynamic process. In practice, many of these candidate terms are insignificant and can be removed. Consequently, the structure-detection problem turns out to be selection of a subset of candidate terms that best predicts the output while maintaining an efficient system description.

NARMAX models also describe nonlinear systems in terms of linear in the parameters difference equations, which represent the current output with present and past inputs and, past outputs. Identifying a NARMAX model requires two distinct steps such as structure detection and parameter estimation. Structure detection can be divided into steps such as model order selection and selection of parameters to include in the model. We consider model order selection as part of structure detection since, theoretically, there are an infinite number of candidate terms that could be considered initially. Establishing the model order limits the choice of terms to be considered. Good parameter estimation methods can be explored if the model order is known. However, there remains a problem in model order selection. Depending on the order of the system, the

number of candidate terms can be very large. Selection of a subset of these candidate terms is necessary for an efficient system description. In fact, many NARMAX systems can be described by only a few terms. A wide range of discrete time multiple variable nonlinear stochastic systems can be represented by the following NARMAX model:

$$\hat{y}(t) = \alpha + F^l\left[y(t-1), ..., y(t-n_x), u(t), ..., u(t-n_y),\ e(t-n_z)\right] + e(k) \tag{1.21}$$

where $y(t)$, $u(t)$ and $e(t)$ represent the system output, input, and prediction error, respectively. Also, l is the degree of nonlinearity, α is a constant dc level, $F^l[.]$ is some vector valued nonlinear function, n_x, n_y and n_z represent the number of lags in the input, output and prediction error, respectively. The prediction error term $e(t)$, defined as $e(t) = y(t) - \hat{y}(t)$, is included in the model to accommodate noise, where $\hat{y}(t)$ is the prediction output. Expanding Eq. (1.21) by defining the function $F^l[.]$ as a polynomial of degree l gives a representation of all the possible combinations of $y(t)$, $u(t)$ and $e(t)$ up to degree l. For example, the current output can be presented as

$$y(t) = \alpha+\theta_1 y(t-1)+\theta_2 u(t-1)+\theta_3 u(t-1)y(t-1)+\theta_4 u(t-1)e(t-1)+\theta_5 e(t-1)+e(t)$$

By defining

$$p_1(t) = y(t-1), p_2(t) = u(t-1),\ p_3(t) = u(t-1)y(t-1),$$
$$p_4(t) = u(t-1)e(t-1), p_5(t) = e(t-1),\ p_0(t) = 1, \text{ and } \theta_0 = \alpha$$

If N input and output measurements are available, and if there are$M$ terms in the model, then the above equation can be written in a matrix form as

$$\mathbf{Y} = \mathbf{A}\theta + \mathbf{e} \tag{1.22}$$

where

$$\mathbf{Y^T} = [y(1)\ y(2)\ \cdots\ y(N)]$$

$$\theta^{\mathbf{T}} = [\theta_0\ \theta_1\ \cdots\ \theta_N]$$

$$\mathbf{e^T} = [e(1)\ e(2)\ \cdots\ e(N)]$$

$$\mathbf{A} = \begin{bmatrix} A_0(1) & A_1(1) & \cdots & A_M(1) \\ A_0(2) & A_1(2) & \cdots & A_M(2) \\ \vdots & \vdots & \vdots & \\ A_0(N) & A_1(N) & \cdots & A_M(N) \end{bmatrix}$$

where $\mathbf{A}$ represents a term in the NARMAX model and θ represents the unknown parameters to be estimated. The parameter vector $\theta^{\mathbf{T}}$ in Eq. (1.22) can be estimated using some well known methods, such as a least-squares-based or prediction error method, Choleski or $U-D$ factorization, the $Q-R$ algorithm, singular value decomposition or principle component regression.

Nonlinear-in-parameter models

This class of models include all parametric descriptions of nonlinear systems whose output is not linearly related to the parameters. Nonlinear-in-parameters models often arise from physical modeling considerations. In general, it can be expressed as $y(t) = F[x(t), \theta]$ where F has a fixed functional form that is parameterized by θ. Neural networks are the most well-known class of nonlinear-in-parameter models [16]. The hidden nodes represent the nonlinear processing units and the link between the nodes represent the weighting factor to the input of each neuron. These weights are the parameters of the neural network model. As the neurons are the nonlinear functions, the output of the network is the nonlinear functions of the parameters. Multi layer perceptrons are nonlinear-in-parameter models so the identification method must include a nonlinear optimization technique such as nonlinear least square, prediction estimation error method etc. It will be discussed later that there remains serious problem of local minima in using these optimization techniques.

The basic block-diagram representation of an identification problem is shown in Fig. 1.1. From the figure it is clear that if the error tends towards zero the estimated output will be same as the desired output. So the identified model

will exactly mimic the original system to be identified.

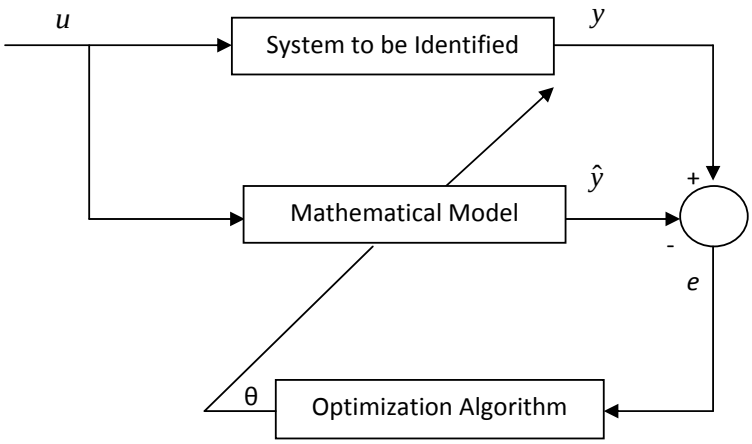

Figure 1.1: Block Diagram on System Identification

As shown in Fig.1.1, given the discrete time, time invariant nonlinear dynamic system of inputs $u(k)$ and outputs $y(k)$. The objective is to develop identification algorithms using several methods such as evolutionary computing techniques and neural networks. The NARX model structure is taken as the nonlinear frame work which is in the form of

$$y_k = f\left(y(k-1), \cdots, y(k-n_y); u(k); u(k-1), \cdots, u(k-n_u)\right) \tag{1.23}$$

$\Re_k = [y(k-1), \cdots, y(k-n_y); u(k); u(k-1), \cdots, u(k-n_u)]$ where $k \in Z^+$is the discrete temporal variable

$u_k \in R^1$ is the input at time k

$y_k \in R^1$ is the output at time k

$f : R^{n_y+n_u}$ is an unknown nonlinear mapping defined on an open set

n_y is an integer denote maximum lag in the output

n_u is an integer denote maximum lag in the input

$\Re_k$ is the regression vector in a NARX model

Since f is unknown, the objective is to use some type of network approximator $\Gamma\left(\Re, \theta\right)$ to approximate $f(\Re)$. In the network, $\Re \in R^n$ is the input to the network and $\theta \in R^d$ is set of adjustable parameter in vector form of d dimension.

The system described in equation (1.23) can be rewritten in the form as

$$y_k = \Gamma\left(\Re(k); \theta^*\right) + e(k) \tag{1.24}$$

where is e the modeling error, defined as

$$e(k) = f\left(\Re(k)\right) - \Gamma\left(\Re(k); \theta^*\right) \tag{1.25}$$

In order to obtain successful identification the identified system must be able to reproduce the output of the physical system for any given input. Let $\Re_k$ belongs to some compact set Z for all $k > 0$, then we define the parameter vector θ^* as the optimal value of θ in the sense that it minimizes the distance between $f$and Γ for all $\Re \in Z$. The optimal parameter vector θ^* is defined as

$$\theta^* = \arg\min \left\{ \sup_{\Re \in Z} |f(\Re) - \Gamma(\Re; \theta)| \right\} \tag{1.26}$$

The optimization problem requires finding a vector $\theta \in \mathbf{S}$, where $\mathbf{S}$ is the search space, so that a certain quality criterion is satisfied, namely that the error norm is minimized. By changing the value of θ it is possible to change the input-output response of the network Γ. The search space $\mathbf{S}$ is defined by a set of maximum and minimum values for each parameter. The vector θ is an d dimensional domain where each element θ_i is bounded with θ_{max} and θ_{min} containing the upper bounds and lower bounds of the d parameters respectively i.e.
$\mathbf{S} = \left\{\theta \in \mathbf{R}^d | \theta_{min,i} \leq \theta_i \leq \theta_{max,i} \ \forall i = 1, 2, \cdots, d\right\}$.

1.2.2 Neural Networks

A neural network [9, 10] consists of a set of processing elements, also known as neurons or nodes, which are interconnected. It can be described as a directed graph in which each node i performs a transfer function $f$of the form

$$y_i = f\left(\sum_{j=1}^{n} w_{i,j} x_j - b_i\right) \tag{1.27}$$

where y_i is the output of the node i, x_j is the j^{th} input to the node i, and $w_{i,j}$ is the connection weight between nodes i and j, b_i is the threshold (or bias) of the node. Usually, f is nonlinear, such as a heaviside, sigmoid, or Gaussian function. NNs can be divided into feed-forward and recurrent classes according to their connectivity. A NN is feed-forward if there exists a method which numbers all the nodes in the network such that there is no connection from a node with a large number to a node with a smaller number. All the connections are from nodes with small numbers to nodes with larger numbers. A NN is recurrent if such a numbering method does not exist. In (1.27), each term in the summation only involves one input. The architecture of a NN is determined by its topological structure, i.e., the overall connectivity and transfer function of each node in the network. Learning NN is otherwise known as training of NN because the learning is achieved by adjusting the connection weights iteratively so that trained NN can perform certain tasks. This Learning is roughly divided into supervised, unsupervised, and reinforcement learning. Supervised learning is based on direct comparison between the estimated output of a NN and the desired correct output, also known as the target output. It is often formulated as the minimization of an error function such as the total mean square error between the actual output and the estimated output summed over all available data. A gradient descent based optimization algorithm such as backpropagation [64] can then be used to adjust connection weights in the NN iteratively in order to minimize the error. Reinforcement learning is a special case of supervised learning where the exact desired output is unknown. It is based only on the information of whether or not the actual output is correct. Unsupervised learning is solely based on the correlations among input data. No information on correct output is available for learning. The essence of a learning algorithm is the learning rule, i.e., a weight-updating rule which determines how connection weights are changed. Examples of popular learning rules include the delta rule and backpropagation. These will be discussed in chapter 2.

1.2.3 Evolutionary Algorithms

Evolutionary algorithms are based on computational models of fundamental evolutionary processes such as selection, recombination and mutation. Fig. 1.2 gives an overview of a general evolutionary algorithm. Individuals, or current approximations are encoded as strings composed over some alphabet(s), e.g. binary, integer, real valued etc., and an initial population is produced by randomly sampling these strings. Once a population has been produced it may be evaluated using an objective function which characterizes an individual performance in the problem domain. The objective function is also used as the basis for selection and determines how well an individual performs in its environment. A fitness value is then derived from the raw performance measure given by the objective function and is used to bias the selection process. Highly fit individuals will have a higher probability of being selected for reproduction than individuals with a lower fitness value. Therefore, the average performance of individuals can be expected to increase as the fitter individuals are more likely to be selected for reproduction and the lower fitness individuals get discarded. Selected individuals are then reproduced, usually in pairs, through the application of genetic operators. These operators are applied to pairs of individuals with a given probability and result in new offspring that contain material exchanged from their parents. The offspring from reproduction are then further perturbed by mutation. These new individuals then make up the next generation. These processes of selection, reproduction and evaluation are then repeated until some termination criteria are satisfied, e.g. a certain number of generations completed, a mean deviation in the performance of individuals in the population or when a particular point in the search space is reached.

```
procedure EA {
t = 0;
initialize P(t);
evaluate P(t);
while not finished do {
t = t + 1;
select P(t) from P(t-1);
reproduce pairs in P(t);
mutate P(t);
evaluate P(t);
}
}
```

Figure 1.2: A Simple Evolutionary Algorithm

In general, most real world optimization problems have several challenging properties. Almost of all problems have a significant number of local optima, and the search space can be so huge that the exact global optimum cannot be found in reasonable time. Further, the problems may have multiple conflicting objectives that should be considered simultaneously (e.g., cost versus quality). Moreover, there may be a number of nonlinear constraints to be fulfilled by the final solution. Furthermore, the problem may have dynamic components altering the location of the optimum during the optimization process. For some problems, variants of the local search approach have proven to be very efficient, e.g., Lin-Kernighan algorithm for the Traveling Salesman Problem. However, deterministic local search algorithms, such as steepest decent, do not allow a decrease in the solutions quality during the search. For this reason, these algorithms often stagnate at a local optimum, which makes local search less desirable for many real-world problems. Valuable alternatives are stochastic search methods such as simulated annealing, Tabu search, and evolutionary

algorithms. Among these techniques, EAs seem to be a particularly promising approach for several reasons. EAs are very general regarding the problem types they can be applied to continuous, mixed-integer and combinatoric type problems. Furthermore, these algorithms can easily be combined with existing techniques such as local search methods. In addition, it is often straightforward to incorporate domain knowledge in the evolutionary operators and in the seeding of the population. Moreover, EAs can handle problems with any combination of the above mentioned challenges in real-world problems i.e. local optima, multiple objectives, constraints, and dynamic components. In this connection, the main advantage lies in the EAs population-based approach. For local optima, the genetic diversity of the population allows the algorithm to explore several areas of the search space simultaneously. There is of course no guarantee on the premature convergence to a local optimum, but the population improves the EAs robustness on such problems. Naturally, EAs do also have some disadvantages. Unfortunately, they are rather computationally demanding, since many candidate solutions have to be evaluated in the optimization process. However, recently there has been an increase interest in dealing with this problem and some techniques have been suggested such a hybrid EAs to make it faster.

1.3 Literature Survey on System Identification

The theory of system identification for linear systems is matured during the last two decades and there exist useful tools based on Least Mean Square (LMS), Recursive Least Square (RLS), Kalman Filtering [1] etc. Many problems in control engineering, signal processing and machine learning can be cast as a system identification problem where the task is to determine a suitable model from a given set of input-output data. The resulting model can then be used for the prediction and control of a "black- box" system. In reality, however, all systems are more or less nonlinear. In recent years there has been a lot of research pursued on nonlinear system identification. A survey of existing techniques of

nonlinear system identification prior to 1980s is given by Billings [2], a survey of the structure detection of input-output nonlinear systems can be obtained in [5], and a survey of nonlinear black-box modeling in system identification can be found in [7]. Several methods have been developed for the identification and control of nonlinear system, including NARMAX, Hammerstein, Wiener or Hammerstein-Wiener structures, but these methods suffer the difficulty of representing the behavior of the system over its full range of operation [6]. For nonlinear system identification, NARX model has been implemented by the authors [4, 8]. The extra complexity associated with nonlinear system identification, particularly when there is no initial information or model structure detail. One successful approach to this problem is the orthogonal Least Squares Regression (LSR) method to find a suitable set of nonlinear terms for the system.

Since eighties, neural networks have been extensively applied to the identification of nonlinear dynamical systems. Most of the works are based on multilayer feed-forward neural networks with back-propagation learning algorithm [68, 70]. In neural network based identification, the selection of the number of hidden nodes and the number of hidden layers (i.e. the structure of the network) corresponds to the model selection stage. The network can be trained in a supervised manner with a back-propagation algorithm, which is based on an error-correction learning rule. The error signal is propagated backward through the network. The back-propagation algorithm utilizes gradient descent to determine the weights of the network and hence corresponds to the parameter estimation stage. Both feed-forward and recurrent networks can be used for identification purposes. The feed-forward network provides a nonlinear static map between inputs and outputs of the neural network. A number of theoretical and practical system identification problems have been solved using neural network approach with multi-layered perceptron (MLP) with back-propagation training [16, 17]. In [18] the author has used a radial basis function

neural network (RBFNN) for the nonlinear system identification problem. The Wavelet networks techniques [19] were also applied to system identification of nonlinear systems in which adaptive techniques such as back propagation algorithm found to provide better accuracy compared to non-adaptive ones such as Volterra series, Wiener-Hammerstein modeling and polynomial methods. A novel multilayer discrete-time neural network is presented in [21] for identification of nonlinear dynamical systems. In [22], a scheme for on-line states and parameters estimation of a large class of nonlinear systems using RBFNN has been designed. An approach to control nonlinear discrete dynamic systems, which relies on the identification of a discrete model of the system by a feed-forward neural network with one hidden layer, is presented in [23]. Nonlinear system identification using discrete-time recurrent single layer and multilayer NNs are studied in [24]. An identification method for nonlinear models in the form of Fuzzy-Neural Networks is introduced in [25]. This Fuzzy-Neural Networks combine fuzzy if-then rules with NNs. An adaptive time delay NN is used for identification of nonlinear systems in [26]. Onder and Kaynak in [27] investigated the identification of nonlinear systems by feed-forward NNs, radial basis function NNs and adaptive neuro-fuzzy inference systems. Authors in [28] have discussed about a least squares support vector machine (SVM) regressor used for generating the control actions, while an SVM-based tree-type neural network is used as the critic.

However, the complexity and the combinatorial growth in the search space mean that exhaustive search is not always feasible and is limited in application. Conventional training algorithms mainly rely on gradient based techniques. Although these techniques suffice in many applications, they require a differentiable performance index or a smooth search space. This condition may not always be satisfied in practical applications because of noisy data or system discontinuity. Even when the derivative or gradient information is available these techniques often result in a local optimum if the solution space

is multi-modal. They may also fail completely, if the space is noisy as found in practical applications. These problems can become more complex if the plant to be identified is multiple-input and multiple-output system. Further, the following difficulties exist with conventional techniques.

- Initial information of the parameters usually need to be known a priori.
- The estimation may be biased if the measurement or process noise is correlated.
- It is difficult to identify the transport delays.
- Input-Output data at steady state may cause problems in matrix manipulations, as they have very close value.
- It is difficult to estimate parameters that are not linearly separable.

Compared to the conventional approaches which search the term space iteratively, building a more and more complex model, the EA based approach conducts a global and robust search of the model space. Thus, the EA has the potential to be more effective in identifying a suitable model structure and hence more general in nature. Contribution to the system identification using EAs is discussed in [29, 30, 31, 32, 33, 34, 35]. In [29] the authors proposed a genetic algorithm based on NARX system identification algorithm. In [30] an inversion control of nonlinear system with an inverse NARX model identification using genetic algorithms has been proposed. Authors in [31] have implemented nonlinear system identification using a subset selection method and LSR using GAs. In [32] the authors have discussed about the identification of structural system using an evolutionary strategy. Models for evolutionary algorithm and their application in system identification are addressed in [33]. Authors in [35] have implemented the genetic algorithms to estimate the Parameter of a robot arm. The GA is used to select a fixed number of terms from a set of possible nonlinear terms and LSR is used to identify the parameters of those terms. Because the EA operates on a population of solution

estimates, the EA produces a family of low-variance models which can be assessed according to different criteria before a final model is chosen. From the previous neural network system identification approaches, it is observed that even neural network has been proved to be a successful technique for nonlinear system identification but there still remains little concern about its convergence and problem of being trapped at local minima. Evolutionary neural networks (EANNs) refer to a special class of neural networks in which evolution is another fundamental form of adaptation in addition to learning [36, 37].

In [38], the author has applied genetic algorithms to obtain the values of the weights of both the feed-forward and feedback connections. It describes the use of genetic algorithms to train the Elman and Jordan networks for dynamic systems identification. In [39] a genetic algorithm is proposed to design wavelet neural networks (WNNs) for nonlinear system identification. By introducing a connection switch to each link between a wavelet and an input node, the decomposition is done automatically during the evolutionary process. GA is used to train the wavelet parameters and the connection switches. In this way, both the structure and wavelet parameters of WNNs are optimized simultaneously. Evolving wavelet neural networks for system identification is discussed by the authors [40]. A new encoding scheme for training RBF networks by genetic algorithms is proposed by the authors [41]. In the proposed encoding scheme, both the architecture (numbers and selections of nodes and inputs) and the parameters (centers and widths) of the RBF networks are represented in one chromosome and evolved simultaneously by GAs so that the selection of nodes and inputs can be achieved automatically. The performance and effectiveness of the presented approach are evaluated using two benchmark time series prediction examples and one practical application example, and are then compared with other existing methods. It is shown by the simulation tests that the developed evolving RBF networks are able to predict the time series accurately with the automatically selected nodes and inputs. In [42], both off-line

architecture optimization and on-line adaptation have been developed for a dynamic neural network (DNN) in nonlinear system identification. A series of GA operations are applied to the connection matrices to find the optimal number of neurons on each hidden layer and interconnection between two neighboring layers of DNN. The hybrid training is adopted to evolve the architecture, and to tune the weights and input delays of DNN by combining GA with the modified adaptation laws. The modified adaptation laws are subsequently used to tune the input time delays, weights and linear parameters in the optimized DNN-based model in on-line nonlinear system identification. An approach to nonlinear system identification using evolutionary Neural Networks and LMS algorithm has been proposed by the authors in [43]. A PSO tuned radial basis function network model is proposed for identification of nonlinear systems in [44]. At each stage of orthogonal forward regression (OFR) model construction process, PSO is adopted to tune one RBF unit's centre vector and diagonal covariance matrix by minimizing the leave-one-out (LOO) MSE. In [45] the author has presented a learning algorithm for dynamic recurrent Elman neural networks based on a modified particle swarm optimization. The proposed algorithm has been applied to perform speed identification and to design a controller to perform speed control for Ultrasonic Motors (USM). The contribution in [46] concerns with the design of a generalized functional-link neural network with internal dynamics and its applicability to system identification by means of multi-input single output nonlinear models of autoregressive with exogenous inputs type. A GA based evolutionary multi-objective optimization in the Pareto-sense is used to determine the optimal architecture of that dynamic network. The contributions in [47] proposed the application of a modified artificial immune network inspired optimization method - the opt-aiNet - combined with sequences generate by Henon map to provide a stochastic search to adjust the control points of a B-spline neural network (BSNN). The numerical results presented here indicate that artificial immune network optimization methods are useful for building good BSNN model for the nonlinear identification of

two case studies: (i) the benchmark of Box and Jenkins gas furnace, and (ii) an experimental ball-and-tube system. Authors in [48] outlined the basic concept and principles of two simple and powerful swarm intelligence tools: the PSO and the BFO. The adaptive identification of an unknown plant has been formulated as an optimization problem and then solved using the PSO and BFO techniques. Using this approach efficient identification of complex nonlinear dynamic plants have been carried out through simulation study. One such evolutionary computation i.e. DE, was first introduced in [49], is successfully applied to many artificial and real world optimization problems with applications. A differential evolution based neural network training algorithm was first introduced in [50]. Authors in [51] proposed an effective DE based learning algorithm for recurrent fuzzy neural network (RFNN) with fuzzy inputs, fuzzy weights and biases, and fuzzy outputs. The effectiveness of the proposed method is illustrated through simulation of benchmark forecasting and identification problems and comparisons with the existing methods. The suggested approach has also been used for real applications in an oil refinery plant for petrol production forecasting.

The major disadvantage of the EANN [36, 37] approach is that it is computationally expensive and has slow convergence. With a view to speed up the convergence of the search process, a number of different gradient methods such as LM and BP are combined with evolutionary algorithms. These are the new class of hybrid algorithms i.e. global evolutionary search supplemented by local search techniques. It may be noted that the local search methods when used alone there may be problem for getting trapped in local minima. The hybridization of these local searches with evolutionary techniques is useful to either accelerate the discovery of good solutions, for which evolution alone would take too long to discover, or to reach solutions that would otherwise be unreachable by evolution or a local method alone. It is assumed that the evolutionary search provides for a wide exploration of the search space while

the local search can somehow zoom-in on the basin of attraction of promising solutions. The natural analogies between human evolution and learning, i.e. EAs and neural networks prompted a great deal of research into the use of hybrid algorithms such as memetic algorithms to evolve the design of NNs.

Memetic algorithms (MAs) have been proven very successful across a wide range of problem domains such as combinatorial optimization [57], optimization of non-stationary functions [52], multi-objective optimization [53], bioinformatics [54], etc. MAs have received various names throughout the literature and scientist not always agree what is and what is not an MA due to the large variety of implementations available. Some of the alternative names used for this search framework are hybrid GAs, Baldwinian EAs, Lamarckian EAs, genetic local search algorithms, etc to cover a wide range of techniques where evolutionary-based search is augmented by the addition of one or more phases of local search. Research in memetic algorithms has progressed substantially, and several Ph.D. dissertations have been written analyzing this search framework and proposing various extensions to it [55, 56, 57, 58]. In [59], the authors have proposed an effective PSO based memetic algorithm for designing artificial neural network where an effective adaptive Meta-Lamarckian learning strategy is employed to decide which local search method to be used so as to prevent the premature convergence and concentrate computing effort on promising neighbor solutions. Authors in [60] propose two hybrid evolutionary algorithms as alternatives to improve the training of dynamic recurrent neural networks.

However, a lot more research is needed to achieve the faster convergence and obtaining global minima. Hence there has been a great interest in combining training and evolution with neural networks in recent years.

1.4 Objectives of the Thesis

The objectives of the thesis are as follows

- To develop efficient nonlinear system identification algorithms using evolutionary computing techniques and neural networks.
- To combine evolutionary algorithm and gradient descent (GD) learning for overcoming the problems of local minima during training of the NNs with GD learning.
- To prove the convergence of the proposed neuro-evolutionary hybrid system identification algorithms.
- To achieve improved identification of nonlinear systems including multi-input multi-output (MIMO) systems introducing a memetic differential evolution algorithm and to compare its performance with other memetic algorithms.
- To devise a new variant differential evolution algorithm with improved search ability for identifying different types of nonlinear systems.
- To propose opposition based mutation differential evolution algorithm based identification algorithm with application to nonlinear system for estimating the parameters of an induction motor.

1.5 Motivation of the Present Work

Determination of efficient structure and weights of a NN become a challenge in the field of nonlinear system identification. The other challenge of applying evolutionary NN is that most evolutionary algorithms [71, 72, 75, 157, 158, 159] will not provide good optimal performance if not fine-tuned in local search although they are good at the global search. Hybridization can improve the efficiency of evolutionary training by incorporating a local search procedure such as BP [78], LM [69] or other random search algorithm into the evolution,

i.e., combining global search ability with local search's ability to fine tune.

1.6 Thesis Organization

Chapter 1, gives an overview of system identification techniques and discusses about the application of neural networks and evolutionary computation for system identification. It describes techniques that are suitable for building models of nonlinear systems. This chapter reviews various representations used to describe linear systems and the methods used to identify them from measurements of input-output data. Subsequently, it considers different descriptions of nonlinear systems and the techniques used in system identification. It also describes about the nonlinear system identification using linear-in-parameter models such as NARMAX modeling followed by nonlinear- in-parameter models i.e. neural models. Subsequently it discusses the contribution of the thesis followed by organization of this thesis.

Chapter 2, starts with discussion about the NNs and their training. A gradient descent-based optimization algorithm such as back-propagation is discussed which is used to adjust connection weights in the NN iteratively in order to minimize the training errors. A weight-updating rule i.e. delta rule which determines how connection weights are changed has been discussed next. Subsequently, the chapter focuses on different types of evolutionary algorithms and population based search strategies i.e. how individuals in a population compete and exchange information with each other in order to perform certain tasks. The essence of this chapter is to discuss about finding a near-optimal set of connection weights for a neural network. The chapter is concluded with discussions about the need of improving training of neural networks so that optimal set of connection weights can be achieved.

Chapter 3, proposes a nonlinear system identification scheme using DE, neu-

ral network and LM. Here, DE and LM in a combined framework are used to train a NN for achieving better convergence of neural network weight optimization. In this chapter the LM is used as a local optimizer after the DE algorithm. This type of algorithms are known as SH where set of algorithms is applied one after another, each using the output of the previous as its input. As DE becomes slow near the basin of the global optimization the function of LM is to enhance the speed of convergence. In this chapter number of examples including a practical case-study has been considered for implementation of this algorithm.

Chapter 4, describes a memetic algorithm approach for the training of artificial neural networks, i.e. how memetic algorithm trained MLP applied to nonlinear system identification. The MAs are used as an alternative to gradient search methods, such as BP, which have shown limitations when dealing with rugged landscapes with many poor local optima. The work described in this chapter aims at designing a training strategy that is able to cope up with difficult error manifolds, and to achieve perfectly trained neural networks that produce small training errors. A rigorous study on the identification of a nonlinear system using seven different algorithms namely BP, GA, PSO, DE, genetic algorithm back-propagation (GABP), paricle swarm optimization back-propagation (PSOBP) along with the proposed differential evolution back-propagation (DEBP) approaches has been done. In the proposed system identification scheme, three global searches have been combined with the gradient descent method i.e. the BP algorithm to overcome the slow convergence of the evolving neural networks. The local search BP algorithm is used as an operator like crossover and mutation operator for GA, PSO and DE. These algorithms have been tested on standard benchmark problems given in [61, 62] for nonlinear system identification to prove their efficacies.

Chapter 5, presents identification of a 1DOF experimental aerodynamic test

rig, a twin rotor multi-input multi-output (TRMS) using SH and MA. The TRMS is a highly nonlinear system which can be considered as an experimental model of a complex air vehicle. Such vehicles are required to be identified precisely to ensure satisfactory control performance to meet the demand for automation. This implies that linear characterization of aircraft's is not good to describe the systems characteristics for control purposes and nonlinear modeling techniques are required. Neural network based nonlinear characterization are promising approaches. This chapter focuses into the development of nonlinear modeling of a TRMS using SH and MA. The system is modeled using a NARX identification scheme with a feed-forward neural network. In this chapter the responses of all the identified models are compared with that of the real TRMS to validate the accuracy of the models.

Chapter 6, discusses a new variant of the DE called ODE. This ODE is combined with LM algorithm for training the feed-forward neural networks applied to nonlinear system identification. The ODE uses opposition based learning that considers simultaneously estimate and its corresponding opposite estimate (i.e., guess and opposite guess) in order to achieve a better approximation for the current candidate solution. The proposed combined opposition based differential evolution neural network (ODE-NN) has been applied to systems given in [61, 62] results obtained envisage that the ODE-NN approach to identification of nonlinear system exhibits better model identification accuracy compared to differential evolution neural network (DE-NN) approach. This ODE-NN approach is applied to obtain dynamics of a twin rotor 1 DOF MIMO system which is usually highly nonlinear.

Chapter 7, describes how DE technique can also be applied to estimate the parameters of a physical system for example the rotor resistance (R_r), stator resistance (R_s), leakage inductance (L_l) and magnetising inductance (L_m) of a three-phase induction machine. Along with a view to obtain better estimation

results different variants of DE such as OMDE are also investigated. A set of steady state equations of the induction motor under consideration were developed to be used for the simulation of this DE based optimization problem. For accomplishing the DE and OMDE based system identification of the induction motor parameters, excitation to the stator and rotor speed were considered as the input and output data respectively.

Chapter 8, summarizes the work described in the thesis. This chapter also includes a brief note on scope of further research that can be pursued in future as extension of this thesis work.

Chapter 2

Neural Networks and Evolutionary Computation Approaches

2.1 Introduction

Neural networks [9, 10] were first studied to understand and imitate the function of the human brain. They consist of highly interconnected processing elements known as neurons that have the ability to respond to input stimuli and to learn to adapt to the environment. They have the advantageous capabilities of learning from training data, recalling memorized information, and generalizing to the unseen patterns. These capabilities do show great potential in such application areas as control [11], signal processing [12], and pattern recognition [13]. There are more than hundred neural network structures and algorithms proposed from varying standpoints [14]. However, the most widely used neural networks are limited to just a few. This chapter describes how training of NNs can be accomplished using evolutionary computing techniques with reference to their applications to system identification. A gradient descent-based optimization algorithm such as back-propagation is discussed which is used to adjust connection weights in the NN iteratively to minimize the error. The chapter discusses different types of evolutionary algorithms that are population based search/optimization such as GA, PSO, BFO and DE. In these individuals in a population compete and exchange information with each

other in order to perform the search tasks. The chapter discusses about finding a near-optimal set of connection weights for a NN using EAs. The scope of formalising of hybrid training for neural networks for finding out best set of connection weights are also hilighted.

Several approaches such as [36, 37] have been applied for combining evolutionary algorithms and neural networks. The most successful of these are hybrid algorithms that combine an evolutionary algorithm with a gradient-descent based training algorithm to optimize the NN weights. Their success lies in the combination of efficient global search with an efficient local search. Gradient-descent based NN training algorithms require the error of an NN over the input training patterns to be computable. The error of an NN is the result of a mean square, sum square or root mean square of the difference between the actual and the expected outputs. It requires the expected output to be known, which makes it most suitable to supervised learning tasks. In NN/evolutionary algorithm approaches applied to supervised learning this error value is often used as the fitness function. However, this is not a mandatory requirement for evolutionary algorithms, where the fitness function can be any appropriate measure of fitness.

2.2 Feed-forward Neural Networks

Figure 2.1 shows the schematic of a single neuron which takes multiple inputs, sum them and then apply an activation function to the sum before putting it as output. The information is stored in the weights. The weights can be positive (excitatory), zero or negative (inhibitory).

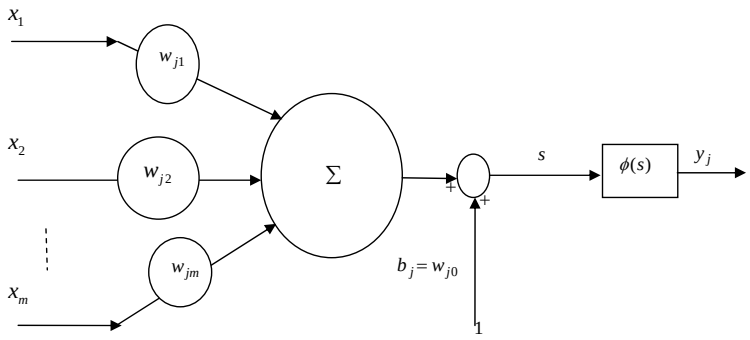

Figure 2.1: Schematic of a single neuron

The argument s of the activation (or squashing) function $\phi(s)$ is related to the inputs through

$$s_j = \sum_{i=0}^{m} w_{j,i} x_i = \sum_{i=1}^{m} w_{j,i} x_i + b_j \tag{2.1}$$

where b_j is the threshold, which is considered to be an additional input of magnitude 1 and weight b_j. x_j is the input to the neuron j. The output of the neuron j is given by $y_j = \phi(s_j)$. The activation functions with range [0; 1] (binary) and [-1; 1] (bipolar) that are normally used are shown in Table 2.1. The constant c represents the slope of the sigmoid functions, and very often it is taken to be unity. The activation function should not be linear so that the effect of multiple neurons cannot be easily combined. For a single neuron the net effect is $y_j = \phi\left(\sum_{i=0}^{m} w_{j,i} x_i\right)$.

2.2.1 Training Artificial Neural Networks

Single-layer feed-forward NN

This is also called a single layer perceptron. Figure 2.2 shows the schematic diagram of a single layer feed-forward neural network.

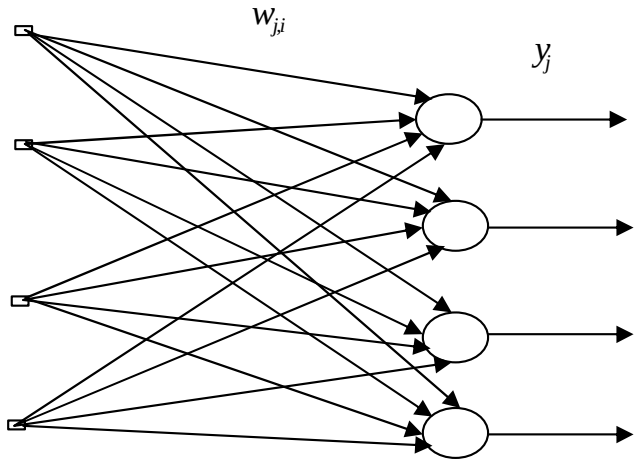

Figure 2.2: Schematic of a single layer network

Multilayer feed-forward NN

MLP is a completely connected feedforward neural network having number of layers. By properly selecting the number of hidden neurons, and the activation function in the hidden layer (i.e., sigmoid) and in the output layer (i.e., purely linear), the output of the MLP can be calculated as follows.

$$y_j = \phi\left(\sum_{k=0}^{m} w_{j,k} h_k\right) = \phi\left(\sum_{k=0}^{n} w_{j,k} \sum_{i=0}^{m} \phi\left(w_{k,i}^{h} x_i\right)\right)$$

where,

$$\phi(s) = 1/\left[1 + \exp(-cs)\right]$$

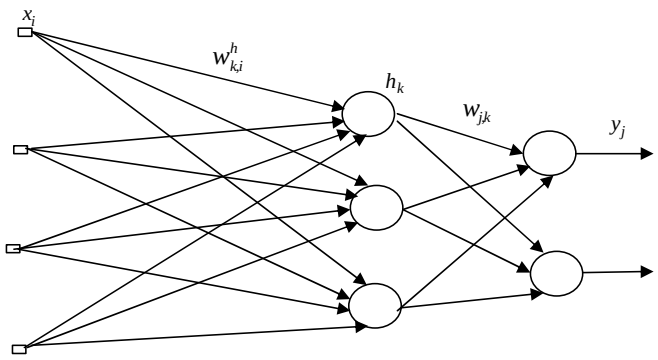

Figure 2.3: Schematic of a multi layer network

Table 2.1: Commonly used activation functions

Function	binary $\phi(s)$	bipolar $\phi(s)$
Step (Heaviside, threshold)	1 if $s > 0$ 0 if $s \leq 0$	1 if $s > 0$ 0 if $s = 0$ -1 if $s < 0$
Piecewise linear	1 if $s > 0.5$ $s + 0.5$ if $-0.5 \leq s \leq 0.5$ 0 if $s < 0.5$	1 if $s > 0.5$ $2s$ if $-0.5 \leq s \leq 0.5$ 0 if $s < 0.5$
Sigmoid	$\{1 + \exp(-cs)\}^{-1}$ Logistic	$\tanh(cs/2)$

2.2.2 Learning Rules

Learning is an adaptive procedure by which the weights are systematically changed under a given rule. Learning in networks may be of the unsupervised, supervised, or reinforcement type. In unsupervised learning the network is also called a self-organizing network. It is provided with a set of data within which patterns or other characteristic features are to be found out. The output of the network is not known and there is no feedback from the environment. The objective is to understand the input data better or extract some information from it. In supervised learning, on the other hand, there is a set of input-outputs pairs called the training set which the network tries to adapt itself to. There is also reinforcement learning with input-output pairs where the change in the weights is evaluated to be in the right or wrong direction. Figure 2.4 gives the flow chart for the neural network learning process.

Delta rule

This is also called the error-correction learning rule. If y_j is the output of a neuron j when the desired value should be d_j, then the error is $e_j = d_j - y_j$. The weights $w_{j,k}$ leading to the neuron are modified in the following manner $\Delta w_{j,k} = \eta e_j h_k$. The learning rate η is a positive value that should neither be too large to avoid runaway instability, nor too small to take a long time for

convergence. One possible measure of the overall error is $E = \frac{1}{2}\sum (e_j)^2$, where the sum is over all the output nodes. For simplicity, we will use the logistics activation function $y = \phi(s) = \frac{1}{1-e^{-s}}$. This has the following derivative

$$\frac{dy}{ds} = \frac{e^{-s}}{(1+e^{-s})^2} = y(1-y)$$

Back-propagation Algorithm

According to the delta rule, $\Delta w_{j,k} = \eta \delta_j h_k$, where δ_j is the local gradient. We will consider neurons that are in the output layer and then those that are in hidden layers.

Neurons in output layer:

If the target output is d_j and the actual output is y_j, then the error is $e_j = d_j - y_j$. The squared output error summed overall output neuron is

$$E = \frac{1}{2}\sum (e_j)^2$$

We can write

$$y_j = \phi\left(\sum_{k=0}^{m} w_{j,k} h_k\right) = \phi\left(\sum_{k=0}^{n} w_{j,k} \sum_{i=0}^{m} \phi\left(w_{k,i}^{h} x_i\right)\right)$$

The rate of change of E with respect to the weight is

$$\begin{aligned}\left(\frac{\partial E}{\partial w_{j,k}}\right) &= \left(\frac{\partial E}{\partial e_j}\right)\left(\frac{\partial e_j}{\partial y_j}\right)\left(\frac{\partial y_j}{\partial h_k}\right)\left(\frac{\partial h_k}{\partial w_{j,k}}\right) \\ &= (e_j)\,(-1)\,(\phi'(y_j))\,(h_k)\end{aligned}$$

Using a gradient descent algorithm $\Delta w_{j,k}$ can be written as

$$\begin{aligned}\Delta w_{j,k} &= -\eta \frac{\partial E}{\partial w_{j,k}} \\ &= \eta e_j \phi'(y_j) h_k \\ &= \eta \delta_j h_k\end{aligned}$$

where $\delta_j = e_j \cdot y_j \cdot (1 - y_j)$ is known as the error term.

Neurons in hidden layer:

Consider neurons k in the hidden layer connected to neurons j in the output

layer. Then the squared error is $E = \frac{1}{2}\sum (e_k)^2$. The error term for the respective hidden unit can be computed as $\delta_k^h = h_k\,(1 - h_k)\sum_{j=1}^{l}(\delta_j w_{j,k})$. The network ia updated using the following rule

$$w_{k,i} = w_{k,i} + \Delta w_{k,i}$$

where, $\Delta w_{k,i} = \eta \delta_k^h x_i$. The local gradients in the hidden layer can thus be calculated from those in the output layer. Figure 2.4 gives the flow chart for learning process.

back-propagation with momentum

Although frequent updates provide a speed up thr back-propagation, there are still other algorithms that can be used for improved performance. One of these is known as standard momentum. Momentum term is a locally adaptive approach to update the weights of a neural network. At the moment of using the momentum the update rule is to be modified as

$$\Delta w_{k,i}(t) = \eta \delta_k^h x_i + \alpha \Delta w_{k,i}(t-1)$$

where α is the momentum term. Adding the momentum term will typically results in aspeed up of the training for many applications. Although momentum speeds up the performence, but an improper selection of the momentum term causes a network to diverge.

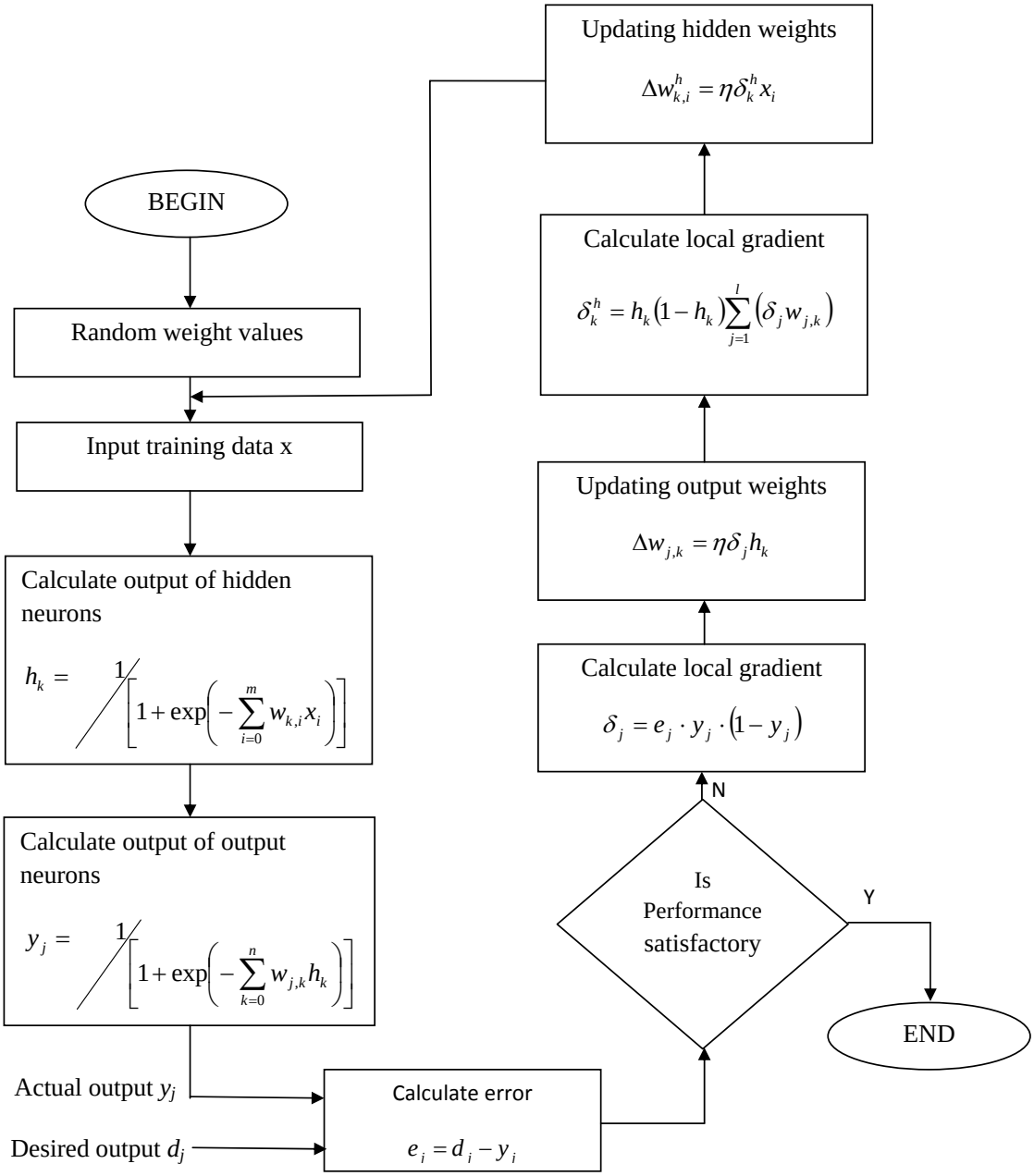

Figure 2.4: Flow chart for learning process

2.3 Variants of Evolutionary Algorithms

The origin of evolutionary algorithms has dated back to early fifties. The earliest EAs that predominated in many engineering and related applications are GA, GP, evolutionary strategy (ES) and evolutionary programming (EP). Each

of these varieties implements an evolutionary algorithm in a different manner which includes the choices of representation for the individual structures, types of selection mechanism used, forms of genetic operators, and measures of performance.

2.3.1 Genetic Algorithms

Genetic Algorithms are adaptive methods which may be used to solve search and optimization problems. Over many generations, natural populations evolve according to the principles of natural selection and survival of the fittest. The basic principles of GAs [112, 114] were first laid down rigorously by Holland, in mid sixties. Thereafter, many researchers have contributed to develope this field. There are many variations of the genetic algorithm but the basic form is the simple genetic algorithm (SGA). The working principle of SGA is described in the following section.

Coding

Before a GA can run, a suitable coding for the problem must be devised. It is assumed that a potential solution to a problem may be represented as a set of parameters. These parameters (known as 'genes') are joined together to form a string of values (often referred as 'chromosome' or 'Individual'). Binary coded strings having ones and zeros are mostly used. For example, if 8 bits are used to code each variable in a two-variable function optimization problem, chromosome would contain two genes, and consists of 16 binary digits. Length of the string is determined according to the desired solution accuracy. It is not necessary to code all variables in equal sub-string length.

Fitness function

As pointed out earlier, GAs mimic the survival of the fittest principle of nature to make a search process. Therefore, GAs are naturally suitable for solving maximization problems. Minimization problems are usually transformed in to maximization problems by suitable transformation. In, general, a fitness

function is first derived from the objective function and used in successive genetic operations. Certain genetic operators require that the fitness function be nonnegative, although certain operators do not have this requirement. For maximization problems, the fitness function can be considered to be the same as the objective function. For minimization problems, the fitness function is an equivalent maximization problem chosen such that the optimum point remains unchanged.

GA operators

The GA works with a set of individuals comprising the population. The initial population consists of P randomly generated individuals, where, P is the size of population. At every iteration of the algorithm, the fitness of each individual in the current population is computed. The population is then transformed in stages to yield a new current population for the next iteration. The transformation is usually done in three stages by sequentially applying the following genetic operators:

Selection : In the first stage, the selection operator is applied as many times as there are individuals in the population. In this stage every individual is replicated with a probability proportional to its relative fitness in the population. The population of P replicated individuals replaces the original population.

Crossover: In the next stage, the crossover operator is applied with a probability p_c, independent of the individuals to which it is applied. Two individuals (parents) are chosen and combined to produce two new individuals (offsprings). The combination is done by choosing at random a cutting point at which each of the parents is divided into two parts; these are exchanged to form the two offsprings which replace their parents in the population. This is known as single point crossover.

Mutation: In the final stage, the mutation operator changes the values in

a randomly chosen location on an individual with a probability p_m.

Convergence

If the GA has been correctly implemented, the population will evolve over successive generations so that the fitness of the best and the average individual in each generation increases towards the global optimum. The algorithm converges after a fixed number of iterations and the best individual generated during the run is taken as the solution. Table 2.2 describes the similarities and dissimilarities between GA and other evolutionary algorithms.

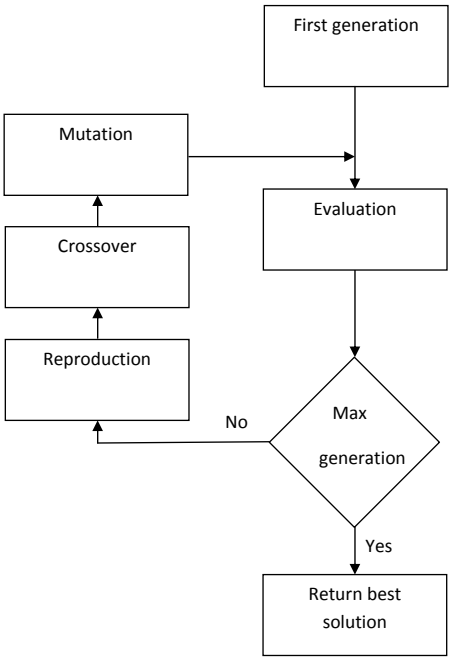

Figure 2.5: Flow chart for GA

Table 2.2: Similarities and dissimilarities between GA and other evolutionary algorithms

Genetic Algorithm	Evolutionary Programing	Evolutionary Strategies
First formulated by Holland for adaptive search and by his students for optimization from mid 1960s to mid 1970s.	First proposed by Fogel et al. in mid 1960s for simulating intelligence.	First proposed by Rechenberg and Schwefel in mid 1960s for numerical optimisation.
Binary strings are used extensively as individuals (chromosomes).	Finite state machines (FSMs) are used to represent individuals, although real-valued vectors have always been used in numerical optimisation.	Real-valued vectors are used to represent individuals.
Simulate Darwinian evolution.	It is closer to Lamarckian evolution.	They are closer to Larmackian evolution.
Search operators are only applied to the genotypic representation (chromosome) of individuals.	Search operators (mutations only) are applied to the phenotypic representation of individuals.	They do have recombination.
Emphasise the role of recombination (crossover). Mutation is only used as a background operator.	It does not use any recombination.	They use self-adaptive mutations.
Often use roulette-wheel selection.	Usually use tournament selection.	

2.3.2 Bacterial Foraging Optimization

The details of bacteria foraging optimization are given in [161]. A group of bacteria move in search of food and away from noxious elements known as foraging. All bacteria try to move upward the food concentration gradient individually. At the initial location they measure the food concentration and then tumble to take a random direction and swim for a fixed distance and measure the concentration there. This tumble and swim make one chemotactic

step. If the concentration is greater at next location, then they take another step in that direction. When concentration at next location is lesser than that of previous location, they tumble to find another direction and swim in this new direction. This process is carried out up to a certain number of steps, which is limited by the lifetime of the bacteria. At the end of its lifetime the bacteria that have gathered good health that are in better concentration region divide into two cells. Thus in the next reproductive step the next generation of bacteria start from a healthy position. The better half reproduces to generate next generation where as the worse half dies. This reproduction step is also carried out a fixed number of times. In the optimization technique we can take the variable we want to optimize as the location of bacteria in the search plane (the plane where the bacteria can move). The specifications such as number of reproductive steps, number chemotactic steps which are consisted of run (or swim) and tumble, swim length, maximum allowable swims in a particular direction are given for a particular problem then the variable can be optimized using this Bacteria Foraging Optimization technique. The *E.coli* bacteria that are present in our intestines have a foraging strategy governed by four processes, namely, chemotaxis, swarming, reproduction, and elimination and dispersal.

Chemotaxis:

This process is achieved through swimming and tumbling. Depending upon the rotation of the flagella in each bacterium, it decides whether it should move in a predefined direction (swimming) or an altogether different direction (tumbling), in the entire lifetime of the bacterium. To represent a tumble, a unit length random direction, $\phi(j)$ say, is generated; this will be used to define the direction of movement after a tumble. In particular,

$$\theta^i\left(j+1,k,l\right)=\theta^i\left(j,k,l\right)+ST(i)\phi(j) \tag{2.2}$$

where $\theta_i(j,k,l)$ represents the i^{th} bacterium at j^{th} chemotactic k^{th} reproductive, and l^{th} elimination and dispersal step. $ST(i)$ is the size of the step taken

in the random direction specified by the tumble. ST is termed as the run length unit.

Swarming:

It is always desired that the bacterium that has searched the optimum path of food should try to attract other bacteria so that they reach the desired place more rapidly. Swarming makes the bacteria congregate into groups and hence move as concentric patterns of groups with high bacterial density. Mathematically, swarming can be represented by

$$\begin{aligned} J_{st} &= \sum_{i=1}^{P} J_{st}^{i}\left(\theta, \theta^{i}(j,k,l)\right) \\ &= \sum_{i=1}^{P}\left[-d_{attract}\,\exp\left(-\omega_{attract}\sum_{m=1}^{d}\left(\theta_m-\theta_m^{i}\right)^2\right)\right]+ \\ &\quad \sum_{i=1}^{P}\left[h_{repellent}\,\exp\left(-\omega_{repellent}\sum_{m=1}^{d}\left(\theta_m-\theta_m^{i}\right)^2\right)\right] \end{aligned} \tag{2.3}$$

where $J_{st}(\theta, P(j,k,l))$ is the cost function value to be added to the actual cost function to be minimized to present a time varying cost function. P is the total number of bacteria. d is the number of parameters to be optimized that are present in each bacterium. $d_{attract}$, $\omega_{attract}$, $h_{repelent}$ and $\omega_{repelent}$ are different coefficients that are to be chosen judiciously.

Reproduction:

The least healthy bacteria die, and the other healthiest bacteria each split into two bacteria, which are placed in the same location. This makes the population of bacteria constant.

Elimination and dispersal:

It is possible that in the local environment, the life of a population of bacteria changes either gradually by consumption of nutrients or suddenly due to some other influence. Events can kill or disperse all the bacteria in a region. They have the effect of possibly destroying the chemotactic progress, but in contrast, they also assist it, since dispersal may place bacteria near good food sources.

Elimination and dispersal helps in reducing the behavior of stagnation (i.e., being trapped in a premature solution point or local optima).

2.3.3 Particle Swarm Optimization

Particle swarm optimization [44, 157] is a stochastic global optimization method which is based on simulation of social behavior. As in GA and ES, PSO exploits a population of potential solutions to probe the search space. In contrast to the aforementioned methods in PSO no operators inspired by natural evolution are applied to extract a new generation of candidate solutions. Instead of mutation PSO relies on the exchange of information between individuals, called particles, of the population, called swarm. In effect, each particle adjusts its trajectory towards its own previous best position, and towards the best previous position attained by any member of its neighborhood. In the global variant of PSO, the whole swarm is considered as the neighborhood. Thus, global sharing of information takes place and particles profit from the discoveries and previous experience of all other companions during the search for promising regions of the landscape. Several variants of PSO [45] have been proposed up to date, following Eberhart and Kennedy who were the first to introduce this method. Initially, assuming that the search space is d dimensional, so the i^{th} particle of the swarm is represented by a d dimensional vector $X_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ and the best particle of the swarm, i.e. the particle with the lowest function value, is denoted by index g. The best previous position (i.e. the position corresponding to the best function value) of the i^{th} particle is recorded and represented as $pos_i = (pos_{i1}, pos_{i2}, \cdots, pos_{id})$ and the position change (velocity) of the i^{th} particle is $V_i = (v_{i1}, v_{i2}, \cdots, v_{id})$. The particles are manipulated according to the following equations (the superscripts denote the iteration):

$$V_i^{k+1} = \chi \left(w V_i^k + c_1 r_{i_1}^k \left(pos_i^k - X_i^k \right) + c_2 r_{i_2}^k \left(pos_g^k - X_i^k \right) \right) \tag{2.4}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{2.5}$$

where $i = 1, \cdots, P$, and P is the size of the population which is used to control and constrict velocities; w is the inertia weight; c_1 and c_2 are two positive constants, called the cognitive and social parameter respectively; r_{i_1} and r_{i_2} are random numbers uniformly distributed within the range [0, 1]. Eq. (2.4) is used to determine the i^{th} particle's new velocity, at each iteration, while Eq. (2.5) provides the new position of the i^{th} particle, adding its new velocity, to its current position. The performance of each particle is measured according to a fitness function. In optimization problems, the fitness function is usually identical with the objective function under consideration. The role of the inertia weight (w) is considered important for the PSOs convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. A large inertia weight facilitates exploration while a small one tends to facilitate exploitation, current search area. A proper value for the inertia weight w provides balance between the global and local exploration ability of the swarm, and, thus results in better solutions.

2.3.4 Differential Evolution

Price and Storn developed differential evolution [49] to be a reliable and versatile function optimizer that is also easy to use. Like nearly all EAs, DE is a population-based optimizer that attacks the starting point problem by sampling the objective function at multiple, randomly chosen initial points. Each vector is indexed with a number from 0 to number of population i.e.P. DE generates new points that are perturbations of existing points, but these deviations are not the samples from a predefined probability density function, like those in the ES. Instead, DE perturbs vectors with the scaled difference of two randomly selected population vectors. To produce the trial vector, DE adds the scaled, random vector difference to a third randomly selected population vector. In the selection stage, the trial vector competes against the population vector of the same index in which the vector with the lower objective function value is marked as a member of the next generation. Once the last trial vector has been tested, the survivors of the competitions become parents for

the next generation in the evolutionary cycle. DE [50, 51] is capable of handling non-differentiable, nonlinear and multimodal objective functions which has been used to train neural networks having real and constrained integer weights. In a population of P potential solutions to an optimization problem within an d-dimensional search space, a fixed number of vectors are randomly initialized, then evolved over time to explore the search space and to locate the minima of the objective function. In DE, individuals are represented as real-valued vectors. For each generation of the evolution process, each individual (target individual) of the population competes against a new individual (trial individual) for survival to the next generation. Only the fitter of the two survives. The trial individual is created by recombining the target individual with another individual created by mutation (mutant individual). Mutation is performed on the best individual found so far in the evolution process. For each target vector $x_{i,g}$ a mutant vector is produced using the following formula

$$\begin{aligned} v_{i,g+1} &= x_{r_1,g} + F z_2 \\ z_2 &= (x_{r_2,g} - x_{r_3,g}) \end{aligned} \tag{2.6}$$

where $i, r_1, r_2, r_3 \in \{1, 2, \cdots, P\}$ are randomly chosen and must be different from each other. Considering equation (2.6), it can be seen that the probability density function (PDF) of the differential population z_2 used during the mutation changes automatically as the generation proceeds and eventually solution converges towards the global minimum. Referring Fig. 2.6 where five populations $x_1, x_2, \cdots, x_5$ produce ten numbers of vector differences in one direction and twenty numbers in both directions which is shown in Fig. 2.7. Similarly for P populations there will be $\frac{P(P-1)}{2}$ vector differences in one direction and $P(P-1)$ in both directions. This implies that the mean of the PDF is always zero and the shape of the distribution changes automatically in successive generations depending on the surface of the objective function being searched which is shown in Fig. 2.8.

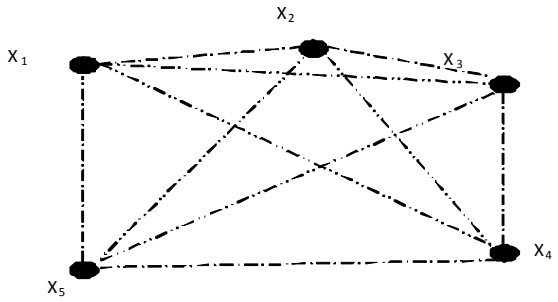

Figure 2.6: Five Populations

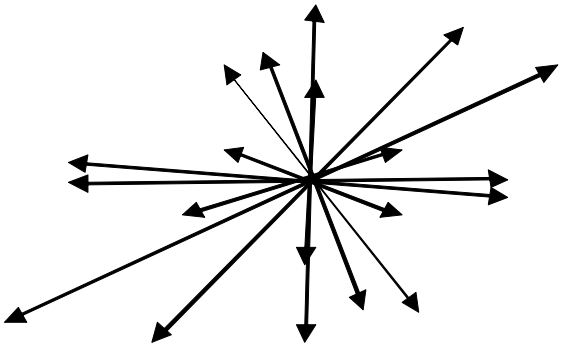

Figure 2.7: Twenty vector differences

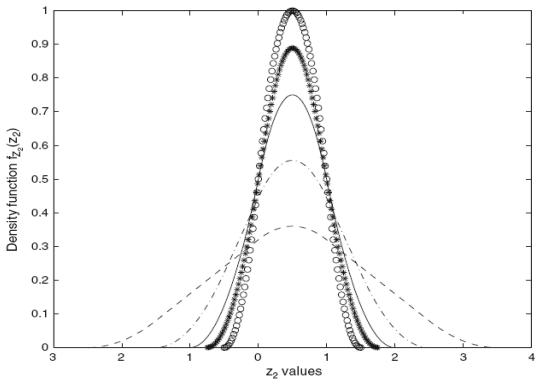

Figure 2.8: Probability density function

In equation (2.6), F is the mutation factor. Recombination creates an offspring (trial individual) by selecting parameters from either the target individual or the mutant individual. There are two methods of recombination

in DE, namely, binomial recombination and exponential recombination. In binomial recombination, a series of binomial experiments are conducted to determine which parent contributes which parameter to the offspring. Each experiment is mediated by a crossover constant, $C, (0 \leq C \leq 1)$. Starting at a randomly selected parameter, the source of each parameter is determined by comparing C to a uniformly distributed random number from the interval [0, 1). If the random number is greater than C, the offspring gets its parameter from the target individual; otherwise, the parameter comes from the mutant individual. In exponential recombination, a single contiguous block of parameters of random size and location is copied from the mutant individual to a copy of the target individual to produce an offspring. A vector of solutions are selected randomly from the mutant individuals when $rand_j$ ($rand_j \in [0, 1]$, is a random number) is less than C. This last operator is referred to as a selection. There are many different variants of DE the variants are as follows. DE/best/1/exp, DE/rand/1/exp, DE/rand-to-best/1/exp, DE/best/2/exp, DE/rand/2/exp. Now we explain the working steps involved in employing a DE cycle.

Step 1: Parameter setup

Choose the parameters of population size P, the mutation factor F, the crossover rate C, and the stopping criterion of the maximum number of generations g and the boundary constraints of optimization variables.

Step 2: Initialization of the population

Set generation $g = 0$. Initialize a population of individuals individuals (real-valued d-dimensional solution vectors) with random values generated according to a uniform probability distribution in the d dimensional problem space. These initial values are chosen randomly within user defined bounds.

Step 3: Evaluation of the population

Evaluate the fitness value of each individual of the population. If the fitness satisfies predefined criteria save the result and stop, otherwise go to step 4.

Step 4: Mutation operation (or differential operation)

Mutation is an operation that adds a vector differential to a population vector of individuals. For each target vector a mutant vector is produced using the following relation,

$$v_{i,g} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g}) \tag{2.7}$$

In Eqn. (2.7), F is the mutation factor, which provides the amplification to the difference between two individuals so as to avoid search stagnation and it is usually taken in the range of [0, 1], where are randomly chosen numbers but they must be different from each other. is the number of population.

Step 5: Recombination operation

Following the mutation operation, recombination is applied to the population. Recombination is employed to generate a trial vector by replacing certain parameters of the target vector with the corresponding parameters of a randomly generated donor (mutant) vector.

$$t_{j,i,g} = \begin{cases} v_{j,i,g} \text{ if } (rand_j \leq C) \text{ or } j = j_{rand} \\ x_{j,i,g} \text{ otherwise} \end{cases} \tag{2.8}$$

$j = 1, 2, \cdots, d$ where d is the number of parameters to be optimized.

Step 6: Selection operation

Selection is the procedure of producing better offspring. If the trial vector has an equal or lower value than that of its target vector, it replaces the target vector in the next generation; otherwise the target retains its place in the

population for at least one more generation.

$$x_{i,g+1} = \begin{cases} t_{i,g}, & if\ f(t_{i,g}) \leq f(x_{i,g}) \\ x_{i,g}, & otherwise \end{cases} \tag{2.9}$$

Once new population is installed, the process of mutation, recombination and selection is replaced until the optimum is located, or a specified termination criterion is satisfied, e.g., the number of generations reaches a predefined maximum. The block diagram for DE is given in Fig.2.9 and and its pseudo code is given in algorithm1.

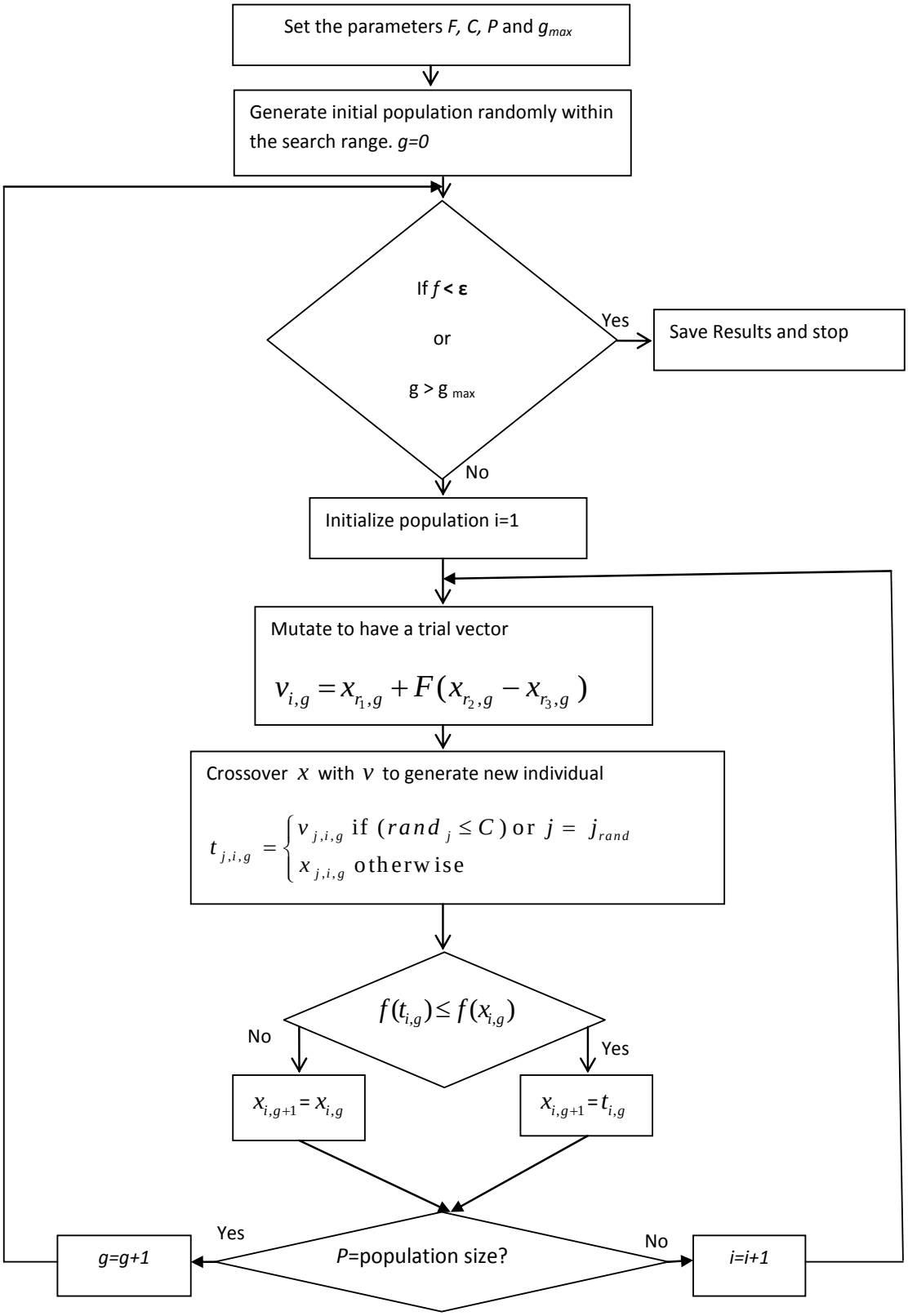

Figure 2.9: Block diagram for DE algorithm

Algorithm 1 DE Algorithm

Require: *pop*: initial population, F: Mutation constant, C: Cross over constant

while Convergence criteria not met **do**
  for $i = 0$ to P **do**
    $r_1 = rand(P)$
    $r_2 = rand(P)$
    $r_3 = rand(P)$
    $i \neq r_1 \neq r_2 \neq r_3$
    $v_{i,g} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g})$
    $t_{j,i,g} = \begin{cases} v_{j,i,g} \text{ if } (rand_j \leq C) \text{ or } j = j_{rand} \\ x_{j,i,g} \text{ otherwise} \end{cases}$
    if $f(t_{i,g}) \leq f(x_{i,g})$ **then**
      $x_{i,g+1} = t_{i,g}$
    else
      $x_{i,g+1} = x_{i,g}$
    end if
  end for
end while

2.4 Evolutionary Algorithms for Neural Networks Training

Neural networks have been applied to a variety of classification and learning tasks. Any nonlinear optimization method, a local or global one, can be applied to the optimization of feed-forward neural networks weights. The training performance varies depending on the objective function and underlying error surface for a given problem and network configuration. Since the gradient information of error surface is available for the most of the applied network configurations, the most popular optimization methods have been variants of gradient based back-propagation algorithms. Most training algorithms, such as BP and conjugate gradient algorithms [64, 65, 66, 67] are based on gradient descent. Naturally, local searches are fundamentally limited to local solutions, while global ones attempt to avoid this limitation. Widely applied methods for training the neural networks are, modified back-propagation [70], back-propagation using the conjugate-gradient approach [67], scaled conjugate-

gradient and its stochastic counterpart [65], the Marquadt algorithm [69], and a concept learning based back-propagation. Many of these gradient based methods are studied and discussed even for large networks in. Several methods have been proposed for network configurations where the gradient information is not available, such as simulated annealing for networks with non-differentiable transfer functions [83].

In many studies only small network configurations are considered in training experiments. Many gradient based methods and especially the Levenberg-Marquadt method are extremely fast for small networks. The disadvantage of using gradient based methods are that it often gets trapped in a local minimum of the error function and is incapable of finding a global minimum if the error function is multimodal and/or non-differentiable. One way to overcome the shortcomings of gradient descent based training algorithms is to adopt EANNs, i.e., to formulate the training process as the evolution of connection weights in the environment determined by the architecture and the learning task. EAs can then be used effectively in the evolution to find a near-optimal set of connection weights globally without computing gradient information. The fitness of a NN can be defined according to different needs. Two important factors which often appear in the fitness function are the error between target and actual outputs and the complexity of the NN. Because EAs can treat large, complex, non-differentiable, and multimodal spaces, which are the typical case in the real world, considerable research and application has been conducted on the evolution of connection weights [74, 75]. Several successful evolving neural approaches such as the EPNet [41] algorithm are hybrid algorithms combining gradient descent methods such as back-propagation learning for connection weight training with evolutionary structure search. In these approaches the fitness function is the generally taken as a mean or sum squared network output error over input training patterns. The field of EANNs can be divided into two major areas of research: the evolution of connection weights

and the evolution of both structure and connection weights. In the first area, the structure of neural networks is fixed before the evolution begins. In the second area, both the structure and the connection weights are determined automatically during the evolutionary process.

The combination of evolutionary algorithm and neural network for weight training consists of three major phases. The first phase is to decide the representation of connection weights, i.e., whether we use a binary strings form or directly use a real number form to represent the connection weights. The second step is the evaluation on the fitness of these connection weights by constructing the corresponding neural network through decoding each genome and computing its fitness function and mean square error function. The third one is applying the evolutionary process such as selection, crossover, and mutation operations by a evolutionary algorithm according to its fitness. The evolution stops when the fitness is greater than a predefined value (i.e., the training error is smaller than a certain value) or the population has converged.

2.4.1 A Comparison between Evolutionary Training and Gradient-Based Training

An advantage of using EAs for training neural networks is that it can handle the global search problem better in a vast, complex, multimodal, and non-differentiable surface. Also, EAs do not rely on calculating the gradient of the fitness function and thus is particularly appealing when this information is unavailable or very costly to obtain or estimate. Because of the stochastic nature of those algorithms the learning process can reach an optimal solution with much higher probability than many standard neural based techniques, which are based on the gradient information of the error surface. The evolutionary approach has been used to train recurrent NNs [75], higher order NNs [76] and fuzzy NNs [77]. Moreover, the same EA can be used to train many different networks regardless of whether they are feed-forward, recurrent, or higher order NNs. The general applicability of the evolutionary approach saves

a lot of human efforts in developing different training algorithms for different types of NNs. The evolutionary approach also makes it easier to generate NNs with some special characteristics. The main disadvantage of EAs is that it is usually slow in comparison with fast variants of BP and conjugate gradient algorithms [78]. However, EAs are generally much less sensitive to initial conditions of training. They always search for a globally optimal solution, while a gradient descent algorithm can only find a local optimum in a neighborhood of the initial solution.

2.5 Hybrid Training

One of the problems through the global search based techniques is the time complexity of the algorithm. Hence there was a further need for an improvement of this approach in terms of the time complexity by fine tuning the search within the local neighborhood area of the global solution obtained. Usually most EAs are rather inefficient in fine-tuned local search although they are good at global search. For getting the optimal connection weights a local search procedure should be incorporated into the evolution using the gradient descent algorithm to find the best connection weights at the local error surface. So the efficiency of evolutionary training can be improved significantly by incorporating a local search procedure into the evolution, i.e., combining EAs global search ability with local searchs ability to fine tune. EAs can be used to locate a good region in the space and then a local search procedure is used to find a near-optimal solution in this region. The local search algorithm could be BP, the variations of BP such as conjugate gradient or Levenberg-Marquardt algorithms. Hybrid training has been used successfully in many application areas [162]. In [163] the authors used GAs to search for a near-optimal set of initial connection weights and then used BP to perform local search from these initial weights. Their results showed that the hybrid GA/BP approach was more efficient than either the GA or BP algorithm used alone.

2.5.1 The Memetic Algorithm

A meme is defined to be a noun that conveys the idea of a unit of cultural transmission, or a unit of imitation [79]. In other words, a meme can be considered as any unit of information, observable in the environment. They are similar to genes in that they are self replicating, but memes differ from genes in that they are transmitted through imitation rather than being inherited. Furthermore, memes replicate in a Lamarckian manner (rather than in a Darwinian manner) in that changes to the meme during its lifetime are passed on. The population-based search algorithm such as GA, PSO and DE are commonly used to solve combinatorial optimization problems where the goal is to find the best solution in a (possibly unknown) solution space. It uses the principle of biological evolution to generate successively better solutions from previous generations of solutions. Memetic algorithm is an extension of the algorithms which incorporates a local-search algorithm for each solution in between generations. According to Pastorino [80], MA is able to improve convergence time, hence making it more favorable over the population based algorithms. In MAs local search is performed in between each generation, in addition to the techniques operators used by the EA. For this reason, Memetic Algorithm is also known as Hybrid-Algorithm [81]. Local search is performed to improve the fitness of the population (in a localized region of the solution space) so that the next generation has better genes from its parents, hence the claim that memetic algorithms can reduce convergence time. Memetic algorithms incorporate the concept of memes by allowing individuals to change before the next population is produced. Individuals may copy parts of genes from other individuals to improve their own fitness.

The local search algorithm adopted in a memetic algorithm is somewhat dependent on the problem being solved. When memes are transmitted, changes to them are also passed on. Memes affect the behavior of an individual, and do not modify the genes themselves. However, as a practical issue, a meme in

a memetic algorithm must be able to modify genes in order to improve fitness during local search. The hybridization is meant to either accelerate the discovery of good solutions, for which evolution alone would take too long to discover, or to reach solutions that would otherwise be unreachable by evolution or a local method alone.

2.6 EC+NN System Identification

Figure 2.10 shows an EC+NN identification scheme where the neural identifier is used to estimate the plant. Evolutionary computing techniques such as GA, PSO and DE are used as optimization algorithm to train the weights of the NN.

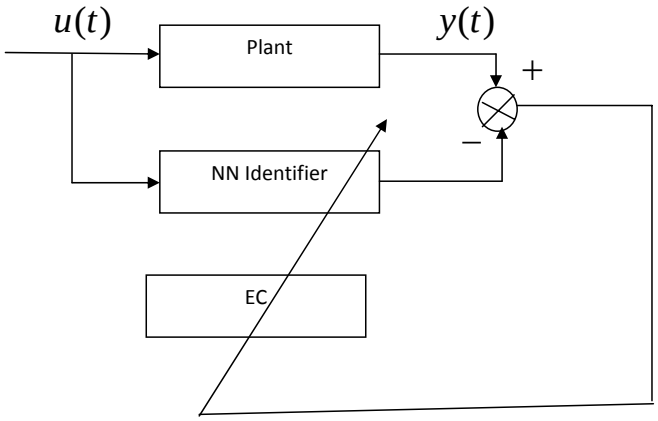

Figure 2.10: EC+NN identification scheme

2.7 Summary

This chapter reviews types of feed-forward neural network, its learning and evolutionary computing techniques. It also presents how to combine the evolutionary computing techniques with the conventional gradient based algorithms. Now a days hybrid algorithms are as good as many advanced EAs in handling noisy problems. One of the hybrid algorithm known as memetic algorithm is discussed here which is used to train the weights of a NN for reducing the convergence time.

Chapter 3

A Differential Evolution Trained Neural Network Scheme for Nonlinear System Identification

3.1 Introduction

This chapter describes a nonlinear system identification scheme using differential evolution, neural network and Levenberg Marquardt algorithm. Here, DE and LM in a combined framework are used to train a neural network for achieving better convergence of neural network weight optimization. In this work , we have combined an EA i.e. DE with a local optimizer i.e. LM in a sequential manner. This type of algorithms are known as sequential hybridization algorithm where set of algorithms is applied one after another, each using the output of the previous as its input. EAs more or less simulate a natural process. As such, they possess a certain dynamics which is inherent to the process, regardless of details related to solution representation or the way solutions are acted upon by operators. Starting with a random population, all its individuals are said to converge i.e. to become more and more uniform after a certain amount of time. Numbers of options are available to hybridize different EAs and EAs with other local search algorithms. In this thesis, we have considered two types of algorithms i.e. sequential hybridization and memetic algorithms for training a MLPNN. The focus of this chapter is on SH algortihm whilst the later will be described in chapter 4.

Sequential hybridization

A fundamental and practical remark on EAs is that after a certain amount of time, the population is quite uniform and the fitness of the population is no longer decreasing. That is, the process has fallen into a basin of attraction from which it has a (very) low probability to escape. This point leads to raise two important issues as follows

- Once the solution falls in a basin, the algorithm is not able to know if it has found the optimal point, or if it has fallen into a local optimum. Hence, we need to find ways to escape the optimum in order to try to find another optimum.
- The exploitation of the already found basin of attraction has to be realized in order to find, as efficiently as possible, the optimal point in the basin.

The first point may be solved by restarting the EA with a new population hoping that the EA will not fall into the same basin. Eshelman [82] reports enhanced results using such a technique. This is quite natural since restarting is equivalent to performing several runs. Hence the odds to find the optimum are multiplied by the number of runs. With regards to the second point, it is experimentally clear that the exploitation of the basin of attraction that has been found may be more efficiently performed by another algorithm than by an EA. Hence, it is much more efficient to use a local search algorithm. This schema of algorithm is known as sequential hybridization. Basically, this phrase means that a set of algorithms is applied one after another, each using the output of the previous as its input, acting in a pipeline fashion. Many authors have used the idea of sequential hybridization. In [83], the authors introduce simulated annealing to improve the population obtained by an EA. In [84], the proposed algorithm starts from simulated annealing and uses EAs to enrich the solutions that have been found. In [85], two genetic algorithms

are pipelined to solve a problem of routing of macro-cell layouts.

In this chapter SH has been utilised as an optimisation algorithm for training NNs applied to nonlinear system identification problems. Here, we have hybridized a global search algorithm i.e. differential evolution with a local search algorithm i.e. LM algorithm for the training of artificial neural networks, more specifically Multilayer Perceptrons, applied to nonlinear system identification. As DE becomes slow near the basin of the global optimization, the function of LM is to enhance the speed of convergence. In this work number of examples including a practical case-study have been considered for implementation of this algorithm. As local search algorithms are greatly dependent on the initialization and the global search algorithms are inherently slow, this type of hybridization enhances the search performed either by the LM or DE alone.

In neural network training, hybrid algorithms can be introduced at various levels. At the lowest level, it can be introduced into weight training, where NN weights are obtained. At the next higher level, it can be introduced into neural network architecture adaptation, where the architecture (number of hidden layers, no of hidden neurons and node transfer functions) is found. At the highest level, the hybrid algorithms can be introduced into the learning mechanism. In weight training, EAs can be used effectively in the evolution to find a near-optimal set of connection weights globally without computing gradient information then a gradient descent algorithm finds the local optimum in a neighborhood of the EAs solution. The fitness of a NN can be defined according to different needs. Two important factors which often appear in the fitness (or error) function are the error between target and actual outputs and the complexity of the NN. The hybrid approach to weight training in NNs consists of three major phases. The first phase is to decide the representation of connection weights, i.e., whether in the form of binary strings or

real strings. The second one is the evolutionary process simulated by an EA, in which search operators such as crossover and mutation have to be decided in conjunction with the representation scheme. Different representations and search operators can lead to quite different training performance. The third phase decides when to apply the local search so that the algorithm can able to find the global optimum with faster convergence.

3.2 Identification Using Neural Network

In the past, most of the system identification problems exploit neural networks either multilayer perceptron neural network (MLPNN) or Radial basis function neural network. Typically, a MLPNN consists of at least two layers of neurons with weighted links connecting the output of neurons in one layer to the input of neurons in the next layer. The weights are updated as follows

$$w_{j,i}(k+1) = w_{j,i}(k) + \eta\delta_j(k)y_j(k) \tag{3.1}$$

where $w_{j,i}$ is the synaptic weight connecting the output of a neuron i to the input of neuron j at time k. η is the learning rate parameter and $\delta_j(k)$ is the local gradient of neuron j at time k. The learning parameter should be chosen to provide minimization of the total error function, E. However, for small η the learning process becomes slow and large value of η corresponds to fast learning but leads to oscillation that prevent the algorithm from converging to the desired solution. A wide class of nonlinear dynamic systems with an input u and an output y_p can be described by the model:
$\hat{y}_p(k) = f\left(\Re(k), \theta\right)$, where $\hat{y}_p$ is the output of the model, $\Re(k)$ is the regression vector and θ is the parameter vector. Depending on the choice of the regressors in $\Re(k)$, different models can be derived:

NFIR (Nonlinear Finite Impulse Response) model:

$$\Re(k) = (u(k-1), u(k-2), \cdots, u(k-n_u))$$

where n_u denotes the maximum lag of the input.

NARX (Nonlinear Auto Regressive with exogenous inputs) model:

$$\Re(k) = (u(k-1), u(k-2), \cdots, u(k-n_u), y_p(k-1), y_p(k-2), \cdots, y_p(k-n_y))$$

where n_y denotes the maximum lag of the output.

NARMAX (Nonlinear Auto Regressive Moving Average with exogenous inputs) model:

$$\Re(k) = (u(k-1), \cdots, u(k-n_u), y_p(k-1), \cdots, y_p(k-n_y), e(k-1), \cdots, e(k-n_e))$$

where $e(k)$ is the prediction error and n_e is the maximum lag of the error.

NOE (Nonlinear Output Error) model:

$$\Re(k) = (u(k-1), u(k-2), \cdots, u(k-n_u), \hat{y}_p(k-1), \hat{y}_p(k-2), \cdots, \hat{y}_p(k-n_y))$$

A neural network based scheme shown in Figure 3.1 is used in this chapter which is the general block scheme of the NARX model. In this type of identifier, the output of the neural-network almost coincides with the output of the plant after learning and the model of the plant is composed in the neural network. In this figure, error e_k denotes the difference between plant actual output and estimated output. Past values of input and output of the plant form the input vector to the neural network; $\hat{y}_p$ corresponds to estimate the plant output given by the neural network at any instant of time k.

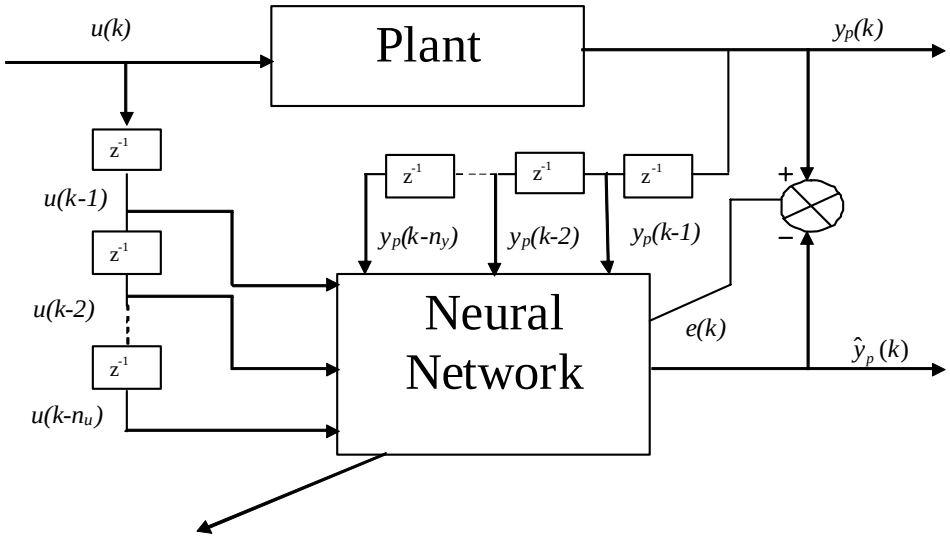

Figure 3.1: The general scheme for NARX model system identification

The two-layer feed-forward neural network with sigmoidal activation function in the hidden layer and linear activation function in output layer has the ability to approximate nonlinear function if the number of neurons in the hidden layer is sufficiently large. The feed-forward neural network (FNN) used in this work is shown in Figure (3.2). The inputs $u(k-1), u(k-2), ..., u(k-n_u)$ and outputs $y(k-1), y(k-2), \ldots, y(k-n_y)$ are multiplied with the weights $w_{u(i,j)}$ and $w_{y(i,j)}$ respectively, and summed at each hidden node. Then, the summed signal at a node activates a nonlinear function (sigmoid function). Thus, the output $\hat{y}_p$ at a linear output node can be calculated from its inputs as follows:

$$\hat{y}_p(k) = \sum_{i=1}^{n_h} w_{k,i} \frac{1}{1 + e^{-\left(\sum_{j=1}^{n_u} u(k-j) w_{u(i,j)} + \sum_{j=1}^{n_y} y(k-j) w_{y(i,j)} + b_i\right)}} + b \qquad (3.2)$$

where $n_u + n_y$ is the number of inputs, n_h is the number of hidden neurons, $w_{u(i,j)}$ is the first layer weight between the input $u(k-j)$ and the i^{th} hidden neuron, $w_{y(i,j)}$ is the first layer weight between the input $y(k-j)$ and the i^{th} hidden neuron, w_i is the second layer weight between the i^{th} hidden neuron and output neuron, b_i is a biased weight for the i^{th} hidden neuron and b is a biased weight for the output neuron. It can be seen from Figure 3.2 that the

FNN is a realization of the NARX model. The difference between the output of the plant $y(k)$ and the output of the network $\hat{y}_p$ is called the prediction error, $e(k) = y_p(k) - \hat{y}_p(k)$. This error is used to adjust the weights and biases in the network via the minimization of the error function $E = \frac{1}{2}\left[y_p(k) - \hat{y}_p(k)\right]^2$.

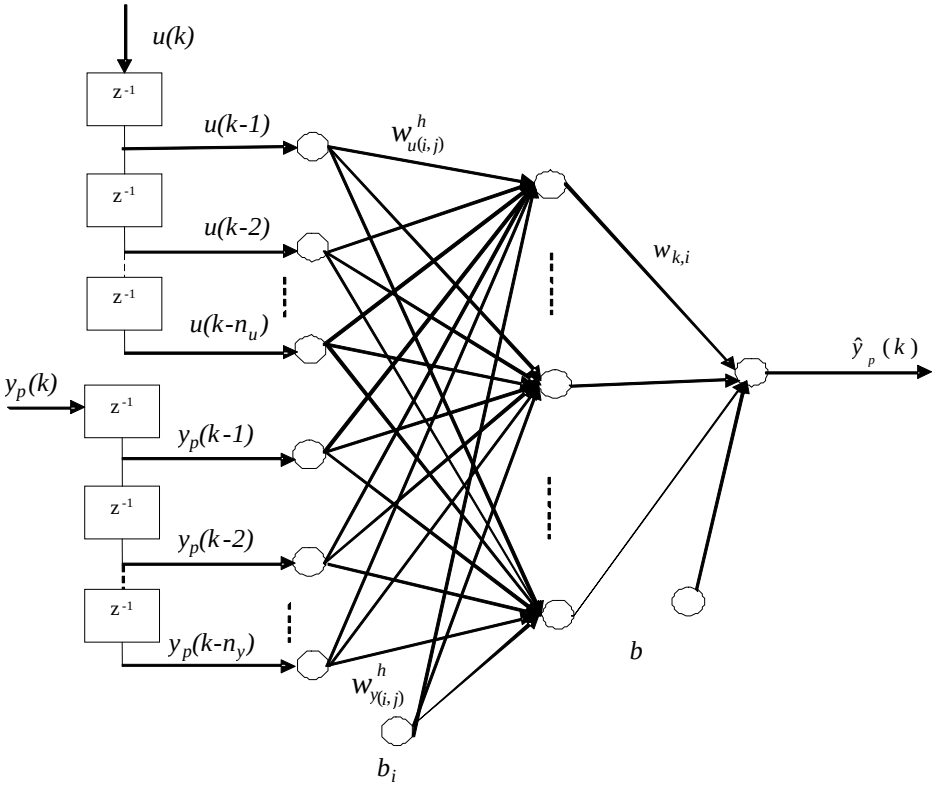

Figure 3.2: Structure of NN for NARX model system identification

We proposed here an improved neural network based nonlinear system identification scheme where the training of the NN employed for the identification has been made faster and accurate. This improved training algorithm was achieved by virtue of two important benefits of hybrid use of two different optimization schemes namely a stochastic evolutionary algorithm i.e. DE algorithm which provides a global search whilst the convergence of the proposed hybrid training has been accelerated by the gradient based optimization technique i.e. LM algorithm. We clearly demonstrated in this work that our proposed DE+LM+NN approach has been found to be successful in providing the best identification performance amongst the other varieties of system

identification schemes such as NN trained by LM, and NN trained by DE. In the proposed identification frame work, differential evolution is used only to find approximate values in the vicinity of the global minimum. These approximate weight values are then used as starting values for a faster convergence algorithm i.e. Levenberg-Marquardt algorithm

3.3 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm [69] is an approximation to the Newton method used also for training NNs. The Newton method approximates the error of the network with a second order expression, which contrasts to the Back-propagation algorithm that does it with a first order expression. LM is popular in the NN domain, although it is not that popular in the meta heuristics field. LM updates the NN weights as follows:

$$\Delta w = \left[J^T(w)\ J(w) + \mu I\right]^{-1} J^T(w)\, e(w) \tag{3.3}$$

where $J(w)$ is the Jacobian matrix of the error vector $e(w)$ evaluated in w, and I is the identity matrix. The vector error $e(w)$ is the error of the network for particular pattern i, that is $e(k) = y_p(k) - \hat{y}_p(k)$. The parameter μ is increased or decreased at each step. If the error is reduced, then μ is divided by a factor β, and it is multiplied by β in other case. Levenberg-Marquardt performs the steps detailed in Fig. 3.3. It calculates the network output, the error vectors, and the Jacobian matrix for each pattern. Then, it computes Δw using Eqn. (3.3) and recalculates the error with $w + \Delta w$ as network weights. If the error has decreased, μ is divided by β, the new weights are maintained, and the process starts again; otherwise, μ is multiplied by β, Δw is calculated with a new value, and it iterates again.

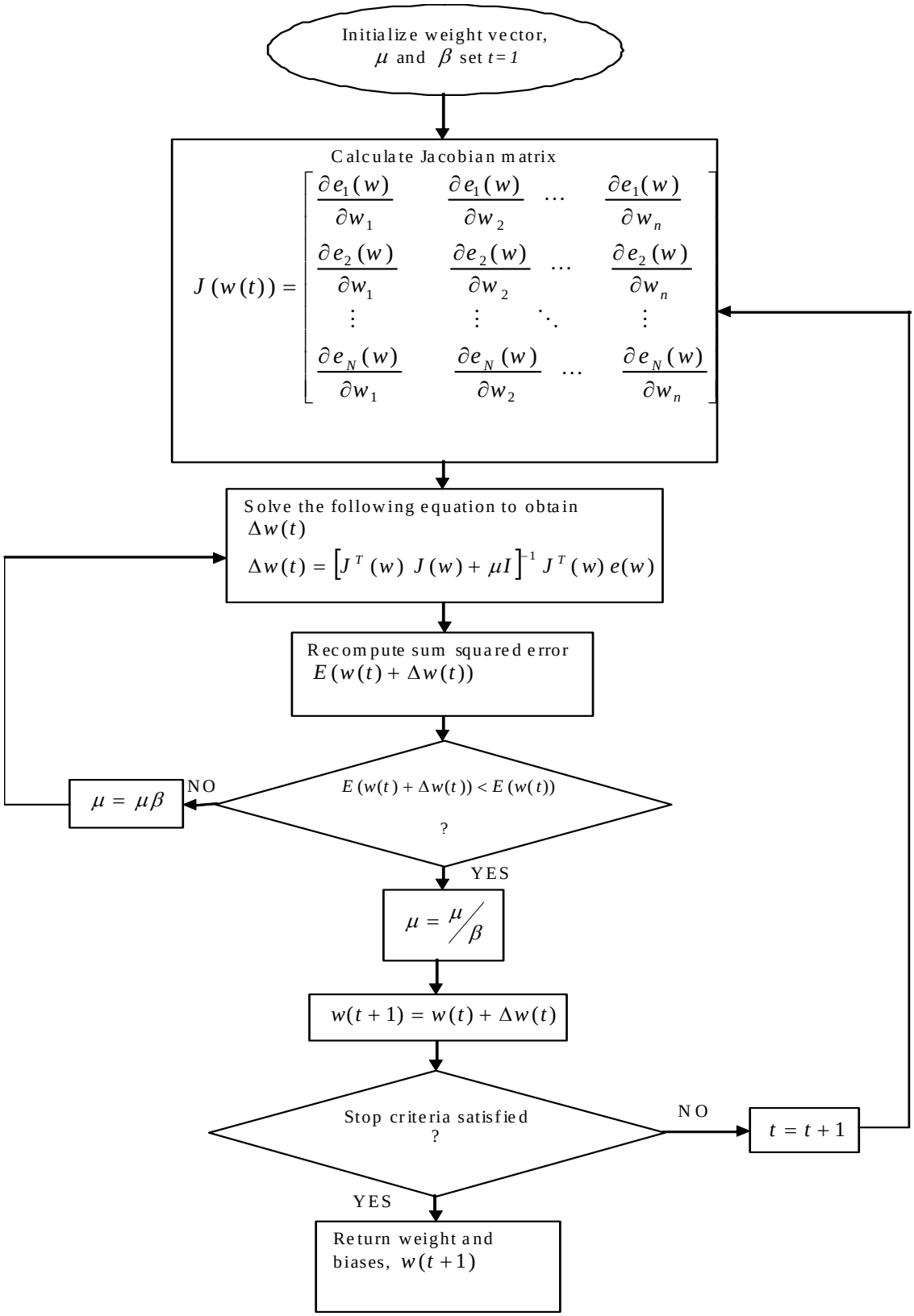

Figure 3.3: Flow chart for LM algorithm

The choice of an appropriate objective function is crucial for any data fitting procedure regardless of the optimization method used. The DE algorithm gives us a great deal of flexibility in this choice since we need only choose a continuous function and do not require the function to have continuous derivatives. The objective function chosen should be fast and simple to calculate.

3.4 Proposed DE+LM+NN Algorithm

In this section we describe how a DE is applied for training neural network in the frame work of system identification. DE can be applied to global searches within the weight space of a typical feed-forward neural network. Output of a feed-forward neural network may be considered as a function of synaptic weights $\mathbf{w}$ and input values $\mathbf{x}$, given by $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$.

The role of LM in the proposed algorithm is to fine tune the search within the local neighborhood area of the solution obtained by DE. In the training processes, given both the input vector $\mathbf{x}$ and the output vector $\mathbf{y}$ the synaptic weights in $\mathbf{w}$ are adapted to obtain appropriate functional mappings from the input $\mathbf{x}$ to the output $\mathbf{y}$. Generally, the adaptation can be carried out by minimizing the network error function $\mathbf{E}$ which is of the form. In this work we have taken $\mathbf{E}$ as mean squared error i.e. $\mathbf{E} = \frac{1}{N}\sum_{k=1}^{N}[\mathbf{y} - f(\mathbf{x}, \mathbf{w})]^2$, where N is the number of data considered. The optimization goal is to minimize the objective function $\mathbf{E}$ by optimizing the values of the network weights $\mathbf{w}$. where

$$\mathbf{w} = (w_1, \cdots, w_d)$$

The algorithm proposed is given below

Step 1.

Initialize population ***pop****: Create a population from randomly chosen object vectors with dimension P.*

$$\mathbf{P}_g = (\mathbf{w}_{1,g}, \cdots, \mathbf{w}_{P,g})\, g = 1, \cdots, g_{\max}$$
$$\mathbf{w}_{i,g} = (w_{1,i,g}, \cdots, w_{d,i,g})\, i = 1, \cdots, P$$

where d is the number of weights in the weight vector and in $\mathbf{w}_{i,g}$*, i is index to the population and g is the generation to which the population belongs.*

Step 2.

Evaluate all the candidate solutions inside ***pop*** *for a specified number of iterations.*

Step 3.

For each i^{th} *candidate in* ***pop*** *select the random variables* $r_1, r_2, r_3 \in \{1, 2, \cdots, P\}$

Step 4.

Apply mutation operator to each candidate in population to yield a mutant vector i.e.

$$v_{j,i,g+1} = w_{j,r_1,g} + F\left(w_{j,r_2,g} - w_{j,r_3,g}\right), \text{for} j = 1, \cdots, d$$

$$(i \neq r_1 \neq r_2 \neq r_3) \in \{1, \cdots, P\} \text{ and} F \in (0, 1+]$$

Step 5.

Apply crossover i.e. each vector in the current population is recombined with a mutant vector to produce trial vector.

$$t_{j,i,g+1} = \begin{cases} v_{j,i,g+1} & if\, rand_j[0,1) \leq C \\ w_{j,i,j} & otherwise \end{cases}$$

$where\ C \in [0, 1]$

Step 6.

Apply selection i.e. between the trial vector and target vector. If the target vector has an equal or lower objective function value than that of its target vector, it replaces target vector in the next generation; otherwise, the target retains its place in the population for at least one more generation

$$\mathbf{w}_{i,g+1} = \begin{cases} t_{i,g+1} \;\; if \;\; \mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}_{i,g+1})) \;\leq\; \mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}_{i,g})) \\ \mathbf{w}_{i,g} \quad otherwise \end{cases}$$

Step 7.

If $\mathbf{E} \leq \varepsilon$ *where* $\varepsilon > 0$ *go to step 8*

Step 8.

Initialize the weight matrix of Levenberg-Marquardt algorithm taking the values of weights obtained after the fixed number of iterations. Find out the value of $\mathbf{E}$*.*

Step 9.

Compute the Jacobian matrix $\mathbf{J}(w)$*.*

Step 10.

Find Δw *using the following equation*

$$\Delta w \;=\; \left[\mathbf{J}^T(w)\, J(w) + \mu I\right]^{-1} \mathbf{J}^T(w)\, \mathbf{E}$$

Step 11.

Recompute $\mathbf{E}$ *using* $w + \Delta w$ *if this new* $\mathbf{E}$ *is smaller than that computed in step 7 then reduce* μ *and go to step1. where* μ *is the damping factor.*

Step 12.

The algorithm is assumed to have converged when the norm of the gradient i.e. $\|\nabla \mathrm{E}\| = \left\|\mathbf{J}^T(w)\mathbf{y} - f(\mathbf{x}, w)\right\|$ *is less than some predetermined value, or when the sum of squares of errors has been reduced to some error goal.*

3.5 Convergence Analysis of DE

Let $\mathbf{P}_g = (\mathbf{w}_{1,g}, \cdots, \mathbf{w}_{P,g}),\ g = 1, \cdots, g_{\max}$ (where g_{max} is the maximum number of generation) be the random population of size P at step $g \geq 0$ and $F_g = \min\{f(\mathbf{P}_{g,i}) : i = 1, \cdots, P\}$ being the best fitness value within the population at step $g \geq 0$. As soon as the random variable F_g attains the value of the global optimum f^*, it is ensured that the population contains an individual representing the global solution of the minimization problem. Ideally, this event should happen after a finite number of steps with probability one regardless of the initialization of the DE algorithm.

Property 1

In DE, it is known that the best solutions found in the process of evolution in the current generation earned over to the next generation. This property of earning good solutions from the previous generation in the next generation guarantees that the global optimum will be found in finite time and never be lost once it has been found out. Thus, the property above shows that the random sequence $(F_g : g \geq 0)$ converges to the limit f^*. Let $(w_1, w_2, \cdots, w_n) \in P^n$ denote the population of parents known as target individuals. An offspring is produced as follows. At first, m number of parents are selected to serve as mates for the mutation process. This operation is denoted as folows

$$mat : P^n \rightarrow P^m$$

where $3 \leq m \leq n$. In calssical DE minimum 3 indivisuals are required for mutation operation so we have taken the minimum value of m as 3. These individuals are then mutated by the following procedure

$$mut : P^m \rightarrow P$$

Thus yielding a mutated individual. Finally by recombination with the

target individual yields a trial individual using the following process

$$reco : P \rightarrow P$$

Now the selection is done by the trial and target individual by a selection procedure given as

$$sec : P \rightarrow P$$

The above selection procedure decides which one will serve as the new parents in the next generation. Thus a single generation of differential evolution can be described as follows.

$$\forall i \in \{1, \cdots, P\} : \; t_i = reco(mut(mat(w_1, \cdots, w_n)))$$
$$\forall i \in \{1, \cdots, P\} : \; y_i = \{sel(w_i, t_i)\}$$

After this operational description differential evolution is in the position of defining some assumptions about the properties of the variation and selection operators.

Assumption (A1)

Every parent may be selected for mating and can be changed to an arbitrary other individual by a finite number of successive mutations. i.e. for every $w \in P$ there exists a finite path such that $p_r \{w_{i+1} = mut(w_i)\} = 1$ for $i = 1, \cdots, (g-1)$

Assumption (A2)

Every mutant individual is altered by the recombination with the minimum probability $p_{cr} > 0$

$$\forall i \in \{1, \cdots, P\} : \; p_r \{t_i = reco(mut(mat(w_1, \cdots, w_n)))\} \geq p_{cr} > 0$$

Assumption (A3)

Every trial individual competing for survival will survive with minimum probability of 0.5.

$$\forall i \in \{1, \cdots, P\} : \ p_r \{sel(w_i, t_i)\} = 0.5$$

Assumption (A4)

The best individual among the competitors in the selection process will survive with probability 1.

Theorem1(convergence analysis of the DE algorithm)

If the assumptions $(A1), (A2)$ and $(A3)$ are valid then the differential evolution visits the global optimum after a finite number of generations with probability one regardless of the initialization.

Definition

Let random variable $T = \{g \geq 0 : F_g = f^*\}$ denote the first hitting time of the global solution. An evolutionary algorithm is said to visit the global optimum in finite time with probability one if $p_r \{T < \infty\} = 1$ regardless of the initialization.

Proof

Let $P^* = \{w \in P : f(w) = f^*\}$ be the set of globally optimal solutions. Using assumption $(A1)$, $\exists$ a finite path from an arbitrary $w \notin P^*$ to some $w^* \in P^*$ that can be traversed by successive mutations. Let t_x be the length of the path between $w \notin P^*$ to the set $w^* \in P^*$.

Now consider an arbitrary parent of some population known as target individual. Assumption $(A1)$ ensures that this parent passes the mutation process

with every change with probability one. The probability that mutated individual transitions to the next point of the path towards $w^* \in P^*$ by recombination is guaranteed to be at least $p_{cr} > 0$ by assumption $(A2)$.

By virtue of assumption $(A3)$ this offspring will survive the selection process at least with probability $p_s = 0.5$. Thus the probability that parent $w \notin P^*$ transitions to a parent representing the next point on the path to $w^* \in P^*$ is at least $p_{cr} \times 0.5 > 0$. Consequently the probability that a globally optimal solution has not been found is $(1 - (p_{cr} \times 0.5))$. A g_w fold repetition of this argumentation shows that a globally optimal solution has not been found after g_w generation at most $(1 - (p_{cr} \times 0.5))^{g_w}$ which converges exponentially fast to zero as $g_w \to \infty$.This immediately implies $p_r \{T < \infty\} = 1$ where $T = \{g \geq 0 : F_g = f^*\}$ thus a global optimum will be visited for the first time after a finite number of iterations with probability one. This proves the theorem.

3.6 Results and Discussions

We present here the performance achieved through using the proposed DE+LM+NN scheme to a number of bench-mark problems as follows.

Example-1:

The nonlinear system [61] to be identified is expressed by

$$y_p(k+1) = \frac{y_p(k)[y_p(k-1)+2][y_p(k)+2.5]}{8.5 + [y_p(k)]^2 + [y_p(k-1)]^2} + u(k) \tag{3.4}$$

where $y_p(k)$ is the output of the system at the k^{th} time step and $u(k)$ is the plant input which is uniformly bounded function of time. The plant is stable at $u(k) \in [-2, 2]$.The identification model be in the form of

$$y_{pi}(k+1) = f(y_p(k), y_p(k-1)) + u(k) \tag{3.5}$$

where $f(y_p(k), y_p(k-1))$ is the nonlinear function of $f(y_p(k)$ and $y_p(k-1)$. The inputs to the neural network are $y_p(k)$ and $y_p(k-1)$. The output from neural network is $y_{pi}(k+1)$. The goal is to train the networks such that when an input $u(k)$ is presented to the network and to the nonlinear system, the network outputs $y_{pi}(k)$ and the actual nonlinear system output $y_p(k)$ will match as close as possible. Table 3.1 gives the parameters for simulation studies.

Table 3.1: Parameters for DE+LM+NN

Population size, P	50
Upper and lower bound of weights	[0 1]
Mutation constant factor , F	0.6
Cross over constant, C	0.5

NN identifier

The neural network identifier structure consisted of eleven numbers of neurons in the hidden layer. After 1000 epochs the training of the neural identifier has been stopped. After the training is over, its prediction capability has been tested for input.

$$\begin{cases} u(k) = 2\ \cos(2\pi k/100) & k \leq\ 200 \\ u(k) = 1.2\ \sin(2\pi k/20) & 200\ <\ k \leq\ 500 \end{cases} \tag{3.6}$$

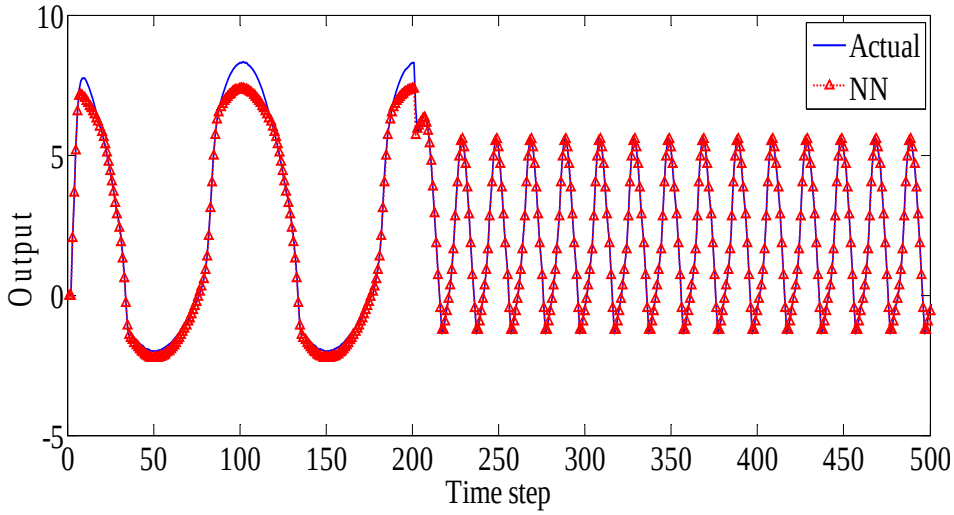

Figure 3.4: Identified and actual models (NN identifier) (Ex-1)

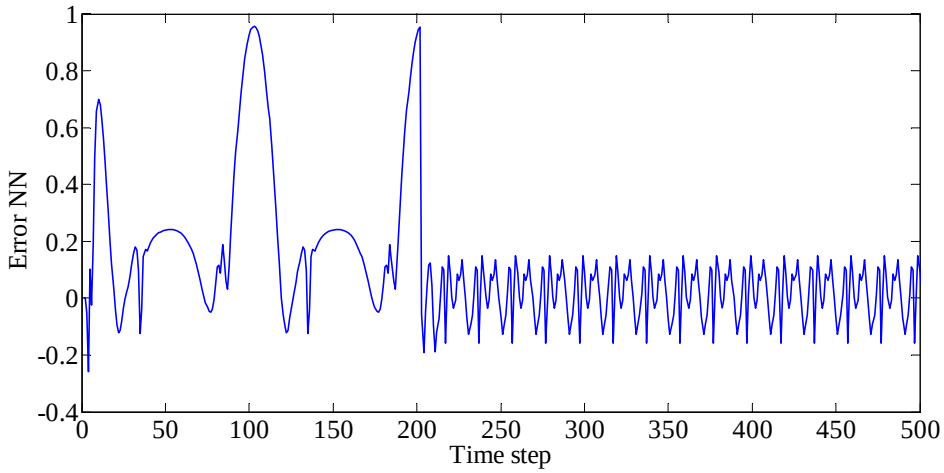

Figure 3.5: Error in modeling (NN identifier)(Ex-1)

Fig. 3.4 shows the system identification results obtained with using NN. The error is more at time steps 100 and 200. Fig. 3.5 shows the identification error.

DE+NN identifier

Figure 3.6 shows the identification performance of the system using DE+NN approach. Here the NN is trained by using differential evolution instead of classical ones such as gradient descent and Levenberg Marquardt algorithm.

The results obtained with DE+NN indicate no significant improvement over the previously discussed NN identifier. Here also eleven number of hidden layer neurons and thousands number of epochs were taken Fig. 3.7 shows the identification error in the case of DE+NN approach.

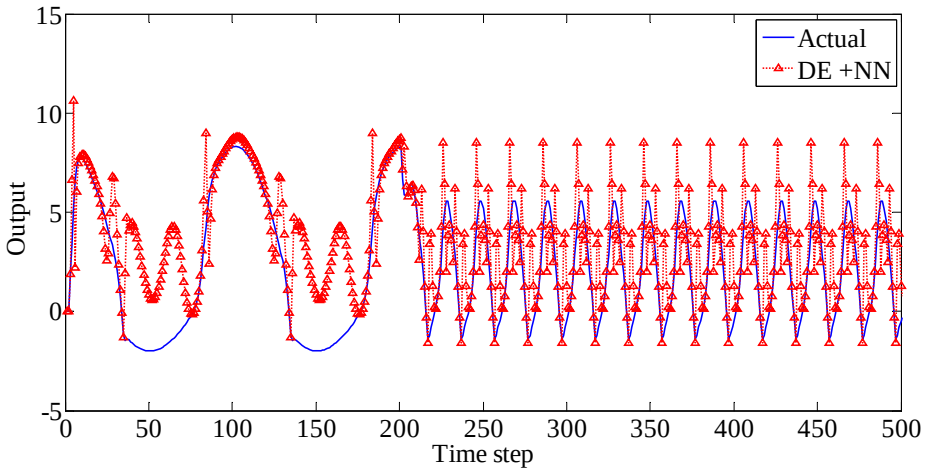

Figure 3.6: Identified and actual models (DE+NN identifier)(Ex-1)

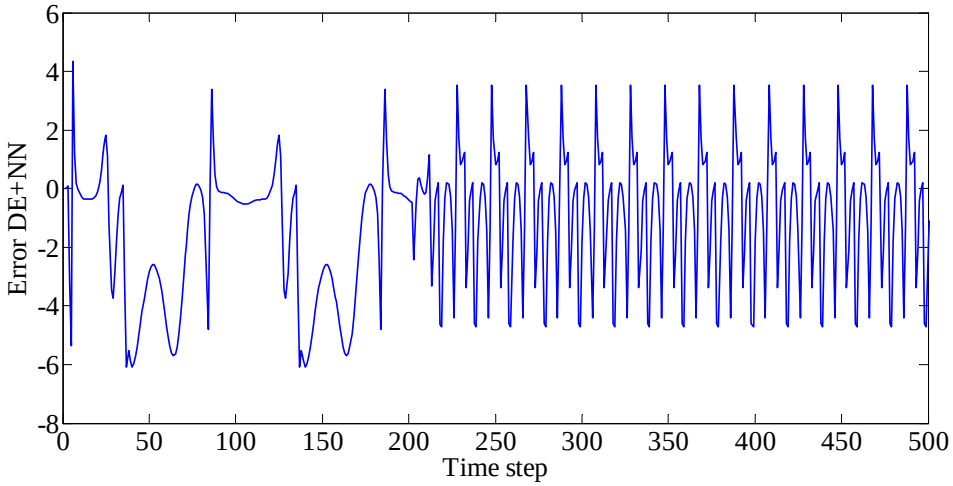

Figure 3.7: Error in modeling (DE+NN identifier)(Ex-1)

DE+NN+LM identifier

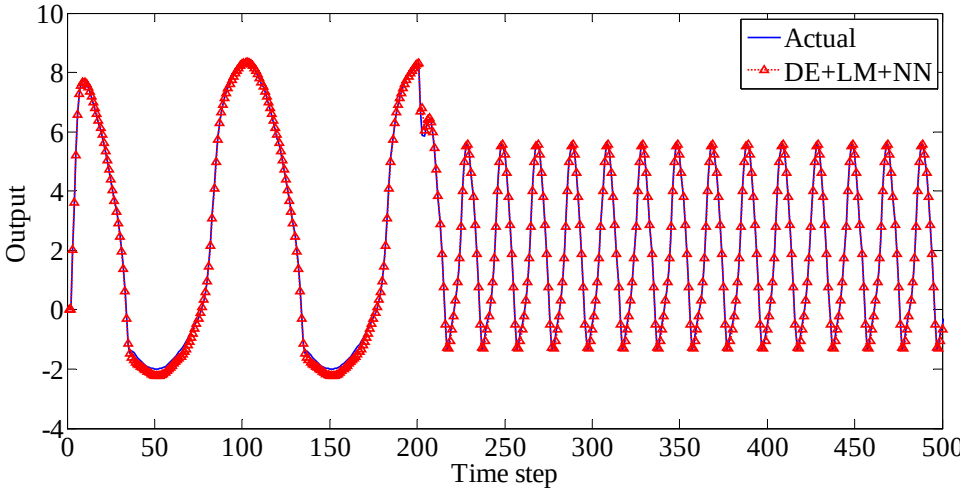

Figure 3.8: Identified and actual models (DE+LM+NN identifier)(Ex-1)

Figure 3.8 gives the result of proposed DE+LM+NN scheme. In this case the network is trained by both DE and Levenberg Marquardt algorithm. Here eleven number of hidden layer neurons are considered. This result (Figure 3.8) clearly indicates accurate identification of nonlinear system is achieved i.e. the superior identification capability of the proposed scheme over the other methods NN and DE+NN.

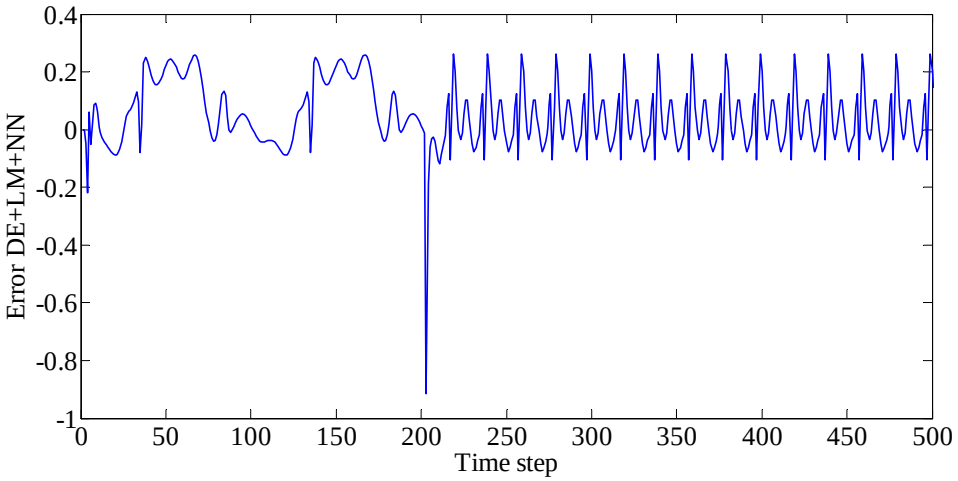

Figure 3.9: Error in modeling (DE+LM+NN identifier)(Ex-1)

Figure 3.9 shows its identification error curve for DE+LM+NN system identification. From Fig. 3.9 it is clear that identification error is smaller compared to error in NN and NN+DE.

Example-2:

The plant [16] to be identified is governed by the difference equation

$$y_p(k+1) \;=\; 0.3\, y_p(k) \;+\; 0.6\, y_p(k-1) \;+\; f\,[u(k)] \tag{3.7}$$

where the unknown function has the form:

$$f(u) \;=\; 0.6\, \sin(\pi\, u) \;+\; 0.3\, \sin(3\pi\, u) + \; 0.1\, \sin(5\pi\, u) \tag{3.8}$$

In order to identify the plant a series parallel model governed by the difference Eq. (3.9) was used.

$$\hat{y}_p(k+1) \;=\; 0.3\, y_p(k) \;+\; 0.6\, y_p(k-1) \;+\; f\,[u(k)] \tag{3.9}$$

NN identifier

The neural network identifier structure consisted of eleven numbers of neurons in the hidden layer. After 1500 epochs, the training of the neural identifier has been stopped. After the training is over, its prediction capability has been tested for input given as

$$u(k) \;=\; \sin(2\pi k/250) \tag{3.10}$$

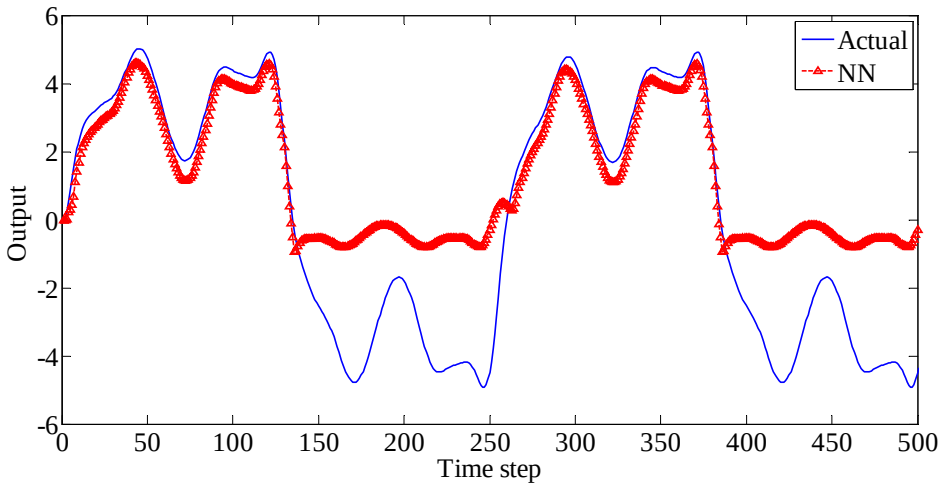

Figure 3.10: Identified and actual models (NN identifier)(Ex-2)

Figure 3.10 shows the actual and identified plant outputs for only NN approach. Figure 3.11 shows the identification error. The figures clearly indicate the poor identification performance of the neural identifier.

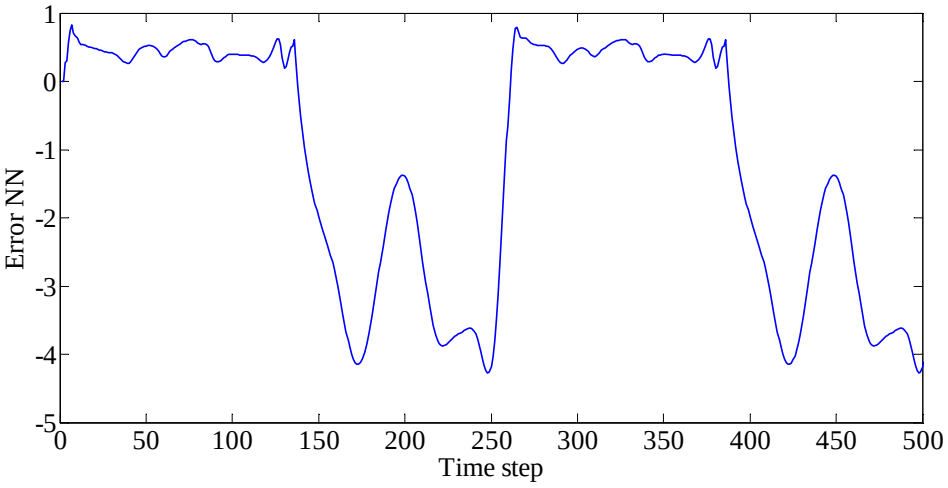

Figure 3.11: Error in modeling (NN identifier)(Ex-2)

DE+NN identifier

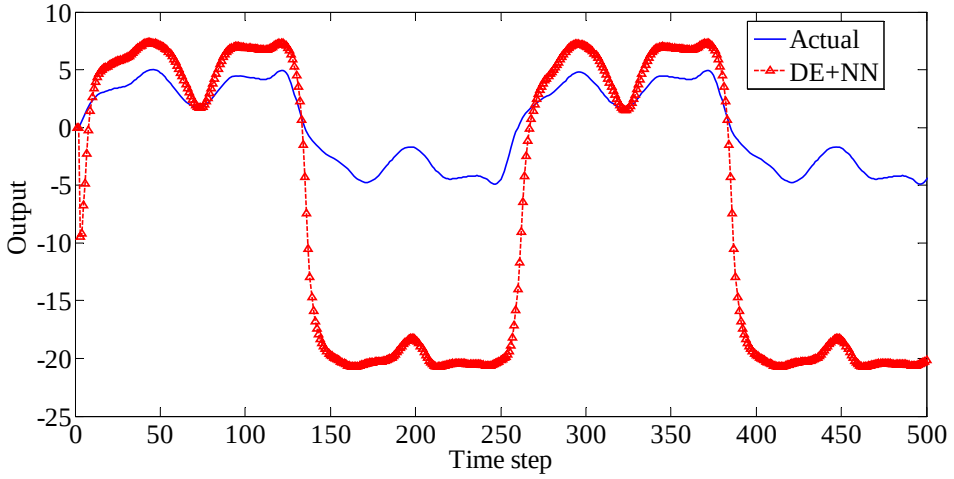

Figure 3.12: Identified and actual models (DE+NN identifier)(Ex-2)

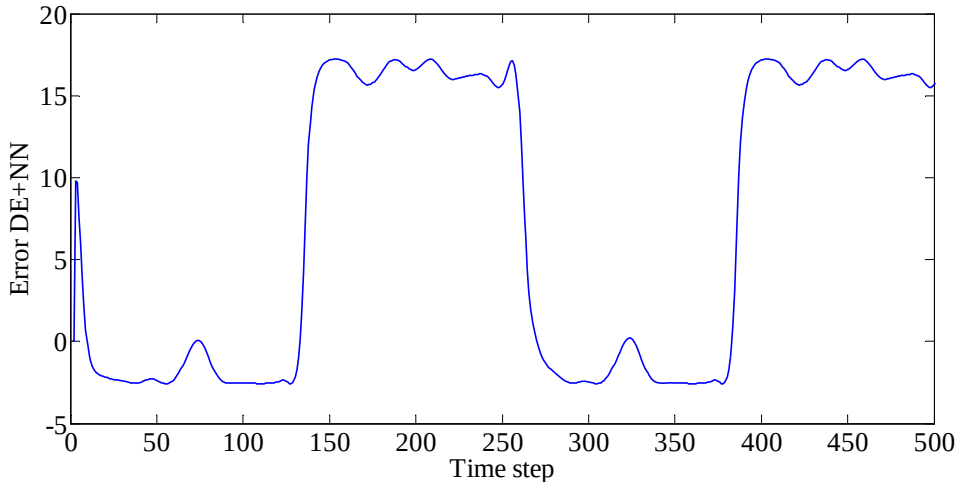

Figure 3.13: Error in modeling (DE+NN identifier)(Ex-2)

Figure 3.12 shows the identification performance of the system using differential evolution. Here the network is trained by using differential evolution instead of classical ones such as gradient descent and LM algorithm. The results obtained with DE+NN indicate no significant improvement over the previously discussed existing ones.

DE+LM+NN identifier

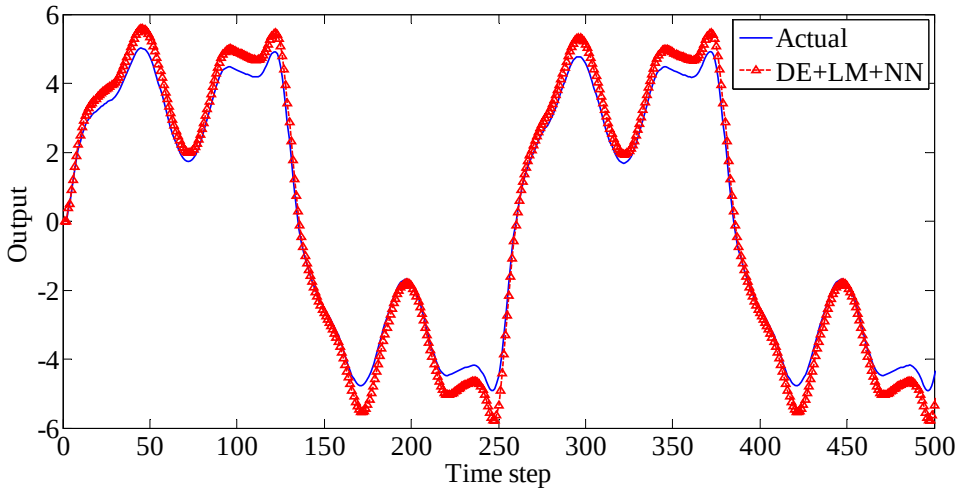

Figure 3.14: Identified and actual models (DE+LM+NN identifier)(Ex-2)

Figure 3.14 gives the result of proposed DE+LM+NN scheme. In this case the network is trained both by DE and LM algorithm. This result clearly indicates the superior identification capability of the proposed scheme over the other two methods discussed i.e. NN and DE+NN approaches. Figure 3.15 shows its identification error curve for DE+LM+NN system identification.

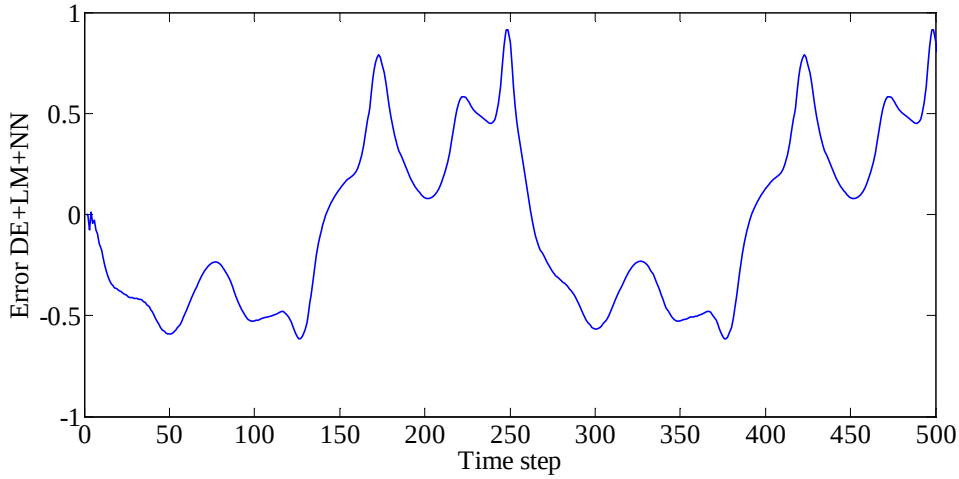

Figure 3.15: Error in modeling (DE+LM+NN identifier)(Ex-2)

Example-3:

Box and Jenkins gas furnace data are frequently used in performance evaluation of system identification methods [62]. The example consists of 296 input-output samples recorded with a sampling period of 9 second. The gas combustion process has one input variable, gas flow $u(k)$, and one output variable, the concentration of carbon dioxide (CO_2), $y(k)$. The instantaneous values of output $y(k)$ have been regarded as being influenced by ten variables $y(k-1), y(k-2), y(k-3), y(k-4), y(k-5), u(k-1), u(k-2), u(k-3), u(k-4), u(k-5)$. In the literature, the number of variables influencing the output varies from 2 to 10. In the proposed method, ten variables were chosen. Results shown gives a comparison of the identification methods such as neural networks trained with conventional methods, neural networks trained with DE+NN and hybrid differential evolution method i.e. DE+LM+NN. For all the methods eleven number of hidden layer neurons were taken and the results obtained after 1000 epochs. The number of training data was taken as 100 for all the cases and rest of the data were taken as test data.

NN identifier

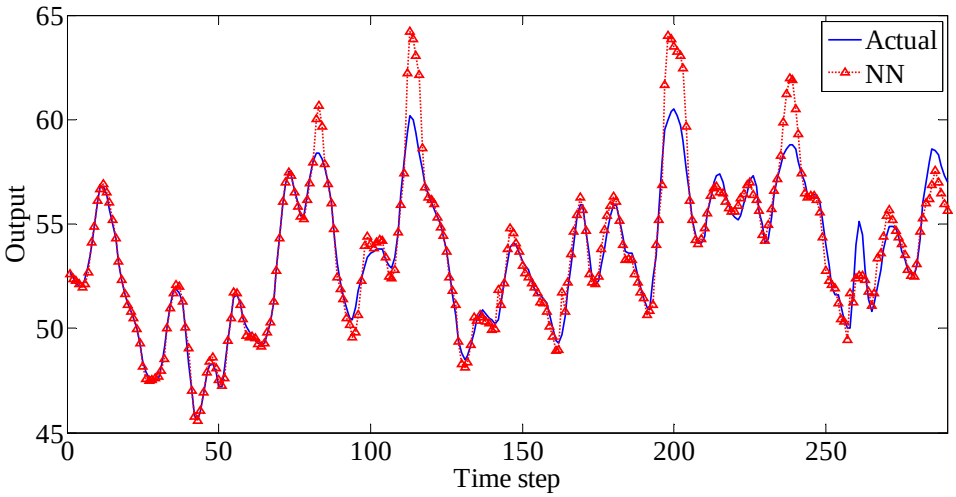

Figure 3.16: Identified and actual models (NN identifier)(Ex-3)

Figure 3.16 shows the graphs of the identified obtained with NN and the actual system. Here, the NN fails to identify the system dynamics at different time steps thus leads to a big identification error.

DE+NN identifier

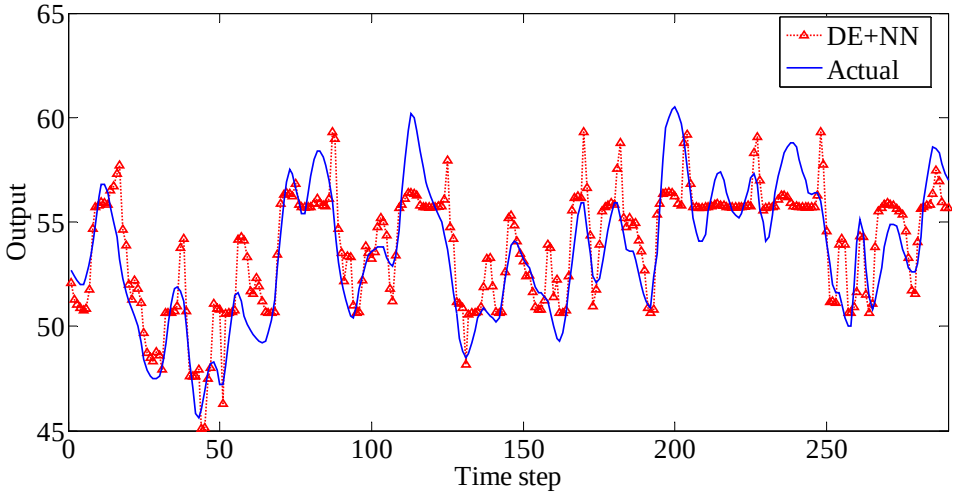

Figure 3.17: Identified and actual models (DE+NN identifier)(Ex-3)

The DE+NN identified system dynamics and the actual system dynamics were plotted in Fig. 3.17. It is observed that there is no improvement in identification with respect to the previous one i.e. the NN identifier.

DE+LM+NN identifier

The DE+LM+NN identified system dynamics and the actual system dynamics were plotted in Fig. 3.18.

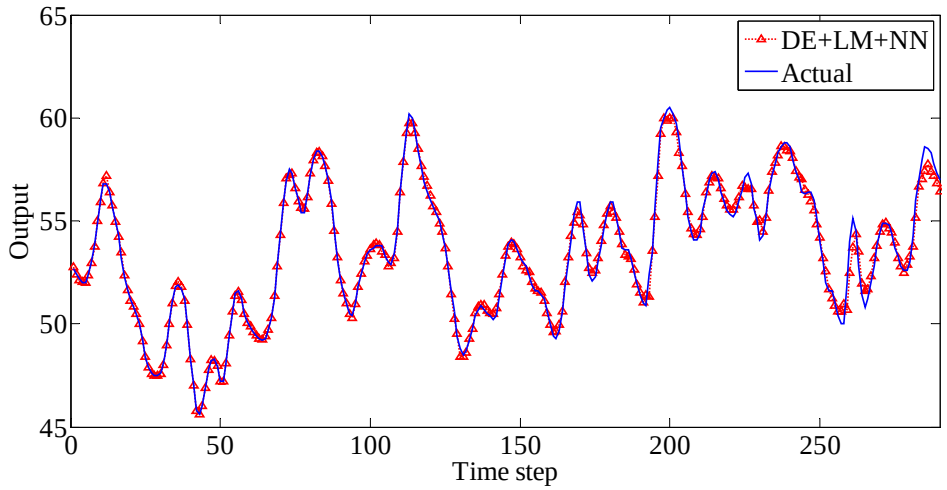

Figure 3.18: Identified and actual models (DE+LM+NN identifier)(Ex-3)

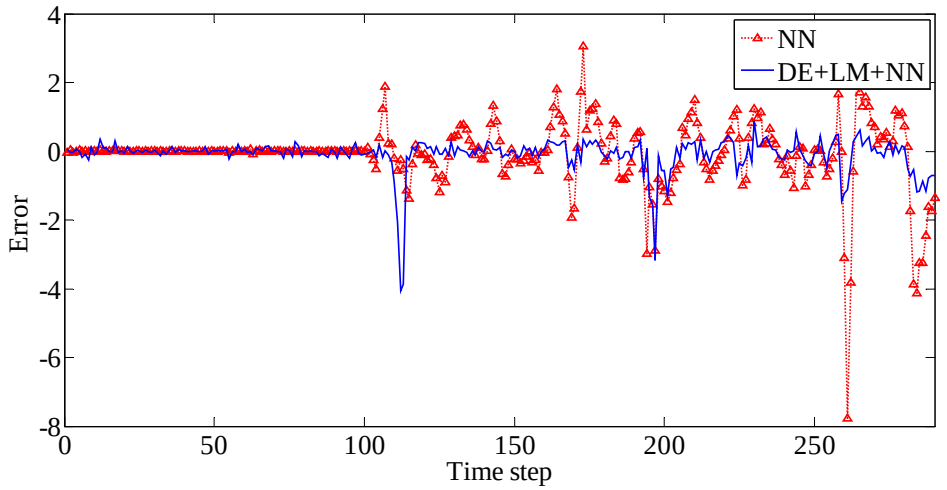

Figure 3.19: Comparison of Error in modeling [NN vs. (DE + LM + NN identifier)](Ex-3)

Figure 3.19 gives the comparison of error in modeling between proposed DE+LM+NN identifier and NN identifier. In this case the network is trained both by DE and Levenberg Marquardt algorithm. Figure 3.18 and Fig. 3.19 concludes that the proposed DE+LM+NN scheme has exhibited the expected identification performance i.e. the error between the true system and the identified one is minimum. Table 3.2 summaries the performance of the proposed method of system identification (DE+LM+NN) over the existing ones (NN,

DE+NN) for different example and case studies.

Hammerstein-Wiener identifier

Hammerstein-Wiener models describe dynamic systems using one or two static nonlinear blocks in series with a linear block. Only the linear block contains dynamic elements. The linear block is a discrete-time transfer function and the nonlinear blocks are implemented using nonlinearity estimators, such as saturation, wavenet, and deadzone. The input signal passes through the first nonlinear block, a linear block, and a second nonlinear block to produce the output signal, as shown in the Fig. 3.20.

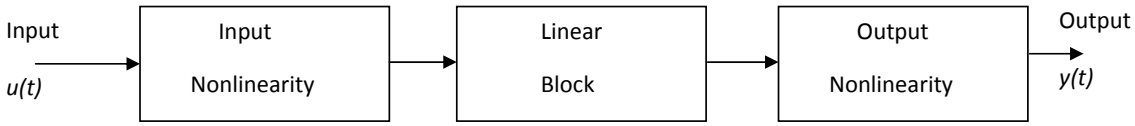

Figure 3.20: Hammerstein-Wiener structure

Figure 3.21-3.23 shows the performance of nonlinear Hammerstein-Wiener model (NLHW) for the examples discussed above. In example-1 and example-2 the model is created based on the first 500 data which is generated randomly then the estimated output is compared to the actual output for data record given by equation 3.6 and 3.10 for validation. In example-3 we load data from box-Jenkins data given in [62]. The input is the gas flow and the output is the concentration of carbon dioxide gas. First a model is created based on the first 100 of the data. The simulated output is then compared to the measured output for the whole data record. The Hammerstein model is tried out; with saturation nonlinearity. From the identified results it is clear that the Hammerstein model gives the less performance as compared to the proposed DE+LM+NN algorithm.

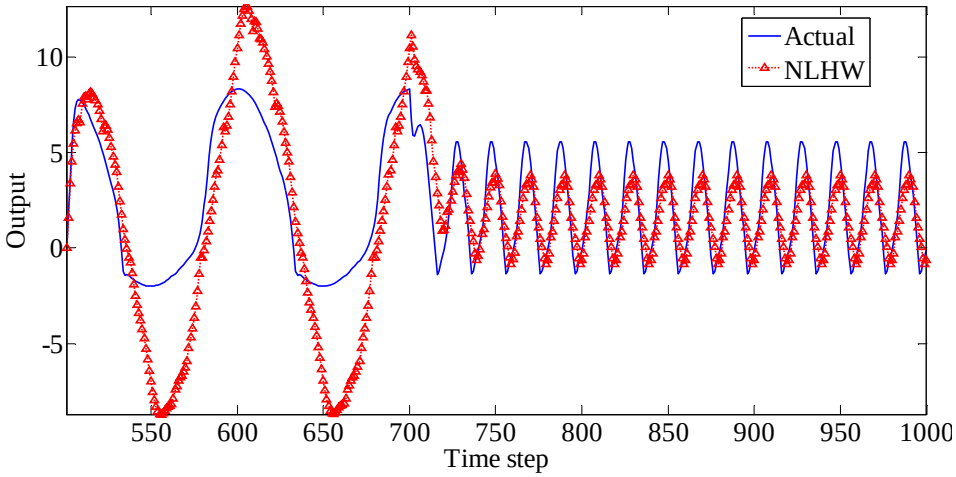

Figure 3.21: Identified and actual models (Hammer Stein-Wiener identifier) Ex-1

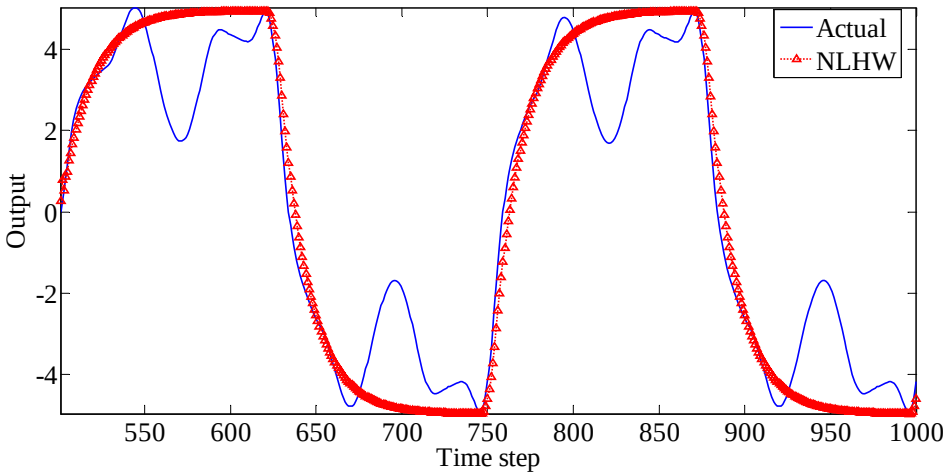

Figure 3.22: Identified and actual models (Hammer Stein-Wiener identifier) Ex-2

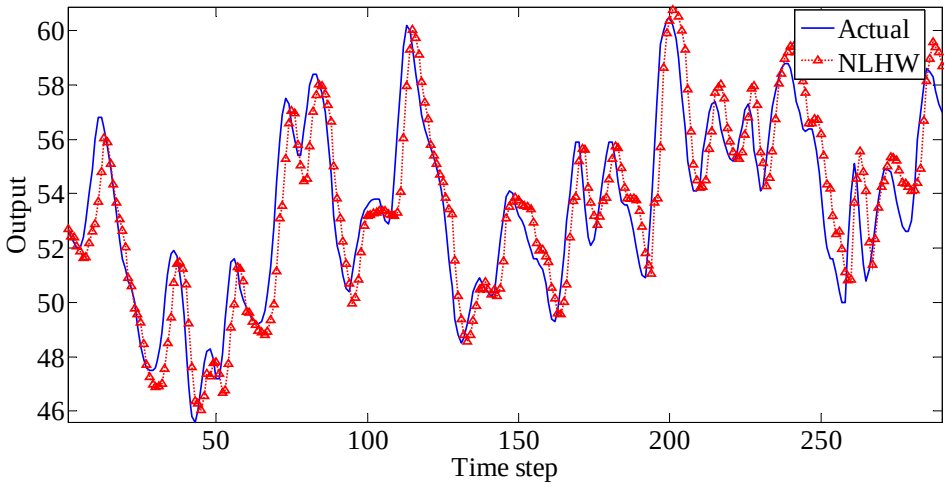

Figure 3.23: Identified and actual models (Hammer Stein-Wiener identifier) Ex-3

Table 3.2: Performance of the proposed methods

Methods	Time of Convergence (Sec)	MSE	Examples
NN	17.490	0.0047	Example-1
DE+NN	50.112	6.8830	Example-1
DE+LM+NN	24.899	0.0030	Example-1
NN	18.991	0.3115	Example-2
DE+NN	121.148	3.5470	Example-2
DE+LM+NN	30.985	0.0059	Example-2
NN	20.929	0.0038	Example-3
DE+NN	102.347	2.6680	Example-3
DE+LM+NN	54.895	0.0001	Example-3

3.7 Summary

The work has described the scope of improving system identification of nonlinear systems by using proposed DE+LM+NN approach. In the proposed identification framework, differential evolution is used only to find approximate values in the vicinity of the global minimum. These approximate weight values are then used as starting values for a faster convergence algorithm i.e. Levenberg Marquardt algorithm. As DE becomes slow near the basin of the

global optimization the function of LM is to enhance the speed of convergence. In this work number of examples including a practical case-study has been considered for implementation of this algorithm. From the results presented in previous section, it is clear that there is certainly an improvement in identification performance for nonlinear systems over the existing approaches. In case of DE+NN and DE+LM+NN 30 independent runs has been taken into consideration and the average of the results are produced here. In comparison to use DE+NN approach proposed DE+LM+NN approach provides better system identification performance in terms of speed of convergence and identification capability. Finally all the results are compared with the conventional Hammerstein-Wiener identifier and it is concluded from the results that the proposed algorithm is having better identification results. Extensive research is still required to implement different concepts of DE, such as opposition based differential evolution (ODE) and to verify whether the proposed learning scheme can bring clear improvement in terms of convergence speedup and better identification capability.

Chapter 4

Nonlinear System Identification Using Memetic Algorithm

4.1 Introduction

This chapter introduces and analyzes a Memetic algorithm approach for the training of artificial neural networks, more specifically MLP applied to nonlinear system identification. The problem of SH (discussed in chapter 3) lies on deciding when to stop one algorithm and trigger the next. Waiting for the stabilization of the search takes more time but it is feasible. However, there may be a time before actual stabilization when the process is already engaged in a basin which it cannot escape. Triggering the next algorithm from that earlier moment might prove more efficient. Furthermore, because of the mutation, there may be further improvements of the solution after a relatively long stable phase. Triggering the next algorithm during the first stabilization may lead to miss this further improvement of the evolutionary search which could be done by the mutation. Our memetic algorithm is proposed as an alternative to SH algorithms which has limitations when dealing with the switching from one algorithm to other. The aim of this work is to develope a memetic training algoritm that is able to deliver trained neural networks that produce small errors. Here we have studied the identification of a nonlinear system [61] using seven different algorithms namely BP, GA, PSO, DE, GABP and PSOBP along with the proposed DEBP approaches. In the proposed system identification scheme, three global search methods such as GA, PSO and DE

have been combined with the gradient descent method i.e. the BP algorithm to overcome the slow convergence of the EANN. The local search algorithm BP is used as an operator for GA, PSO and DE. These algorithms have been tested on some standard benchmark problems for nonlinear system identification to prove their efficacies. As discussed in chapter 1, there are a number of benefits that can be gained by combining the global search features of EAs with the local search for improving and refining an individuals solution. However, as there are no free lunches these benefits must be balanced against an increase in the complexity in the design of the algorithm. That is, a careful consideration must be made on exactly how the hybridization will be done. Consider for example the memetic algorithm template in Fig. 4.1. This is a particular structure of memetic algorithm that has been considered in this work. The hybridization could be done in many ways of applying the local search inside the global algorithm. For example, the initial population could be seeded with solutions arising from sophisticated problem specific heuristics, the crossover (mutation) operator could be enhanced with domain specific and representation specific constrains as to provide better search ability to the EA. Moreover, local search could be applied to any or all of the intermediate sets of solutions. However, the most popular form of hybridization is to apply one or more phases of local search, based on some probability parameter, to individual members of the population in each generation. As shown in Fig. 4.1 the local search is applied before the selection operator i.e. a fine search is applied to the offsprings before entering to the next generation.

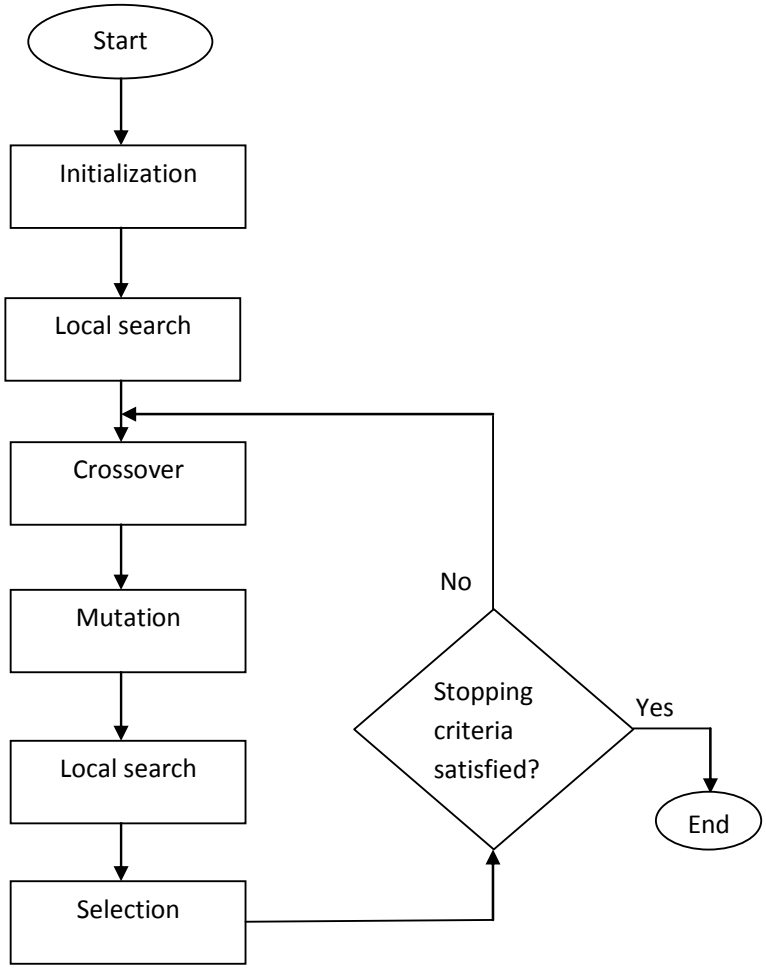

Figure 4.1: Scheme of the memetic algorithm

4.2 Lamarckianism and Baldwinian Effect in MA

When integrating the local search with the evolutionary search we face with the dilemma of what to do with the improved solution that is produced by the local search. That is, suppose that individual i belongs to the population P in generation g and that the fitness of i is $f(i)$. Suppose that the local search produces a new individual i^{new} with $f(i^{new}) < f(i)$ for a minimization problem. The designer of the algorithm must now choose between two alternative options.

- Repacing i with i^{new}, in which case $P = P - i + i^{new}$ and the genetic information in i is lost and replaced with that of i',
- The genetic information of i is kept but its fitness altered:$f(i) = f(i^{new})$.

The first option is commonly known as Lamarckian learning while the second option is referred to as Baldwinian Learning [73]. It is difficult to decide a priori what method is the best, and probably no one is better in all cases. In our study we have considered the Lamarckianism which tends to substantially accelerate the evolutionary process.

4.3 Mechanism of Proposed Memetic Algorithm

To describe this mechanism with more details, the following formal framework is introduced. Let us first introduce the number of population as P and the population at the g^{th} generation, $\Pi_g = \left\{\underline{c}_{1,g}, \cdots, \underline{c}_{P,g}\right\}$. Let H be the operator may be (mutation/crossover/reproduction) defined as

$$H|\Pi_g \in R^{d*P} \rightarrow \underline{x}(x_1, \cdots, x_P) \in R^P$$

which is associated to each population the fitness vector of its elements. Let RE, M be the recombination and the mutation operators respectively. These operators are called the reproduction operators as well and are defined as

$$RE|\Pi \in R^d \times R^P \rightarrow \Pi' \in R^d \times R^P$$

$$M|\Pi' \in R^d \times R^P \rightarrow \Pi'' \in R^d \times R^P$$

Let us denote LS as the local search operator, i.e., the operator which produces a new population by applying the LS with starting points equal to the individuals in the current population:

$$LS|\Pi'' \in R^d \times R^P \rightarrow \Pi''' \in R^d \times R^P$$

where d is the number of parameters and P is the number of population.

Then applying the selection operator the next generation population can be determined.

$$\Pi_{g+1} = selection\left\{RE\left(M\left[LS\left(\Pi_g\right)\right]\right)\right\}$$

In a Hybrid Evolutionary Algorithm, the role of the Evolutionary Algorithm is essentially to explore the searching space and locate the more promising regions. Figure 4.2 shows how to produce next generation of the proposed algorithm.

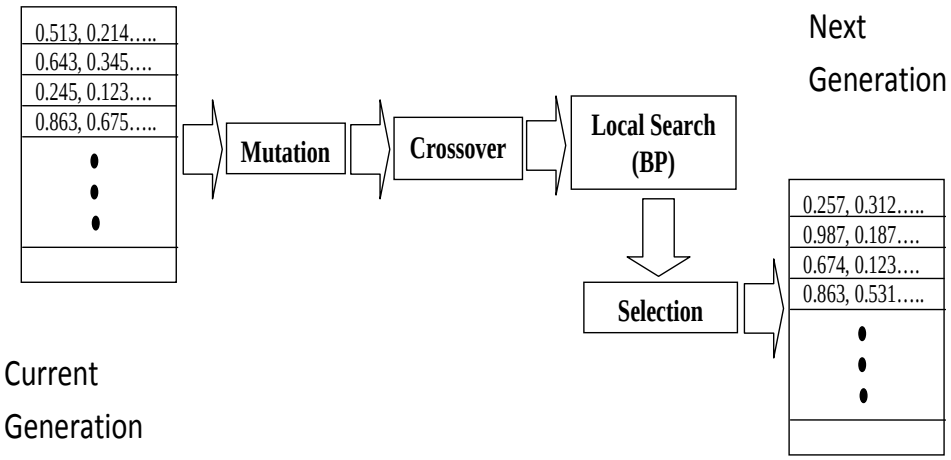

Figure 4.2: Template for proposed Memetic algorithm

Figure 4.3 shows a NN identification scheme where memetic algorithm is used as an optimization algorithm to find the best set of neural network weights.

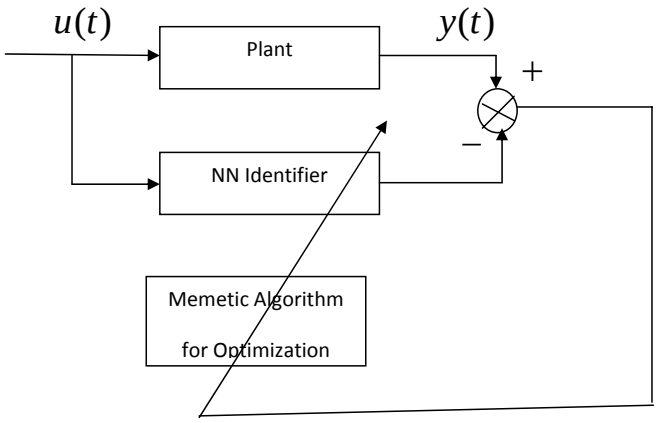

Figure 4.3: Memetic identification scheme

4.4 Proposed DEBP Training Algorithm for Nonlinear System Identification

Here, we describe how a DE is applied for training neural network in the frame work of system identification. Output of a feed-forward neural network is a function of synaptic weights $\mathbf{w}$ and input values $\mathbf{x}$, i.e. $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$. The role of BP in the proposed algorithm is to fine tune the weigts in the current population. In the training processes, both the input vector $\mathbf{x}$ and the output vector $\mathbf{y}$ are known and the synaptic weights in are adapted to obtain appropriate functional mappings from the input to the output. Generally, the adaptation can be carried out by minimizing the network error function $\mathbf{E}$ which is of the form $\mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}))$. In this work we have taken $\mathbf{E}$ as mean squared error i.e. $\mathbf{E} = \frac{1}{N}\sum_{k=1}^{N}[\mathbf{y} - f(\mathbf{x}, \mathbf{w})]^2$, where N is the number of data considered. The optimization goal is to minimize the objective function $\mathbf{E}$ by optimizing the values of the network weights $\mathbf{w}$. where

$$\mathbf{w} = (w_1, \cdots, w_d)$$

where d is the number of weights in the weight vector

The proposed DEBP algorithm

Step 1.

Initialize population **pop**: Create a population from randomly chosen object vectors with dimension P, where P is the number of population

$$\mathbf{P}_g = (\mathbf{w}_{1,g}, \cdots, \mathbf{w}_{P,g})^T, \;\; g = 1, \cdots, g_{\max}$$

$$\mathbf{w}_{i,g} = (w_{1,i,g}, \cdots, w_{d,i,g}), \quad i = 1, \cdots, P$$

In $\mathbf{w}_{i,g}$, i is index to the population and g is the iteration (generation) to which the population belongs.

Step 2.

Evaluate all the candidate solutions inside the **pop** for a specified number of iterations.

Step 3.

For each i^{th} candidate in **pop**, select the random population members, $r_1, r_2, r_3 \in \{1, 2, \cdots, P\}$

Step 4.

Apply a mutation operator to each candidate in a population to yield a mutant vector i.e.

$$v_{j,i,g+1} = w_{j,r1,g} + F(w_{j,r2,g} - w_{j,r3,g}), \text{ for } j = 1, \cdots, d$$
$$(i \neq r_1 \neq r_2 \neq r_3) \in \{1, \cdots, P\} \text{ and } F \in (0, 1+]$$

where F denotes the mutation factor.

Step 5.

Apply crossover i.e. each vector in the current population is recombined with a mutant vector to produce trial vector.

$$t_{j,i,g+1} = \begin{cases} v_{j,i,g+1} & \text{if } rand_j[0,1) \leq C \\ w_{j,i,g} & \text{otherwise} \end{cases} \qquad \text{where} \quad C \in [0,1]$$

Step 6.

Apply Local Search back propagation algorithm i.e. each trial vector will produce a lst-trial vector

$$lst_{j,i,g+1} = bp : (t_{j,i,g+1})$$

Step 7.

Apply selection i.e. between the local search trial (lst-trial) vector and the target vector. If the lst-trial vector has an equal or lower objective function value than that of its target vector, it replaces the target vector in the next generation; otherwise, the target retains its place in the population for at least one more generation

$$\mathbf{w}_{i,g+1} = \begin{cases} lst_{i,g+1} & \text{if } \mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}_{i,g+1})) \leq \mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}_{i,g})) \\ \mathbf{w}_{i,g} & otherwise \end{cases}$$

Step 8.

Repeat steps 1 to 7 until stopping criteria (i.e. maximum number of generation) is reached.

4.5 Results and Discussions

This section compares the performance of the proposed DEBP identification algorithm with other memetic approaches such as GABP and PSOBP using the simulation studies on a bench mark problem for the identification. The

given nonlinear discrete system [61] expressed by

$$y_p(k+1) = \frac{y_p(k)[y_p(k-1)+2][y_p(k)+2.5]}{8.5+[y_p(k)]^2+[y_p(k-1)]^2} + u(k) \tag{4.1}$$

where $y_p(k)$ is the output of the system at the k^{th} time step and $u(k)$ is the plant input which is uniformly bounded function of time. The plant is stable at $u(k) \in [-2, 2]$. For the identification of the plant described in Eqn. (4.1), let the neural model be in the form of

$$y_{pi}(k+1) = f(y_p(k), y_p(k-1)) + u(k) \tag{4.2}$$

where $f(y_p(k), y_p(k-1))$ is the nonlinear function of and the inputs to the neural network are $y_p(k)$ and $y_p(k-1)$. The output from neural network is $y_{pi}(k+1)$. In the following discussions, we will present our observation studies on the nonlinear plant identification scheme using seven different identification algorithms and will present their comparative results. Figure 4.3 shows the scheme of neural identifier for the given plant utilizing proposed memetic algorithm as an optimizer to train the weights of the neural network.

For plant identification, the morphology of the neural network consisted of 21 numbers of neurons in the hidden layer. After 100 epochs the training of the neural identifier has been stopped. During training period, input u(k) was a random white noise signal, but after the training is over, its prediction capability were tested for input given by

$$\begin{cases} u(k) = 2\ \cos(2\pi k/100) & k \leq\ 200 \\ u(k) = 1.2\ \sin(2\pi k/20) & 200\ <\ k \leq\ 500 \end{cases} \tag{4.3}$$

Identification using BP

Figure 4.4 shows the system identification results obtained using the back propagation algorithm for training the feed-forward neural network. Figure 4.5 shows the identification error plot obtained from BP identification.

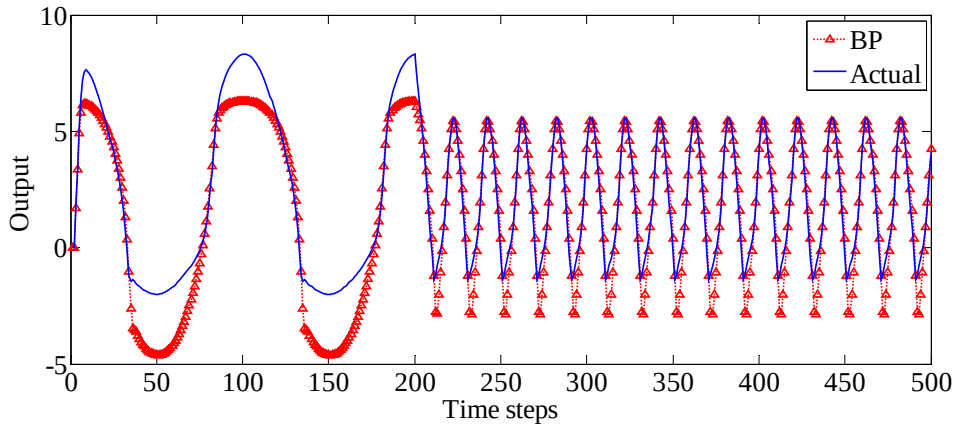

Figure 4.4: BP identification performance

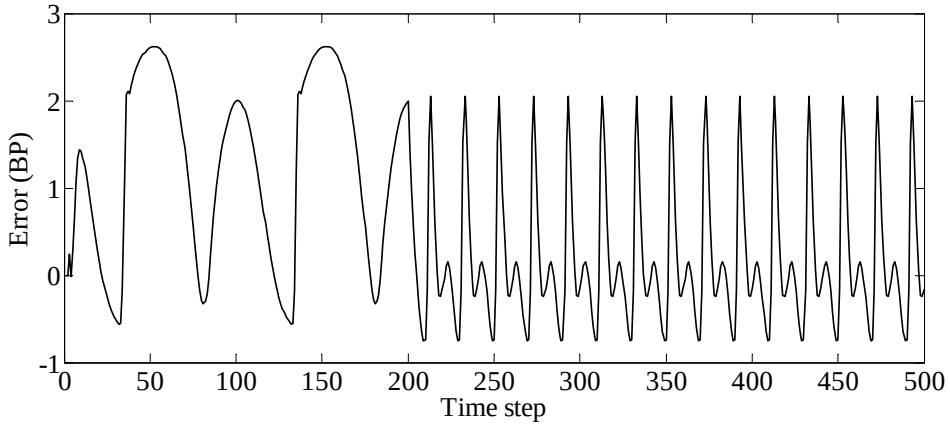

Figure 4.5: Error in modeling (BP identification)

Identification using GA

Figure 4.6 shows the identification performance of the system using genetic algorithm as learning algorithm for the given neural network. The same neural network configuration i.e. twenty one number of neurons are taken into account. After 100 epochs it was found that the squared error is more than conventional back propagation also taking more time to converge. Figure 4.7 shows the error between actual and GA identification.

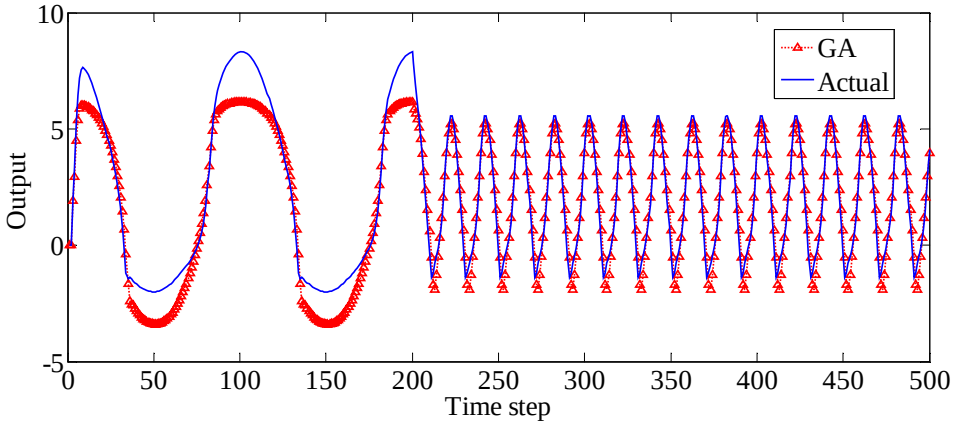

Figure 4.6: GA identification performance

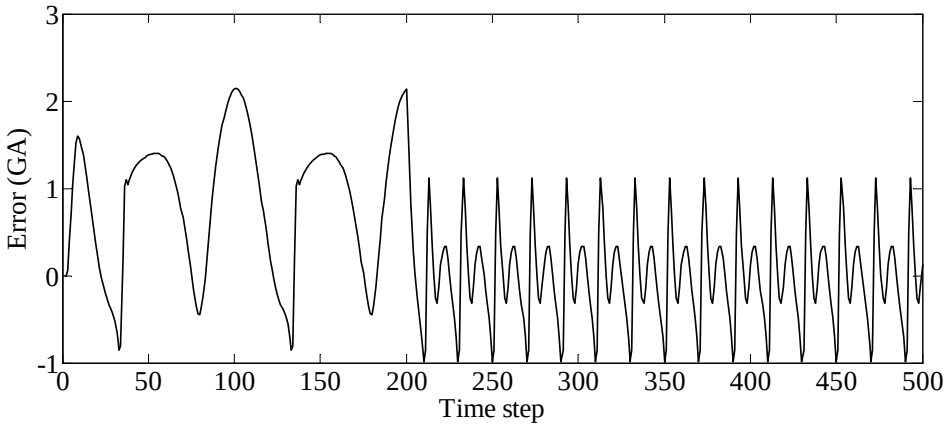

Figure 4.7: Error in modeling (GA identification)

GABP identification

Figure 4.8 shows the identification performance between the GABP algorithm and the actual output. The identification error between the actual output and the GABP output is shown in Fig. 4.9.

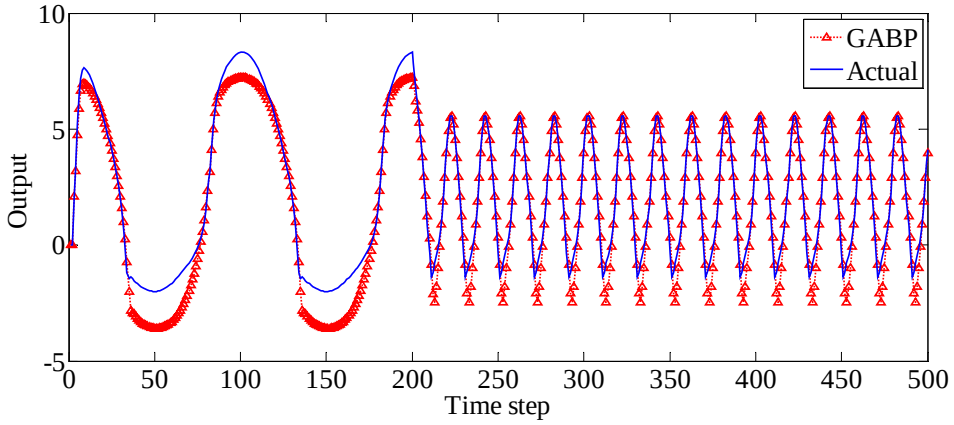

Figure 4.8: GABP identification performance

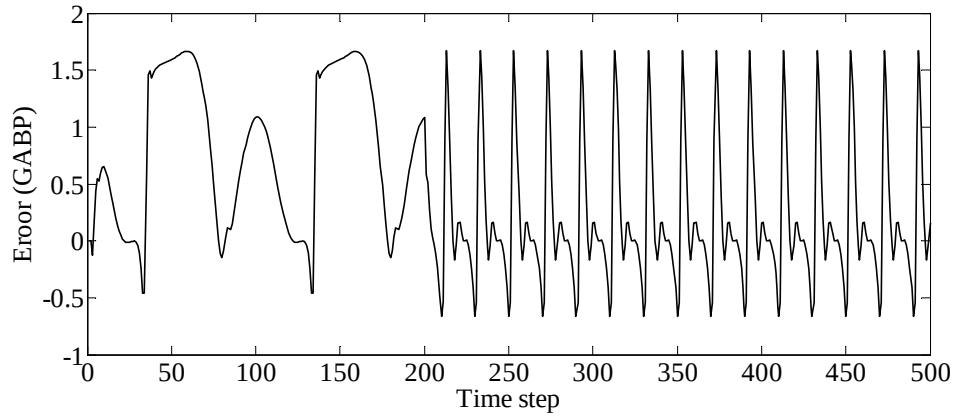

Figure 4.9: Error in modeling (GABP identification)

Identification using PSO

Figure 4.10 shows the identification performance between the particle swarm optimization and the actual output. The identification error between the actual output and the PSO output is shown in Fig. 4.11.

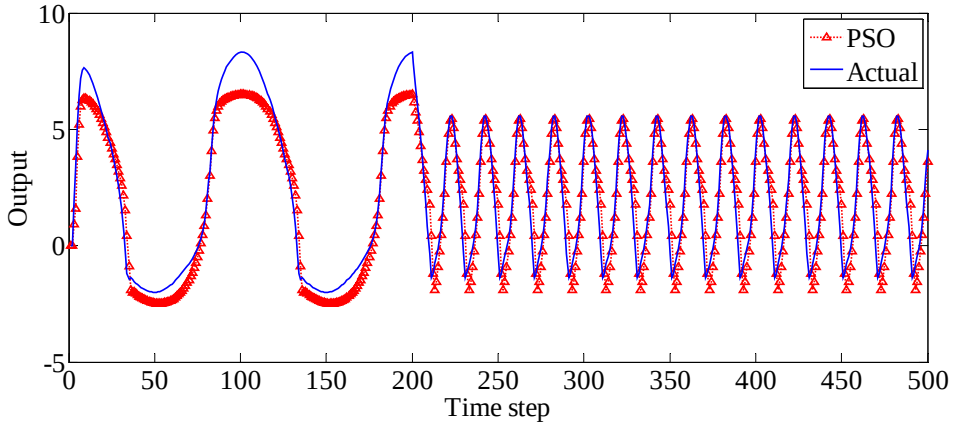

Figure 4.10: PSO identification performance

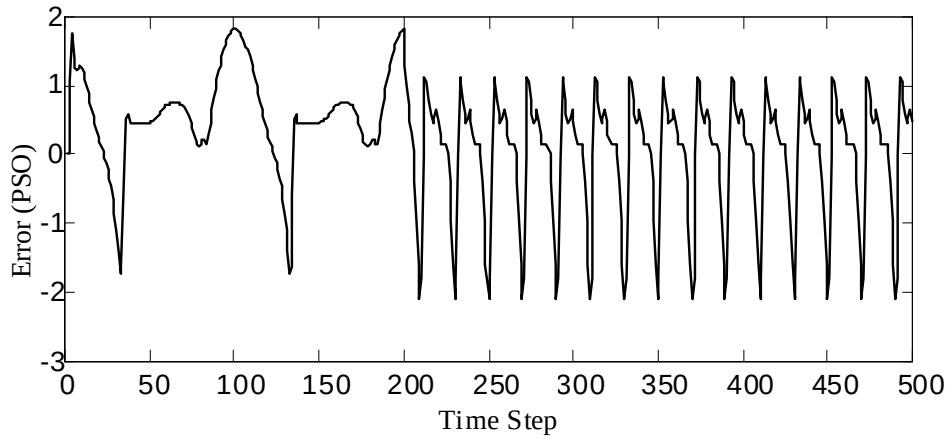

Figure 4.11: Error in modeling (PSO identification)

PSOBP identification

Figure 4.12 shows the result of memetic scheme PSOBP where particle swarm optimization is hybridized with back propagation. The result clearly indicates the above scheme does not give better identification of nonlinear system compared to only PSO algortihm. Figure 4.13 shows identification error curve between actual and PSOBP system output.

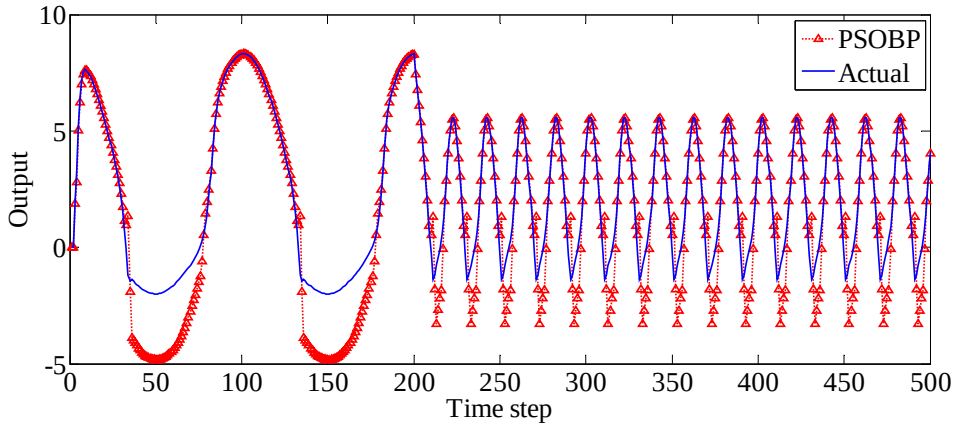

Figure 4.12: PSOBP identification performance

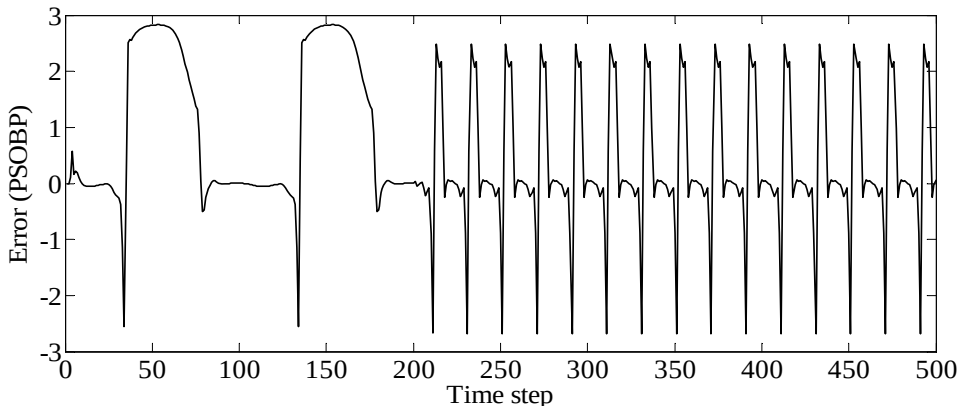

Figure 4.13: Error in modeling (PSOBP identification)

DE identification

Figure 4.14 shows the identification performance between the differential evolution and the actual output. It was found that the performance is better than GA and PSO but worst than GABP. The identification error between the actual output and the DE output is shown in Fig. 4.15.

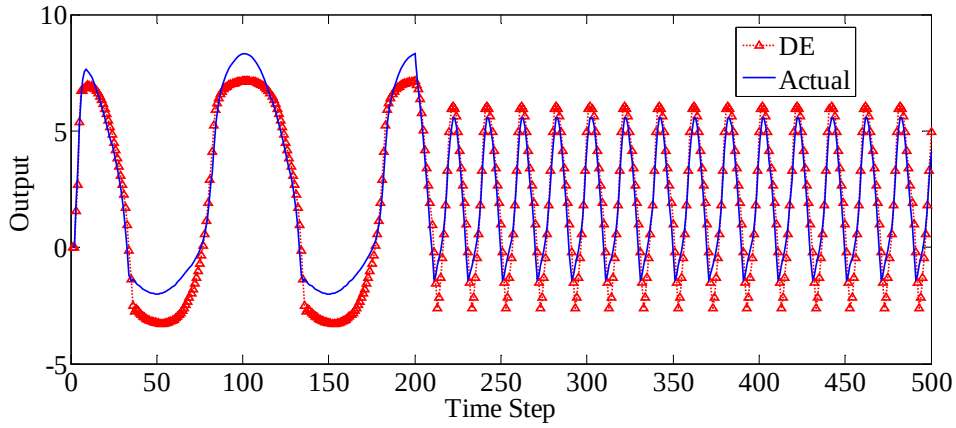

Figure 4.14: DE identification performance

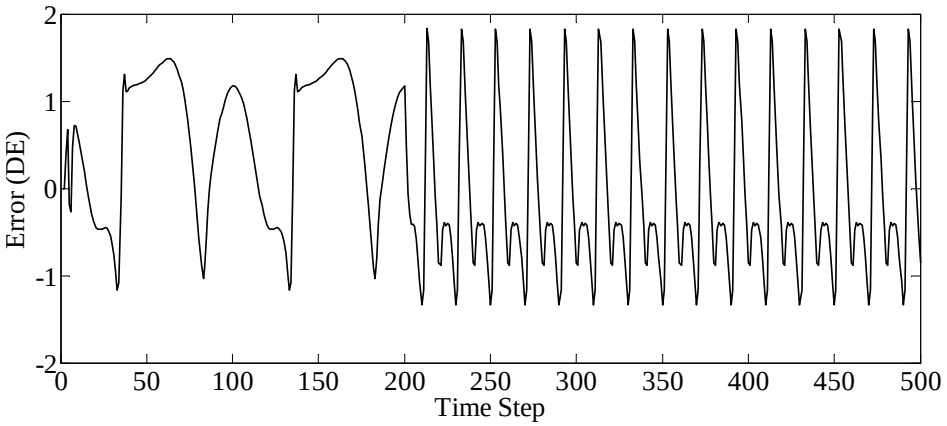

Figure 4.15: Error in modeling (DE identification)

DEBP identification

From Fig.4.16 it is clear that the proposed method i.e. DEBP identification is more effective than other mentioned approaches as per as identification performance and speed of convergence is concerned. Figure 4.17 shows the error between the proposed one and the actual one.

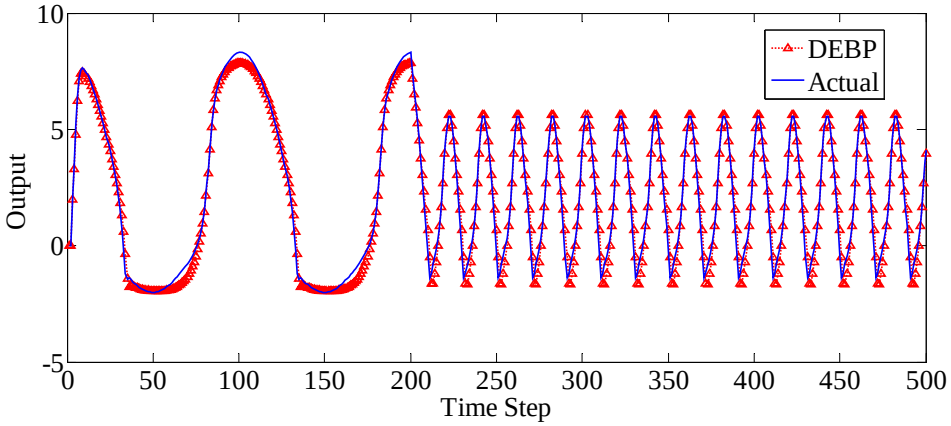

Figure 4.16: DEBP identification performance

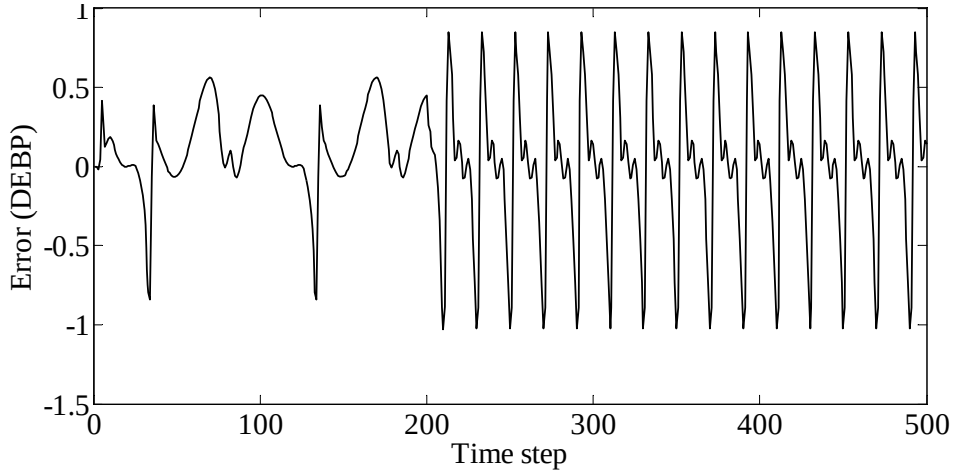

Figure 4.17: Error in modeling (DEBP identification)

Performance comparison of all the seven identification methods

Figure 4.18 depicts the MSE profiles for all the seven different identification methods (BP, GA, GABP, PSO, PSOBP, DE and DEBP). In these seven methods, a new identification scheme, namely the DEBP identification approach is proposed. From this figure it is clear that the MSE with the proposed method DEBP converges to zero very fast taking only about 20^{th} iteration while the error curves with the other system identification methods (BP, GA,GABP, PSO, PSOBP and DE) converges to zero taking over 50^{th} iterations. Hence it

is important to note that the proposed DEBP system identification exhibits better convergence characteristics. All the simulations have been performed in MATLAB using same set of parameters i.e. population size, number of generations, upper and lower bounds of weights and number of hidden layer neurons given in Table 4.1.

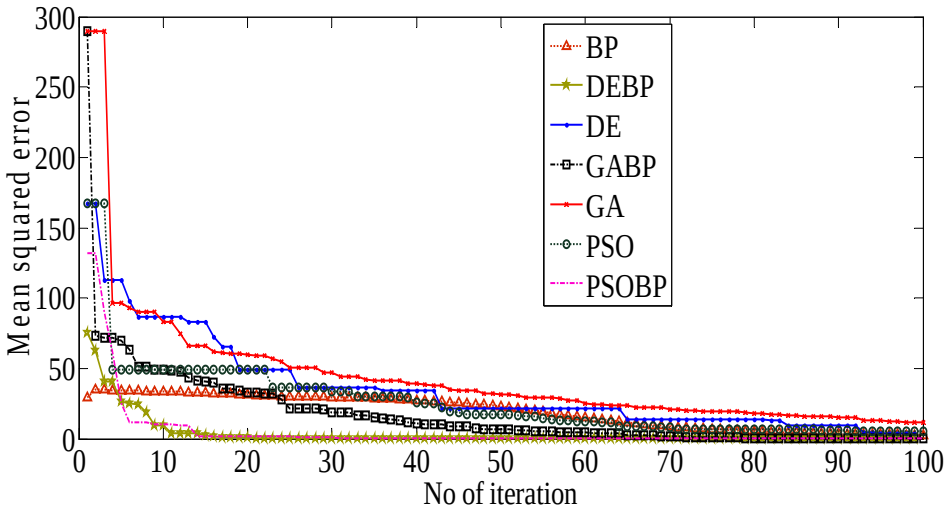

Figure 4.18: A comparisons on the convergence on the MSE for all the seven methods

Table 4.1: Parameters used in simulation studies

Total sampling period, T	500
Population size, P	50
Number of generations	100
Upper and lower bound of weights	[-1 1]
BP learning parameter, η	0.55
Number of hidden layer neurons	21
Parameters for DE and DEBP Algorithms	
Mutation constant factor, F	0.6
Cross over constant, C	0.5
BP learning Parameter, η	0.55
Parameters for GA and GABP Algorithms	
Mutation probability, p_m	0.002
Cross over constant probability, p_c	1
BP learning Parameter, η	0.55
Parameters for PSO and PSOBP Algorithms	
Learning factor, C_1	1.9
Learning factor, C_2	1.9
BP learning Parameter, η	0.55

Table 4.2: Comparison of performance of seven methods.

SL NO	Identification method	Computation time in seconds (Sec)	MSE	Number of generation at which the MSE converges to zero
1	BP	4.76	2.6086	> 100
2	GA	40.42	11.4156	> 100
3	GABP	131.42	0.2852	70
4	PSO	42.15	5.49	> 100
5	PSOBP	142.79	0.2074	50
6	DE	42.19	3.9645	> 100
7	DEBP	136.73	0.0625	20

Table 4.2 gives the comparison of performance of all the seven methods in terms of MSE which has been obtained after taking the avegarage results

of 30 independent runs. From the results it is clear that for a particular number of iteration i.e. 100, the proposed DEBP algorithm has a MSE of 0.0625. It is found that the memetic approaches GABP and DEBP are having faster convergence in comparison to GA and DE. Finally it is concluded that the proposed memetic DEBP is having better identification performance and faster convergence in comparison to memetic GABP and PSOBP algorithm which indicates DE is outperforming than its counterpart GA and PSO.

4.6 Summary

This chapter provides an extensive study of memetic algorithms applied to nonlinear system identification. From the results presented in this chapter it has been found that the proposed DEBP memetic algorithm applied to neural network learning exhibits better result in terms of fast convergence and lowest MSE amongst all the seven methods (i.e. BP, GA, GABP, PSO, PSOBP, DE. and DEBP). For each evolutionary approaches the results obtained is the average of 30 independent runs. The proposed method DEBP exploits the advantages of both the local search and global search. It is interesting to note that the local search pursued after the mutation and crossover operation that helps in intensifying the region of search space which leads to faster convergence. The overall performance of memetic DE was found to be better than the other memetic approaches i.e. GABP and PSOBP. This shows it is advantageous to use DE over other evolutionary computation such as GA and PSO in nonlinear system identification.

Chapter 5

Identification of Twin Rotor MIMO System (TRMS)

5.1 Introduction

This chapter presents identification of a one degree of freedom experimental aerodynamic test rig, a twin rotor multi-input-multi-output system using sequential hybridization algorithm and memetic algorithm both developed in this work. The TRMS is a highly nonlinear system which can be considered as an experimental model of a complex air vehicle. Such vehicles are required to be identified precisely to ensure satisfactory control performance to meet the demand for automation. This implies that linear characterization of aircrafts is not well enough to describe the systems characteristics for control purposes and nonlinear modeling techniques are required. Neural network based nonlinear characterization look promising in this regard. This chapter focuses into the development of nonlinear modeling a TRMS system. The system is modeled using a NARX identification scheme with a feed-forward neural network. Two different types of algorithms discussed in previous chapters used in this work for supervised leaning of the network and their performances are compared in terms of identification capability and speed of convergence. One uses Differential evolution algorithm and Levenberg-Marquardt applied one after other known as SH to train the weights of the neural network. The second one is the memetic algorithm where three global searches i.e. GA, PSO and DE are successfully hybridized with gradient based BP algorithm. Here the

BP acts as an operator in each generation after the crossover and mutation. The responses of all the identified models are compared with that of the real TRMS to validate the accuracy of the models.

5.2 Description of TRMS

In classical aircraft applications, the role of system identification is to estimate the parameters of nonlinear or linearized 6 DOF equation of motion from flight data, having a known structure. A considerable effort has been made in the past few years to find methodologies for identifying and control the systems with nonlinearity and uncertain dynamics. Now a days neural network with different architecture and learning algorithms has become a successful tool in this regard, with application to various types of nonlinear systems including air vehicles. In [86] neural networks have been employed for estimating the aerodynamic coefficients of unmanned air vehicles (UAVs). Neural networks were utilized by [87], [88] for dynamic modeling and control of super maneuvering delta wing aircraft. Lately, B-splines have been investigated in modeling and identification of nonlinear aerodynamic functions [89]. In all these cases the model structure is known. However, in the present work, no model structure was assumed a priori i.e. black-box modeling. Such an approach yields input-output models with neither a prior defined model structure nor specific parameter settings reflecting any physical aspects. Nonlinear modeling of a TRMS using radial basis function networks has been addressed in [90], which presents nonlinear system identification method for modeling air vehicles of complex configuration. Authors in [91] have carried out dynamic modeling and optimal control of a TRMS. The extracted model is employed in the design of a feedback Linear Quadratic Gaussian (LQG) compensator. Performance analysis of 4 types of conjugate gradient algorithms in the nonlinear dynamic modeling of a TRMS using feed-forward NNs has been reported by the authors in [92]. Dynamic modeling of a TRMS has also been presented in [93], which has investigated the utilization of NNs and parametric linear approaches for

modeling the system in hovering position. In [94] a parametric modeling of a TRMS using GA has been proposed. In their approach the global search technique of GA has been used to identify the parameters of the TRMS based on one-step-ahead prediction. Nonlinear dynamic modeling of a TRMS using resilient propagation algorithm (RPROP) algorithm with feed-forward Neural Networks is discussed in [95]. There the author has proposed a NARX approach with feed-forward neural network and a RPROP to model the network. In [96] the authors have proposed dynamic modeling of a TRMS using Analytical and Empirical Approaches where the TRMS is modeled in terms of vertical and horizontal 1DOF dynamics using Newtonian and Lagrangian methods based analytical approaches and neural networks based empirical approaches.

The scope of this chapter is to find out an efficient neural model for a highly nonlinear TRMS system. The modeling is done assuming no prior knowledge of model structure or parameters relating to physical phenomena, i.e. black-box modeling. This is realized by minimizing the prediction error of the actual plant output and the model output. Different identification algorithms such as sequential hybridization algorithm and memetic algorithm and neural networks discussed previously have been used to model the system. The various attractive features of DEs such as simplicity and faster convergence motivate utilization of a DE for this purpose.

5.3 Modeling of the TRMS

The TRMS used in this work is supplied by Feedback Instruments designed for control experiments. This TRMS setup serves as a model of a helicopter. It consists of two rotors placed on a beam with a counterbalance. These two rotors are driven by two D.C motors. The main rotor produces a lifting force allowing the beam to rise vertically making the rotation around the pitch axis. The tail rotor which is smaller than the main rotor is used to make the beam

turn left or right around the yaw axis. Both the axis of either or both axis of rotation can be locked by means of two locking screws provided for physically restricting the horizontal and or vertical plane of the TRMS rotation. Thus, the system permits both 1 and 2 DOF experiments. Although the TRMS system permits MIMO experiments, this work addresses the problem of identifying the system in a single-input single-output (SISO) mode in the pitch axis (i.e. vertical movement) or yaw axis (i.e. horizontal movement). The yaw and pitch movement caused by the tail and main rotor can be physically locked and as a result there is no cross-coupling effect between the two channels of the TRMS. The 1 DOF around the pitch and yaw axis is identified by SH method discussed in chapter 3. The memetic approach is applied to identify only 1 DOF pitch movement. The schematic diagram of the laboratory setup is shown in Fig. 5.1

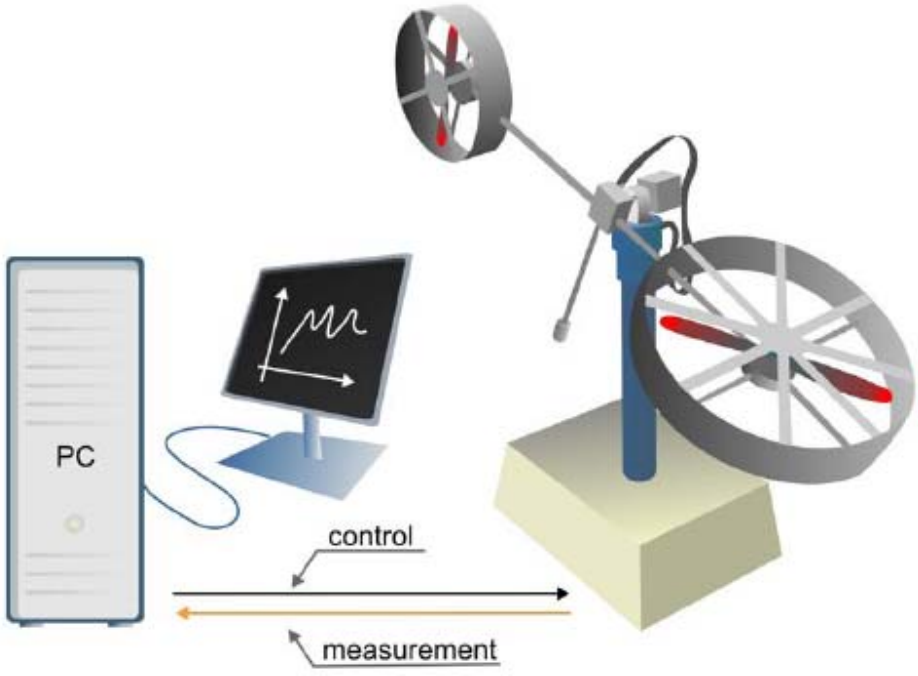

Figure 5.1: The laboratory set-up: TRMS system

The modeling in elevation of helicopter is carried out using standard physics laws of angular momentum [98]. Considering the free body diagram of helicopter model, different torque produced by different forces is balanced about

the pivot point. The different torques produced is gravitational torque, frictional torque, centrifugal torque, main rotor torque and gyroscopic torque.

5.3.1 Gravitational and Centrifugal Torque

Consider the free body diagram shown in Figure (5.2), the weight of the helicopter and centrifugal force produce respective torques about the pivot point. Equation (5.1) describes the gravitational torque produced by the model weight [98].

$$\tau_w = lw \sin\psi = M_{gr} \sin\psi \tag{5.1}$$

where τ_w = Gravitational torque $(N.m)$

ψ = Elevation angle (rad)

w = Weight of the helicopter (Kg)

l = Moment arm (m)

M_{gr} = Gravity momentum parameter$(N.m)$

Equation (5.2) describes the centrifugal torque produced by centrifugal force during rotation in horizontal plane.

$$\tau_c = lF_c \cos\psi \tag{5.2}$$

$$F_c = ml\dot{\varphi}^2 \sin\psi \tag{5.3}$$

$\dot{\varphi}$ = Angular Velocity in horizontal plane (rad/sec)

F_c = Centrifugal Force (N)

5.3.2 Main Rotor Torque

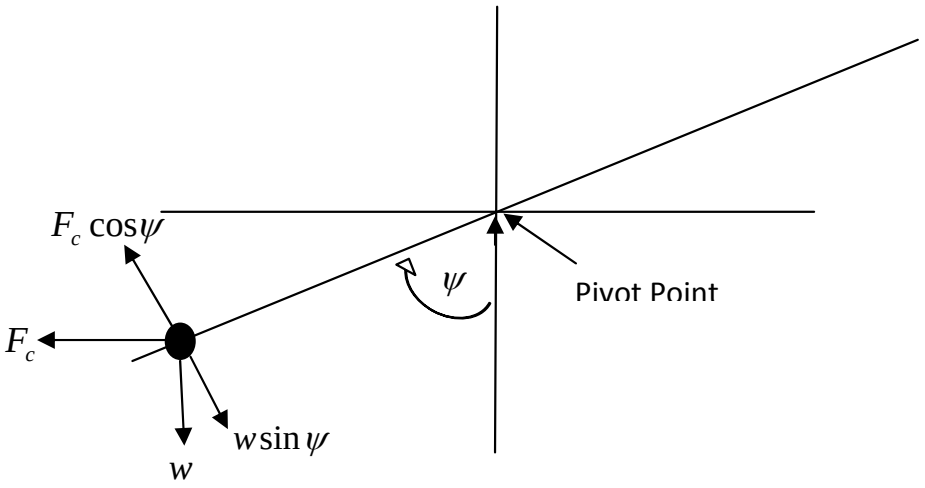

Figure 5.2: Gravitational and centrifugal forces acting of the helicopter in the vertical plane

The main rotor force produced is the consequence of its angular speed as shown in Fig. 5.2. The more the angular speed of rotor the more force will be induced on the helicopter body, which will produce angular torque about the pivot point. Therefore we can say that

$$\tau_1 \propto F_1(\omega_1) \tag{5.4}$$

F_1 = Main rotor Force (N)

ω_1 = Main rotor angular velocity (rad/sec)

τ_1 = Main rotor torque $(N.m)$

The main motor is approximated by the first order transfer function which can be described as

$$G_1 = \frac{k_1}{T_{11}s + T_{10}} u_1 \tag{5.5}$$

The nonlinearity caused by the rotor can be estimated as second order polynomial and finally the torque induced in helicopter body via motor can be given in equation (5.6).

$$\tau_1 = a_1 G_1^2 + b_1 G_1 \tag{5.6}$$

where a_1 and b_1 are the nonlinearity constant parameters.

5.3.3 Gyroscopic Torque

Gyroscopic torque occurs as a result of Coriolis forces acting on helicopter elevation dynamics. This torque results when moving main rotor changes its position in azimuth. Thus resultant gyroscopic torque caused by the main rotor and azimuth rotation can be calculated from the Fig. 5.3 as

$$\tau_G = k_{gy}\dot{\varphi}\tau_1 \cos\psi \tag{5.7}$$

where φ = Azimuth Angle (rad)

$\dot{\varphi}$ = Angular velocity in Azimuth (rad/sec)

k_{gy} = Constant of proportionality (sec/rad)

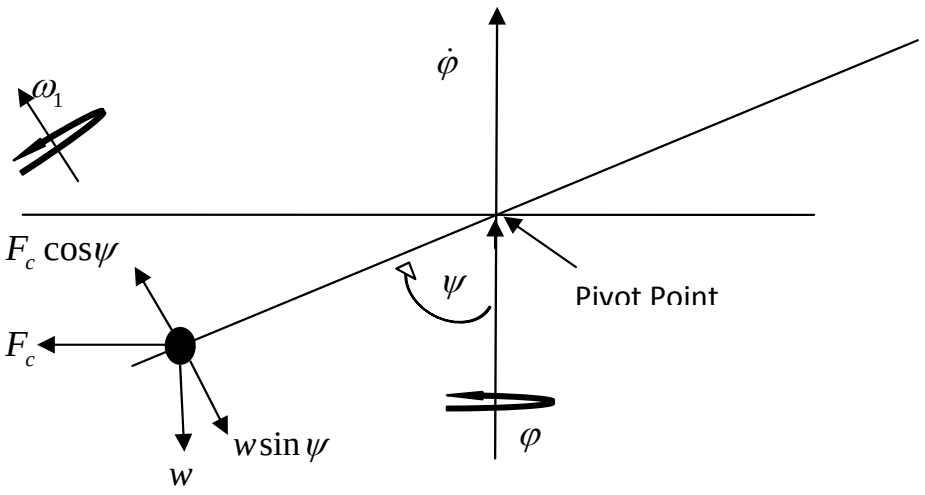

Figure 5.3: Gyroscopic torque due to rate of change of azimuth in vertical plane

5.3.4 Frictional Torque

The frictional torque can be estimated from the following equation

$$\tau_{f_1} = B_1\dot{\psi} \tag{5.8}$$

where B_1 = Damping Constant $(N.m.s/rad)$

All the torques produced in the helicopter body discussed above shown in Fig.5.4. The net torque is given by

$$I_1\ddot{\psi} = \tau_1 + \tau_c + \tau_G - \tau_w - \tau_{f_1} \tag{5.9}$$

I_1 = Moment of inertia of the helicopter body around horizontal axis $(kg.m^2)$

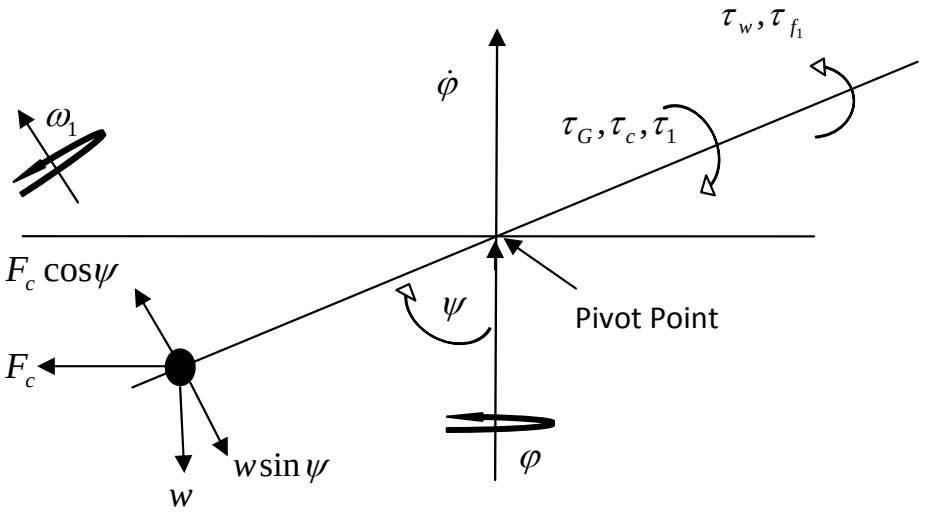

Figure 5.4: Net torques acting on the helicopter in the vertical plane

5.3.5 Azimuth Dynamics

Similar equations can be written for the horizontal plane motion. The net torques produced in horizontal plane is shown in Fig. 5.5.

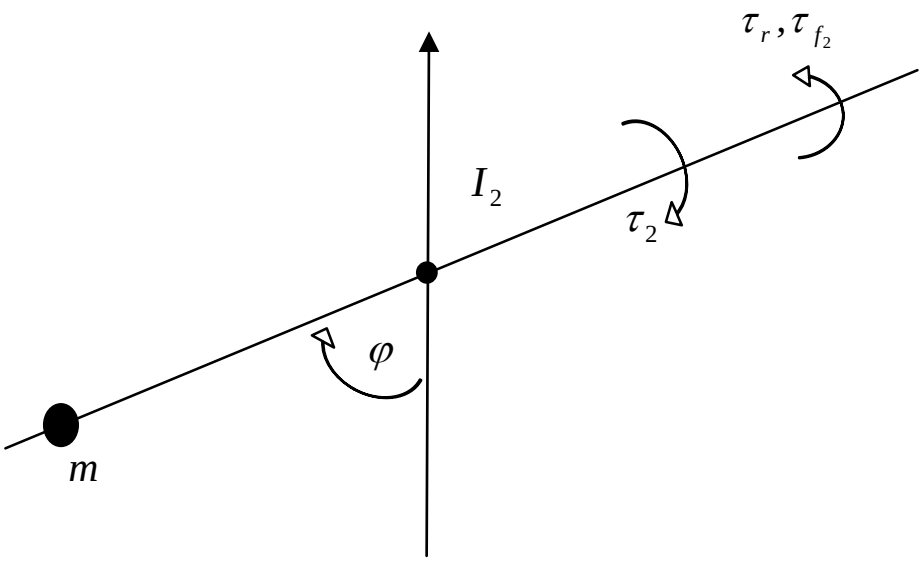

Figure 5.5: Mechanical torques produced in horizontal plane

$$I_2\ddot{\varphi} = \tau_2 - \tau_r - \tau_{f_2} \tag{5.10}$$

τ_2 = Side rotor torque $(N.m)$

τ_{f_2} = Frictional torque (N)

τ_r = Main motor reaction torque $(N.m)$

I_2 = Moment of Inertia in vertical plane $(Kg.m^2)$

The fictional torque and side rotor torque are calculated similarly to elevation dynamics. As they are proportional to rate of change of angular position frictional torque can be estimated from the following equation

$$\tau_{f_2} = B_2\dot{\varphi} \tag{5.11}$$

where B_2 = Damping Constant $(N.m.s/rad)$

The main rotor reaction torque acting on azimuth can be estimated by first order transfer function shown in equation (5.12).

$$\tau_r = \frac{k_c\,(T_o s + 1)}{T_p s + 1} \tag{5.12}$$

where T_o and T_p are the cross reaction momentum parameter.
The tail motor is approximated by the first order transfer function which can be described as

$$G_2 = \frac{k_2}{T_{21}s + T_{20}}u_2 \tag{5.13}$$

The nonlinearity caused by the rotor can be estimated as second order polynomial and finally the torque induced in helicopter body via motor can be given in equation (5.14)[98].

$$\tau_2 = a_2 G_2^2 + b_2 G_2 \tag{5.14}$$

where a_1 and a_2 are the nonlinearity constant parameters and u_1 , u_2 are the inputs to main and tail motors respectively.

5.4 Experimental Set-up

The TRMS plant has two degrees of freedom. There are rotors (the main and tail rotors), driven by DC motors, at both ends of the beam. The two rotors can rotate the unit about vertical and horizontal axis. With the horizontal axis locking screw removed the larger rotor weight should cause the rotor arm to rest at an angle approximately 28 degrees to the horizontal axis. A counterbalance arm with a weight at its end is fixed to the beam at the pivot. The state of the beam is described by four process variables: horizontal and vertical angles measured by position sensors fitted at the pivot, and two corresponding angular velocities. The input to the TRMS are the rotor voltages i.e. V_p, V_y. The position of the beam is measured with the help of incremental encoders which provide relative position signal. The model is interfaced with desktop computer via PCI1711 card which is accessible in MATLAB Simulink environment through Real-time Toolbox and Real Time Windows Target Toolbox. These toolboxes provide us the liberty to access the encoder values and issue commands to DC motors and servo system. The PCI1711 consists of two blocks namely Feedback encoder block and Feedback DAC block through which the external equipments are interfaced. Table 5.1 gives some specification of the TRMS used.

Table 5.1: Parameter values for modeling TRMS

Symbols	Parameter	Value
I_1	Moment of inertia of the vertical rotor	$6.8 \times 10^{-2} kg - m^2$
I_2	Moment of inertia of the horizontal rotor	$2 \times 10^{-2} kg - m^2$
a_1	Static characteristic parameter	0.0135
b_1	Static characteristic parameter	0.0924
a_2	Static characteristic parameter	0.02
b_2	Static characteristic parameter	0.09
$M_g r$	Gravity momentum	$0.32N - m$
B_1	Fictional function parameter	$6 \times 10^{-3} kg.m/sec$
B_2	Fictional function parameter	$1 \times 10^{-1} kg.m/sec$
$K_g y$	Gyroscopic parameter	$0.05N.m/sec$
k_1	Motor 1 gain	1.1
k_2	Motor 2 gain	0.8
$T_1 1$	Motor 1 denominator parameter	1.1
$T_1 0$	Motor 1 denominator parameter	1
$T_2 1$	Motor 2 denominator parameter	1
$T_2 0$	Motor 2 denominator parameter	1
T_p	Cross reaction momentum parameter	2
T_o	Cross reaction momentum parameter	3.5
k_c	Cross reaction momentum gain	-0.2

5.5 Neuro Modeling of TRMS

In order to model the TRMS in terms of 1DOF pitch and yaw dynamics a MLP neural network model of $5 \times 11 \times 1$ configuration has been used for SH algorithm but for the memetic approach the configuration is taken as $5 \times 21 \times 1$. In other words, the NN-based model has 5 inputs, 11 or 21 neurons in hidden layer and 1 neuron in output layer. To find a suitable configuration it is common to start from a simple configuration, usually only one hidden layer, and then increase the number of neurons and even the number of layers if necessary. The inputs are main rotor voltage at present time, $V_p(t)$, main rotor voltage at previous time, $V_p(t-1)$, main rotor voltage at two samples before, $V_p(t-2)$, pitch angle of the beam at previous time, $\Psi_p(t-1)$ and pitch angle of the beam

at two samples before, $\Psi_p(t-2)$. Figure 5.6 shows the structure of the NN based model in terms of 1DOF pitch dynamics. It is noted that the activation functions used in the hidden layer and output layer are sigmoid and pure linear respectively. The first 300 data were used for training and the whole 500 data were use for the test. The Sample time of data is set to be 0.1 second which implies that the frequency of sampling is 10 Hz. The NN has been trained with both the algorithms i.e. sequential hybrid algorithm and memetic algorithm and the results are obtained by taking the average of 30 independent runs. It is noted that all data for training and testing have been extracted from the real TRMS.

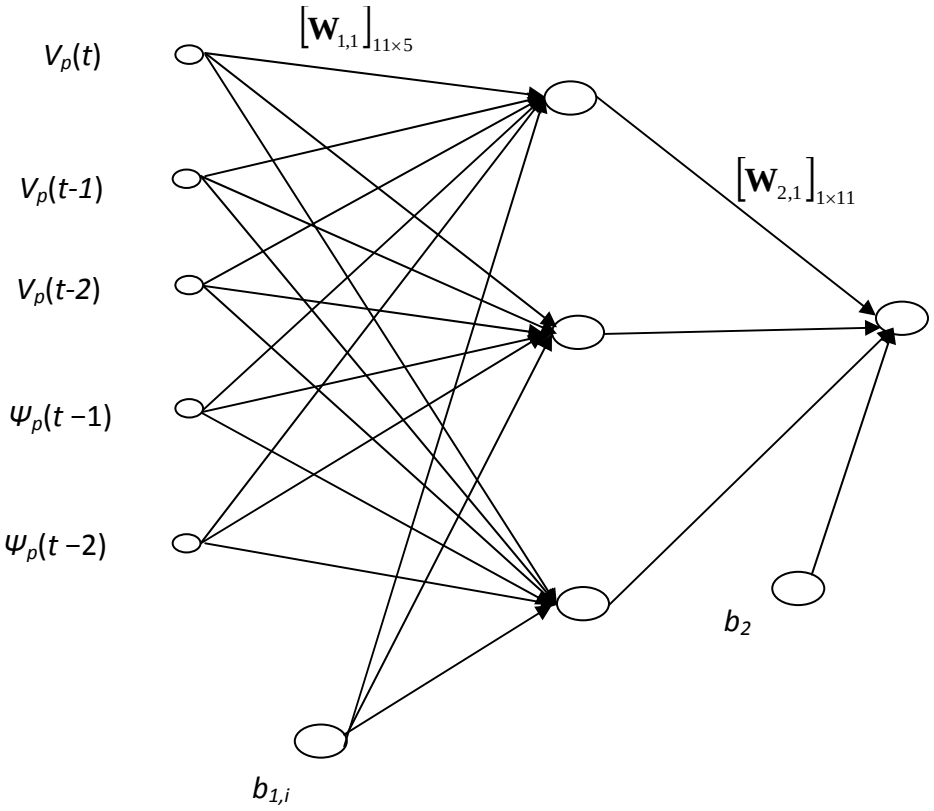

Figure 5.6: The structure of the NN based model in terms of 1DOF horizontal

5.6 Experimental Input-Output Data

In nonlinear system identification, the type of input signal to be used plays a crucial role and has a direct bearing on the fidelity of the resulting identified model. The excitation signal should have two important characteristics:

- It should be able to excite all the dynamic modes of interest, that is, the signal should be persistently exciting.
- It should be rich in amplitude level, that is, have different levels of input amplitudes over the whole range of operation.

In this work we have taken the input as a summation of sinusoidal signals which is a noise signal shown in Fig. 5.7 given as:

$$1.2 + 0.2 \left[\begin{array}{l} 0.5 \sin\left((2\pi \times 0.5)\, t - 20\right) + 0.4 \sin\left((2\pi \times 0.067)\, t - 17\right) \\ +0.42 \sin\left(2\pi \times 0.15\right) t + 0.6 \sin\left((2\pi \times 0.24)\, t - 40\right) \end{array} \right] + rand$$

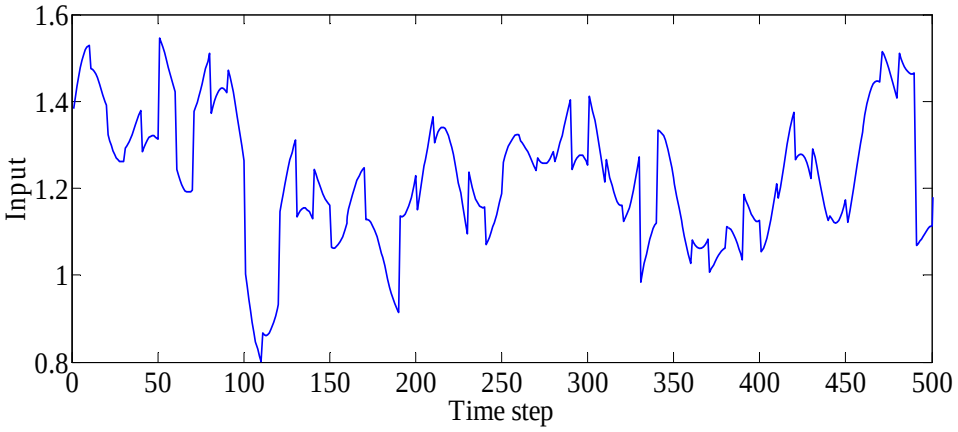

Figure 5.7: Applied input signal to TRMS

5.7 Results and Discussions

5.7.1 SH Algorithm

In this example we load data from a real time TRMS system. One data set consists of 1 DOF pitch dynamics where the input is the voltage to the main motor and the output is the pitch angle. The other data set consist of 1 DOF yaw dynamics where the input is the voltage to the tail motor and the output is the yaw angle. First a Hammerstein-Wiener model is tried out, with a sigmoidal nonlinearity. The simulated output is then compared to the measured output for the whole data record. Figure 5.8 and Fig. 5.9 shows the measured output and the model simulated output for pitch and yaw dynamics. It is clear from the results that the conventional Hammerstein-Wiener model is not adequate to identify the dynamics of a real time TRMS system. Table 5.2 gives the parameter values for DE+LM+NN algorithm.

Table 5.2: Parameters for DE+LM+NN

Total number of iterations	1000
Population size, P	50
Upper and lower bound of weights	[0 1]
Mutation constant factor , F	0.6
Cross over constant, C	0.5

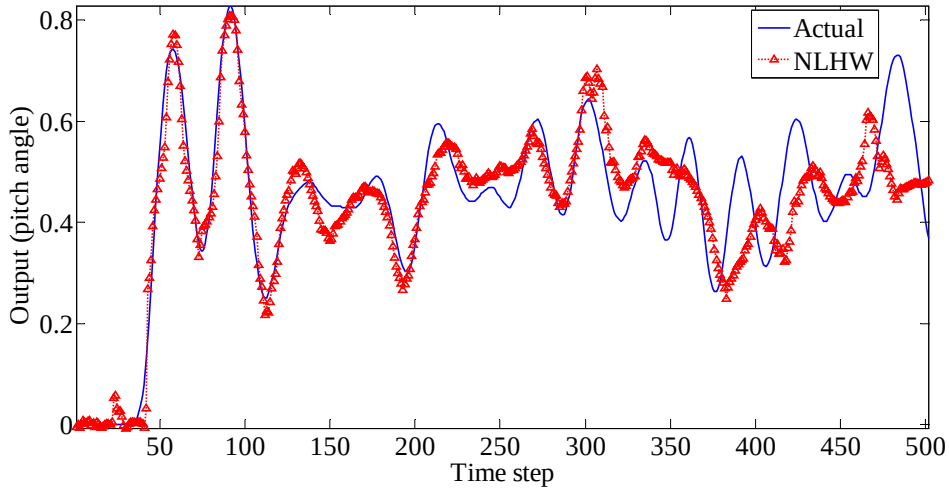

Figure 5.8: NLHW identification performance (pitch angle)

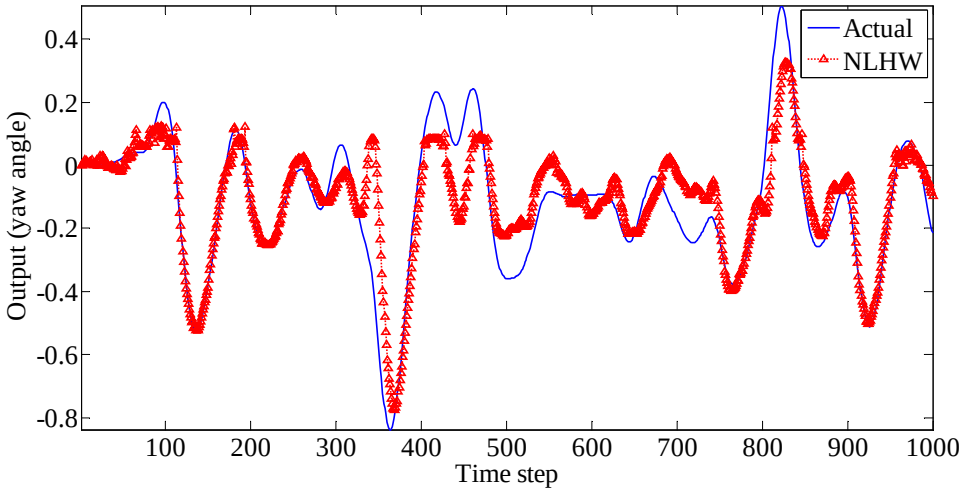

Figure 5.9: NLHW identification performance (yaw angle)

Figure 5.10 shows the identification performance of the TRMS using proposed DE+LM+NN, only DE and NN methods. Figure 5.11 shows the zoomed version of the results between the time steps of 413 to 435. From the results it is clear that DE+LM+NN identification results are better than only DE and NN. Figure 5.12 shows the error between the actual and identified outputs for NN and DE+LM+NN.

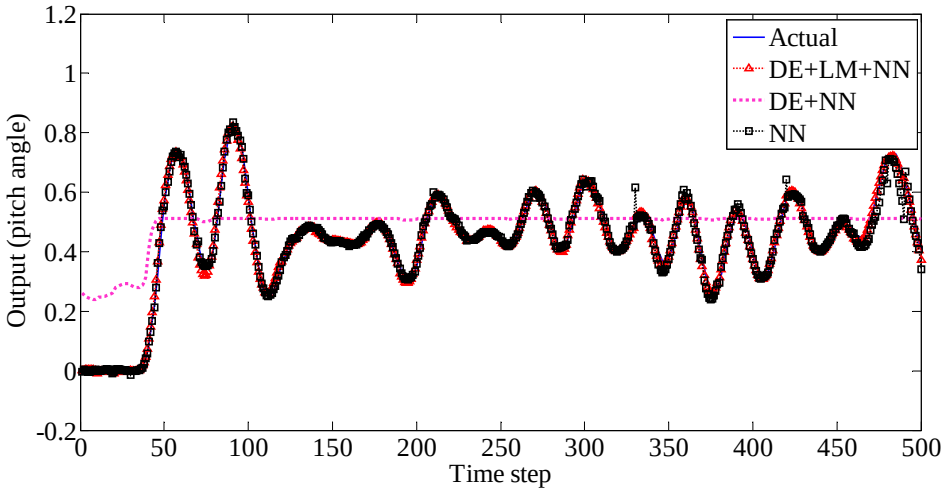

Figure 5.10: DE+LM+NN dentification performance (pitch angle)

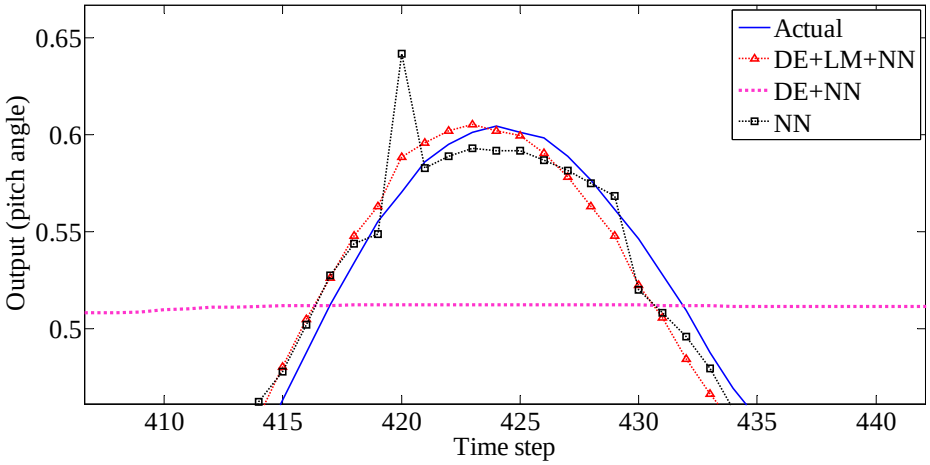

Figure 5.11: DE+LM+NN zoomed identification performance (pitch angle)

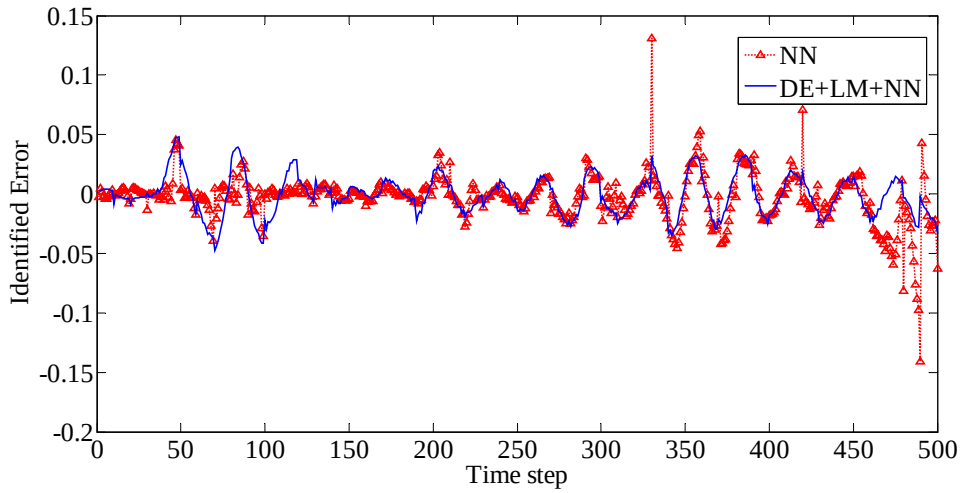

Figure 5.12: Identification error (pitch angle)

Figure 5.13 shows the identification performance of the yaw dynamics DE+LM+NN, only DE and NN methods. Figure 5.14 shows the zoomed version of the results between the time steps of 345 to 385. Clearly the DE+LM+NN model gives best performance compared to other model. Figure 5.15 shows the error between the actual and identified outputs for NN and DE+LM+NN.

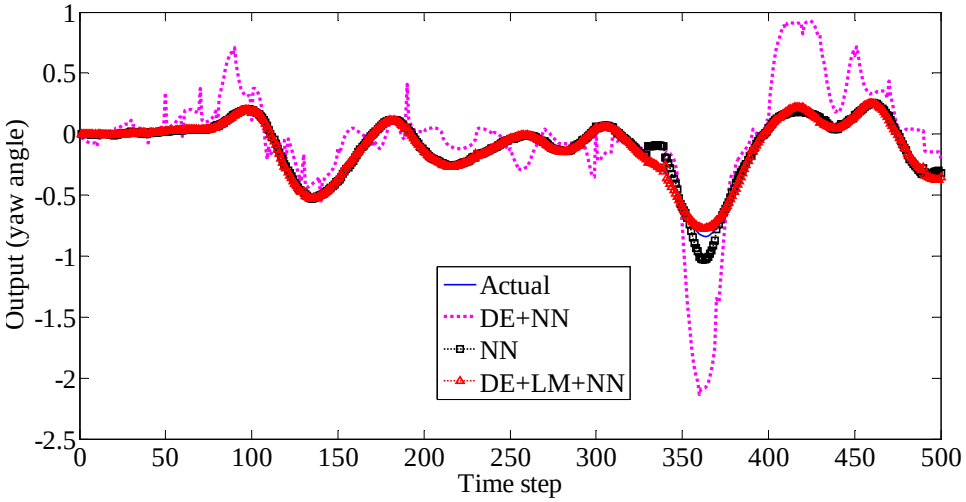

Figure 5.13: DE+LM+NN identification performance (yaw angle)

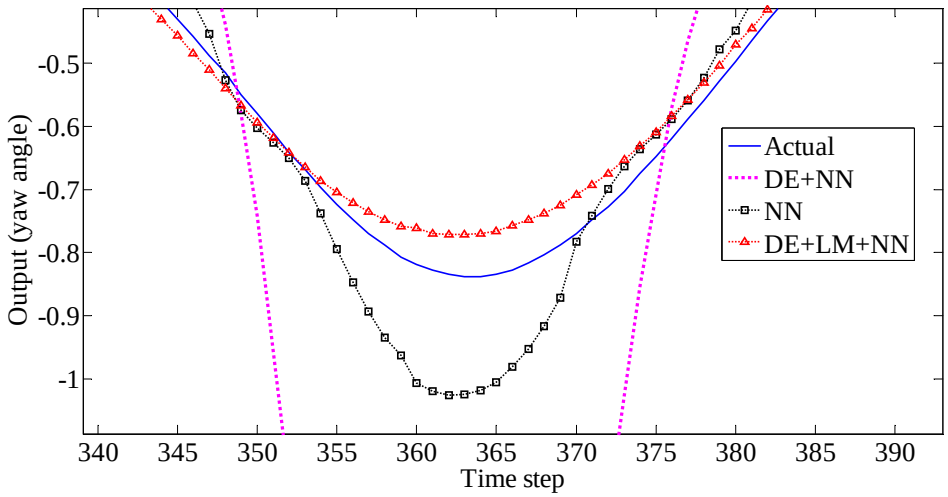

Figure 5.14: DE+LM+NN zoomed dentification performance (yaw angle)

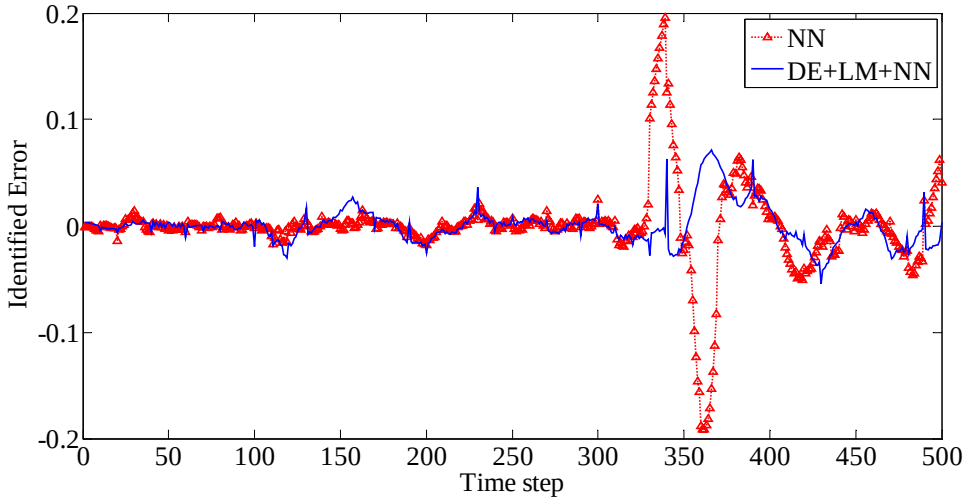

Figure 5.15: Identification error (yaw angle)

Figure 5.16 and 5.17 show the Power spectral densities (PSDs) for DE+LM+NN model and the real time TRMS. It is obvious from these figures that PSDs of the model and system responses are closely overlapped.

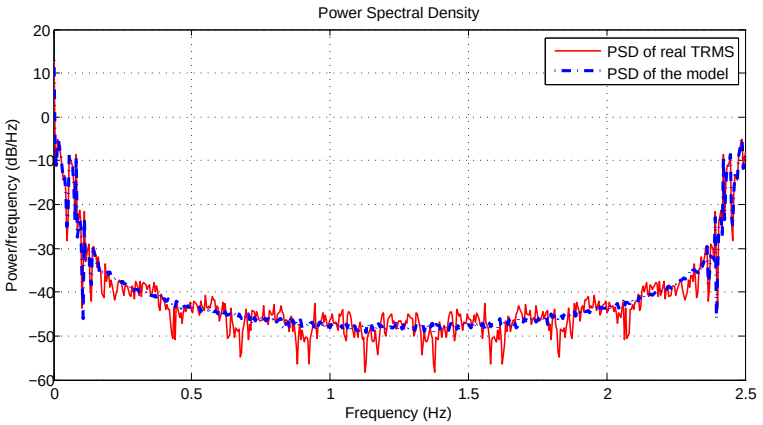

Figure 5.16: Power spectal density for pitch

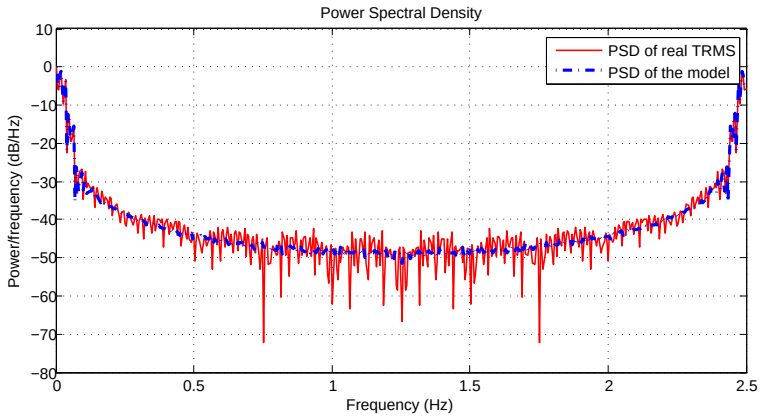

Figure 5.17: Power spectal density for yaw

5.7.2 Memetic Algorithm

DE and DEBP identification

Here we have taken 21 number of hidden neurons and 1000 epochs to train the neural network. Figure 5.18 compares the actual output $y(t)$, and identified plant output $\hat{y}(t)$ of a 1 DOF pitch dynamics of a TRMS within the time step of 0 to 500. As the identification performances shown in Figure 5.18 are overlapping each other, in Figure 5.19 we have shown the results within the time step of 86 to 96. From this it is clear that the proposed DEBP exhibits better identification ability compared to DE approach. Figure 5.20 and 5.21

shows the error between the actual and identified model. Figure 5.22 gives the comparison of SSE between DE and DEBP, where it is found that the value of SSE for DEBP is 0.0036 whereas for DE identification is 0.0110. The parameters used in simulation study was same as Table 4.1 given in chapter 4 except the number of epochs which has been taken as 1000 here.

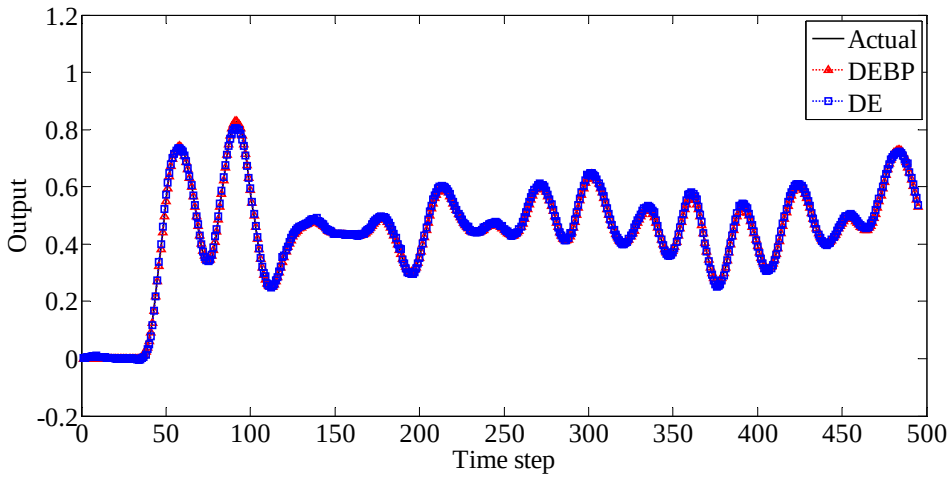

Figure 5.18: DE and DEBP identification performance

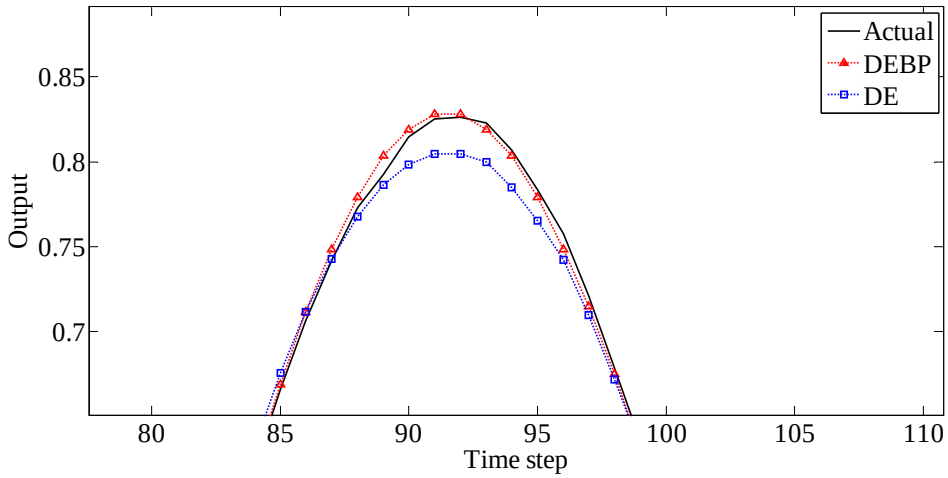

Figure 5.19: DE and DEBP zoomed identification performance

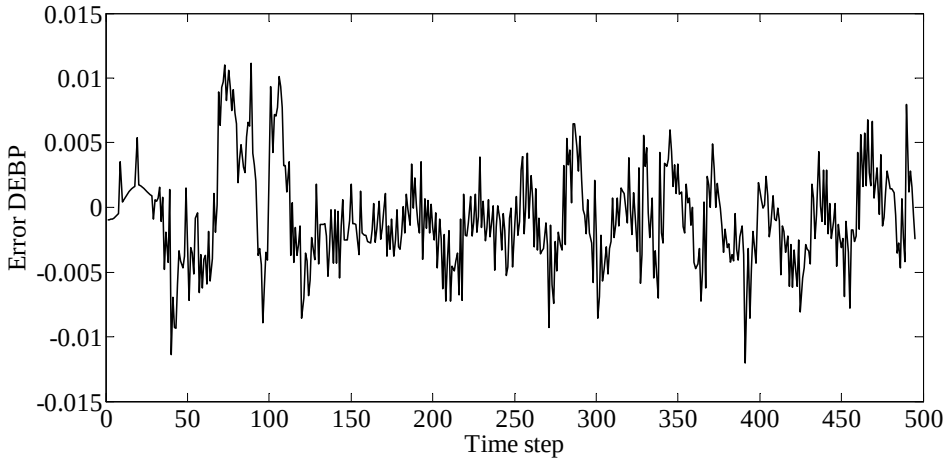

Figure 5.20: Error in modeling (DEBP identification)

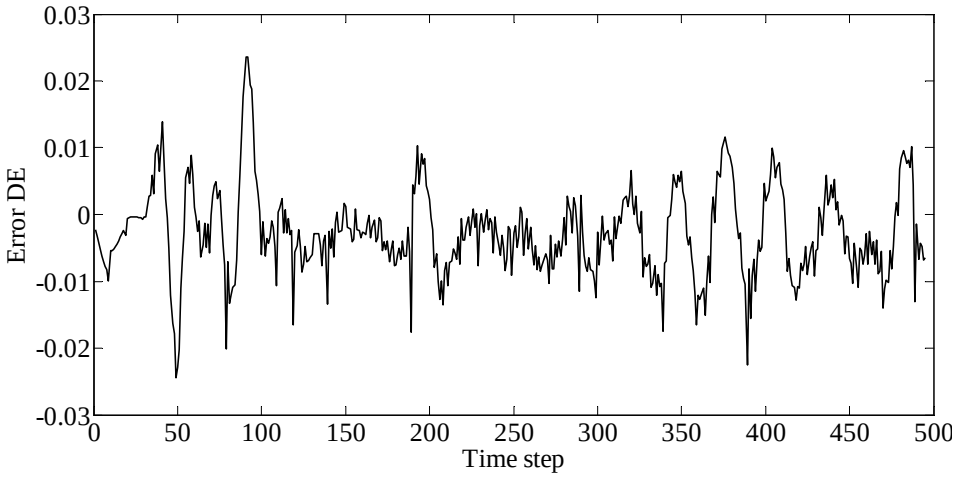

Figure 5.21: Error in modeling (DE identification)

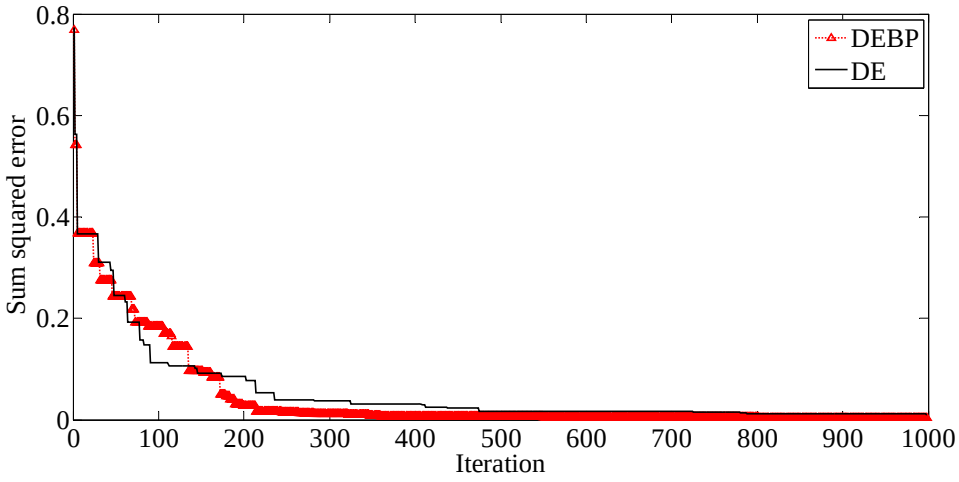

Figure 5.22: A comparisons on the convergence on the SSE (DE, DEBP)

GA and GABP identification

Figure 5.23 shows the identification performance of 1 degree of freedom (DOF) vertical TRMS by GA and GABP based model. As the identification performances shown in Fig. 5.23 are overlapping each other, in Figure 5.24 we have shown the results within the time step of 208 to 221. From this it is clear that the GABP identification approach exhibits better identification ability compared to GA approach. Figure 5.25 gives the SSE where it is found that the value of SSE for GABP is 0.0197 whereas for GA identification is 0.0327. Figure 5.26 and 5.27 show the error between the actual and identified model for both the identification scheme.

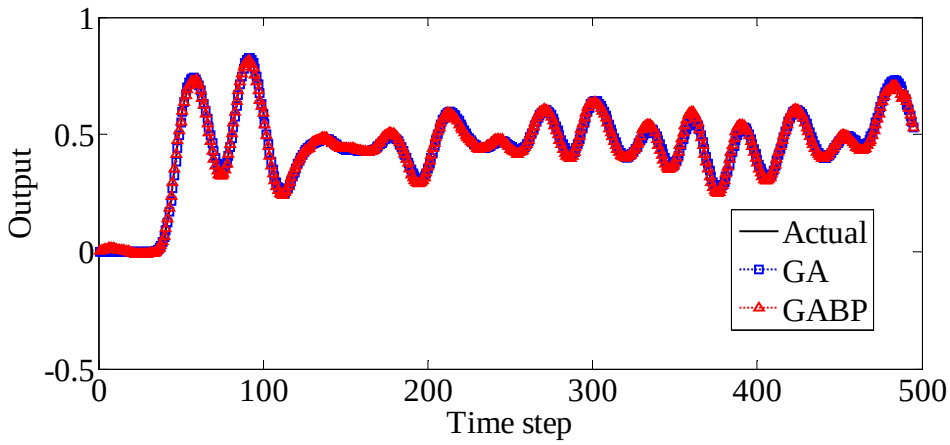

Figure 5.23: GA and GABP identification performance

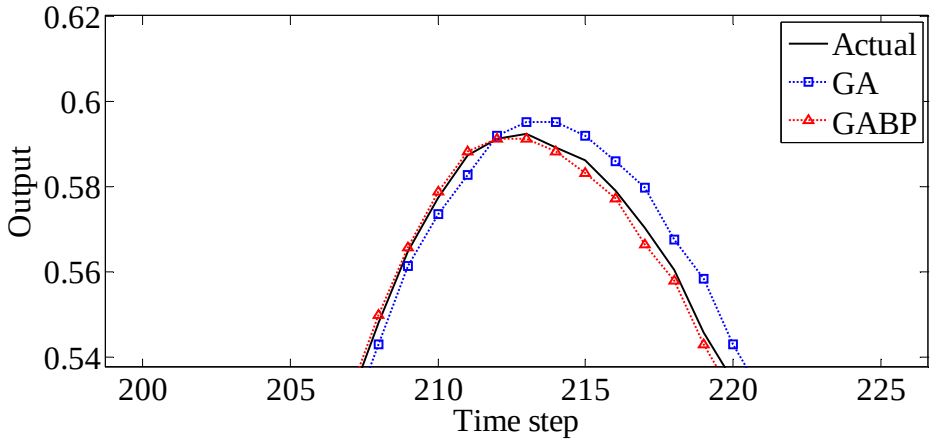

Figure 5.24: GA and GABP zoomed identification performance

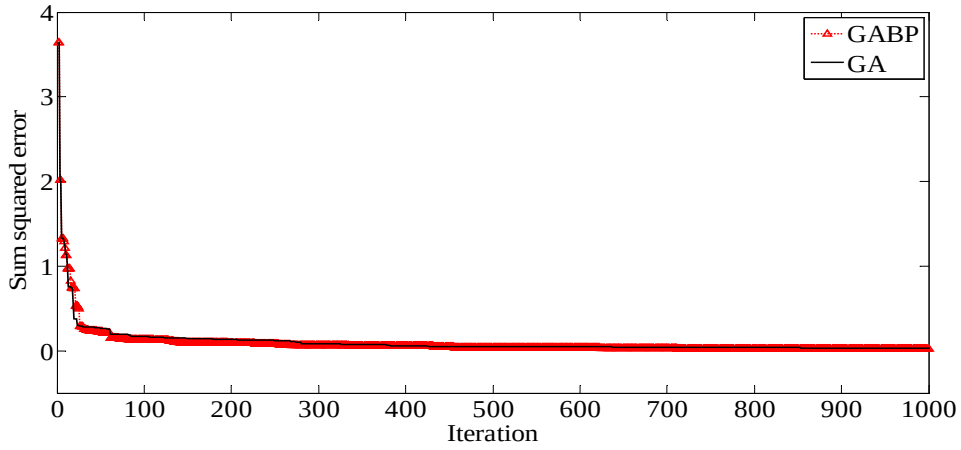

Figure 5.25: A comparisons on the convergence on the SSE (GA, GABP)

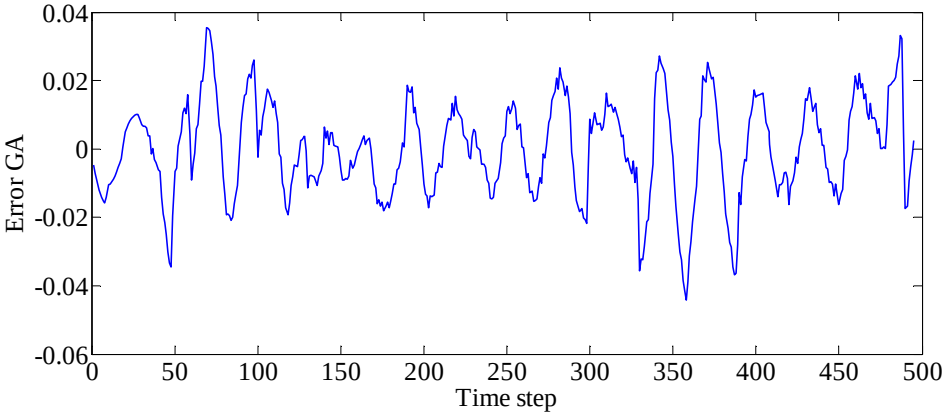

Figure 5.26: Error in modeling (GA identification)

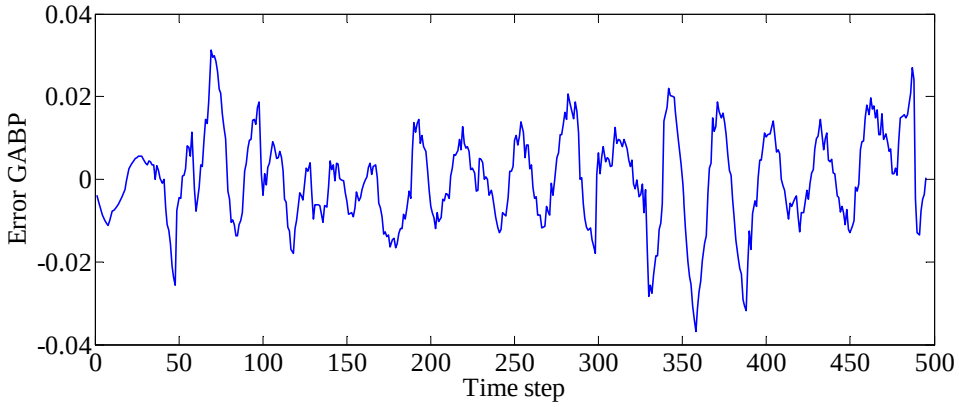

Figure 5.27: Error in modeling (GABP identification)

PSO and PSOBP identification

Figure 5.28 shows the identification performance of 1 DOF vertical TRMS by PSO and PSOBP based model. In Fig. 5.29 we have shown the zoomed results within the time step of 87 to 96. From this it is clear that the PSOBP approach exhibits better identification ability compared to PSO approach. Figure 5.30 gives the SSE where it is found that the value of SSE for PSOBP is 0.0235 whereas for PSO identification is 0.0505. Figure 5.31 and 5.32 shows the error between the actual and identified model for both the identification scheme.

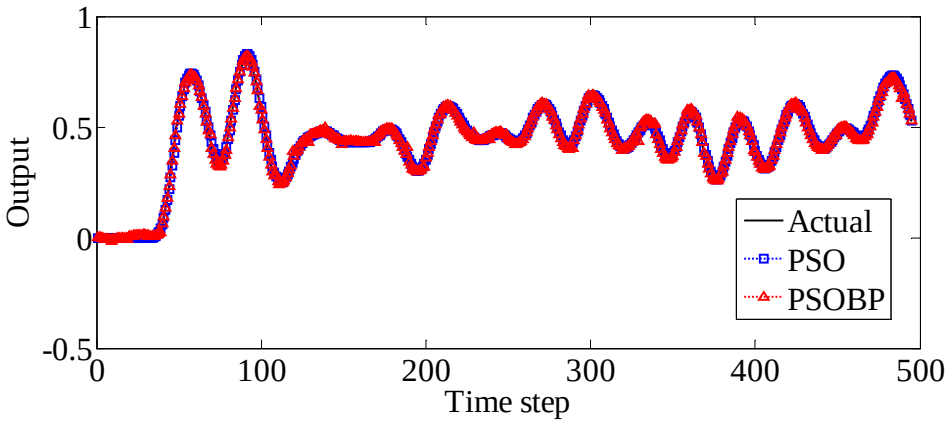

Figure 5.28: PSO and PSOBP identification performance

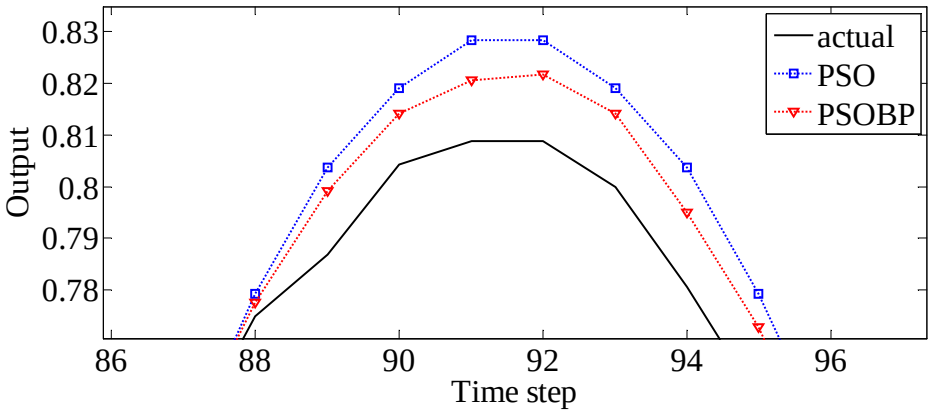

Figure 5.29: PSO and PSOBP zoomed identification performance

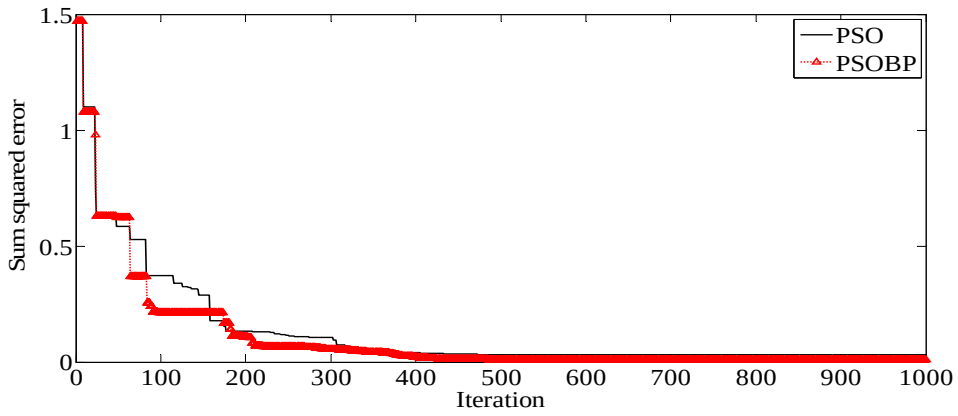

Figure 5.30: A comparisons on the convergence on the SSE (PSO, PSOBP)

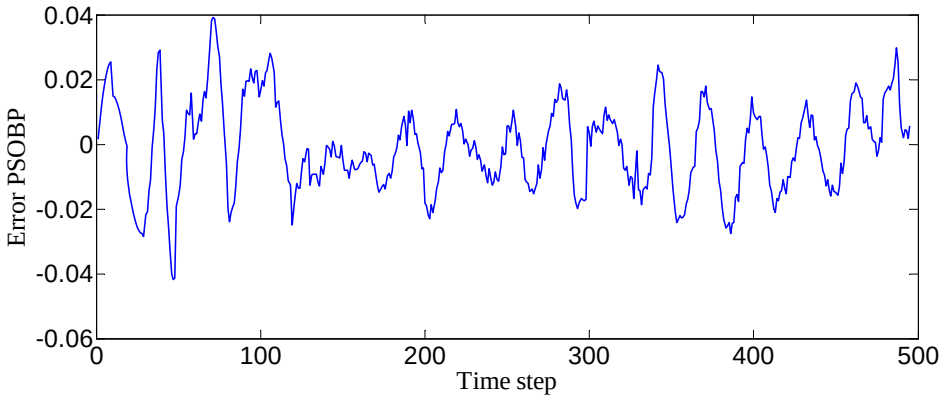

Figure 5.31: Error in modeling (PSOBP identification)

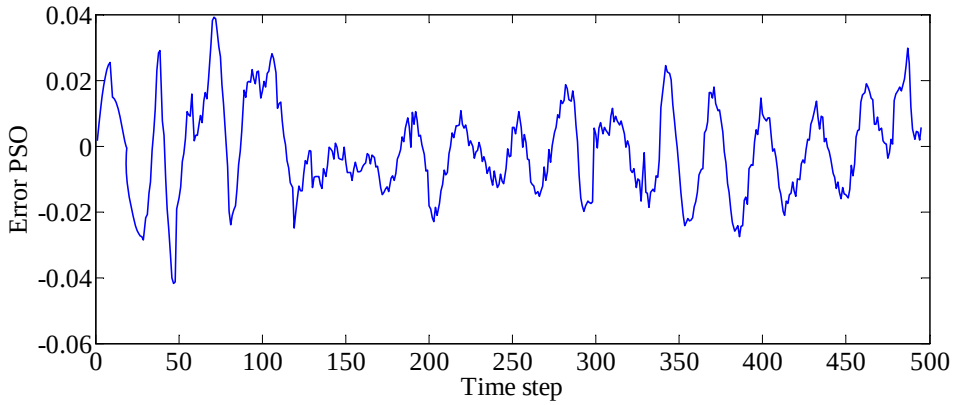

Figure 5.32: Error in modeling (PSO identification)

Finally it has been seen that among all the methods the proposed DEBP method is having lowest SSE i.e. 0.0036 amongst all the methods disused. Table 5.3 gives the SSE for different methods.

Table 5.3: SSE for different methods

Sl No	Methods	SSE
1	PSO	0.0505
2	PSOBP	0.0235
3	GA	0.0327
4	GABP	0.0197
5	DE	0.0110
6	DEBP	0.0036

In this section we have provided an extensive study of MAs applied to nonlinear system identification. The proposed method DEBP exploits the advantages of both the local search and global search. It is interesting to note that the local search pursued after the mutation and crossover operation helps in intensifying the region of search space which leads to faster convergence. We investigated the performance of the proposed version of the DEBP algorithm using a real time multi input multi output highly nonlinear TRMS system. The simulation studies showed that the proposed algorithm of DEBP outperforms in terms of convergence velocity among all the discussed algorithms. This shows it is advantageous to use DE over other evolutionary computation such as GA and PSO in nonlinear system identification.

5.8 Summary

This chapter discusses the identification of nonlinear systems using different hybrid approaches. Hybridization of a global search algorithm with a local search is a challenging approach for optimization problems where the individual methods without hybridization may suffer from slow convergence and trapped by local minima. From the identification results and the error graphs it is found that the proposed approaches are able to identify accurate models of

different nonlinear systems. The identification performances obtained by both the SH and MA algorithm were found to be similar but SH found to provide faster convergence than MAs.

Chapter 6

An Opposition Based Differential Evolution Approach to Nonlinear System Identification

6.1 Introduction

The concept of opposition-based learning (OBL) was introduced by Tizhoosh [116]. It is applied to accelerate reinforcement learning [117] and back-propagation learning in neural networks [118]. The main idea behind OBL is the simultaneous consideration of an estimate and its corresponding opposite estimate (i.e., guess and opposite guess) in order to achieve a better approximation for the current candidate solution. In this work, OBL has been used to accelerate the convergence rate of the DE. Hence, the proposed approach described in this chapter is called opposition-based differential evolution. ODE uses opposite numbers during population initialization and also for generating new populations during the evolutionary process. The opposite numbers have been utilized to speed up the convergence rate of DE optimization algorithm. Purely random resampling or selection of solutions from a given population has the chance of visiting or even revisiting unproductive regions of the search space. It has been demonstrated in [116, 119] that the chance of this occurring is lower for opposite numbers than it is for purely random ones. In fact, a mathematical proof has already been proposed to show that opposite numbers are

more likely to be closer to the optimal solution than purely random ones [120]. In [124], the usefulness of opposite numbers is investigated by replacing them with random numbers and it is applied for population initialization and generation jumping for different versions of DE. However, a little work has been reported on applying ODE to system identification and its use in training neural network employed as nonlinear system identifiers. Therefore, it attracts the attention of the present work for exploiting the use of OBL for effective neural network training. In this work, an opposition based differential evolution has been applied as a global optimization method for improving learning of feed-forward neural networks used for identification nonlinear systems.

Nonlinear system as considered in [61, 62] has been chosen in this work for demonstrating the efficacy of the proposed ODE-NN system identification approach in comparison to DE-NN approach. In this chapter, an opposition based differential evolution method combined with LM has been applied as a global optimization method for training feed-forward neural networks. In the proposed scheme, the ODE is used to train the neural network that is chosen as a suitable candidate for nonlinear system identification. Then the network is trained using LM after observing the trends of training towards minimum through ODE. The role of the ODE here is to find the basin of global minimum and then LM is used to move forward to locate the exact minimum point. This switch over from one algorithm to other can be done after satisfying a predefined criteria i.e. number of generations or particular value of error criterion. As LM is a gradient based algorithm, it can be exploited to increase the convergence speed for reaching the global minimum.

6.2 Opposition based Differential Evolution

In most of the situations evolutionary algorithms (e.g. GA, PSO and DE), weight optimization in a neural network, the learning begins at a random point. In each iteration, the solution obtained moves towards the optimal solution

and the search process terminates when some predefined criteria is satisfied. The time of computation generally depends on the initial guess i.e. more is the distance between the initial guess to optimal solution more time it will take to terminate and vice versa. If the initial random guess is far away from the optimal solution, assuming that it is in the exact opposite location of the optimal solution than the search will take considerably more time to converge. So in the absence of a-priori knowledge about the solution, random guess cannot be a best initial guess. Hence, at starting it is always better to look in all directions simultaneously i.e. more efficiently in opposite direction. Opposition based learning improves the chance of starting with better initial population by checking the opposite solutions. According to the probability theory, there is a 50 percent chance that the random guess is at larger distance than the opposite guess. So instead of taking the random guess as initial population the opposite guess is to be found out and the closer of these two guseses are taken as initial population. Starting with the closer of the two guesses (as judged by its fitness) has the potential to accelerate convergence. The same approach can be applied not only to initial solutions but also continuously to each solution in the current population. Before applying OBL to the problem of system identification, we first define the concept of opposite numbers [116].

6.2.1 Definition of opposite number and opposite point

Let $x \in [a, b]$ be a real number. The opposite number $\tilde{x}$ is defined by

$$\tilde{x} = a + b - x$$

Let $p = (x_1, x_2, \cdots, x_d)$ be a point in the d dimensional space, where $x_1, x_2, \cdots, x_d \in R$ and $x_i \in [a_i, b_i]$. The opposite point $\tilde{p} = (\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_d)$ where $\tilde{x}_i = a_i + b_i - x_i$. Figure 6.1 gives the illustration of a point and its opposite point in one and two dimension.

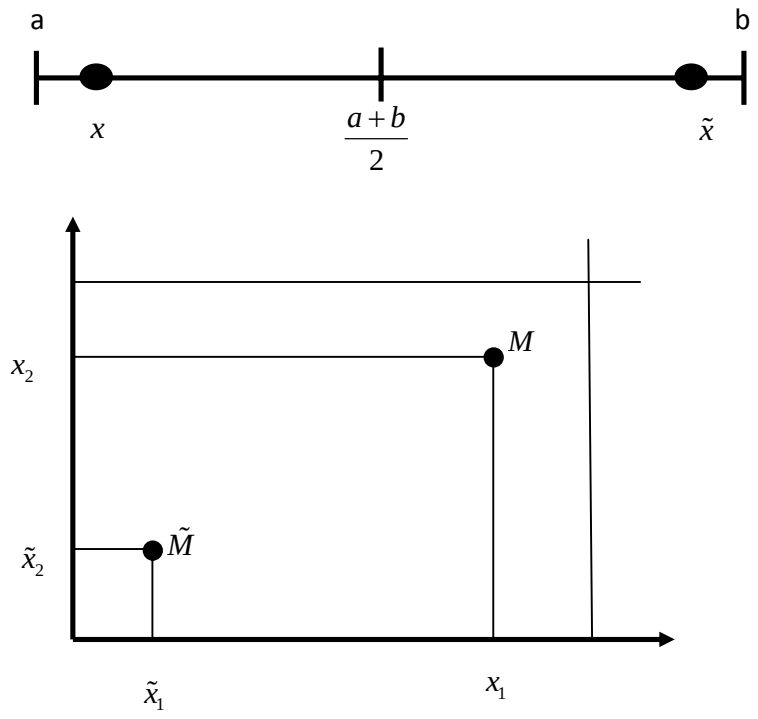

Figure 6.1: Illustration of a point and its corresponding opposite in one and two dimensional spaces

6.2.2 OBL optimization

Let $p = (x_1, x_2, \cdots, x_d)$ be a point in the d dimensional space i.e. a candidate solution. Assume $f(.)$ is the fitness function which is used to measure the candidates fitness. According to the definition of the opposite point, $\tilde{p} = (\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_d)$ is the opposite of $p = (x_1, x_2, \cdots, x_d)$. Now if $f(\tilde{p}) \geq f(p)$ then point p can be replaced by $\tilde{p}$ otherwise we will continue with p. Hence the point and its opposite point are evaluated simultaneously in order to continue with the more fit ones.

6.3 Proposed ODE-NN Algorithm

Similar to all population-based optimization algorithms, two main steps are distinguishable for DE, namely, population initialization and producing new generations by evolutionary operations such as mutation, crossover, and selec-

tion. The objective of using the OBL scheme in SI is to enhance the above two steps in EAs. The DE is chosen as a parent algorithm and the proposed opposition-based ideas are embedded into it to accelerate its convergence speed. The pseudo code for the proposed approach ODE-NN is presented in algorithm 1.

6.3.1 Opposition-Based Population Initialization

After having a review on evolutionary optimization literature discussed in chapter 2, random number generation, in absence of a priori knowledge, is the common choice to create an initial population. Therefore, by utilizing OBL, we can obtain fitter starting candidate solutions even when there is no a priori knowledge available about the solution(s). The following steps are presented to describe opposition-based initialization for the ODE. Initialize the population (P) represented as *pop* randomly, Calculate opposite population using the formula given below.

$$opop_{i,j} = a_j + b_j - pop_{i,j}$$

$$i = 1, 2, \cdots, P \quad j = 1, 2, \cdots, d$$

where $pop_{i,j}$ and $opop_{i,j}$ denote the j^{th} variable of the i^{th} vector of the population and opposite population respectively. Select P fittest individual from the total of *pop* and *opop* i.e. ($popUopop$) as initial population.

6.3.2 Opposition-Based Generation Jumping

By applying a similar approach to the current population, the evolutionary process can be forced to jump to a new solution candidate, which is ideally fitter than the current one. Based on a jumping rate (i.e., jumping probability), after generating new populations by mutation, crossover, and selection, the opposite population is calculated and the fittest individuals are selected from the union of the current population and the opposite population. Unlike opposition-based initialization, generation jumping calculates the opposite population dynamically. In each generation, the search space is reduced so that

we have to calculate the opposite points by using variables in the current interval in the population using the following expression.

$$onpop_{i,j} = \min(npop_j) + \max(npop_j) - npop_{i,j}$$

Instead of calculating the opposite points dynamically if it would be calculated using the initial static boundaries then we would jump outside of the already shrunken search space and the knowledge of the current reduced space (converged population) would be lost. Hence, we calculate opposite points by using variables current interval in the population which depends on the maximum and minimum value of the current population. As the search does progress, the search space shrinks and the variables will remain within the search space.

Algorithm 2 ODE Algorithm

Require: *pop*: initial population, F: Mutation constant, C: Cross over constant, J_r: Random number

{Opposition based Initialization }

for $i = 0$ to P **do**

  for $j = 0$ to d **do**

    $opop_{i,j} = a_j + b_j - pop_{i,j}$

  end for

end for

Select P fittest individual from $(pop \cup opop)$

while Convergence criteria not met **do**

  for $i = 0$ to P **do**

    $r_1 = rand(P)$

    $r_2 = rand(P)$

    $r_3 = rand(P)$

    $ov_{i,g} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g})$

    $t_{j,i,g} = \begin{cases} ov_{j,i,g} \text{ if } (rand_j \leq C) \text{ or } j = j_{rand} \\ x_{j,i,g} \text{ otherwise} \end{cases}$

    if $f(t_{i,g}) \leq f(x_{i,g})$ **then**

      $x_{i,g+1} = t_{i,g}$

    else

      $x_{i,g+1} = x_{i,g}$

    end if

  end for

  Store in new population *npop* {Opposition based Generation jumping }

  if $rand < J_r$ **then**

    for $i = 0$ to P **do**

      for $j = 0$ to d **do**

        $onpop_{i,j} = max(npop_j) + min(npop_j) - npop_{i,j}$

      end for

    end for

  end if

  Select P fittest individual from the set $(onpop_{i,j} \cup npop_{i,j})$

end while

6.4 A Combined ODE-NN Approach to System Identification

In this section, a brief description is given how an ODE is applied for training the neural networks in the frame work of system identification. According to step 10 given below in the proposed algorithm, the value of the cost function after reaching a particular value of ϵ, or the number of generation the algorithm

is switched from global search such of the evolutionary algorithm (ODE) to local search, LM. In opposition based differential evolution, at the moment of starting, the differential term is very high. As the solution approaches to global minimum, the differential term automatically reduces to a low value. So at the initial period, the convergence speed is faster and search space is very large but in latter stages nearer to the optimum, due to small differential term, the algorithm becomes slower which will take more time to converge. As LM is a gradient based algorithm at that point the role of LM is to increase the convergence speed for reaching the global minimum. Thus, ODE can be applied to obtain global searches within the weight space of a typical feed-forward neural network.

Steps of ODE-NN Algorithm

Step 1.

Initialize population **pop**: Create a population from randomly chosen object vectors.

Step 2.

Find out the opposite population **opop**: Create an opposite population from the population *pop*.

Step 3.

Create a fittest population **npop** from both **pop U opop** with dimension P.

$$\mathbf{P}_g = (\mathbf{w}_{1,g}, \cdots, \mathbf{w}_{P,g})^T, \;\; g = 1, \cdots, g_{\max}$$

$$\mathbf{w}_{i,g} = (w_{1,i,g}, \cdots, w_{d,i,g}), \quad i = 1, \cdots, P$$

where d is the number of weights in the weight vector $\mathbf{w}_{i,g}$, i is index to the population and is the generation to which the population belongs.

Step 4.

Evaluate all the candidate solution inside **npop** for a specified number of generations.

Step 5.

For each candidate in **npop** select the random variables $r_1, r_2, r_3 \in \{1, 2, \cdots, P\}$.

Step 6.

Apply mutation operator to each candidate in population to yield a mutant vector i.e.

$$v_{j,i,g} = w_{j,r_1,g} + F(w_{j,r_2,g} - w_{j,r_3,g}),\ for\ j = 1, \cdots, d$$
$$(i \neq r_1 \neq r_2 \neq r_3) \in \{1, \cdots, P\}\ and\ F \in (0, 1+]$$

Step 7.

Apply crossover i.e. each vector in the current population is recombined with a mutant vector to produce trial vector.

$$t_{j,i,g} = \begin{cases} v_{j,i,g} & if\ rand_j[0,1) \leq C \\ w_{j,i,j} & otherwise \end{cases}$$
$$where\ C \in [0, 1]$$

Step 8.

Apply selection i.e. between the trial vector and target vector. If the target vector has an equal or lower objective function value than that of its target vector, it replaces target vector; otherwise, the target retains its place in the population.

$$\mathbf{w}_{i,g} = \begin{cases} t_{i,g}\ \ if\ \mathbf{E}(\mathbf{y}, f(\mathbf{x}, t_{i,g})) \leq \mathbf{E}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}_{i,g})) \\ \mathbf{w}_{i,g} \quad otherwise \end{cases}$$

Step 9.

If $rand_j < J_r$ Find the opposite population of $\mathbf{w}_{i,g}$ i.e. $\mathbf{ow}_{i,g}$ Select P fittest individuals from $\mathbf{w}_{i,g} \cup \mathbf{ow}_{i,g}$ which gives the populations for the next generation which is represented by $\mathbf{w}_{i,g+1}$. Else $\mathbf{w}_{i,g+1} = \mathbf{w}_{i,g}$

Step 10.

Evaluate for the weights obtained from step-9 *If* $\mathbf{E} \leq \varepsilon$ where $\varepsilon > 0$ go to step-8 *Else* go to step-5.

Step 11.

Initialize the weight matrix of Levenberg-Marquardt algorithm taking the values of weights obtained after the fixed number of iterations. Find out the value

of **E**.

Step 12.

Compute the Jacobian matrix $\mathbf{J}(w)$.

Step 13.

Find Δw using the following equation

$$\Delta w = \left[\mathbf{J}^T(w)\, J(w) + \mu I\right]^{-1} \mathbf{J}^T(w)\, \mathbf{E}$$

Step 14.

Recompute using if this new is smaller than that computed in step 7 then reduce and go to step1.where is the damping factor.

Step 15.

The algorithm is assumed to have converged when the norm of the gradient i.e. is less than some predetermined value, or when the sum of squares of errors has been reduced to some error goal.

6.5 Results and Discussions

We present here the system identification results of different approaches such as DE-NN and ODE-NN applied to the systems given in equation (6.1) and Box and G.M. Jenkins, Time Series Analysis, and an real time TRMS system. The results obtained on this section are taken as the average of thirty independent runs with different initialization.

Example: 1

The nonlinear system to be identified is given in [61] expressed by

$$y_p(k+1) = \frac{y_p(k)[y_p(k-1)+2][y_p(k)+2.5]}{8.5+[y_p(k)]^2+[y_p(k-1)]^2} + u(k)$$

The identification model be in the form of

$$y_{pi}(k+1) = f(y_p(k), y_p(k-1)) + u(k) \tag{6.1}$$

where $f(y_p(k), y_p(k-1))$ is the nonlinear function of and the inputs to the neural network are $y_p(k)$ and $y_p(k-1)$. The output from neural network is $y_{pi}(k+1)$. The neural network identifier structure consisted of eleven numbers of neurons in the hidden layer. After 500 epochs the training of the neural identifier has been stopped. After the training is over, its prediction capability has been tested for input given below.

$$u(k) = \begin{cases} 2\cos(2\pi k/100) \text{ if} & k \leq 200 \\ 1.2\sin(2\pi k/20) \text{ if} & 200 < k \leq 500 \end{cases}$$

Table-6.1 gives the parameters considered for DE and ODE identification scheme. Figure 6.2 and 6.3 gives the identification performance between actual and identified model for DE-NN and ODE-NN respectively. Figure 6.4 and 6.5 gives the identification error and figure 6.6 gives the comparison of MSE for both the system identification scheme. From the figures it is clear that both the results are nearly same. The value of MSE is given in Table-6.2. From this it is clear that the prediction error is slightly less in case of ODE-NN approach in comparison to the DE-NN system identification technique and the performances are found to be comparable.

Table 6.1: Parameters for DE and ODE

Population size, P	50
Upper and lower bound of weights	[0 1]
Mutation constant factor , F	0.6
Cross over constant, C	0.5
Random number J_r	0.3

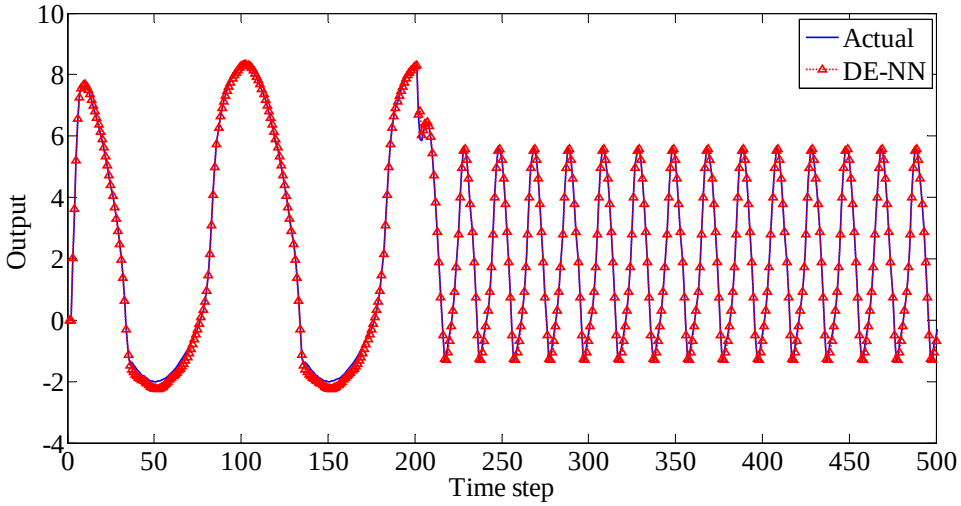

Figure 6.2: DE-NN Identification performance(Ex-1)

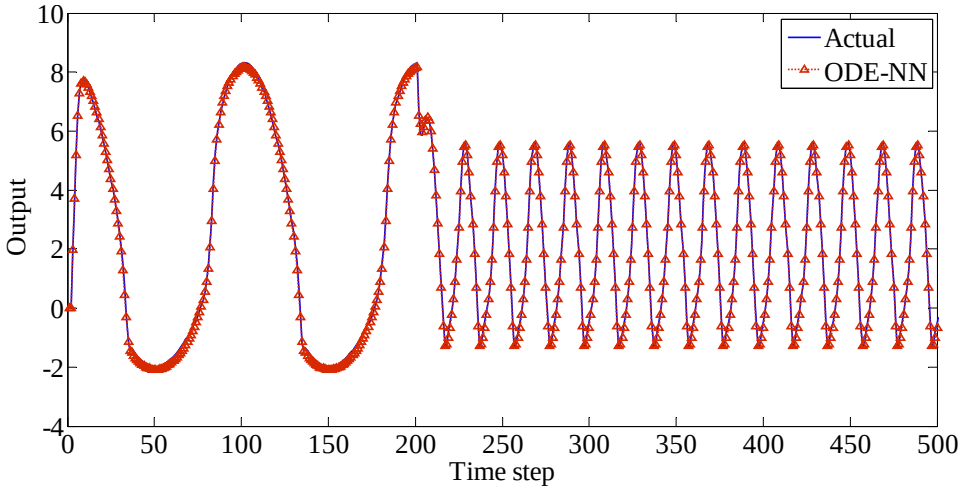

Figure 6.3: ODE-NN Identification performance(Ex-1)

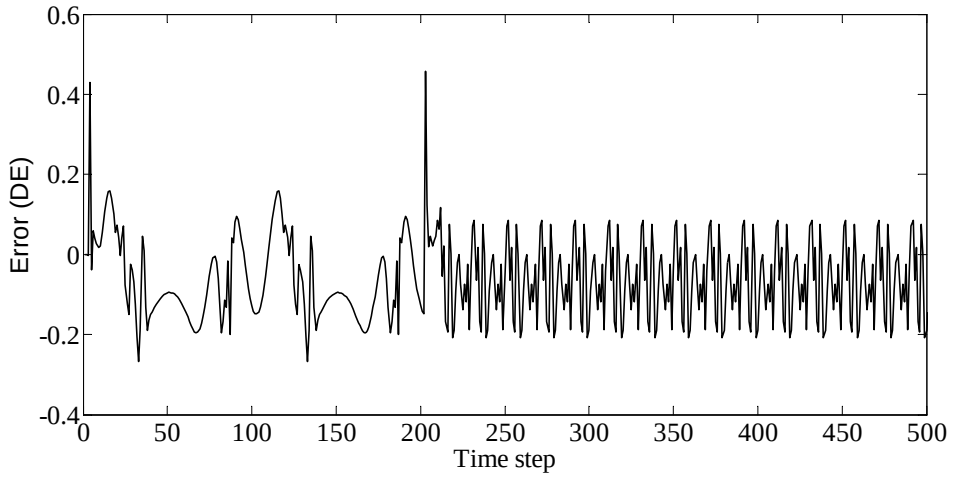

Figure 6.4: DE-NN Identification error(Ex-1)

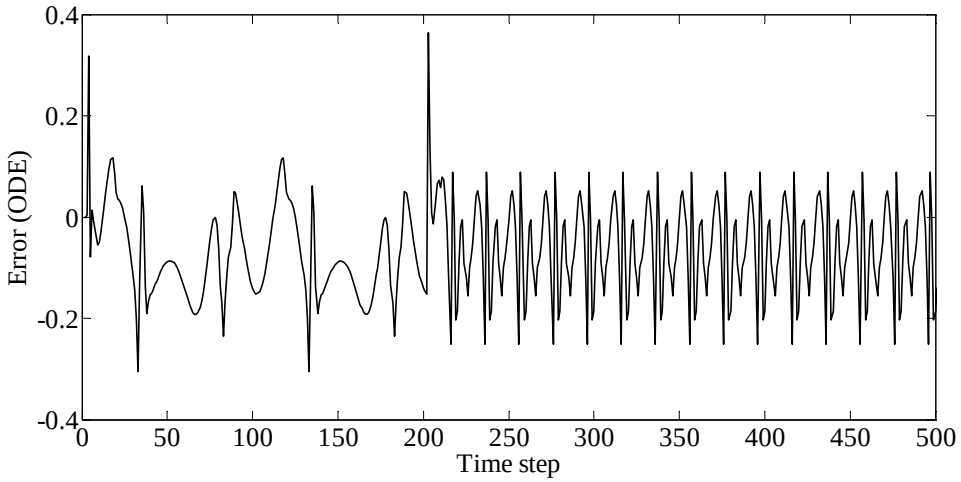

Figure 6.5: ODE-NN Identification error(Ex-1)

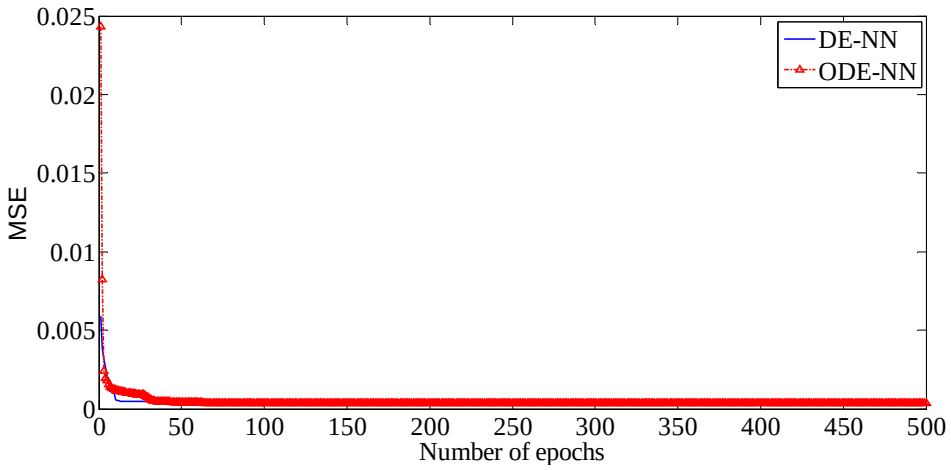

Figure 6.6: MSE(Ex-1)

Example: 2 (Box Jenkins Gas Furnace Problem)

Box and Jenkins gas furnace data are frequently used in performance evaluation of system identification methods. This is a time series data set for a gas furnace. The data consists of 296 input-output samples recorded with a sampling period of 9 s. The gas combustion process has one variable, gas flow $u(t)$, and one output variable, the concentration of carbon dioxide (CO_2), $y(t)$. The instantaneous values of output $y(t)$ are being influenced by ten past input and output variables such as $y(t-1), y(t-2), y(t-3), y(t-4), y(t-5), u(t-1), u(t-2), u(t-3), u(t-4), u(t-5)$. The original data set contains 296 $[u(t), y(t)]$ data pairs. The number of training data was taken as 100 and the rest data were considered as the test data. For simplicity two inputs were considered as follows one is from the furnace output and other is from the furnace input so we have build 24 models of different input and output. Table 6.1 gives different parameter values and their ranges. These data have been used for both DE and ODE. Both for DE-NN and ODE-NN approaches eleven number of hidden layer neurons were taken and the results obtained after 100 epochs.

From Table 6.2, we can conclude that model with $y(t-1)$ and $u(t-3)$ as input has the smallest training and testing error for both the DE-NN and

ODE-NN identification schemes. It is also clear that in case of ODE training and testing errors are lower as compared to its DE counterpart. The MSE for testing data turned out to be the least for twenty cases in ODE-NN approach whereas it was found to be better only for four cases for DE-NN approach. Similarly it was found that the training MSE is least for sixteen cases for ODE-NN and for the rest eight cases, DE-NN was found to be better in terms of less training error. In some cases even if the training error is less for DE-NN but the testing error is better for ODE-NN. As it is not possible to show the identification performance and error curve for all the 24 cases given in Table 6.2. We have taken three cases to analyze the MSE and their performances.

Figure 6.7 gives the actual and the identified outputs for a input combination of $(y(t-1), u(t-3))$. The closer version of this result within the time step 111 to 116 is shown in Fig. 6.8. From Fig. 6.9, it is clear that MSE of the proposed ODE-NN approach is converging faster than DE-NN approach. Further the MSE of ODE-NN starting from a lower vale i.e. around 2.2 where as for DE-NN it is starting from 2.9. The result shows ODE-NN is having better identification performance than DE-NN approach.

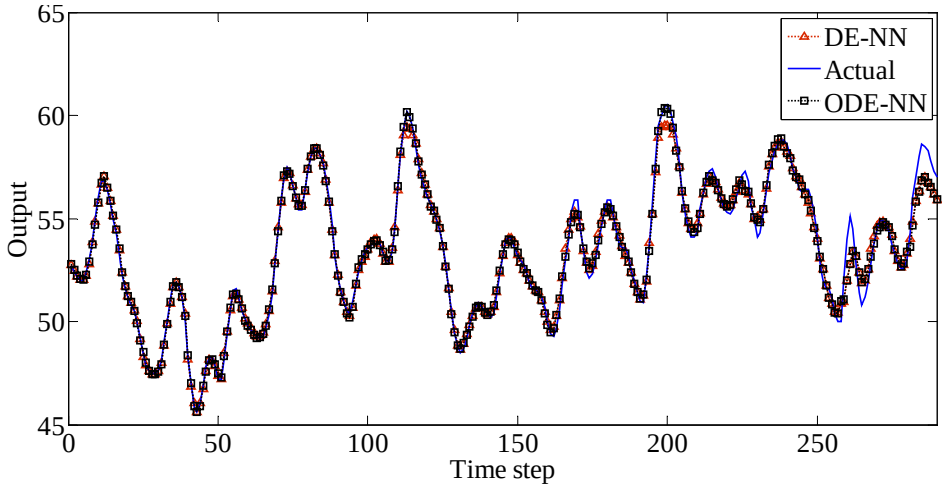

Figure 6.7: Identification performance$(y(t-1), u(t-3))$ (Ex-2)

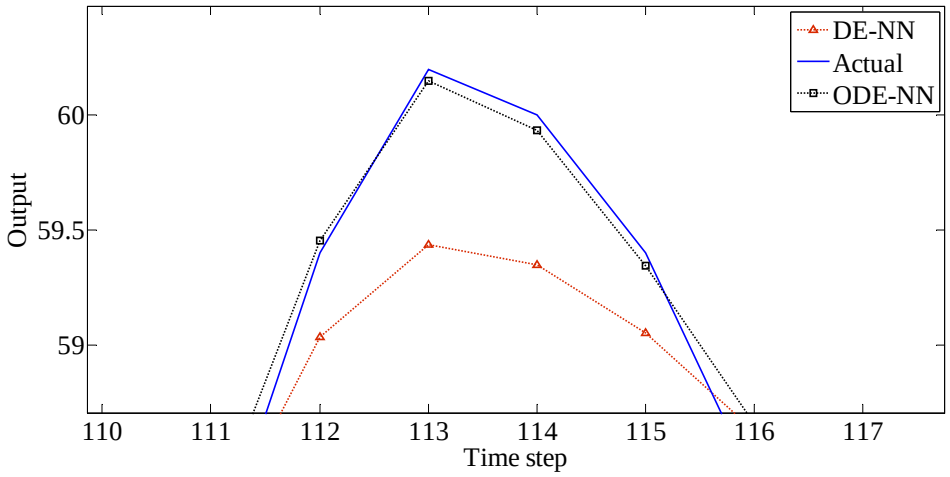

Figure 6.8: Zoomed identification performance $(y(t-1), u(t-3))$ (Ex-2)

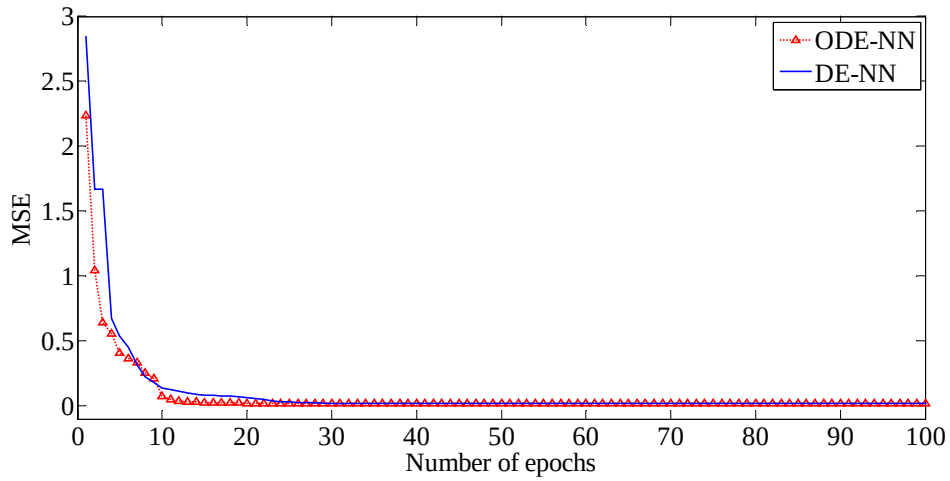

Figure 6.9: MSE $(y(t-1), u(t-3))$ (Ex-2)

Figure 6.10 gives the MSE for the input $y(t-4)$ and $u(t-5)$. We have considered 20 epochs because there was no change in MSE after 20 epochs. For the input combination of $y(t-4)$ and $u(t-5)$, the ODE-NN MSE starts from a lower value and also converges to a lower value as compared to the to DE-NN approach. Figure 6.11 gives the identification performance for the input combination of $y(t-4)$ and $u(t-5)$. The zoomed version of the identification performance within time step 54 to 58 is shown in Fig. 6.12.

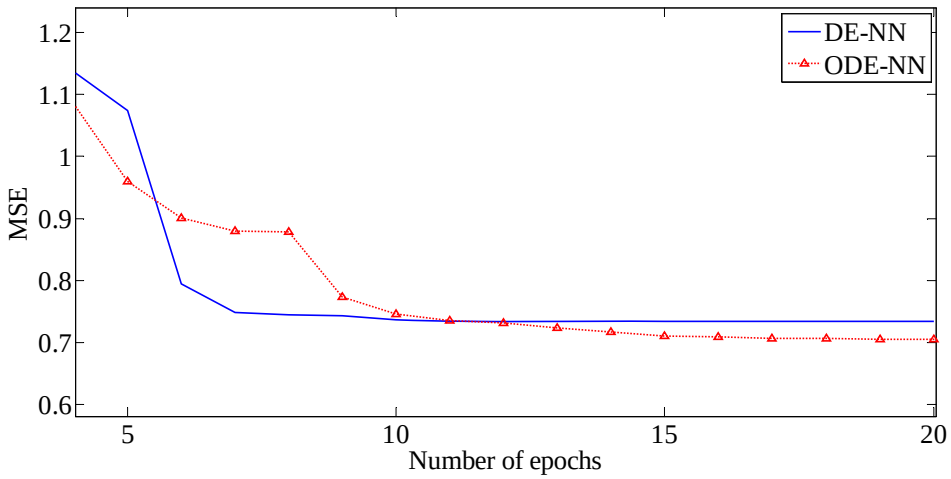

Figure 6.10: MSE $(y(t-4), u(t-5))$ (Ex-2)

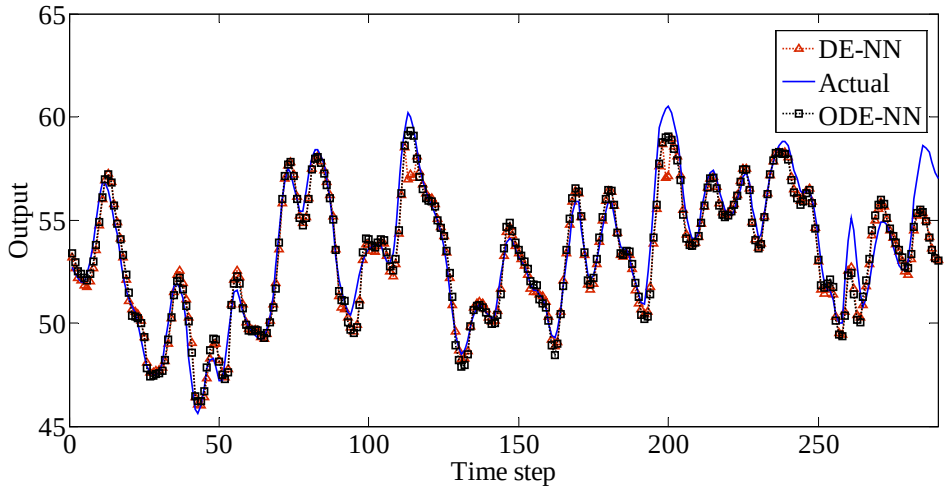

Figure 6.11: Identification performance $(y(t-4), u(t-5))$ (Ex-2)

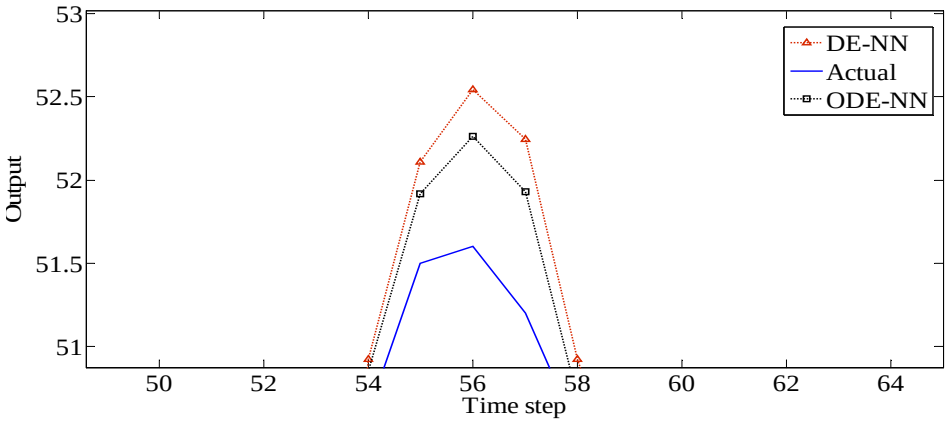

Figure 6.12: Zoomed identification performance $(y(t-4), u(t-5))$ (Ex-2)

Figure 6.13 shows the MSE for the input $y(t-4)$ and $u(t-4)$ from which it is clear that the MSE for ODE-NN exhibits faster convergence speed but eventually gives a slight loer value of MSE compared to DE-NN approach. The numerical values of training and testing MSE are mentioned in Table 6.2. Figure 6.14 shows the identification performance for the input $y(t-4)$ and $u(t-4)$ from which it is found that even if the training error for ODE-NN approach is slightly higher than that of the DE-NN approach but the former provides better identification in comparison to DE-NN approach.

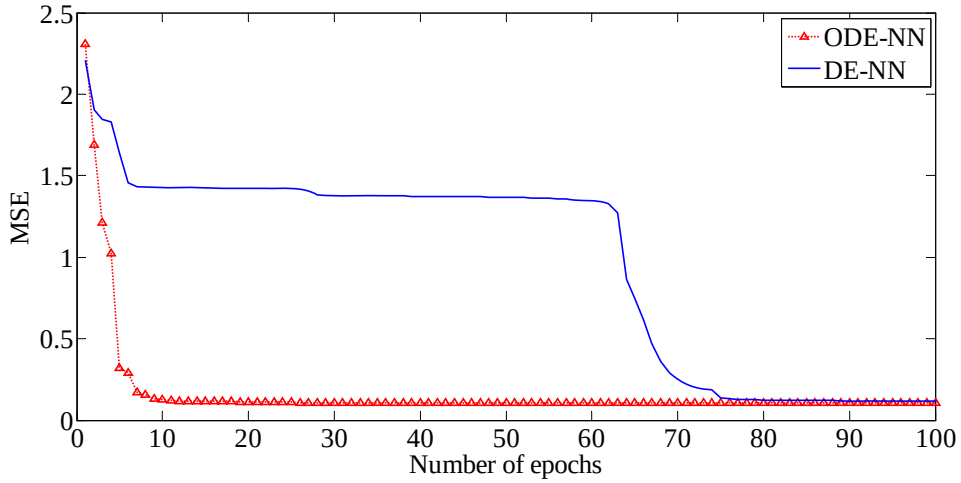

Figure 6.13: MSE $(y(t-4), u(t-4))$ (Ex-2)

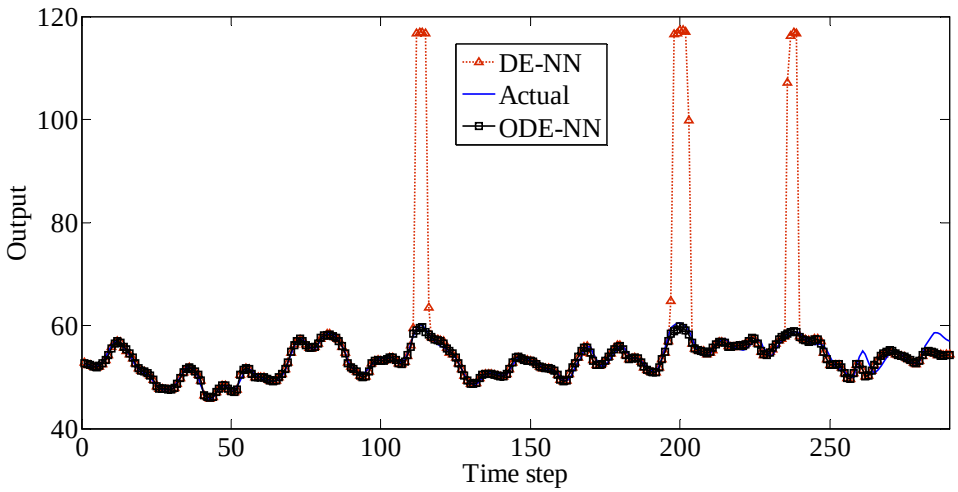

Figure 6.14: Identification Performance $(y(t-4), u(t-4))$ (Ex-2)

Table 6.2: Comparison of training and testing errors

	Training Error (MSE)		Testing Error (MSE)	
Example-1				
Input	DE	ODE	DE	ODE
y(k),y(k-1), u(k)	0.0207	**0.0190**	0.1186	**0.1137**
Example-2				
Input	DE	ODE	DE	ODE
y(t-1),u(t-3)	0.4400	**0.4194**	0.1501	**0.1411**
y(t-3),u(t-4)	0.7838	**0.7773**	0.3402	**0.2805**
y(t-2),u(t-4)	0.6733	**0.6602**	0.3256	**0.2898**
y(t-1),u(t-2)	**0.4906**	0.6801	**0.2909**	0.2924
y(t-1),u(t-4)	0.5430	**0.5132**	0.2991	**0.2926**
y(t-4),u(t-4)	12.259	**0.8894**	**0.3274**	0.3428
y(t-2),u(t-3)	1.1340	**0.7199**	**0.2968**	0.3051
y(t-1),u(t-1)	0.6183	**0.6056**	0.4638	**0.4151**
y(t-4),u(t-3)	**1.2405**	1.2771	0.7266	**0.4301**
y(t-1),u(t-6)	0.8469	**0.8410**	0.6012	**0.5661**
y(t-3),u(t-3)	**1.0067**	1.0347	**0.5172**	0.5176
y(t-2),u(t-2)	0.9889	**0.9753**	0.6314	**0.6261**
y(t-1),u(t-5)	0.6873	**0.6518**	**0.6220**	0.6303
y(t-4),u(t-5)	1.0149	**0.9698**	0.7038	**0.6373**
y(t-2),u(t-1)	1.8368	**1.2726**	0.8934	**0.6844**
y(t-2),u(t-5)	**0.9176**	1.1808	0.7222	**0.6804**
y(t-3),u(t-5)	**0.9536**	1.0470	**0.7138**	0.7338
y(t-3),u(t-2)	1.8184	**1.4138**	0.8766	**0.8600**
y(t-4),u(t-6)	1.7628	**1.4677**	1.3988	**1.1126**
y(t-2),u(t-6)	1.3352	**1.2639**	1.6264	**1.1945**
y(t-4),u(t-2)	1.6725	**1.6377**	**1.1799**	1.1963
y(t-3),u(t-6)	27.468	**1.4641**	**1.2063**	1.2424
y(t-3),u(t-1)	1.7123	**1.6475**	1.5725	**1.2702**
y(t-4),u(t-1)	2.0821	**2.0217**	**1.4250**	1.4352

Table-6.2 gives the comparison of training and testing MSEs of two approaches namely DE-NN and ODE-NN.

For the first example it is found that the training and testing errors are less in the case of proposed ODE-NN approach. From the table the results marked in bold indicates less training and testing error for the corresponding input combinations.

In example-2 we have considered all the possible input combinations i.e. twenty

four. It is found that the testing error is less for 19 combinations of ODE-NN approach compared to DE-NN approach. Thus, it is clear from above discussion that ODE-NN identifier is a better identifier compared to DE-NN one.

Example: 3 (Twin Rotor MIMO System)

Next we studied a TRMS considering only 1 DOF around the pitch axis and identified the system using proposed ODE-NN method. The model has three inputs and eleven neurons in the hidden layer. The inputs are the main rotor voltage at the present time $V(t)$, main rotor voltage at previous time $V(t-1)$ and the pitch angle of the beam at previous time instant $\psi(t-1)$. Figure 6.15 shows the identification performance of 1 DOF vertical ODE-NN based model. A more convincing method of the identification model validation is to use correlation tests. If the model of a system is adequate then the residuals should be unpredictable from (uncorrelated with) all linear and nonlinear combinations of past inputs and outputs. A number of auto-correlation and cross-correlation tests between the input and residual given below has been recommended by the authors in [97].

$$\xi_{\varepsilon\varepsilon} = E[\varepsilon(t-\tau)\varepsilon(t)] = \delta(\tau)$$

$$\xi_{u\varepsilon} = E[u(t-\tau)\varepsilon(t)] = 0 \qquad \forall\tau$$

$$\xi_{u^2\varepsilon^2} = E[u^2(t-\tau) - \bar{u}^2\varepsilon^2(t)] = 0 \qquad \forall\tau$$

$$\xi_{\varepsilon(\varepsilon u)} = E[\varepsilon(t)\varepsilon(t-1-\tau)u(t-1-\tau)] = 0 \qquad \tau \geq 0$$

where $\xi_{u\varepsilon}$ indicates the cross-correlation between $u(t)$ and $\varepsilon(t)$ and $\delta(t)$ is an impulse function. The test results are given below. In general, if the correlation functions are within the 95 percent confidence intervals, $1.96/N$, where, N is the total number of data points, the model is regarded as accurate. The correlation analysis of the above model is given in Fig. 6.16-6.19. If the resid-

uals (model errors) contain no information about the past residuals or about the dynamics of the system, it is likely that all information has been extracted from the training set and the model approximates the system well. It is found that all four correlation functions; Cross-correlation of input and residuals (Fig. 6.16), Auto-correlation of residuals (Fig. 6.17), Cross-correlation of input square and residuals square (Fig. 6.18), Cross-correlation of residuals and input×residuals (Fig. 6.19) are within 95 percent of the confidence band indicating that the model is adequate, i.e. the model behavior is closed to the real system performance.

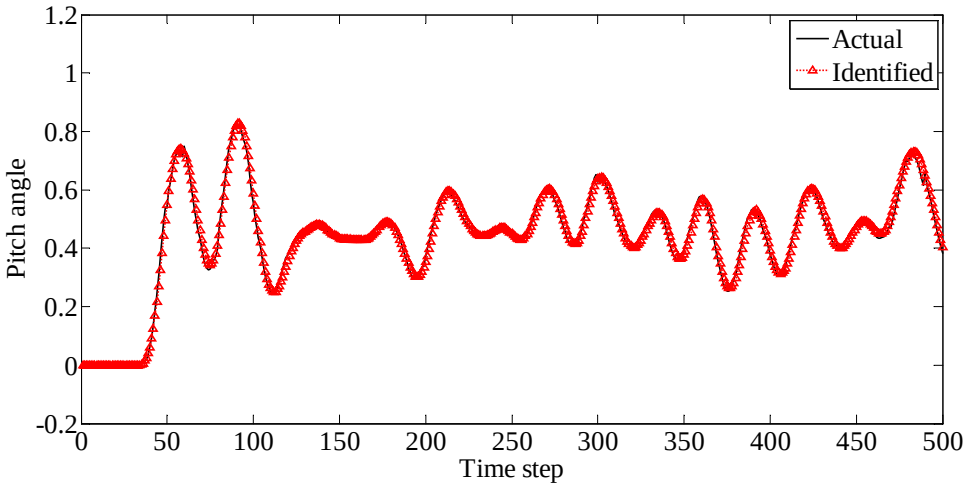

Figure 6.15: Identification Performance(TRMS)

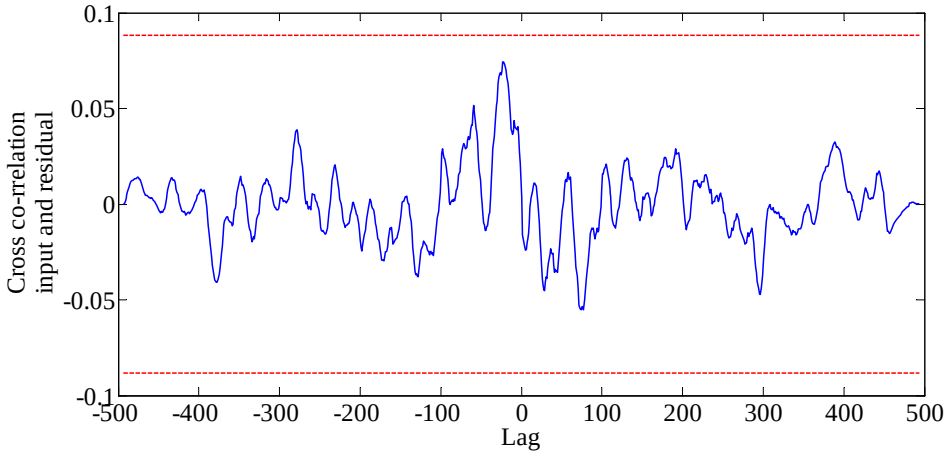

Figure 6.16: Cross-correlation of input and residuals

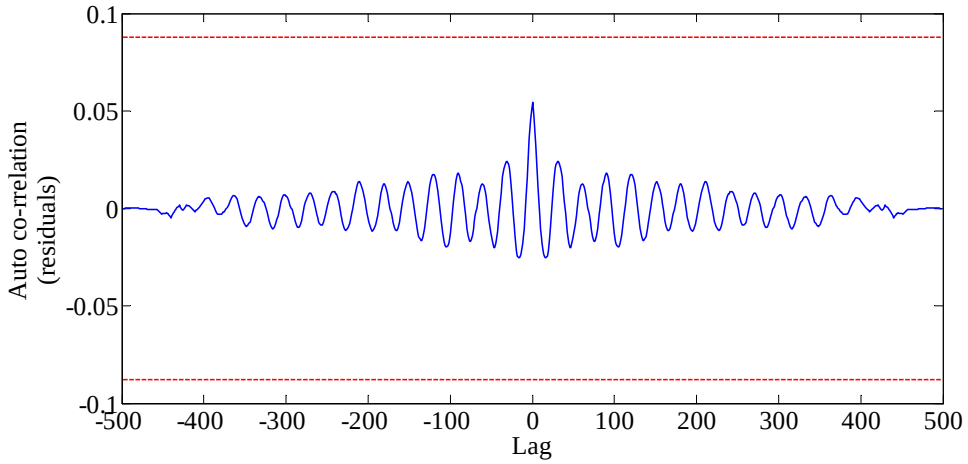

Figure 6.17: Auto-correlation of residuals

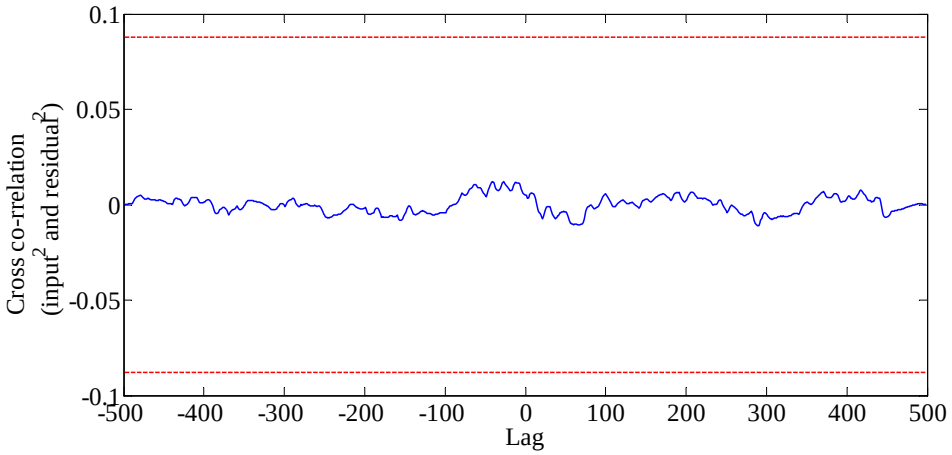

Figure 6.18: Cross-correlation of input square and residuals square

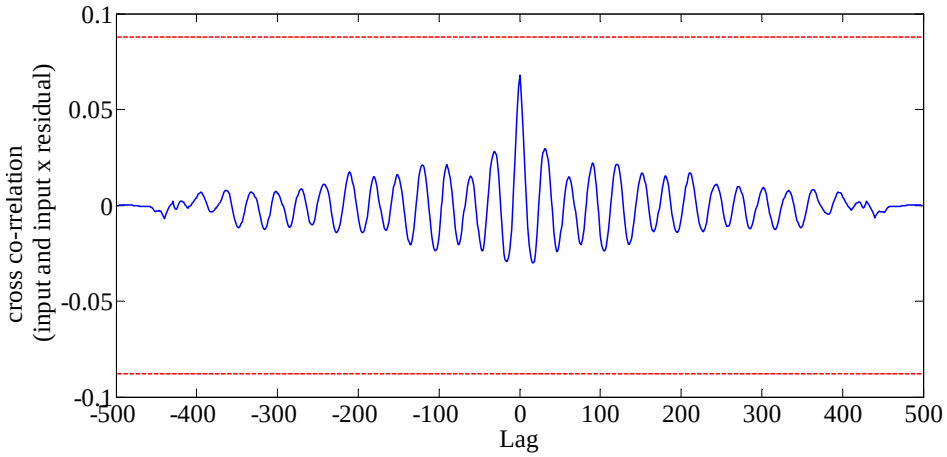

Figure 6.19: Cross-correlation of residuals and input and residuals

6.6 Summary

The chapter has described the scope of improving system identification of nonlinear systems by using proposed ODE-NN approach. The proposed ODE-NN approach is tested on a real time TRMS identification for testing its effectiveness. Results proposed demonstrate how the opposition based optimization can be employed to accelerate the convergence speed of DE by embedding opposition based population initialization and opposition based generation jumping. From the results presented in section 6.5, it is clear that there is certainly an improvement in identification performance for nonlinear systems over the existing DE-NN approach. These results envisage that ODE provides a higher performance than the classical DE approach. Further, the proposed ODE-NN approach provides better system identification performance in terms of speed of convergence and accuracy in compared to the DE-NN approach. The above proposed ODE-NN approach is tested on a real time TRMS system. It is shown that the models obtained ODE-NN methods can generally be considered adequate in representing the system.

Chapter 7

Parameter Estimation of Induction Motor Using DE and OMDE Algorithm

7.1 Introduction

This chapter describes an evolutionary methodology of identifying the parameters of an induction motor in electric drive applications. Here the stator currents (i_s) and voltages (v_s) and rotor angular speed (ω_r) are taken as input-output data that are used to estimate the parameters of the induction motor. This chapter investigates the use of differential evolution technique to estimate the rotor resistance (R_r), stator resistance (R_s), leakage inductance (L_l) and magnetising inductance (L_m) i of a three-phase induction machine. In order to obtain results with maximum accuracy, some variations of DE known as OMDE estimates are investigated.

The parameters of the induction motor model vary as operating conditions change. Accurate knowledge of these parameters and their dependency on operating conditions is critical for field oriented control. Several approaches are available for the estimation of the parameter vector. A rich variety of estimation procedures were reported in literature for induction motor parameter estimation are found in [127, 128, 129]. The simultaneous estimation of induction machine parameters and states are presented in [130, 131, 132]. The

use of linear techniques based on the dynamic model of the induction motor is proposed in [139]. The use of NNs and fuzzy methods for induction motor parameter estimation were proposed respectively in [140] and [141]. The extended Kalman filter has been employed to accomplish the joint estimation of the state variables and the machine parameters [142, 143]. The on-line tuning of the stator resistance, stator inductance, transient inductance, and rotor resistance has been discussed in [144, 145]. In [146] adaptive identification of rotor resistance is proposed for an indirect stator flux oriented induction motor drive. All these investigations demonstrate that the performance of the drive can be improved through accurate estimation of the machine parameters. For analytical identification, a model is developed from the steady-state equations of induction motor dynamics. The identification procedure, based on a simple algorithm derived from least squares techniques, uses only the information of stator currents and voltages and rotor angular speed as input-output data. The machine equations can be expressed in the form, $y\,(k) = \theta^T(k).x(k)$ where k is the sample at which a measurement is taken and θ is the vector of unknown parameters. Many investigations have been presented on nonlinear models that incorporate nonlinear effects such as magnetic saturation effects [147] and the induction machine parameters were obtained with various traditional optimization methods such as least squares and local search. However, the fundamental problem with traditional techniques is their dependence on unrealistic assumptions such as unimodal landscapes, differentiability and continuity of the objective function. Consequently nonlinear problems are often over simplified to fulfill such assumptions. In contrast EAs seem to be promising alternative to traditional approaches as they are capable of addressing problems with nonlinear effects, multimodality, non-differentiability and time varying components. Authors in [148, 149, 150, 151] have investigated parameter identification of induction motor using genetic algorithm. Parameter identification of induction motor using evolutionary algorithm and stochastic optimization algorithm has been discussed in [154] and [153]. Differential evo-

lution is one of the variant of EAs which has been prefered in many real world problems due to its simplicity and less number of operational parameters. It has the advantage of incorporating a relatively simple and efficient form of self-adaptive mutation strategy. With its esae of implementation and proven efficiency, DE is idealy suited to estimate the parameter of an induction motor. Authors in [155] have used DE to identify the parameters of an induction motor. This chapter focuses on parameter identification of induction motor using a different version of DE i.e. OMDE. Here the parameters are identified in simulation models based on nonlinear differential equations. For analytical identification, a model is developed from the steady-state equations of induction motor dynamics. The identification procedure, based on DE and OMDE algorithm, uses only the information of stator currents and voltages and rotor angular speed as input-output data. The computer simulation using MATLAB is used to prove the efficacy of the proposed method.

7.2 Review of some parameter estimation algorithms applied to induction motor

The least mean squares algorithm

The least mean squares algorithm is a gradient-descent method

$$e\left(k\right) = y\left(k\right) - \theta^{T}\left(k\right) x\left(k\right) \tag{7.1}$$

$$\theta\left(k+1\right) = \theta\left(k\right) + \mu x\left(k\right) e\left(k\right) \tag{7.2}$$

where $y(k)$ is the plant output vector (in this case the voltage vector), $\theta^{T}(k)$ is the unknown parameter vector, $x(k)$ is the input vector, $e(k)$ is the error between the plant and the estimator, and μ is a diagonal gain matrix, where because of the nonlinear plant inputs each of the diagonal elements is a different constant scalar value. The selection of the values of each diagonal element μ is input dependant and can be given as $0 < \mu < \lambda_{\max}$ where $\lambda_{\max}$ is the maximum Eigen value of the autocorrelation matrix of the input vector $x(k)$. The problem with this is that for the nonlinear case the maximum Eigen

value spread (the ratio of minimum to maximum Eigen values) is in general very large, giving slow convergence rates. The advantage of this algorithm is computational simplicity.

Recursive least squares algorithm

The RLSs algorithm provides an adaptive solution to the least-squares regression problem

$$\hat{y}(k) = \hat{\theta}^T(k-1)\, x(k) \tag{7.3}$$

$$e(k) = y(k) - \hat{y}(k) \tag{7.4}$$

$$Cov(k) = Cov(k-1) - \frac{Cov(k-1)\, x(k)\, x^T(k)\, Cov(k-1)}{1 + x^T(k)\, Cov(k-1)\, x(k)} \tag{7.5}$$

$$\hat{\theta}(k-1) = \hat{\theta}(k-1) + Cov(k)\, x(k)\, e(k) \tag{7.6}$$

where $\hat{y(k)}$ is the plant output vector (in this case the voltage vector), $\theta^T(k)$ is the unknown parameter vector, $x(k)$ is the input vector, $e(k)$ is the error between the plant and the estimator. Cov is the covariance matrix. The RLS algorithm provides superior convergence properties to the LMS algorithm, however this is at the cost of additional computational expense.

Extended Kalman Filter Algorithm

To apply the Kalman filter to estimation of the induction motor parameters, measurement of some state variables is necessary [142]. In this case, the measurable state variables are the stator currents, i_{ds} and i_{qs}. This measurement can be expressed

$$Z = Hy + v \tag{7.7}$$

v is the assumed to be a zero-mean white Gaussian noise with covariance R. Kalman filter algorithm as follows:

- Set the initial value $\hat{y(0)}$ of the estimated state vector $\hat{y}$ and the initial value $Cov(0)$ of covariance of estimation error Cov.

- Calculate the Kalman gain K_k at time stage k :

$$K_k = Cov_k H^T R^{-1} \tag{7.8}$$

- Measure the values of z_k and u_k
- Calculate the linearization matrix F_k

$$F_k = \left[\frac{\partial f(y,u)}{\partial y}\right]_{y=y_k, u=u_k} \tag{7.9}$$

- Calculate the estimate value

$$\hat{y}(k) = f\left(y_k, u_k\right) + K_k\left(Z_k - Hy_k\right) \tag{7.10}$$

$$\hat{Cov}_k = F_k \hat{Cov}_k + \hat{Cov}_k F_k^T - K_k H \hat{Cov}_k + Q \tag{7.11}$$

An extended Kalman filter approach to rotor time constant measurement in PWM induction motor drive is discussed in [142]. However, difficulties are encountered in expressing the equations in regressor form and in turn this method may not be a viable choice for all situations in induction motor parameter estimation problems. Therefore, an alternative way of solving the parameter estimation problem, by using evolutionary method which does not require the description of equation $y\left(n\right) = \theta^T(n)x(n)$. In spite of the considerable theoretical foundation of induction motors, few studies have used EAs and other stochastic search techniques to identify the model parameters. EA investigations on parameter identification of induction motors can be categorized in two groups i.e. identification of parameters in simulation models based on nonlinear differential equations and identification of parameters from equations describing the load torques. The approach used in the first group determines the parameters from time-series data, which typically leads to high-accuracy models that can be used to control the motor. The approach used in the second group is less precise, but is independent of available time-series data. In [145], the authors investigated a 1kW motor and showed that the evolutionary algorithm outperformed least squares fitting. They determined stator resis-

tance (R_s), stator inductance (L_s), leakage inductance (L_l), motor load torque (τ_r), and moment of inertia (J_m) using a state space model with scaled rotor flux. In a similar study, authors in [146] identified stator and rotor resistance (R_s and R_r), stator and rotor self-inductance (L_s and L_r), and magnetizing inductance (L_m) using a motor model not accounting for saturation effects. In their paper, they compared the performance of a genetic algorithm to a random search algorithm under four levels of simulated measurement noise. In [148] the authors, determined the parameters in a motor load model using a GA. Authors in [149] used a simple evolutionary algorithm to determine stator resistance (R_s), rotor resistance (R_r), and combination of stator and rotor reactance X_{lr} from the motors specifications provided by the manufacturer. In a follow-up study, authors in [150] used a GA to determine the parameters of three motors from equations for the full load torque, the lock rotor torque, and the breakdown torque. Additionally, they used genetic programming (GP) to evolve equations modeling these torques the results obtained are somewhat inconclusive since the GA outperformed the GP in some cases, but not in other cases.

7.3 Induction Motor Modeling

A transformation of stator ABC variables to dq variables can be carried out using Park's transformation given below. a) 3-phse to stationary $d^s - q^s$ reference frame

$$\begin{bmatrix} v_{qs}^s \\ v_{ds}^s \\ v_{os}^s \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos\theta & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ \sin\theta & \sin\left(\theta - \frac{2\pi}{3}\right) & \sin\left(\theta + \frac{2\pi}{3}\right) \\ 0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \end{bmatrix} \tag{7.12}$$

or $v^s_{qdo} = K_s v_{abcs}$

where K_s is the transformation matrix. b) $d^s - q^s$ to 3-phase transformation

$$\begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \end{bmatrix} = \begin{bmatrix} \cos\theta & sin\theta & 1 \\ \cos\left(\theta - \frac{2\pi}{3}\right) & \sin\left(\theta - \frac{2\pi}{3}\right) & 1 \\ \cos\left(\theta + \frac{2\pi}{3}\right) & \sin\left(\theta + \frac{2\pi}{3}\right) & 1 \end{bmatrix} \begin{bmatrix} v^s_{qs} \\ v^s_{ds} \\ v^s_{os} \end{bmatrix} \tag{7.13}$$

or $v_{abcs} = K_s^{-1} v^s_{qdo}$

It is convenient to set $\theta = 0$. Ignoring zero sequence component

$$v^s_{qs} = v_{as} \tag{7.14}$$

$$v^s_{ds} = \frac{1}{\sqrt{3}} (v_{cs} - v_{bs}) \tag{7.15}$$

Stationary ($d^s - q^s$) to rotating ($d^e - q^e$) and vice versa

$d^e - q^e$ axes are rotating at speed ω_e and $d^s - q^s$. axes are stationary. Angle, θ_e is given by

$$\theta_e = \omega_e . t \tag{7.16}$$

$$v_{qs} = v^s_{qs} \cos\theta_e - v^s_{ds} \sin\theta_e \tag{7.17}$$

$$v_{ds} = v^s_{qs} \sin\theta_e + v^s_{ds} \cos\theta_e \tag{7.18}$$

$$v^s_{qs} = v_{qs} \cos\theta_e + v_{ds} \sin\theta_e \tag{7.19}$$

$$v^s_{ds} = -v_{qs} \sin\theta_e + v_{ds} \cos\theta_e \tag{7.20}$$

Assuming three phase balanced voltages

$$v_{as} = V_m \cos(\omega_e . t + \phi) \tag{7.21}$$

$$v_{bs} = V_m \cos(\omega_e.t - \frac{2\pi}{3} + \phi) \tag{7.22}$$

$$v_{cs} = V_m \cos(\omega_e.t + \frac{2\pi}{3} + \phi) \tag{7.23}$$

We get $d^s - q^s$ variables

$$v_{qs}^s = v_{as} = V_m \cos(\omega_e.t + \phi) \tag{7.24}$$

$$v_{ds}^s = \frac{1}{\sqrt{3}} (v_{cs} - v_{bs}) = -V_m \sin(\omega_e.t + \phi) \tag{7.25}$$

A complex space vector, represented by $d^s - q^s$ variables

$$\overrightarrow{V} = v_{qs}^s - j v_{ds}^s$$

$$= V_m \left[\cos(\omega_e.t + \phi) + j \sin(\omega_e.t + \phi)\right]$$

$$= V_m e^{j(\omega_e.t + \phi)} \tag{7.26}$$

The voltage vector rotates at the speed, ω_e, from the initial angle ϕ. Magnitude of the voltage vector

$$\left|\overrightarrow{V}\right| = \sqrt{{v_{qs}^s}^2 + {v_{ds}^s}^2} = V_m \tag{7.27}$$

In the synchronously rotating $d^e - q^e$ frame

$$v_{qs} = v_{qs}^s \cos\theta_e - v_{ds}^s \sin\theta_e$$

$$= V_m \cos(\omega_e.t + \phi) \cos(\omega_e.t) + V_m \sin(\omega_e.t + \phi) \sin(\omega_e.t)$$

$$= V_m \cos\phi \tag{7.28}$$

$$v_{ds} = v_{qs}^s \sin\theta_e + v_{ds}^s \cos\theta_e$$

$$= V_m \cos(\omega_e.t + \phi) \sin(\omega_e.t) - V_m \sin(\omega_e.t + \phi) \cos(\omega_e.t)$$

$$= -V_m \sin \phi \tag{7.29}$$

$v_{qs} = -V_m \cos \phi$ and $v_{ds} = -V_m \sin \phi$ are the constant quantities independent of time. Thus, three sinusoidal variables appear as dc quantities in synchronously rotating reference frame. This is the main advantage of reference frame theory dynamic model of induction machine in synchronously rotating reference frame. The stator voltage components v_{ds}, v_{qs} and rotor voltages v_{dr}, v_{qr} with synchronously rotating reference frame variables are given by

$$v_{ds} = R_s i_{ds} + \frac{d\psi_{ds}}{dt} - \omega_e \psi_{qs} \tag{7.30}$$

$$v_{qs} = R_s i_{qs} + \frac{d\psi_{qs}}{dt} + \omega_e \psi_{ds} \tag{7.31}$$

$$v_{dr} = R_r i_{dr} + \frac{d\psi_{dr}}{dt} - (\omega_e - p\omega_r)\, \psi_{ar} \tag{7.32}$$

$$v_{qr} = R_r i_{qr} + \frac{d\psi_{qr}}{dt} + (\omega_e - p\omega_r)\, \psi_{dr} \tag{7.33}$$

$i_{ds}(i_{qs})$ = d-axis (q-axis)stator current

$i_{dr}(i_{qr})$ = d-axis (q-axis)rotor current

$\psi_{ds}(\psi_{qs})$ = d-axis (q-axis)stator flux linkage

$\psi_{dr}(\psi_{qr})$ = d-axis (q-axis)stator flux linkage

ω_e = speed of the reference frame (radian/second)

ω_r = mechanical radians/seconds (speed)of the rotor

p = number of poles

The flux linkage expression can written as

$$\psi_{ds} = L_s i_{ds} + L_m i_{dr} \tag{7.34}$$

$$\psi_{qs} = L_s i_{qs} + L_m i_{qr} \tag{7.35}$$

$$\psi_{dr} = L_r i_{dr} + L_m i_{ds} \tag{7.36}$$

$$\psi_{qr} = L_r i_{qr} + L_m i_{qs} \tag{7.37}$$

Replacing the flux linkage terms in equations (7.30-7.33) by their expressions in equations (7.34-7.37), the electrical transient model in terms of voltages and currents is

$$\begin{bmatrix} v_{qs} \\ v_{ds} \\ v_{qr} \\ v_{dr} \end{bmatrix} = \begin{bmatrix} R_s + sL_s & \omega_e L_s & sL_m & \omega_e L_m \\ -\omega_e L_s & R_s + sL_s & \omega_e L_m & sL_m \\ sL_m & (\omega_e - p\omega_r) L_m & R_r + sL_r & (\omega_e - p\omega_r) L_r \\ (\omega_e - p\omega_r) L_m & sL_m & (\omega_e - p\omega_r) L_m & R_r + sL_r \end{bmatrix} \begin{bmatrix} i_{qs} \\ i_{ds} \\ i_{qr} \\ i_{dr} \end{bmatrix} \tag{7.38}$$

where s is the Laplace operator, which is replaced by $\frac{d}{dt}$. For rotor fed machine, with rotor short circuited $v_{qr} = v_{dr} = 0$

The torque developed by the machine is

$$T_e = \frac{3}{2} p L_m \left(i_{qs} i_{dr} - i_{ds} i_{qr} \right) \tag{7.39}$$

The torque balance equation is

$$T_e = T_L + J\frac{d\omega_r}{dt} + B\omega_r \tag{7.40}$$

Equations (7.38) and (7.39) represent the 5th order model of the electrical dynamics of induction motor equations where (7.38) represents the mechanical dynamics of the motor. Equation (7.38), (7.39) and (7.40) represent the complete model of induction motor, which is of 6th order. Knowing stator input voltages v_{ds} and v_{qs} these equations can be solved to find the transient response of current and speed. Equation (7.39) can be arranged in state space form with

stator currents and rotor flux linkages as state variables. From equation (7.39)

$$i_{dr} = \frac{\psi_{dr}}{L_r} - \frac{L_m}{L_r} i_{ds} \tag{7.41}$$

Substituting equation (7.41) in equation (7.30) and rearranging

$$\frac{d\psi_{dr}}{dt} = \frac{R_r L_m}{L_r} i_{ds} - \frac{R_r}{L_r}\psi_{dr} + \omega_{se}\psi_{qr} \tag{7.42}$$

From equation (7.37)

$$i_{qr} = \frac{\psi_{qr}}{L_r} - \frac{L_m}{L_r} i_{qs} \tag{7.43}$$

Substituting equation (7.43) in equation (7.31) and rearranging

$$\frac{d\psi_{qr}}{dt} = \frac{R_r L_m}{L_r} i_{qs} - \frac{R_r}{L_r}\psi_{qr} - \omega_{se}\psi_{dr} \tag{7.44}$$

Substituting equation (7.43) in equation (7.34) and rearranging

$$\psi_{ds} = L_s i_{ds} + \frac{L_m}{L_r}\psi_{dr} - \frac{L_m^2}{L_r} i_{ds}$$

$$= L_s \left(1 - \frac{L_m^2}{L_r L_s}\right) i_{ds} + \frac{L_m}{L_r}\psi_{dr}$$

$$= \sigma L_s i_{ds} + \frac{L_m}{L_r}\psi_{dr} \tag{7.45}$$

where, $\sigma = \left(1 - \frac{L_m^2}{L_r L_s}\right)$ is the leakage coefficient Substituting equation (7.43) in equation (7.45)

$$\psi_{ds} = \sigma L_s i_{qs} + \frac{L_m}{L_r}\psi_{qr} \tag{7.46}$$

Substituting equation (7.45) and (7.46) in equation (7.30)

$$v_{ds} = R_s i_{ds} + \sigma L_s \frac{di_{ds}}{dt} + \frac{L_m}{L_r}\frac{d\psi_{dr}}{dt} - \omega_e \left(\sigma L_s i_{qs} + \frac{L_m}{L_r}\psi_{qr}\right) \tag{7.47}$$

Substituting equation (7.44) in equation (7.47)

$$\sigma L_s \frac{di_{ds}}{dt} = v_{ds} - R_s i_{ds} - \frac{L_m}{L_r}\left(\frac{R_r L_m}{L_r} i_{ds} - \frac{R_r}{L_r}\psi_{dr} + \omega_{se}\psi_{qr}\right) - \omega_e \left(\sigma L_s i_{qs} + \frac{L_m}{L_r}\psi_{qr}\right)$$

$$= -\left(R_s + \frac{R_r L_m^2}{L_r^2}\right) i_{ds} + \sigma L_s \omega_e i_{qs} + \frac{R_r L_m}{L_r^2}\psi_{dr} + \frac{L_m}{L_r}\left(\omega_e - \omega_r\right)\psi_{qr} + v_{ds}$$

$$\frac{di_{ds}}{dt} = -\frac{(R_s L_r^2 + R_r L_m^2)}{\sigma L_r^2 L_s} i_{ds} + \omega_e i_{qs} + \frac{R_r L_m}{\sigma L_s L_r^2}\psi_{dr} + \frac{L_m}{\sigma L_s L_r} p\omega_r \psi_{qr} + \frac{v_{ds}}{\sigma L_s} \quad (7.48)$$

Similarly, substituting equation (7.47),(7.48)and (7.46) in equation (7.31)

$$\frac{di_{qs}}{dt} = -\frac{(R_s L_r^2 + R_r L_m^2)}{\sigma L_r^2 L_s} i_{qs} - \omega_e i_{ds} + \frac{R_r L_m}{\sigma L_s L_r^2}\psi_{qr} - \frac{L_m}{\sigma L_s L_r} p\omega_r \psi_{dr} + \frac{v_{qs}}{\sigma L_s} \quad (7.49)$$

Organizing equation (7.48), (7.49), (7.44) and (7.43) in vector matrix form ,the state space model is obtained as

$$\frac{d}{dt}\begin{bmatrix} i_{ds} \\ i_{qs} \\ \psi_{dr} \\ \psi_{qr} \end{bmatrix} = \begin{bmatrix} -a_1 & \omega_e & a_2 & a_3 p\omega_r \\ -\omega_e & -a_1 & -a_3 p\omega_r & a_2 \\ a_5 & 0 & -a_4 & \omega_{se} \\ 0 & a_5 & -\omega_{se} & -a_4 \end{bmatrix} \begin{bmatrix} i_{ds} \\ i_{qs} \\ \psi_{dr} \\ \psi_{qr} \end{bmatrix} + \begin{bmatrix} \frac{1}{\sigma L_s} & 0 \\ 0 & \frac{1}{\sigma L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_{ds} \\ v_{qs} \end{bmatrix} \quad (7.50)$$

where, $a_1 = \frac{\left(R_s L_r^2 + R_r L_m^2\right)}{\sigma L_r^2 L_s}$, $a_2 = \frac{R_r L_m}{\sigma L_s L_r^2}$, $a_3 = \frac{L_m}{\sigma L_s L_r}$, $a_4 = \frac{R_r}{L_r}$, $a_5 = \frac{R_r L_m}{L_r}$

$$T_e = \frac{3}{2} p \frac{L_m}{L_r}\left(i_{qs}\psi_{dr} - i_{ds}\psi_{qr}\right) \quad (7.51)$$

7.4 Problem Formulation

Figure 7.1 gives the scheme for parameter estimation of an induction motor.

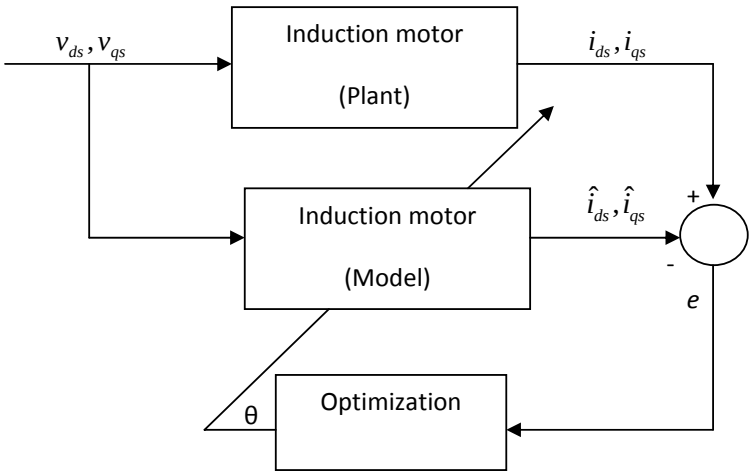

Figure 7.1: Formulation of Parameter Estimation Problem

The basic idea in parameter estimation is to compare the time dependent response of the system and a parameterized model by a norm or some performance criterion giving a measure of how well the model response fits the system response. Normally, the dynamic response of the induction motor is given by the solution to a vector differential equation of the form given in equations (7.50,7.51) which can be written in compact form as:

$$\dot{X} = AX + BU$$
$$Y = CX$$

where

$$X = [i_{qs}\ i_{ds}\ \lambda_{qr}\ \lambda_{dr}]^T$$

$$A = \begin{bmatrix} -\frac{R_s+R_r}{L_l} & 0 & \frac{R_r}{L_l L_m} & -\frac{\omega_r}{L_l} \\ 0 & -\frac{R_s+R_r}{L_l} & \frac{\omega_r}{L_l} & \frac{R_r}{L_l L_m} \\ -R_r & 0 & -\frac{R_r}{L_m} & \omega_r \\ 0 & R_r & -\omega_r & -\frac{R_r}{L_m} \end{bmatrix}$$

$$B = \frac{1}{L_l}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} v_q \\ v_d \end{bmatrix}$$

The parameter vector θ is given as follows

$$\theta = \begin{bmatrix} R_s & R_r & L_l & L_m \end{bmatrix}^T$$

where L_l is the leakage inductance and L_m is the mutual inductance. The initial condition of the model was established as given below

$$X(0) = \begin{bmatrix} i_{qs(0)} & i_{ds(0)} & \lambda_{qr(0)} & \lambda_{dr(0)} \end{bmatrix}^T$$

Normally, the system is affected by noise in both states and measurement, which may be real noise or noise caused by unmodeled dynamics. The parameter vector θ is unknown for real systems. Hence, the objective in parameter estimation is to determine this vector as accurately as possible. The system response and the model response can then be compared by a performance criterion, which in the simple case can be quadratic given as follows.

$$e = \sqrt{\sum_{k=1}^{N} \left(\hat{i}_{ds(k)} - i_{ds(k)}\right)^2 + \left(\hat{i}_{qs(k)} - i_{qs(k)}\right)^2} \tag{7.52}$$

where, e is the error between model and measurements, $\hat{i}_{ds(k)}$ and $\hat{i}_{qs(k)}$ denotes the estimated values. The objective of the estimation problem is to determine θ such that the error given by (7.52) is minimized. In Fig. 7.1, we present the parameter scheme of the induction motor drive system, where the optimization

is to be performed using different variants of DE and OMDE algorithm. The DE and OMDE will start searching the search space with random initialization. In each successive generation the search space shrinks and the algorithm will converge after some generation if the performance index will reach certain predefined value.

7.5 Opposition Based Mutation Differential Evolution (OMDE)

The concept of OBL has been discussed in chapter 6. As, it also known that if the crossover is applied between two good parents then there is fair chance of reproducing better offsprings. So the same opposition based approach can be applied to each solution after the mutation and before the crossover in the current population. Two main steps are distinguishable for OMDE, namely, opposition based initialization and opposition based mutation. We will enhance these two steps using the OBL scheme. The original DE is chosen as a parent algorithm and the proposed opposition-based ideas are embedded in DE to accelerate its convergence speed. The main advantage of OMDE over ODE is that it requires less number of tuning parameters i.e. some extra parameters such as jumping rate J_r to be tuned properly. The convergence of the ODE algorithm is highly dependent on the jumping rate J_r. The advantage of OMDE is that it does not require any extra parameters to be tuned which provides more flexibility than its counterpart i.e. ODE. Corresponding algorithm for the proposed OMDE is explained in algorithm1.

7.5.1 Opposition-Based Population Initialization

According to our review of optimization literature, random number generation, in absence of a priori knowledge, is the common choice to create an initial population. Therefore, by utilizing OBL, we can obtain fitter starting candidate solutions even when there is not a priori knowledge about the solution(s). The following steps present opposition-based initialization for OMDE

that procedure. Initialize the population $pop(P)$ randomly Calculate opposite population

$$opop_{i,j} = a_j + b_j - pop_{i,j}$$

$$i = 1, 2, \cdots, P \quad j = 1, 2, \cdots, d$$

where $pop_{i,j}$ and $opop_{i,j}$ denote the j^{th} variable of the i^{th} vector of the population and opposite population respectively. Select P fittest individual from the union of *pop* and *opop* as initial population i.e. *mpop*.

7.5.2 Opposition-Based Mutation

By applying the same approach described above, to the current population, after the mutation, the evolutionary process can be forced to create new solution candidate, which ideally is fitter than the current one. After generating populations using mutation, the opposite population is calculated and the fittest individuals are selected from the union of the current population and the opposite population. Unlike opposition-based initialization, opposition based mutation calculates the opposite population dynamically i.e. instead of using variables within predefined boundaries, opposition based mutation calculates the opposite of each variable based on minimum and maximum values of that variable in the current population as given by the equation below.

$$ompop_{i,j} = \min(mpop_j) + \max(mpop_j) - mpop_{i,j}$$

In each generation the maximum and minimum values of the variables changes dynamically, as generation increases the search space is reduced so the boundary values changes for each generation. By staying within variables interval static boundaries (i.e. initial boundary values), we would jump outside of the already shrunken search space and the knowledge of the current reduced space (converged population) would be lost. Hence, we calculate opposite points by using variables current interval in the population

which shrinks with higher generation. Like DE, there exits ten variants of OMDE which depend on the type of cross over and mutation. The variants of the OMDE are OMDE/best/1/exp, OMDE/rand/1/exp, OMDE/rand-to-best/1/exp, OMDE/best/2/exp, OMDE/rand/2/exp, OMDE/best/1/bin, OMDE/rand/1/bin, OMDE/rand-to-best/1/bin, OMDE/best/2/bin, OMDE/rand/2/bin.

Algorithm 3 OMDE Algorithm

Require: *pop*: initial population, F: Mutation constant, C: Cross over constant

{Opposition based Initialization }
for $i = 0$ to P **do**
  for $j = 0$ to d **do**
    $opop_{i,j} = a_j + b_j - pop_{i,j}$
  end for
end for
Select P fittest individual from the set $(pop_{i,j} \cup opop_{i,j})$
while Convergence criteria not met **do**
  for $i = 0$ to P **do**
    $r_1 = rand(P)$
    $r_2 = rand(P)$
    $r_3 = rand(P)$
    $v_{i,g} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g})$ {Srore in mutation population *mpop* }
    {*/ Opposition based mutaion starts */ }
    $ompop_{i,j} = min(mpop_j) + max(mpop_j) - mpop_{i,j}$
    Select P fittest individual from the set $(mpop_{i,j} \cup ompop_{i,j})$ which is denoted by ov
    {*/ Opposition based mutaion ends */ }
    $t_{j,i,g} = \begin{cases} ov_{j,i,g} \text{ if } (rand_j \leq C) \text{ or } j = j_{rand} \\ x_{j,i,g} \text{ otherwise} \end{cases}$
    if $f(t_{i,g}) \leq f(x_{i,g})$ **then**
      $x_{i,g+1} = t_{i,g}$
    else
      $x_{i,g+1} = x_{i,g}$
    end if
  end for
end while

7.6 Parameter Identification of Induction Motor using DE and OMDE

The objective of the parameter estimation of induction motor is to determine a mathematical model of the motor with sufficient accuracy. To develop robust methods for parameter estimation, it is important to quantify the information content about machine parameters on measured signals. This is of particular importance when we restrict only to electrical terminal quantities, such as stator voltages and currents. Most of the existing parameter estimation methods such as LMS and RLS methods use the regressor equation i.e.

$$y = x^T\theta + \varepsilon \tag{7.53}$$

where y is the output vector, x is the regressor matrix, θ is the parameters to be estimated and ε is the system noise. However, difficulties are encountered in the regression equation (7.53) and in turn this method may be a viable choice for all situations in induction motor parameter estimation problems. Therefore, we explore an alternative way of solving the parameter estimation problem by using evolutionary method i.e. the DE and OMDE which do not require the description of equation (7.53) for parameter estimation.

It may be noted that recently DE algorithm has been considered as a novel evolutionary computation technique used for optimization problems. The DE has been preferred to many other evolutionary techniques such as GA and PSO due to its attractive characteristics such as its simple concept, easy implementation and quick convergence. Generally speaking, all population-based optimization algorithms, no exception for DE, suffer from long computational times because of their evolutionary/stochastic nature. This crucial drawback sometimes limits their application to offline problems with little or no real-time constraints. The parameter identification of induction motor using differential evolution and stochastic optimization algorithm is discussed in [155], [154]. The concept of using OBL for initialization and for generation jumping was given by

[116] which is known as ODE. Our proposed approach has been called OMDE. OMDE uses opposite numbers during population initialization and also for generating new populations after mutation during the evolutionary process. As in differential evolution the cross over is done after the mutation, we have put the OBL just after mutation so that better individuals can take part in crossover as a result we could able to get fitter individuals in the next generation. The focus of parameter estimation using evolutionary computation here are as follows.

- Instead of being confronted with difficulties in finding expressions to represent the system by $y = x^T\theta+\varepsilon$, the DE and OMDE method estimate the parameters directly.
- An extensive study on finding of an efficient OMDE strategy with a view of obtaining faster convergence for parameter estimation of induction motor has been pursued.

7.7 Results and Discussions

The parameter estimation schemes such as DE and OMDE have been applied to the induction motor by using the input-output data i.e. the stator voltage (transformed d-q axis, equation) and the stator current (d-q transformed, equation) to estimate motor resistance and inductance. Table 7.1 gives the rating of the induction motor and the actual value of the parameters to be identified. The results obtained is the average of 30 independent runs for each case.

Table 7.1: Specification of the induction motor

Voltage	220 V
Power	5 HP
Frequency	50 Hz
Stator resisteance	0.3900 Ohm
Rotor resistance	0.2200 Ohm
Leakage Inductance	0.0060 Henry
Magnetising Inductance	0.0680 Henry
RPM	1750

All the ten variants of the DE and OMDE schemes for identifying the motor parameters R_s, R_r, L_l and L_m have been implemented using the following common parameters given in table 7.2.

Table 7.2: Parameters of the proposed DE and OMDE

Number of generation	50
Population size, P	20
Upper and lower bound of stator resistance	[0 1]
Upper and lower bound of rotor resistance	[0 1]
Upper and lower bound of leakage inductance	[0 1]
Upper and lower bound of magnetizing inductance	[0 1]
Mutation factor, F	0.6
Crossover constant, C	0.5

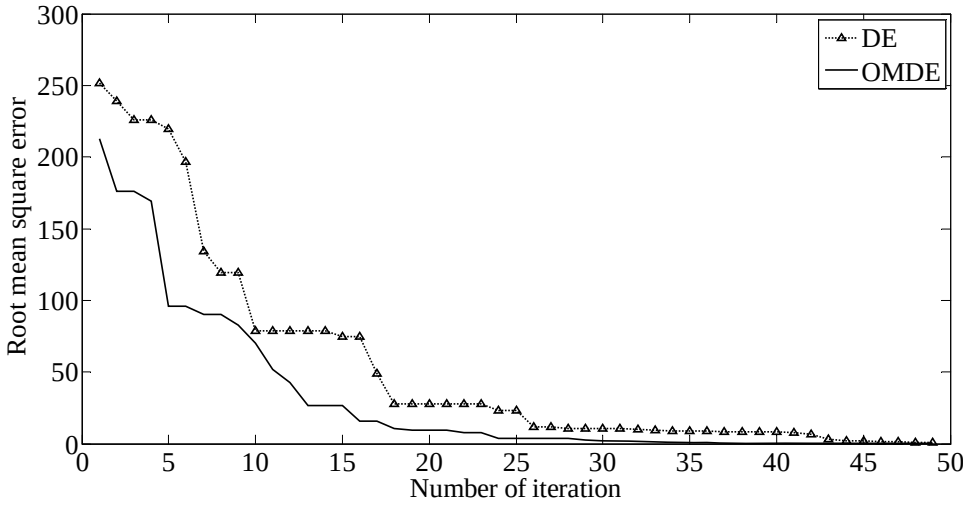

Figure 7.2: RMSE for DE/best/1/exp and OMDE/best/1/exp

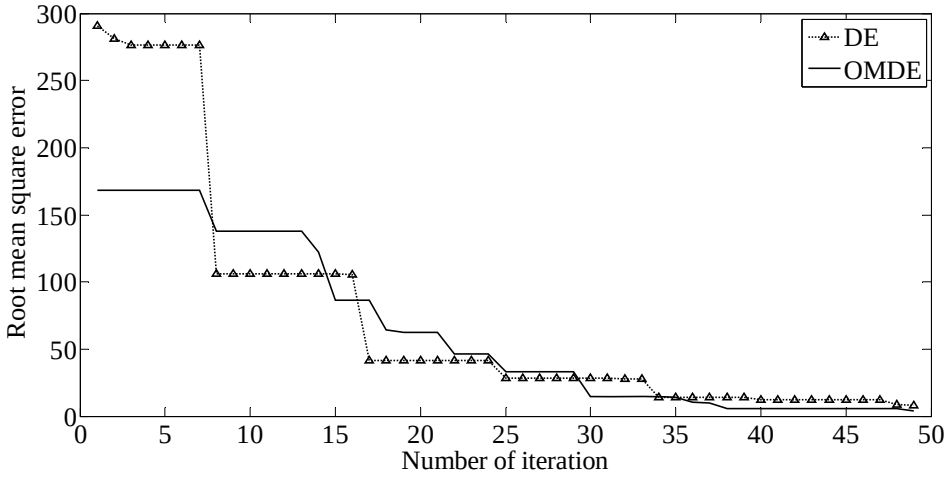

Figure 7.3: RMSE for DE/rand/1/exp and OMDE/rand/1/exp

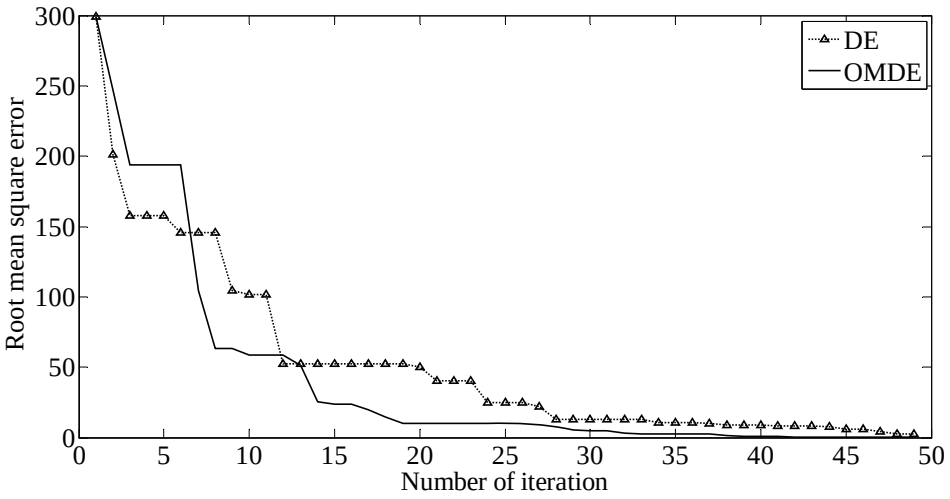

Figure 7.4: RMSE for DE/rand-to-best/1/exp and OMDE /rand-to-best/1/exp

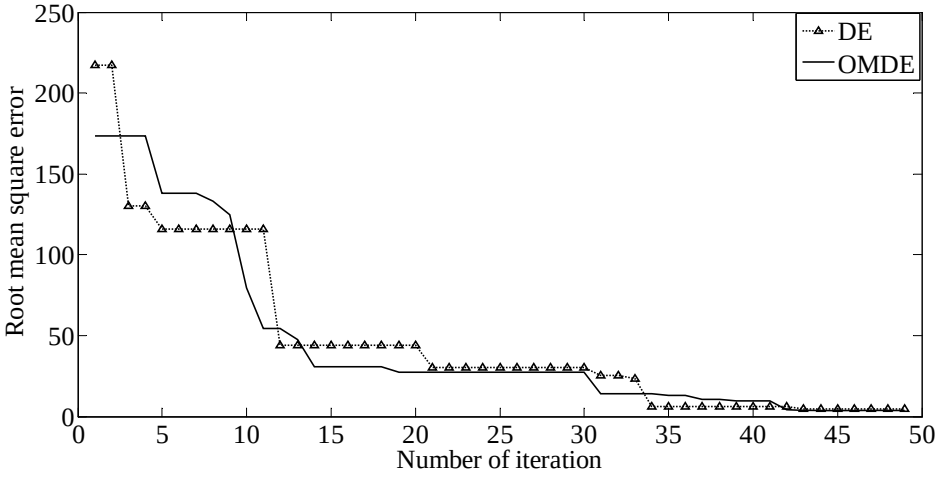

Figure 7.5: RMSE for DE/best/2/exp and OMDE/best/2/exp

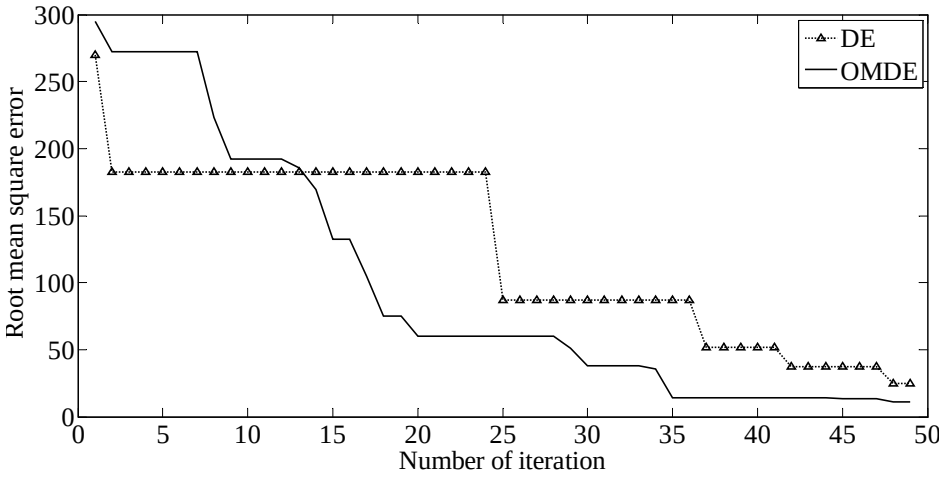

Figure 7.6: RMSE for DE/rand/2/exp and OMDE/rand/2/exp

Figure 7.2-7.6 shows the RMSE plot for the exponential crossover scheme. From these figures it is clear that the OMDE strategies outperform over the corresponding DE strategies in terms of faster convergence and less estimated error, the RMSE is maximum for DE/rand/2/exp i.e. 25.08 and minimum for OMDE/best/1/exp i.e. 0.0307.

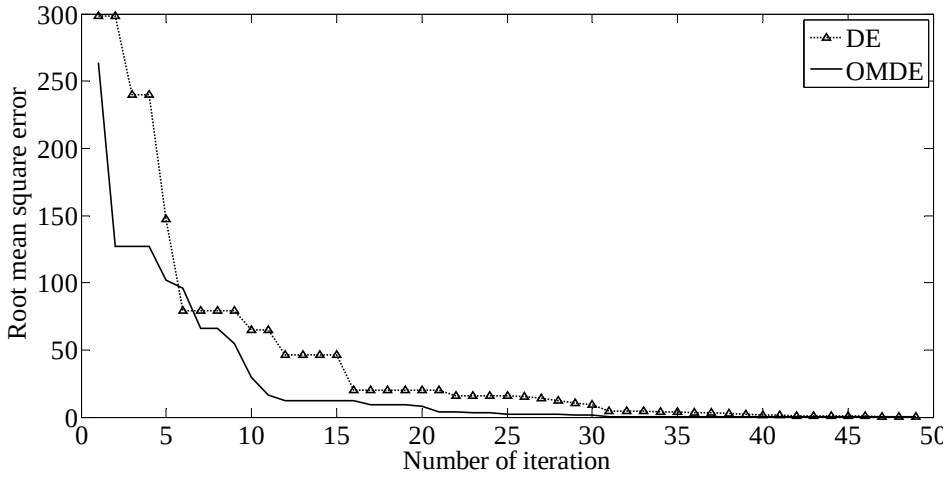

Figure 7.7: RMSE for DE/best/1/bin and OMDE/best/1/bin

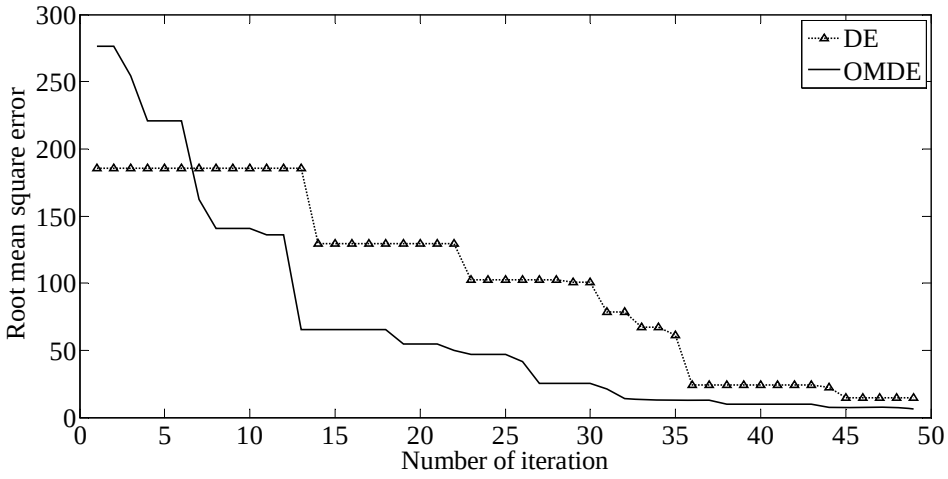

Figure 7.8: RMSE for DE/rand/1/bin and OMDE/rand/1/bin

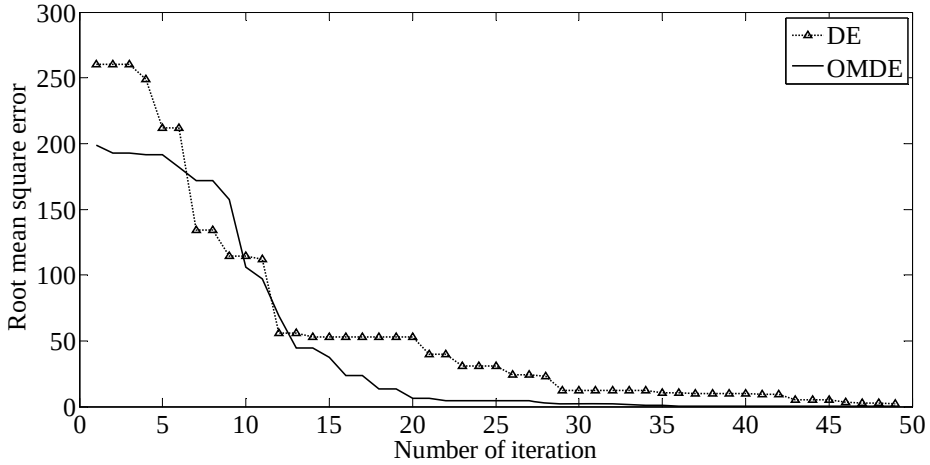

Figure 7.9: RMSE for DE/rand-to-best/1/bin and OMDE /rand-to-best/1/bin

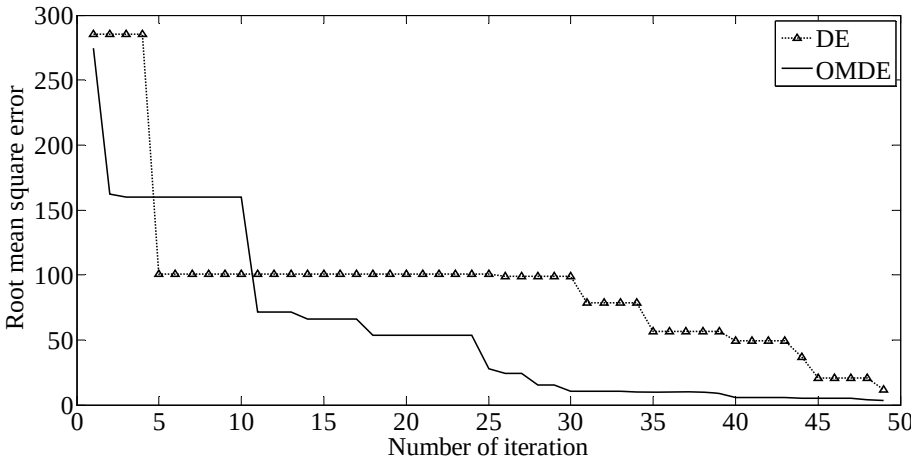

Figure 7.10: RMSE for DE/best/2/bin and OMDE/best/2/bin

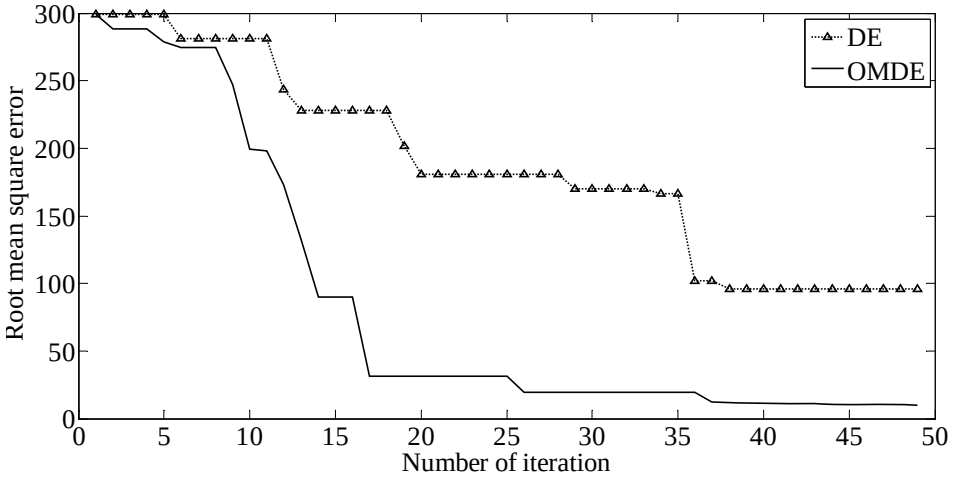

Figure 7.11: RMSE for DE/rand/2/bin and OMDE/rand/2/bin

Figure 7.7-7.11 gives the RMSE plot for different DE and OMDE strategies for the binomial cross scheme. From all the Figures 7.3-7.12 it is clear that the OMDE has the better convergence characteristic in comparison to classical DE both for exponential and binomial crossover scheme.

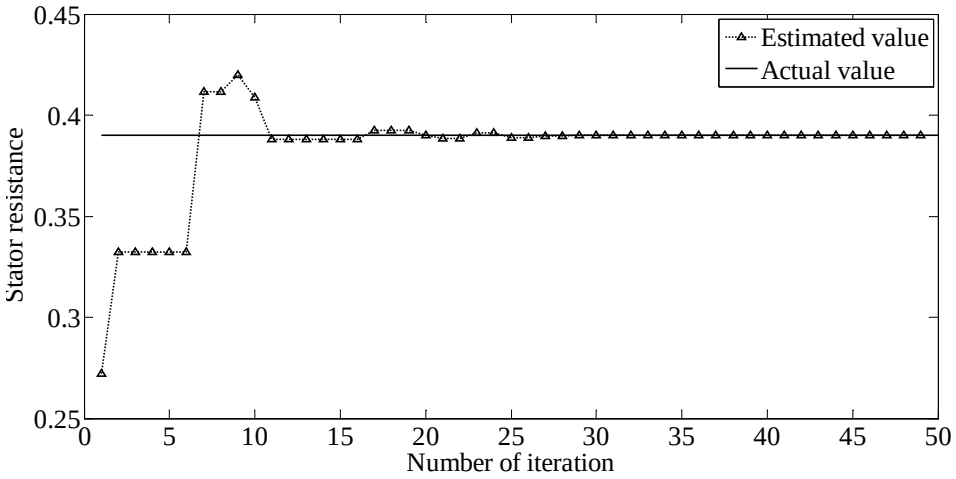

Figure 7.12: Estimation of stator resistance

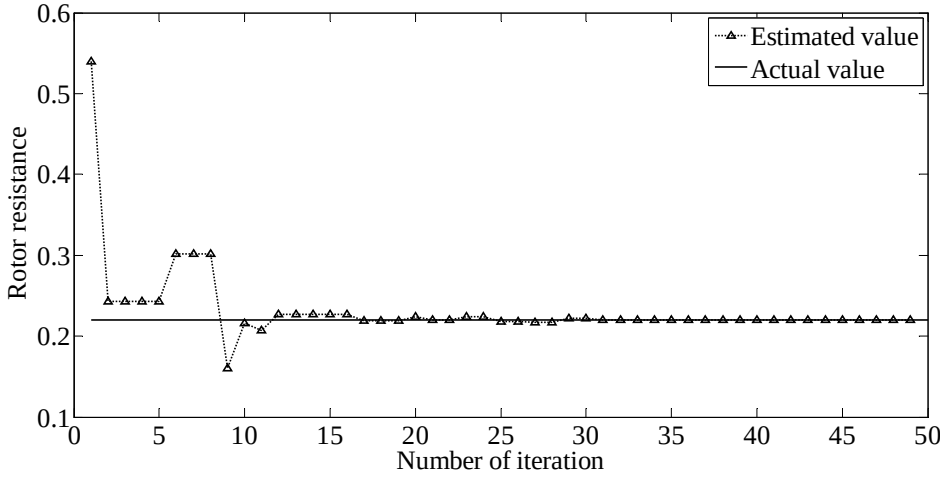

Figure 7.13: Estimation of rotor resistance

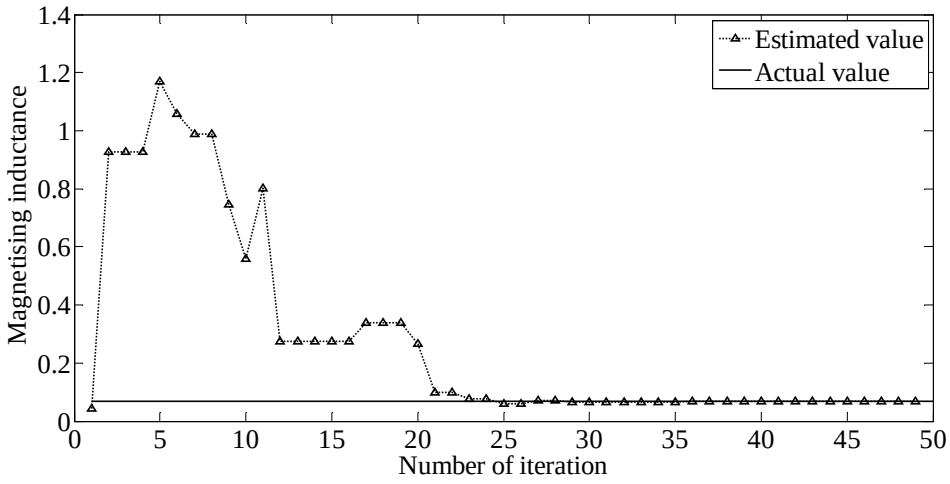

Figure 7.14: Estimation of magnetizing inductance

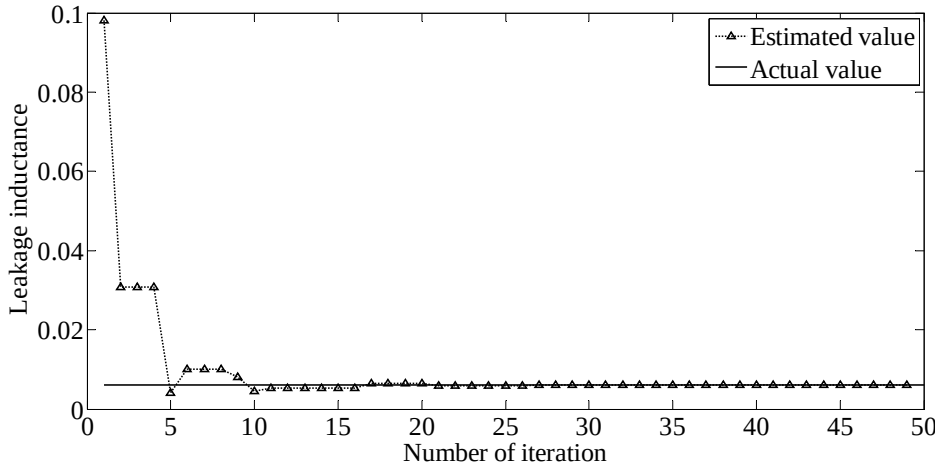

Figure 7.15: Estimation of leakage inductance

Figures 7.12-7.15 gives the value of the estimated stator resistance, rotor resistance, magnetizing inductance and leakage inductance respectively, for the strategy OMDE/best/1/bin those values becomes approximately equal to its actual value after 30 iterations. Table-7.3 shows the comparison of the performance of all the ten variants of DE and OMDE terms of the means squared error after fifty iterations. From the results it is found that for DE/best/1/bin the RMSE is minimum i.e. 0.4060 among all the DE strategies similarly OMDE/best/1/bin gives the minimum RMSE i.e. 0.0168 amongst all the

OMDE strategies. This shows that the superiority of proposed OMDE over the conventional DE. In some cases shown in Table-7.3 the value of the parameters exceed their boundary values because we have defined the boundaries only for initialization, but when the evolutionary process continues the parameter values depend on the type of mutation scheme. The process converges to its optimal value after few numbers of iterations i.e. only 30 number of iterations as shown in Fig 7.7 for the strategy OMDE/best/1/bin.

Table 7.3: Comparison of estimation results for different strategies

Strategies of DE and OMDE	Stator resistance in Ohm	Rotor resistance in Ohm	Leakage Inductance in Henry	Magnetizing Inductance in Henry	RMSE
DE/best/1/exp	0.3903	0.2204	0.0060	0.0728	0.7778
DE/rand/1/exp	0.3920	0.2157	0.0052	0.0564	7.4396
DE/rand-to-best/1/exp	0.3894	0.2192	0.0061	0.0503	2.3332
DE/best/2/exp	0.3913	0.2250	0.0063	0.0688	4.8499
DE/rand/2/exp	0.3979	0.1956	0.0077	7.5864	25.0840
DE/best/1/bin	**0.3900**	**0.2202**	**0.0060**	**0.0694**	**0.4060**
DE/rand/1/bin	0.3902	0.2051	0.0065	3.1812	14.4818
DE/rand-to-best/1/bin	0.3899	0.2180	0.0060	0.0855	2.0627
DE/best/2/bin	0.3902	0.2135	0.0066	6.7598	11.7129
DE/rand/2/bin	0.4455	0.1534	0.0145	19.6302	95.6588
OMDE/best/1/exp	0.3900	0.2200	0.0060	0.0683	0.0307
OMDE/rand/1/exp	0.3903	0.2256	0.0062	0.0734	4.0755
OMDE/rand-to-best/1/exp	0.3900	0.2201	0.0060	0.0672	0.1249
OMDE/best/2/exp	0.3923	0.2194	0.0058	0.0555	3.6979
OMDE/rand/2/exp	0.3884	0.2264	0.0063	-26.3119	10.7830
OMDE/best/1/bin	**0.3900**	**0.2199**	**0.0060**	**0.0680**	**0.0168**
OMDE/rand/1/bin	0.3888	0.2196	0.0061	0.2271	6.2502
OMDE/rand-to-best/1/bin	0.3900	0.2200	0.0060	0.0685	0.0611
OMDE/best/2/bin	0.3902	0.2222	0.0062	0.0980	3.1331
OMDE/rand/2/bin	0.3906	0.2207	0.0059	22.7209	10.1365

7.8 Summary

This chapter presents a new differential evolution algorithm called opposition based mutation differential evolution. The application of the DE and OMDE strategies for efficiently solving the identification problem of an induction motor is described in this chapter. Here ten different DE and OMDE formulations is considered towards estimating the parameters i.e. stator and rotor resistances, leakage inductance and magnetizing inductance of the Induction Motor Drive System. For a given induction motor, the unknown parameters are successively evolved by using DE and OMDE algorithm to approximate the actual parameters accurately. From results obtained above, it is concluded that OMDE/best/1/exp gives the better result in terms of faster convergence and accuracy in estimating parameters.

Chapter 8

Conclusion

This chapter concludes the thesis and some future research problems that may be attempted by an interested reader are outlined.

8.1 Summary of the Thesis Work

The thesis has mainly investigated on identification and parameter estimation of nonlinear systems. In this work a number of neuro-evolutionary hybrid system algorithms based on evolutionary strategy have been investigated. We have used two different fusion strategies for hybridizing EA and GD i.e. sequential and memetic hybridization.

- A new SH known as DE+LM+NN approach has been employed for identification of nonlinear plants where LM is used after DE. This study presents a promising hybrid optimization technique. Based on a strong coupling approach between a DE and the LM, a new hybrid optimizer has been developed. This tool exploits the main advantages of both DE and LM, namely the ability to deal with problems exhibiting several local minima for the former, and the quick convergence to the optimal solution for the latter. Another advantage of this hybrid optimizer with respect to a pure DE is that, for a similar quality of results, it allows the use of a smaller number of function evaluations, and thereby an important reduction of the computational time. This hybrid optimizer has successfully solved various nonlinear system identification problems. It

is shown through extensive study using simulation and analysis that the new DE+LM +NN has taken less computational overhead and better identification performance compared to the DE+NN and conventional Hammer Stein-Wiener identification. The main disadvantage of those type of SH algorithm is the decision of the point of switching from one algorithm to other. We have taken the MSE as criteria to switch from DE to LM.

- The convergence of the DE algorithm has been proved in a stochastic framework.

- One of the possibilities for hybridization of an evolutionary algorithm with local search algorithm is explored which is known as MA. From an optimization point of view, MAs are hybrid evolutionary algorithms that combine global and local search by using an evolutionary algorithm to perform exploration while the local search method performs exploitation. In MAs either some or all the solutions obtained by EA are improved through local search. In this work all the solutions obtained by GA, PSO and DE are improved by BP. Different memetic algorithms such as GABP, PSOBP and DEBP have been developed for identification of nonlinear systems. Here BP is applied to all populations after the mutation and cross over. In terms of convergence behavior, and identification error, it is observed that the DEBP memetic algorithm trained system identification scheme offer improved identification performance compared to BP, GA, GABP, PSO, PSOBP, and DE based methods. Further it is observed that memetic algorithm trained NN approach to system identification using DE is an efficient method of offering better optimal solutions compared to GA and PSO based identification schemes.

- The TRMS is a highly nonlinear system which can be considered as an experimental model whose behavior in certain aspects resembles that of

a helicopter, with a significant cross coupling between longitudinal and lateral directional motions, so, it is an interesting identification problem. A dynamic model of a TRMS is extracted using a black-box system identification technique. Two neural network based models are developed using hybrid evolutionary algorithms i.e. sequential hybridization and memetic approach has been applied to a TRMS system for modeling with a NARX model structure. The identification results are compared in terms of identification capability and speed of convergence. The responses of all the identified models are compared with that of the real TRMS to validate the accuracy of the models. It is found that the identification using the above two methods are almost similar but the DE+LM+NN sequential hybridization method is comparatively taking less computation time than the memetic DEBP identification.

- Subsequently an opposition-based learning as a new scheme for evolutionary computation is introduced. Optimizing neural network weights by using the concept of opposite weights is the foundation of this approach. A variant of the differential evolution ODE basing on OBL is applied to train a feed-forward neural network that is used for system identification nonlinear systems. This approach is applied to the nonlinear systems [61], [62] including the real time TRMS. From the simulation results it was found that the ODE based system identification is faster and the identification error is less compared to DE based SI approach.

- A new differential evolution algorithm called OMDE have been proposed to develop a parameter estimation scheme. In this work; the application of the DE and OMDE strategies have been described for efficiently solving the identification problem of an induction motor. Here ten different DE and OMDE formulations towards estimating the parameters i.e. stator and rotor resistances, leakage inductance and magnetizing inductance of the induction motor drive system are considered. For a given induction motor, the unknown parameters are successively evolved by

using DE and OMDE algorithm to approximate the actual parameters accurately. Results obtained envisage that OMDE/best/1/exp gives the better result in terms of faster convergence and accuracy in estimating parameters. One of the advantages of using DE over other algorithms is that it requires less number of tuning parameters. The ODE converges faster than DE but it requires some extra parameters such as jumping rate J_r to be tuned properly which is difficult to find and varies for different problems. The advantage of OMDE over ODE is that the proposed method does not require any extra parameters to be tuned which provides more flexibility than its counterpart i.e. ODE.

The objectives of the thesis proposed in section 1.4 have been thus met in overcoming the difficulties of existing neural network based system identification strategies by demonstrating the efficacies of new variants of neuro-evolutionary system identification algorithms.

8.2 Thesis Contributions

The following are the salient contributions of the thesis.

- Introduction of a new identification framework which combines both DE and LM, for optimizing the weights of a feed-forward neural network whose minimization yields an estimated model [168].
- Convergence analysis of the DE algorithm for nonlinear system identification in a stochastic frame work has been made.
- Development of a memetic algorithm with the hybridization of DE and BP for optimizing the weights of neural network applied to nonlinear system identification problems [169].
- Development of a variation of DE based on opposition based learning i.e. for enhancing the convergence of DE [170].

- Improvement of nonlinear system identification performance by proposing an ippositon based mutation differential evolution.

8.3 Future Scope of Work

- The DE applications may divide into two main areas: off-line design-aid tools and robust on-line search improvement algorithms. A training method for designing of hybrid DE computing models for nonlinear system identification has been proposed here, which is an off-line training method. Possible improvement and generalization of the algorithm for the use of online training are needed. The rate at which the DE is applied to real-world problems is predicted to increase still further during the next few years so different variants of DE and hybrid DE for speeding up the training speed needed to be further studied. We have successfully applied our methods to the identification and control of nonlinear systems, e.g., some bench mark problems, Box Jenkinns furnace system and TRMS system. The next step is to evaluate the effectiveness of our methods in other industrial plants.

- Neural network ensembles are receiving increasing attention in recent neural network research, due to their interesting features. They are a powerful tool especially when facing complex problems. Although theoretically, a single neural network with a sufficient number of neurons in the hidden layer would suffice to solve any problem, in practice many real-world problems are too hard to construct the appropriate network that solve them. In such problems, neural network ensembles are a successful alternative. Network ensembles are usually made up of a linear combination of several networks that have been trained using the same data, although the actual sample used by each network to learn can be different. Each network within the ensemble has a potentially different weight in the output of the ensemble. Several works have shown that the network ensemble has a generalization error generally smaller than

that obtained with a single network and also that the variance of the ensemble is lesser than the variance of a single network. Identification of nonlinear system using ensemble hybrid evolutionary learning could be the future extension of this work.

- Power system small signal, transient, and dynamic stability studies should be as accurate as possible which depends heavily on the accuracy of the model parameters of the system components. In practice, the parameters commonly used in stability studies are manufacturer specified values, or typical values. These typical values may be grossly inaccurate, as various parameters may drift over time or with operating condition. Thus, it is desirable to develop methods for estimating exact component parameters of an excitation system. Parameter estimation of synchronous machines has been well documented, parameter estimation of excitation systems has only begun to receive thorough attention. This work presents a new method of off-line estimation for the stator and rotor resistances, magnetic and leakage inductances of an induction motor drive, using DE and its variation i.e. OMDE. In this context, it would be quite interesting to use the same algorithm for estimating the parameters of an excitation system which includes nonlinear saturation function and nonlinear regulating rectifier function of the main exciter. The method will utilize the data signals in the time domain and will be used to estimate the parameters of an IEEE excitation system [166].

Bibliography

[1] L. L. Jung, *System Identification - Theory for the User*. 2nd edition, Prentice-Hall. Upper Saddle River, N.J., 1999.

[2] S. A. Billings, Identification of nonlinear systems-a survey, *IEE Proceedings*, Part D, vol.127, pp.272-285, 1980.

[3] S. A. Billings and G. N. Jones, Orthogonal least-squares parameter estimation algorithms for nonlinear stochastic systems, *International Journal of Systems Science*, vol.23, no.7, pp.1019-1032, 1992

[4] J. Leontaritis and S. A Billings, Input-output parametric models for nonlinear systems, Part I: Deterministic nonlinear systems, *International Journal of Control*, vol.41, pp.303-328, 1985.

[5] H. Haber, and H. Unbehauen, Structure identification of nonlinear dynamic systems-A survey on input/output approaches, *Automatica*, vol.26, no.4, pp.615-677, 1990.

[6] S. Chen, S. A. Billings, Representation of nonlinear systems: the NARMAX model, *International Journal of Control*, vol.49, pp.1013-1032, 1989.

[7] S. berg, Q. Zhang, L. L. Jung, A. Benveniste, B. Delyon, P. Glorennec, H. Hjalmarsson and A. Juditsky, Nonlinear black-box modeling in system identification: a unified overview, *Automatica*, vol.31, no.4, pp.1691-1724, 1995.

[8] A. Boukhris, G. Mourot and J. Ragot, Nonlinear dynamic system identification: a multi-model approach, *International Journal of Control*, vol.72, no.7/8, pp.591-604, 2000.

[9] R. Hecht-Nielsen, *Neurocomputing*. Reading, MA: Addison-Wesley, 1990.

[10] M. L. Minsky and S. Papert, *Perceptrons*. Cambridge, MA: MIT Press, 1969.

[11] K. J. Hunt, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, Neural networks for control systems-a survey, *Automatica*, vol.28, pp.1083-1112, June 1992.

[12] B. Widrow and R. Winter, Neural nets for adaptive filtering and adaptive pattern recognition, *Computer*, vol.21, no.3, pp.25-39, 1988.

[13] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[14] J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation*. New York, NY: Addison-Wesley, 1991.

[15] D. R. Hush and B. G. Horne, Progress in supervised neural networks, IEEE Signal Processing Mag., vol.10, pp.8-39, 1993.

[16] K. S. Narendra and K. Parthaasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans Neural Networks*, vol.1, no.1, pp.4-27, 1990.

[17] S. Chen and S. A. Billings, Neural networks for nonlinear dynamic system modeling and identification, *International Journal of Control*, vol.56, pp.319-346, 1992.

[18] S. Chen, S. A. Billings, C. F. N. Cowan and P. M. Grant, Practical identification of NARMAX models using radial basis functions, *International Journal of Control*, vol.52, pp.1327-1350, 1990.

[19] S. A. Billings and H. L. Wei, A new class of wavelet networks for nonlinear system identification, *IEEE Trans Neural Networks*, vol.16, no.4, pp.862-874, 2005.

[20] David westwick, *Methods for identification of multiple input nonlinear system*, PhD Thesis, Department of Electrical Engineering and Biomedical Engineering Department McGill University, Montreal, 1995.

[21] S. Jagannathan and F. L. Lewis, Identification of nonlinear dynamical systems using multilayered neural networks, *Automatica*, vol.32, no.12, pp.1707-1712, 1996.

[22] G. Kenn, T. Ahmed-Ali, F. Lamnabhi-Lagarrigue and H. N. kwawo, Nonlinear systems parameters estimation using radial basis function network, *Control Engineering Practice*, vol.14, no.7, pp.819-832, 2006.

[23] J. I. Canelon, L. Shieh and N. B. Karayiannis, A new approach for neural control of nonlinear discrete dynamic systems, *Information Sciences*, vol.174, no.3-4, pp.177-196, 2005.

[24] W. Yu, Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms, *Information Sciences*, vol.158, pp.131-147, 2004.

[25] S.-K. Oh, W. Pedrycz and H.-S. Park, Hybrid identification in fuzzy-neural networks, *Fuzzy Sets and Systems*, vol.138, no.2, pp.399-426, 2003.

[26] A. Yazdizadeh and K. Khorasani, Adaptive time delay neural network structures for nonlinear system identification, *Neurocomputing*, vol.47, no.1-4, pp.207-240, 2002.

[27] M. O. Efe and O. Kaynak, A comparative study of neural network structures in identification of nonlinear systems, Mechatronics, vol.9, no.3, pp.287-300, 1999.

[28] A. K. Deb, Jayadeva, M. Gopal and S. Chandra, SVM-Based Tree-Type Neural Networks as a Critic in Adaptive Critic Designs for Control , *IEEE Trans. Neural Network*, vol.18, no.4, pp.1016-1030, 2007.

[29] G. C. Luh and C. Y. Wu, Nonlinear system identification using genetic algorithms, *In Proc. of Institute of Mechanical Engineering Part I: Journal of System and Control Engineering*, vol.213, pp.105-117, 1999.

[30] G. C. Luh and C. Y. Wu, Inversion control of nonlinear system with an inverse NARX model identified using genetic algorithms, *In Proc. of Institute of Mechanical Engineering Part I: Journal of System and Control Engineering*, vol.214, pp.259-271, 2000.

[31] C. M. Fonseca, E. M. Mendes, P. J. Fleming and S. A. Billings, Nonlinear model term selection with genetic algorithms, *In Proc. of IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, pp.1-8, 1993.

[32] G. Franco, R. Betti and H. Lus, Identification of structural systems using an evolution strategy, *Journal of Engineering Mechanics*, vol.130, no.10, pp.1125-1139, 2004.

[33] R. K. Ursem, *Models for evolutionary algorithms and their applications in system identification and control Optimization*, PhD Thesis, Faculty of Science of the University of Aarhus, 2003.

[34] R. Belea, *Contribution to the system identification using genetic algorithms*, PhD Thesis, DUNAREA DE JOS - University of Galati, 2004.

[35] Zakharov and S. Halasz, Parameter identification of a robot arm using Genetic Algorithms, Periodica Politehnica Ser. Eng. vol.45, no.3-4, pp.195-209, 2001.

[36] X. Yao, A review of evolutionary artificial neural networks, *Int. J. Intell. Syst.*, vol.8, no.4, pp.539-567, 1993.

[37] X. Yao, Evolutionary artificial neural networks, *Int. J. Neural Syst*, vol.4, no.3, pp.203-222, 1993.

[38] 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, April 1999.

[39] J. Xu, GA-Optimized wavelet neural networks for system identification, *First International Conference on Innovative Computing, Information and Control ,(ICICIC'06)*, vol.1, pp.214-217, 2006.

[40] Y. Chen and S. Kawaji , Evolving wavelet neural networks for system identification, *In Proc. International Conf. on Electrical Engineering*, Kitakyushu, Japan, pp.279-282, 2000.

[41] H. Du and N. Zhang, Time series prediction using evolving radial basis function networks with new encoding scheme, *Neurocomputing*, vol.71, no.7-9, pp.1388-1400, 2008.

[42] W. F. Xie, Y. Q. Zhu, Z. Y. Zhao and Y. K. Wong, Nonlinear system identification using optimized dynamic neural network, *Neurocomputing*, vol.72, no.13-15, pp.3277-3287, 2009.

[43] A. Ali, H. M. Mojtaba and K. Hamid, An effective approach to nonlinear Hammerstein model identification using evolutionary neural network, *In Proc. International Joint Conference on Neural Networks,* Budapest, Hungary, 25-29, July, 2004.

[44] S. Chen, X. Hong, B. L. Luk and C. J. Haris, Nonlinear system identification using particle swarm optimisation tuned radial basis function models, *International Journal of Bio-Inspired Computation*, vol.1, no.4, pp.246-258, 2009.

[45] G. H. Wei, L. Yan-Chun and M. Maurizio, A modified particle swarm optimization-based dynamic recurrent neural network for identifying and

controlling nonlinear systems, *Computers and structures*, vol.85, no.21-22, pp.1611-1622, 2007.

[46] T. Marcu, B. K. Seliger, P. M. Frank and S. X. Ding, Dynamic functional-link neural networks genetically evolved applied to system identification, *ESANN'2004, proc. European Symposium on Artificial Neural Networks*, Bruges (Belgium), pp.115-120, April 28-30, 2004.

[47] L. S. Coelho and M. W. Pesso, Nonlinear identification using a B-spline neural network and chaotic immune approaches, *Mechanical Systems and Signal Processing*, vol.23, pp.2418-2434, 2009.

[48] B. Majhi and G. Panda, Development of efficient identification scheme for nonlinear dynamic systems using swarm intelligence techniques, *Expert Systems with Applications: An International Journal*, vol.37, no.1, pp.556-566, 2010.

[49] R. Storn and K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, vol.11, no.4, pp.341-359, 1997.

[50] J. Ilonen and J. K. Kamarainen, Differential evolution training algorithm for feed forward neural networks, *Neural. Proc. Lett.* vol.17, pp.93-105, 2003.

[51] Zelinka and J. Lampinen, An evolutionary learning algorithm for neural networks, *In: 5th International Conference on Soft Computing MENDEL99*, pp.410-414, 1999.

[52] F. Vavak, T. Fogarty and K. Jukes, A genetic algorithm with variable range of local search for tracking changing environments. I*n Proc. of the 4th Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science* 1141, Voigt, H.M., Ebeling, W., Rechenberg, I. and Schwefel, H.P. (eds.). Springer,1996.

[53] J. Knowles and D. Corne, A comparative assessment of memetic, evolutionary and constructive algorithms for the multi-objective d-msat problem, *In proc. Genetic and Evolutionary Computation Workshop*, 2001.

[54] N. Krasnogor, Self-generating metaheuristics in Bioinformatics: The protein structure comparison case, *Genetic Programming and Evolvable Machines*, vol.5, pp.181-201, 2004.

[55] W. Hart, *Adaptive global optimization with local search*, PhD thesis, University of California, San Diego, USA, 1994.

[56] M. Land, *Evolutionary algorithms with local search for combinatorial optimization*, PhD thesis, University of California, San Diego, USA, 1998.

[57] P. Merz, *Memetic algorithms for combinatorial optimization problems: fitness landscapes and efective search strategies*, PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2000.

[58] N. Krasnogor, *Studies in the theory and design space of memetic algorithms*, PhD thesis, University of the West of England, Bristol, U.K, 2002.

[59] B. Liu, L. Wang, Y. Jin and D. Huang, Designing neural networks using PSO based Memetic Algorithm, Advances in Neural Networks, pp.219-224, 2007.

[60] M. Delgado, M. C. Pegalajar and M. P. Cullar, Memetic evolutionary training for recurrent neural networks: an application to time-series prediction, *Expert Systems*, vol.23, no.2, pp.99-115, 2006.

[61] C.-T. Lin and C. S. George Lee, *Neural Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems*, Prentice Hall International, Inc., New Jersey: 1996.

[62] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden Day, San Francisco (1970).

[63] D. E. Rumehart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol.1, Cambridge, MA: MIT Press, 1986.

[64] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representations by error propagation, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol.1, MIT Press, 1986, pp.318-362.

[65] M. F. Moller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks, vol.6, no.4, pp.525-533, 1993.

[66] M. Liang, S.-X. Wang and Y.-H. Luo, Fast learning algorithms for multilayered feedforward neural network, *In IEEE 1994 National Aerospace and Electronics Conference NAECON1994*, vol.2, pp.787-790,1994

[67] C. Charalambous, Conjugate gradient algorithm for efficient training of artificial neural networks, Circuits, Devices and Systems, vol.139, no.3, pp.301-310, 1992.

[68] M. Moller, *Efficient training of feed-forward neural networks*, Ph.D. thesis, Computer Science Department, Aarhus University, Arhus, Denmark, 1997.

[69] M. T. Hagan and M. B. Menhaj, Training feedforward networks with the Marquadt algorithm, *IEEE Transactions on Neural Networks*, vol.5, no.6, pp.989-993, 1994.

[70] N. Japkowicz and S. J. Hanson, Adaptability of the back-propagation procedure, *In IJCNN99 International Joint Conference on Neural Networks*, vol.3, pp.1710-1715, 1999.

[71] S. P. Day and D. S. Camporese, A stochastic training technique for feed-forward neural networks, *In IJCNN International Joint Conference on Neural Networks*, vol.3, pp.607-612, 1990.

[72] D. Whitley, T. Starkweather and C. Bogart, Genetic algorithms and neural networks: Optimizing connections and connectivity, *Parallel Computing*, vol.14, no.3, pp.347-361, 1990.

[73] L. Whitley, S. Gordon and K. Mathias, Lamarkian evolution, the Baldwin effect, and function optimisation. *In Proc. of the 3rd Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 866, Davidor, Y., Schwefel, H.P. and Manner, R. (eds). Springer, 1994.

[74] D. Montana and L. Davis, Training feedforward neural networks using genetic algorithms, *In Proc. 11th Int. Joint Conf. Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, pp.762-767, 1989.

[75] G. W. Greenwood, Training partially recurrent neural networks using evolutionary strategies, *IEEE Trans. Speech Audio Processing*, vol.5, pp.192-194, 1997.

[76] D. J. Janson and J. F. Frenzel, Application of genetic algorithms to the training of higher order neural networks, *J. Syst. Eng*, vol.2, pp.272-276, 1992.

[77] M. Lehotsky, V. Olej and J. Chmurny, Pattern recognition based on the fuzzy neural networks and their learning by modified genetic algorithms, Neural Network World, vol.5, no.1, pp.91-97, 1995.

[78] S. E. Fahlman, Faster-learning variations on back-propagation: An empirical study, *In Proc. Connectionist Models Summer School*, pp.38-51, 1988.

[79] R. Dawkins, *The Selfish Gene*, Oxford: Oxford University Press, 1979.

[80] M. Pastorino, Reconstruction algorithm for electromagnetic imaging, *IEEE Transactions on Instrumentation and Measurement*, vol.53, pp.692-699, 2004.

[81] P. Moscato, Memetic Algorithms, Home Page, *http* : *//www.densis.fee.unicamp.br /moscato/memetic − home.html*.

[82] L. J. Eshelman, The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, *In proc. Genetic Algorithms in Search, Optimisation, and Machine Learning*, 1989.

[83] S. W. Mahfoud and D. E. Goldberg, Parallel recombinative simulated annealing: A genetic algorithm, *Parallel computing*, vol.21, pp.1-28, 1995.

[84] F. T. Lin, C. Y. Kao and C. C. Hsu, Incorporating genetic algorithms into simulated annealing, *Proc. of the Fourth International Symposium on AI*, pp.290-297, 1991.

[85] H. Esbensen, A macro-cell global router based on two genetic algorithms, *In Proc. of the European Design Automation Conference*, pp.428-433, 1994.

[86] P. W. Blythe and G. Chamitoff, Estimation of aircrafts aerodynamic coefficients using recurrent neural networks. *In Proc. Second Pacific International Conference on Aerospace Science and Technology*, 1995.

[87] B. S. Kim and A. J. Calise, Nonlinear flight control using neural networks, *Journal of Guidance, Control, and Dynamics*, vol.20, no.1, pp.26-33, 1998.

[88] K. Rokhsaz and J. E. Steck, Use of neural networks in control of High-Alpha maneuvers, *Journal of Guidance, Control, and Dynamics*, vol.16, no.5, pp.934-939, 1993.

[89] P. D. Bruce and M. G. Kellet, Modeling and identification of nonlinear aerodynamic functions using B-splines, *In Proc. of Institution of Mechanical Engineers*, vol.2, no.14, Part G, pp.27-40, 2000.

[90] S. M. Ahmad, M. H. Shaheed, A. J. Chipperfield and M. O. Tokhi, Nonlinear modeling of a twin rotor MIMO system using radial basis function networks, *In Proc. of the 2000 IEEE International Conference on National Aerospace and Electronics*, pp.313-320, 2000.

[91] S. M. Ahmad, A. J. Chipperfield and M. O. Tokhi, Dynamic modeling and optimal control of a twin rotor MIMO System, *In Proc. of the 2000 IEEE International Conference on National Aerospace and Electronics*, pp.391-398, 2000.

[92] M. H. Shaheed, Performance analysis of 4 types of conjugate gradient algorithm in the nonlinear dynamic modeling of a TRMS using feedforward neural networks, *In Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, pp.5985-5990, 2004.

[93] F. M. Aldebrez, I. Z. M. Darus and M. O. Tokhi, Dynamic modeling of a twin rotor system in hovering position, *In Proc. of the 2004 International Symposium on Control, Communications and Signal Processing*, pp.823-826, 2004.

[94] I. Z. M. Darus, F. M. Aldebrez and M. O. Tokhi, Parametric modeling of a twin rotor system using genetic algorithms, *In Proc. of the 2004 International Symposium on Control, Communications and Signal Processing*, pp.115-118, 2004.

[95] M. H. Shaheed, Nonlinear Dynamic Modeling of a TRMS using RPROP Algorithm with Feedforward Neural Networks, *10th IEEE International Conference on Methods and Models in Automation and Robotics*, 30 August- 2 September, Poland, pp.1303-1308, 2004.

[96] A. Rahideh, M. H. Shaheed and H. J. C. Huijberts, Dynamic Modeling of a TRMS using Analytical and Empirical Approaches, *Control Engineering Practice*, vol.3, no.16, pp.241-259, 2009.

[97] S. A. Billings and W. S. F. Voon, Correlation based validity tests for nonlinear models, *International Journal of Control*, vol.44, no.1, pp.235-244, 1986.

[98] Qadeer Ahmed, *Robust control algorithms for twin Rotor system*,A Thesis Master of Science in Electronic Engineering, Faculty of Engineering and Applied Sciences Muhammad Ali Jinnah University, Islamabad, 2009.

[99] H. Hujiberts, H. Nijmeijer and R. willems, System identification in communication with chaotic Systems, *IEEE Trans on Circuits and Systems-I*, vol.47, no.6, pp.800-808, 2000.

[100] M. Adjrad and A. Belouchrani, Estimation of multi-component polynomial phase signals impinging on a multisenser array using state-space modeling, *IEEE Trans. Signal Processing*. vol.55, no.1, pp.32-45, 2007.

[101] K. Watanbe, I. Matsuura, M. Abe, M. Kebota and D. M. Himelblau, Incipient Fault Diagnosis of Chemical Processes via Artificial Neural Networks, *AICHE Journal*, vol.35, no.11, pp.1803-1812.

[102] Y. Xie, B. Guo, L. Xu, J. Li and P. Stoica, Multistatic adaptive microwave imaging for early breast cancer detection, *IEEE Trans. Biomedical Eng.*, vol.53, no.8, pp.1647-1657, 2006.

[103] M. Schetzmen, *The Voltera and Winner Theories on Nonlinear Systems*, Newyork, Wiley.

[104] F. Dinga and T. Chenb, Identification of Hammerstein nonlinear ARMAX systems, *Automatica*. vol.41, pp.1479-1489, 2005.

[105] E. Hernandez and Y. Arkun, Control of nonlinear systems using Polynomial ARMA models, *AICHE Journal*, vol.39, no.3, pp.446-460, 1993.

[106] T. T. Lee and J. T. Jeng, The Chebyshev polynomial-based unified model neural networks for functional approximation, *IEEE Trans. Syst., Man Cybern.-B*, vol.28, pp.925-935, 1998.

[107] N. Sadegh, A perceptron based neural network for identification and control of nonlinear systems, *IEEE Trans. Neural Networks*, vol.4, pp.982-988, 1993.

[108] K. H. Chon and R. J. Cohen, Linear and nonlinear ARMA model parameter estimation using an artificial neural network, *IEEE Trans. Biomed. En.* vol.44, pp.168-174, 1997.

[109] X. N. Stef and J. R. Simons, Nonlinear dynamic system identification using radial basis function networks, *In Proc. IEEE on Decision and Control*, Kobe, Japan, pp.935-936, 1996.

[110] Q. Zhang, Using wavelet network in nonparametric estimation, *IEEE Trans. Neural Networks*, vol.8, pp.227-236, 1997.

[111] R. Stron, System design by constraint adaptation and differential evolution, *IEEE Trans. Evol. Comput.*, vol.3, pp.22-34, 1999.

[112] E. Goldberg and J. Richardson, Genetic llgorithms with sharing for multimodal function optimization, *In Grefenstette, editor, Genetic Algorithms and their Applications (ICGA87)*, pp.41-49, 1987.

[113] M. L. Davis, Training feedforward neural networks using genetic algorithms, *In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, pp.762-767, 1989.

[114] E. Goldberg, Genetic algorithms and Walsh functions: Part 2, Deception and its analysis, *Complex Systems*, vol.3, pp.153-171, 1989.

[115] K. Kristinsson and G. A. Dumont, System identification and control using genetic algorithms, *IEEE Trans System Man and Cybernetics*, vol.22, pp.1033-1046, 1992.

[116] H. R. Tizhoosh, Opposition-based learning: A new scheme for machine intelligence, *In Proc. Int. Conf. Comput. Intell. Modeling Controland Autom.*, Vienna, Austria, vol.1, pp.695-701, 2005.

[117] M. Shokri, H. R. Tizhoosh and M. Kamel, Opposition-based $Q - \lambda$ algorithm, *In Proc. IEEE World Congr. Comput. Intell.*, Vancouver,BC, Canada, pp. 646-653, 2006.

[118] M. Ventresca and H. R. Tizhoosh, Improving the convergence of backpropagation by opposite transfer functions, *In Proc. EEE WorldCongr. Comput. Intell.*, Vancouver, BC, Canada, pp.9527-9534, 2006.

[119] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, Opposition versus randomness in soft computing techniques, *Elsevier J. AppliedSoft Comput.*, vol.8, no.2, pp.906-918, 2008.

[120] H. Tizhoosh and M. Ventresca, Oppositional Concepts in Computational Intelligence, *Studies in Computational Intelligence*, Springer-Verlag, 2008.

[121] M. Ventresca and H. R. Tizhoosh, Numerical condition of feed-forward networks with opposite transfer Functions, *International Joint Conference on Neural Networks (IJCNN)*, pp.3232-3239, Hong Kong, China, 2008.

[122] M. Ventresca and H. R. Tizhoosh, Simulated annealing with opposite neighbors, I*EEE Symposium on Foundations of Computational Intelligence*, pp.186-192, Honolulu, USA, 2007.

[123] M. Ventresca and H. R. Tizhoosh, Opposite transfer functions and backpropagation through time, *IEEE Symposium on Foundations of Computational Intelligence*, pp.570-577, Honolulu, USA, 2007.

[124] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, Opposition-based differential evolution, *IEEE Trans. Evolutionary Computation*, vol.12, no.1, pp.64-79, 2008.

[125] J.-S. R. Jang, ANFIS: Adaptive network based fuzzy inference systems, I*EEE Transactions on Systems, Man, and Cybernetics*, vol.23, no.3, pp.665-685, 1993.

[126] J.-S. R. Jang, C.-T. Sun and E. Mizutani, *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Eastern Economy Edition, New Delhi: PHI, 2003.

[127] D. J. Atkinson, P. P. Acarnley and J. W. Finch., Estimation of rotor resistance induction motor, *In Proc. IEE Elect Power Appl.* vol.143, no.1, pp.87-94, 1996.

[128] M. Cirrincione, M. Pucci, G. Cirrincione and G. A. Capolino, A new experimental application of least-squares techniques for the estimation of the induction motor parameters, *IEEE Trans. Ind. Appl.*, vol.39, no.5, pp.1247-1256, 2003.

[129] Y. Koubaa, Induction machine drive parameters estimation, *In: CD-ROM of the IEEE International Conference on Systems, Man and Cybernetics (SMC02)*, Hammamet, Tunisia 2002,

[130] S. Moonl and A. Keyhani, Estimation of induction machine parameters from standstill time-domain data, *IEEE Trans. Ind. Appl.*, vol.30, no.6, pp.1609-1615, 1994.

[131] A. Ba-Razzouk, A. Cheriti and P. Sicard, Implementation of a DSP based real-time estimator of induction motors rotor time constant, *IEEE Trans. Power Electron*, vol.17, no.4, pp.534-542, 2002.

[132] J. Stephan, M. Bodson and J. Chiasson, Real-time estimation of the parameters and fluxes of induction motors, *IEEE Trans. Ind. Appl.*, vol.30, no.3 pp.746-759,1994.

[133] B. A. Carli and M. L. Cava, Parameter identification for induction motor simulation, *Automatica*, vol.12, no.4, pp.383-386, 1976.

[134] J. Holtz and T. Thimm, Identification of the machine parameters in a vector-controlled induction motor drives, I*EEE Trans. Ind. Appl.*, vol.27, pp.1111-1118, 1991.

[135] Y. Xing, M. W. Dunnigan and B. W. Williams, A novel rotor resistance identification method for an indirect rotor flux-orientated controlled induction machine system, *IEEE Trans. Power Electron.*, vol.17, no.3, pp.353-364, 2002.

[136] D. J. Atkinson, P. P. Acarnley and J. W. Finch, Observers for induction motor state and parameter estimation, *IEEE Trans. Ind. Appl.*, vol.27, no.6, pp.1119-1127, 1991.

[137] A. Garc-ia-Cerrada and J. L. Zamora, On-line rotor-resistance estimation for induction motors, *In Proc. EPE97*, vol.1, pp.542-547, Trondheim, Norway, 1997.

[138] Y. Koubaa, Parametric identification of induction motor with HG diagram, *In International Conference on Electrical Drives and Power Electronics*, pp.433-437, High Tatras, Slovak Republic, 2001.

[139] L. A. Ribeiro, C. B. Jacobina and A. M. N. Lima, Linear parameter estimation for induction machines considering the operating conditions, *IEEE Trans. Power Electron.*, vol.14, no.1, pp.62-73, 1999.

[140] L. A. Cabrera, M. E. Elbuluk and I. Husain, Tuning the stator resistance of induction motors using artificial neural network, *IEEE Trans. Power Electron.*, vol.12, no.5, pp.779-787, 1997.

[141] K. B. Bimal and R. P. Nitin, Quasi-fuzzy estimation of stator resistance of induction machines, *IEEE Trans. Power Electron.*, vol.13, no.3, pp.401-409, 1988.

[142] Z. L. Cheng, C. L. DeMarco and T. A. Lipo, An extended Kalman filter approach to rotor time constant measurement in PWM induction motor drives, *IEEE Trans. Ind. Appl.*, vol.28, no.1, pp.96-104, 1992.

[143] E. D. Mitronikas, A. N. Safacas and E. C. Tatakis, A new stator resistance tuning method for stator-flux oriented vector controlled induction motor drive, *IEEE Trans. Ind. Electron.*, vol.48, no.6, pp.1148-1157, 2001.

[144] J. Ha and H. L. Sang, An on-line identification method for both stator and rotor resistances of induction motors without rotational transducers, *IEEE Trans. Ind. Electron.*, vol.47, no.4, pp.842-853, 2000.

[145] S. H. Jeon, K. K. Oh and J. Y. Choi, Flux observer with online tuning of stator and rotor resistances for induction motors, *IEEE Trans. Ind. Electron.*, vol.49, no.3, pp.653-664, 2002.

[146] Y. Koubaa and M. Boussak, Adaptive rotor resistance identification for indirect stator flux oriented induction motor drive, *In CD-ROM of the Second International Conference on Signals, Systems Decision and Information Technology (SSD03)*, Sousse, Tunisia, 2003.

[147] A. B. Proca and A. Keyhani, Identification of variable frequency induction motor models from operating data, *IEEE Trans. Energy Convers.*, vol.17, no.1, pp.24-31, 2002.

[148] F. Alonge, F. D. Ippolito and F. M. Raimondi, Least squares and genetic algorithms for parameter identification of induction motors, *Control Eng. Pract.*, vol.9, no.6, pp.647-657, 2001.

[149] F. Alonge, F. D. Ippolito, G. Ferrante and F. M. Raimondi, Parameter identification of induction motor model using genetic algorithms, *In Proc. Control Theory Application*, vol.145, no.6, pp.587-593, 1998.

[150] K. S. Huang, W. Kent, Q. H. Wu and D. R. Turner, Parameter identification for FOC induction motors using genetic algorithms with improved mathematical model, *Electr. Power Components Syst.*, vol.29, no.3, pp.247-258, 2001.

[151] P. Ju and E. Handschin, Parameter estimation of composite induction motor loads using genetic algorithms, *In Proc. Stockholm Power Technology International Symposium on Electric Power Engineering*, vol.3, pp.97-102, 1992.

[152] T. Haque, R. Nolan, P. Pillay and J. Reynaud, Parameter determination for induction motors, *In Proc. IEEE SOUTHEASTCON94*, pp.45-49, 1994.

[153] P. Nangsue, P. Pillay and S. Conry, Evolutionary algorithms for induction motor parameter determination, *IEEE Trans. Energy Conversion*, vol.14, no.3, pp.447-453, 1999.

[154] R. K. Ursem and P. Vadstrup, Parameter identification of induction motors using stochastic optimization algorithms, *Applied Soft Computing*, vol.4, no.1, pp.49-64, 2004.

[155] R. K. Ursem and P. Vadstrup, Parameter identification of induction motors using differential evolution, *In Proc. of the 2003 Congress on Evolutionary Computation CEC2003*, pp.790-796, 8-12 December, Canberra, 2003.

[156] E. Merchan-Cruz and A. S. Morris, Fuzzy GA based trajectory planner for robot manipulators sharing a common workspace, *IEEE Trans. Rob. and Autom.*, vol.22, no.4, pp.613-624, 2006.

[157] B. K. Panigrahi, S. Agrawal and M. K. Tiwari, Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch, *IEEE Trans. Evolutionary Comput.*, vol.12, no.5, pp.529-541, 2008.

[158] M. N. H. Siddique and B. P. Amavasai, Bio-inspired behavior-based control, Artificial intelligence review, *Special Issue on Hybrid Systems in AI*, Springer Journal, vol.27, no.2-3, pp.131-147, 2008.

[159] S. Das and A. Konar, A swarm intelligence approach to the synthesis of two-dimensional IIR filters, *Engineering Applications of Artificial Intelligence*, vol.20, no.8, pp.1023-1062, 2007

[160] J. M. Martins, Z. Mohamed, M. O. Tokhi, J. Costa and M. A. Botto, Approaches for dynamic modeling of flexible manipulator systems, *In Proc. of IEE-Control Theory and Applications*, vol.150, no.4, pp.401-411, 2003.

[161] W. Lin and P. X. Liu, Hammerstein model identification based on bacterial foraging, *Electronics Letters*, vol.42, no.23, pp.1332-1333, 2006.

[162] W. Kinnebrock, Accelerating the standard backpropagation method using a genetic approach, *Neurocomput.*, vol.6, no.56, pp.583-588, 1994.

[163] I. Erkmen and A. Ozdogan, Short term load forecasting using genetically optimized neural network cascaded with a modified Kohonen clustering process, *In Proc. 1997 IEEE Int. Symp. Intelligent Control*, pp.107-112.

[164] S. K. Aditya and D. Das, Design of load frequency controllers using genetic algorithm for two area interconnected hydro power system, *International Journal of Electric Power Components and Systems*, vol.31, pp.81-94, 2003.

[165] Y. R. Sood, Evolutionary programming based optimal power flow and its validation for deregulated power system analysis, *International Journal of Electrical Power and Energy Systems*, vol.29, no.1, pp.65-75, 2007.

[166] J. C. Wang, H. -D. Chiang, C. -T. Huang and Y. -T. Chen, Identification of excitation system models based on on-line digital measurents, *IEEE Transactions on Power Systems*, vol.10, no.3, pp.1286-1293, 1995.

[167] M. F. Azeem, M. Hanmandlu and N. Ahmad, Structure identification of generalized adaptive neuro-fuzzy inference system, *IEEE Transactions on Fuzzy Systems*, vol.2, no.5, pp.666-681, 2003.

[168] B. Subudhi and D. Jena, Differential Evolution and Levenberg Marquardt Trained Neural Network Scheme for Nonlinear System Identification, *Neural Processing Letters, Springer*, vol.27, no.3, pp.285-296, 2008.

[169] B. Subudhi and D. Jena, M. M. Gupta, Memetic Differential Evolution Trained Neural Networks for Nonlinear System Identification, *In Proc. of IEEE-Intrnational Conference on Industrial and Information Systems ICIIS-08*, IIT, Kharagpur, 8-10 Dec, 2008.

[170] B. Subudhi, D. Jena, A differential evolution based neural network approach to nonlinear system identification, *Applied Soft Computing Elsevier*, doi:10.1016/j.asoc.2010.01.006, 2010.

Publications

In Journals

[1] B. Subudhi and D. Jena, Differential Evolution and Levenberg Marquardt Trained Neural Network Scheme for Nonlinear System Identification, *Neural Processing Letters, Springer*, vol.27, no.3, pp.285-296, 2008.

[2] B. Subudhi and D. Jena, An Improved Differential Evolution Trained Neural Network Scheme for Nonlinear System Identification, *Intl. Journal of Automation and Computing, Springer*, vol.6, no.2, pp.137-144, 2009.

[3] B. Subudhi and D. Jena, Nonlinear System Identification using Opposition Based Learning Differential Evolution and Neural Network Techniques, *IEEE Intelligent Cybernatic Systems Journal*, vol.5, no.1, pp.1-19, 2009.

[4] B. Subudhi and D. Jena, Differential Evolution Applied to Parameter Estimation of Induction Motor, *Archives of Control Sciences*, vol.19, no.1, pp.59-80, 2009.

[5] B. Subudhi, D. Jena, A differential evolution based neural network approach to nonlinear system identification, *Applied Soft Computing Elsevier*, doi:10.1016/j.asoc.2010.01.006, 2010.

In Conferences

[1] B. Subudhi, D. Jena, An Improved Differential Evolution trained Neural Network Scheme for Nonlinear System Identification, *In Proc. National System Conference*, Manipal, 14-15 Dec, 2007.

[2] B. Subudhi and D. Jena, A Combined Differential Evolution and Neural Network Approach to Nonlinear System Identification, *In Proc. of Intl. Conf. IEEE-TENCON*, University of Hyderabad, 19-21 Nov, 2008.

[3] B. Subudhi and D. Jena, M. M. Gupta, Memetic Differential Evolution Trained Neural Networks for Nonlinear System Identification, *In Proc. of IEEE-Intrnational Conference on Industrial and Information Systems ICIIS-08*, IIT, Kharagpur, 8-10 Dec, 2008.

[4] B. Subudhi and D. Jena, Nonlinear System Identification of a Twin Rotor System using Memetic Algorithm trained Neural Netwroks, *In Proc. of Intl. Conf. IEEE-TENCON*, Singapore, 23-26 Nov, 2009.

Debashisha Jena

Asst. Professor
Department of Electrical and Electronics Engineering
National Institute of Technology Karnataka, Surathkal
Mangalore – 575025, India.

Ph: +91-824-2474000 Extn-3457

e-mail: bapu4002@gmail.com

Qualification

- Ph.D. (Continuing)
  NIT Rourkela
- M.Tech. (Power System)
  Biju Patnaik University of Technology [First division]
- B.E (Electrical)
  Sambalpur University, Orissa [First division]
- +2 (Science)
  Council of Higher Secondary Education, Orissa, [First division]
- 10th
  Board of Secondary Education, Orissa, [First division]

Publications

- 05 Journal Articles
- 04 Conference Papers

Permanent Address

Indira Nagar, Industrial Estate,
Berhampur – 760008, Orissa, India.

Date of Birth

June 13, 1973