Hybrid algorithm for NARX network parameters' determination using differential evolution and genetic algorithm

To cite this article: M J E Salami *et al* 2013 *IOP Conf. Ser.: Mater. Sci. Eng.* **53** 012062

Hybrid algorithm for NARX network parameters' determination using differential evolution and genetic algorithm

M J E Salami[1,4], I B Tijani[1, 2], A I Abdullateef[1] and M A Aibinu[3].
[1] Department of Mechatronics Engineering, Faculty of Engineering, IIUM, Malaysia
[2] Electronics Eng. Dept., ADMC, HCT, Abu Dhabi, UAE
[3] Federal University of Technology, Minna, Nigeria

E-mail: momoh@iium.edu.my

Abstract. A hybrid optimization algorithm using Differential Evolution (DE) and Genetic Algorithm (GA) is proposed in this study to address the problem of network parameters determination associated with the Nonlinear Autoregressive with eXogenous inputs Network (NARX-network). The proposed algorithm involves a two level optimization scheme to search for both optimal network architecture and weights. The DE at the upper level is formulated as combinatorial optimization to search for the network architecture while the associated network weights that minimize the prediction error is provided by the GA at the lower level. The performance of the algorithm is evaluated on identification of a laboratory rotary motion system. The system identification results show the effectiveness of the proposed algorithm for nonparametric model development.

1. Introduction

Non parametric modeling approach based Nonlinear Autoregressive with eXogenous inputs Network (NARX-network) has been a successful technique for time series prediction, filtering, system modeling and identification in almost all fields of studies, [1], [2]. However, one of the major challenges in the application of this NARX network is in the choice of appropriate network architecture in terms of number of input and output tapped delays, TD, and number of hidden neurons (*H*) for a single hidden layer network, [3]. The decision has been based majorly on iterative experimentation (trial and error basis). This problem has initiated the use of evolutionary optimization algorithm in neural network design for optimal/sub-optimal performances [3], [4], and by extension to NARX network. Many research works have been reported on network (MLN) architecture optimization using simulation annealing and tabu search, [5], using GA, [4], and DE, [6],and recently a multi-objective Differential Evolution (MODE) was proposed for network architecture optimization, [7]

In this paper, a hybrid optimization algorithm involving DE and GA is proposed to determine optimal NARX network architectures and weight. The motivation is to explore the optimization strength of both DE and GA to search simultaneously for both model architecture and weight.

The rest of the paper is organized as follows. A brief overview of both NARX network is given in Section 2 followed by the basic description of DE and GA algorithms in Section 3. The proposed

[4] Prof. Momoh Jimoh Eyiomika Salami

hybrid algorithm is presented in Section 4. The application to system identification of a practical system is reported in Section 5 while the results and discussion are given in Section 6. The paper is concluded in Section 7.

2. NARX network overview

NARX network generally describes a discrete nonlinear system as linear combination of its past output values as well as those of the input, [8]:

$$\hat{y}(k) = f\left[y(k-1), y(k-2), \ldots, y(k-n+1);\ \delta(k), \delta(k-1), \ldots, \delta(k-m+1)\right] \quad (1)$$

where $\delta(k)$ and y(k) represent input and output of the system as time step k, n and m are the input and output orders respectively, while the function $f[\bullet]$ is a nonlinear mapping function which is given by the Multilayer Perceptron (MLP) network.

As shown in Figure 1, the input-output mapping implemented with two layers MLP is expressed as:

$$\hat{y}(k) = \Gamma(u(k), y(k)) = \Gamma\left[[W^L \Pi[W^I(\delta^*, y^*)]\right] \quad (2)$$

where $\Gamma(\bullet)$ is mapping function implemented by the MLP, $\Pi(\bullet)$ is nonlinear activation function usually a sigmoid function presents in both hidden and output layers, W^I and W^L represent the connection weights for input and hidden layers respectively.

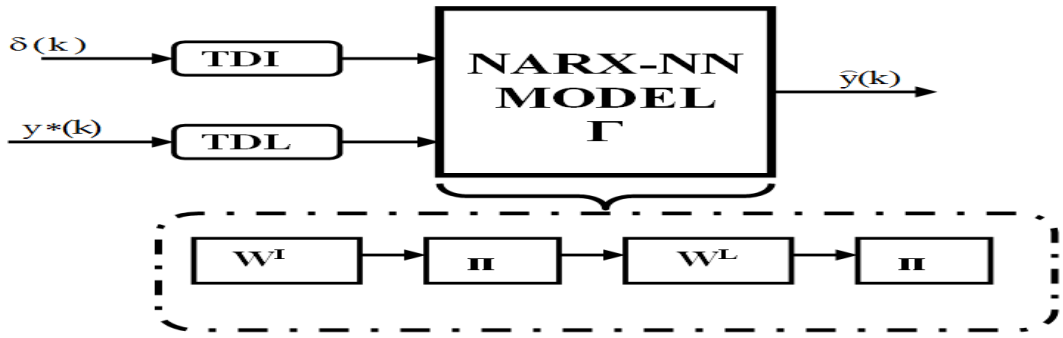


Figure 1. Architecture of series-parallel NARX-Network, [7]

A general configuration of NARX network for multi-inputs and multi-outputs (MIMO) systems with two inputs, two outputs and one hidden layer with three nodes together with biases is shown in Figure 2.

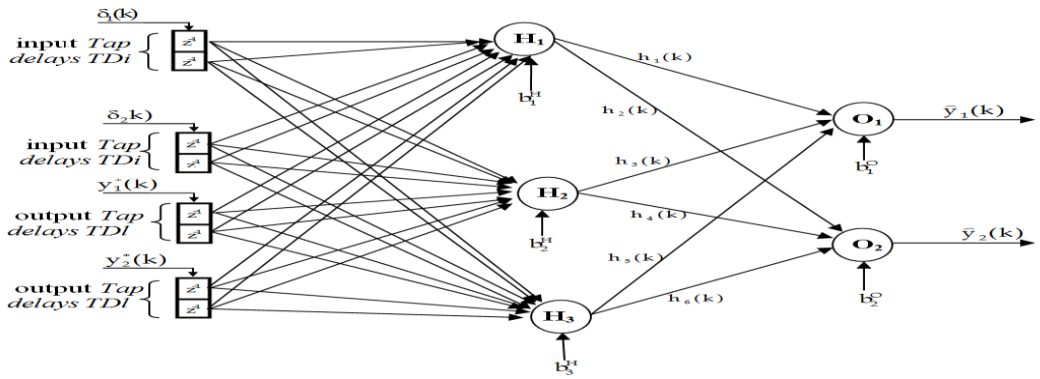


Figure 2 A three layer, two inputs, two outputs NARX network [7]

Given *n* as number of inputs, *m* as number of outputs with O nodes, assuming a single layer network with H hidden nodes, the network parameters consist of: (i) input states $\delta = [\delta_1, \delta_2, \ldots, \delta_\Phi]$ where Φ is total number of the states given as:

$$\Phi = H[(n * TDI) + (m * TDL)] \tag{3}$$

TDI and TDL are number of input and out time delays (z^{-1}), respectively.

(ii) input biases, $b^H = [b_1, b_2, \ldots, b_H]$; (iii) input weights W^I:connection weight between input state, j and hidden layer nodes, i, expressed as $w_{i,j}$. Note that total number of this weights is equal to Φ; (iv) layer weights, W^L: connection weight between the hidden layer nodes i and output nodes κ expressed as $w_{\kappa,i}$. The total of this connection weight is given by $H * O$; and (v) output biases $b^O = [b_1, b_2, \ldots, b_O]$.

Thus, the total network parameters N_p to be adjusted during identification process is given as sum of total input connection weights, input biases, total hidden layer connection weight and output biases:

$$N_p = \Phi + H * O + b^H + b^O \tag{4}$$

The output of the hidden layer is given by:

$$h_i = \Pi\left[\sum_{j=1}^{N} w_{i,j} x_j + b_i^H\right] \tag{5}$$

while the estimated model output is:

$$y_\kappa = \Pi\left[\sum_{i=1}^{H} w_{\kappa,i} h_i + b_i^O\right] \tag{6}$$

The network learning process generally involves determination of the network parameters W^I and W^L that minimizes the prediction error between the actual and predicted output given as:

$$V_N = \frac{1}{2N}\sum_{k=1}^{N} \varepsilon^T \varepsilon \tag{7}$$

where ε is prediction error expressed as $\varepsilon = y(k) - \hat{y}(k)$

Conventionally, the training process is carried out with back propagation learning algorithm with Levenberg-Marquardt learning function [10].

3. Overview of GA and DE algorithm

GA and DE are among the prominent evolutionary optimization algorithms with vast practical applications as reported in the literatures, [10], [11]. The GA was originally proposed by John Holland, [12], and represents the pioneer evolutionary approach to practically optimization problem. It mimics biological process of natural selection mechanism in which stronger individuals have higher probability to win in a competing environment. DE on the other hand, was latter proposed by Storn and Price, [13]. Unlike GA which uses genetic operator in evolving new solutions/generation, DE uses differential operator to evolve new solutions/generation. Also, DE uses real values to represent the

solution candidates while GA originally represents the parameters in form of chromosomes (a binary encoding of the actual parameters). Generally, both are population based algorithm and shared similar major concepts of: population initialization, crossover, mutation and selection, [14].

Initialization involves generation of initial population using a random process for the optimization process. In GA, the individuals are encoded as a string of bits (0 & 1) using the process of encoding. While in DE the random generated real parameters values are used directly.

Mutation and crossover are two consecutive operators for generation of new vector known as child/trial vector. The process begins by selecting two individuals from the current generation, then, crossover and mutation operators are applied to generate child vector. In GA, a crossover point is randomly set and the portions of the two genes beyond this set point are swapped to form new vector (child). This process is controlled by a pre-specified probability rate known as crossover rate usually between 0.6 and 1.0. Then, mutation is applied to each child string to alter each bit randomly with a probability determined by the pre specified mutation rate which is usually between 0.1 and 0.001, [14].

The DE mutation comes first and involves addition of weighted differential of two or more randomly selected vectors to another randomly selected based vector. Though there are several strategies of computing the weighted differential, the basic and common one is "DE/rand/1/bin", [13]. Then a recombination process known as crossover is used to alter randomly the new vector. It ensures that each parameter of the differential mutant vector is accepted into the trial vector with some probability know as crossover constant which is usually between 0 and 1. It determines the fraction of parameter values that are copied from the mutant into the trial or child vector.

Selection operator generally involves comparison of the objective values known as fitness function of both parent and child vector to determine who survives in the next generation

Once the new population is generated, the process of mutation, crossover (recombination) and selection is repeated until the pre-specified termination criterion is satisfied. Based on this routine, the general flowchart of the DE and GA algorithms are shown in Figure 3 (a) and Figure 3(b), respectively.

4. Proposed hybrid DE-GA algorithm

The proposed hybrid algorithm explores the optimization strength of both DE and GA sub optimization algorithms in determining the optimal parameters of NARX network. The algorithm comprises two level optimization processes. The first level (upper level) comprises the DE sub algorithm which searches for the optimal NARX network architecture, while the second level involves GA sub algorithm searches for the optimal network weights that minimizes a given objective function. The flowchart of the proposed algorithm is shown in Figure 4. As shown in Figure 4, the GA sub algorithm is integrated into the DE algorithm to compute the overall objective function.

The problem is formulated as follows: given a set of input and output data pairs (*u(k),y(k)*) of a system to be modeled, the objective of the DE upper level optimization is to search for set of network architecture parameters, φ,

$$\varphi := [TD, H] \tag{8}$$

that minimizes the objective function, $f(\varphi)$ as follows:

$$\min f(\varphi) = 0.5*(\Gamma(W) + p_c) \tag{9}$$

where p_c is the percentage of network connection used, and $\Gamma(W)$ is the objective function values returns by the GA sub-algorithm.

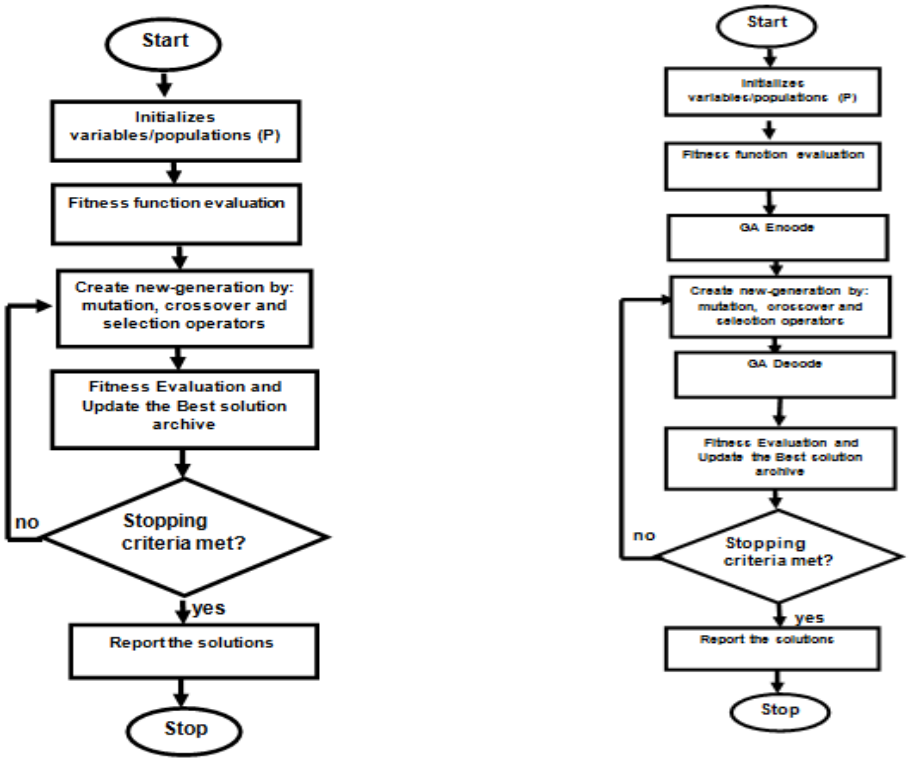


Figure 3 Flowchart of (a) DE algorithm, (b) GA Algorithm

The GA sub-algorithm searches for a set of network weights, W, which minimizes the objective function, $\Gamma(W)$, where $\Gamma(W)$ is defined as prediction error between the actual response and predicted response for a given data, length, *N:*

$$\min \Gamma(W) = \frac{1}{2N}\sum_{k=1}^{N}\varepsilon^T\varepsilon \qquad (10)$$

where $\varepsilon = y(k) - \hat{y}(k)$

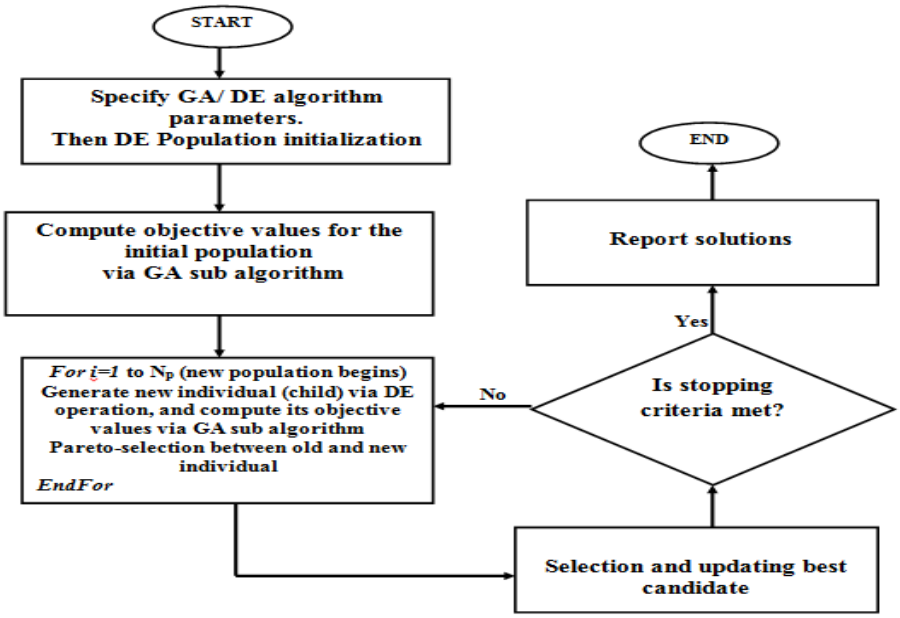


Figure 4 Flowchart of the proposed hybrid DE-GA Optimization algorithm

The major algorithm procedures are summarized as follows:

Step 1: Specifying the optimization parameters for both DE and GA sub algorithms: DE parameters are: crossover constant, *CR*, mutation Constant, *F*, population size, β_{DE}, and maximum generation g_{max}^{DE} while GA main parameters are: probability crossover, p_c , mutation rate, m_c, population size, β_{GA} , and maximum generation size, g_{max}^{GA}

Step 2: Initialize the DE population, P_o, of size, β_{DE}, for $g^{DE}=0$. The population comprises of individuals, $\varphi_i^g \ \forall i, i=1,...,\beta_{DE}$, where $g=g^{DE}$ and . Given a specified lower, *L*, and upper, *U*, bound on the network parameters, an individual of the population is initialized using:

$$\varphi_j^g = randint(1,1,[L,U]) \tag{11}$$

where "*randint*" is a function that generates uniformly distributed random integers, within the range specified by the bounds *L* and *U*.

Step 3: Compute vector of objective values, *Fit*, for all the individuals in the population. This is achieved by evaluating the objective function given in equation (1), hence,

$$Fit_i = \min f(\varphi_i^g) = 0.5*(\Gamma(W_i)+p_{ci}) \quad \forall i, i=1,...,\beta_{DE} \tag{12}$$

where $f(\varphi_i^g)$ is the objective functions for a given parameter vector φ_i^g .

As earlier stated, the objective functions expressed in equation (5) comprises the accuracy and complexity of the NARX network. The accuracy is specified by the sub-objective values, $\Gamma(W)$ which is the optimal prediction error returns by the GA sub-algorithm. The GA sub-algorithm is called by the DE sub-algorithm at this point to search for optimal network weight, *W* that minimizes $\Gamma(W)$.

For a given network architecture, φ produced by DE sub-algorithm, GA sub algorithm is called to search for corresponding optimal weights that minimizes prediction error given in equation (3). A nested MATLAB function (m-files) incorporating the MATLAB inbuilt GA algorithm, [15] is developed to integrate the GA into the DE process. The total weight for the given, φ is computed using equation (3), and this with other pre-specified parameters as stated in step 1 are passed into the GA sub algorithm.

Step 4: Begins DE optimization process

while g< g_{max}^{DE}

Step 5: Starts DE operation to generate new population, P_{new} and update best candidate

For *i*= 1 to β_{DE}

1. Using the DE process of mutation and cross-over, generate new candidate, v_i^{g+1} called child vector from a randomly selected three individuals, r_1, r_2, r_3, from the previous population such that $r_1, r_2, r_3 \in [1; N_p]$, *where* $r_1 \neq r_2 \neq r_3 \neq i$ as follows:

i. Mutation: compute the mutant vector υ using DE operation*:*

$$\upsilon_i = \varphi_{r_3} + (\varphi_{r_2} - \varphi_{r_1}), \ F=1 \tag{13}$$

ii. Perform crossover to generate child vector, v_i^{g+1} using:

Generate a random number $Rand_j = floor(rand(0,1)*d)+1$

for j=1: d

$$v_{i,j}^{g+1} = \begin{cases} \upsilon_{i,j} & if\ (rand(0,1) \le CR\ \ or\ \ j = Rand_j \\ \varphi_{i,j}^{g} & otherwise \end{cases} \quad (14)$$

EndFor

2. Check for violation of the bounds *L* and *U*, and effect correction if needed

For j=1: d

If $v_{i,j}^{g+1} < L \,||\, v_{i,j}^{g+1} > U$

$v_{i,j}^{g+1} = randint(1,1,[L,U])$

EndIf

EndFor

3. Evaluate the objective function, Fit_i^{g+1} of the child vector, $f(v_i^{g+1})$ as described in step 3.

4. Selection process: let the generated child vector, v_i^{g+1} compete with the parent vector φ_i^g as follows

$$\varphi_i^{g+1} = \begin{cases} v_i^g : if\ \ v_i^{g+1}\ wins\ \varphi_i^g \\ \varphi_i^g\ : if\ \ v_i^{g+1}\ not\ wins\ \varphi_i^g \end{cases} \quad (27)$$

where: $v_i^{g+1}\ wins\ \varphi_i^g \quad if\ only\ and\ if\ \ : f(v_i^g) \le f(\varphi_i^g)$ (15)

Update new population and Best candidate index:

$P_{new}(i) = \varphi_i^{g+1}$

$iBest_{new} = i : if\ Fit(iBest_{new}) <= Fit(iBest_{old})$

EndFor

Step 6: set *g=g+1* , and go to Step 4

EndWhileLoop

Step 7: Report the solutions: Best Fit and Best parameters set: φ and *W*

This hybrid algorithm, subsequently refers to as DE-GA-NARX network algorithm, is written in MATLAB which facilitate the integration of the MATLAB inbuilt GA and NARX network algorithm in the hybrid optimization algorithm process.

5. Application to system identification

The hybrid algorithm is used to identify model of a laboratory rotary motion control system shown in Figure 5. As shown in Figure 5, the DC motor driven rotary motion system represents a typical practical problem in system identification. It consists of servo motor driven by an amplifier and position encoder attached to the shaft as the feedback sensor. Although, an approximate lump mathematical model based on the physical laws of motion and Kirchhoff's laws could be derived for the system as reported in [16], the actual model order and parameters of the complete system components are not always known a priori. In the absence of actual system model detailed physical parameters, the NARX network technique provides an effective approach for the system nonlinear model identification.

The system is excited with Pseudo Random Binary Signal (PBRS) to obtain input-output data pairs requires for system identification process. Given this input and output data pair, the proposed algorithm is applied to identify NARX network parameters the optimally modeled the system.

Two set of data were collected for training with data size, *NT*=503 and validation with data size, *NV*=120. Sample of the input and measured output data is shown in Figure 6.

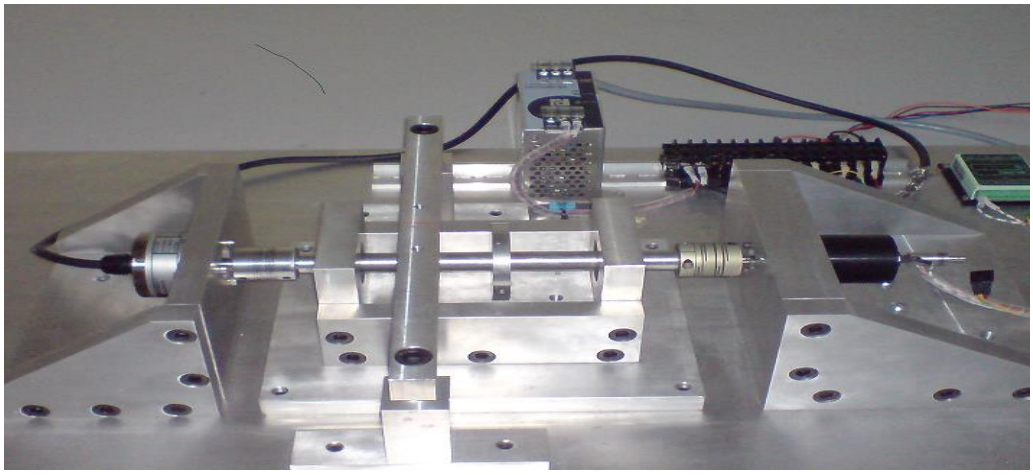


Figure 5 DC-Motor driven rotary motion systems [16].

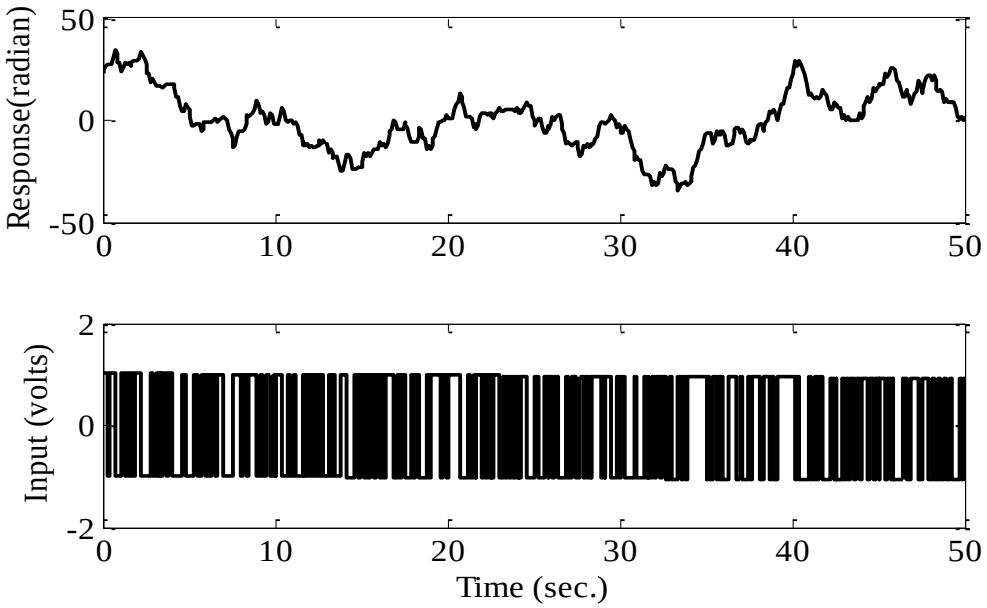
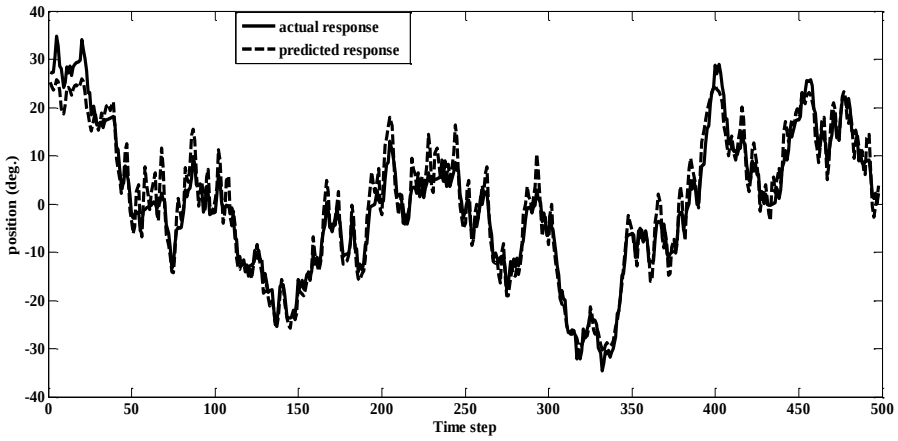


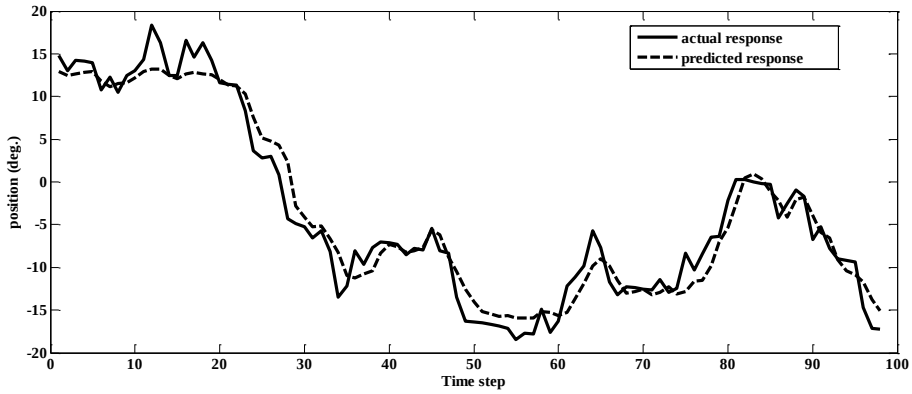
Figure 6 Experimental input and output data pair

6. Results and discussion

The primary optimization parameters are: population size, β_{DE}=30; β_{GA}=20; maximum generation size, g_{max}^{DE}=100, g_{max}^{GA}=20; maximum network size, TD=10, H=10; DE mutation factor, 0.75 and cross over constant, 0.25; GA mutation rate, 0.01, probability cross over, 0.8. Figure 7 shows the comparison of the actual and predicted output with validated data. The results of the optimal network produced by the algorithm are given in Table 1. As shown in Figure 7, and Table 1, the prediction error is considerable minimized for both training and validation data, and yet provides network with less than 50% (17 size) full connection.



(a)



(b)

Figure 7. Measures and predicted plant responses with (a) training and (b) validation data

Table 1: Results of Optimized NARX Network for the plant model

Network size	**% connection**	**Prediction error** [1] **Training data**	**Prediction error Validation data**
[3,3,2]	7.69	0.0089	0.0135

7. Conclusion

This study has presented an hybrid algorithm for optimal determination of a NARX network parameters. The proposed hybrid algorithm explored the evolutionary optimization strength of both DE and GA to search for both the network architecture and weights which yield optimal model accuracy and least network complexity.

The algorithm is applied to identify model of a rotary motion control system, the results shows the effectiveness of the proposed algorithm in terms of model accuracy with average prediction error of 0.011, while utilizing only 50% of the full network size. Generally, the proposed hybrid algorithm provides automatic network architecture and parameters determination for nonlinear, nonparametric NARX network model. This is expected to reduce the developmental time involves in autonomous helicopter project.

References

[1] Vijaya K M P, Sampath S S, Omkar S N and Ranjan G 2011 J. **83** 283–295.
[2] José M P, Menezes Jr., Guilherme A B 2008 J. *Neurocomputing*, **71**, 3335–3343
[3] Teresa B, Ludermir AY, and Cleber Z 2006 *IEEE Transactions*, **17**
[4] Jinn-Tsong T, Jyh-Horng C, and Tung-Kuan L 2006 *IEEE Transaction on Neural Networks*
[5] Chia-Nan K, *Engineering Applications of Artificial Intelligence* **25** 533–543
[6] Subudhi B, Jena D 2008 *Neural Processing Letters*, **27** 285-296
[7] Tijani I B, Akmeliawati R, Legowo A, Budiyono A 2012 *Under review* (EAAI), Elsevier.
[8] Narendra K S and Parthasarathy K 1990 *IEEE Trans. on neural networks* **1** 4-27
[9] MATLAB, 2009a. Neural Network Toolbox, User Guide. Mathworks
[10] Yoel T, 2012 *Engineering Applications of Artificial Intelligence* **25** 1009–1021
[11] Xin-SheYang, 2010 Wiley, A John Wiley & Sons, Inc., Publication.
[12] Holland J, The University of Michigan Press, Ann Arbor, MI
[13] Storn R, and Price K, 1997 J 341–359 *Kluwer Academic Publishers*.
[14] Man K F, Tang K S, and Kwong S 1996 *IEEE Transactions On Industrial Elec*tronics **43**
[15] MATLAB, 2009a, User Guide's, Mathworks, 2009
[16] Tijani I B, and Akmeliawati R 2012 J. Elsevier **25**