A Hybrid Method Based on Conjugate Gradient Trained Neural Network and Differential Evolution for Non Linear systems Identification

Chiha Ibtissem
Laboratory of Automatic Image and Signal Processing (ATSI)
National School of Engineers of Monastir (ENIM), Monastir,Tunisia
chiha.ibtissem@yahoo.fr

Liouane Nouredine
Laboratory of Automatic Image and Signal Processing (ATSI)
National School of Engineers of Monastir (ENIM), Monastir,Tunisia
nourliouane@yahoo.fr

***Abstract*—A hybrid method based on Differential Evolution and Neural Network training algorithms is presented in this paper for improving the performance of neural network in the non linear system identification. For this purpose, the local optimization algorithm of conjugate gradients (CG) is combined with the differential evolution algorithm (DE), which is a population-based stochastic global search method, to yield a computationally efficient algorithm for training multilayer perceptron networks for nonlinear system identification. After, a series of simulation studies of our method on the different nonlinear systems it has been confirmed that the proposed CG+DE algorithm has yielded better identification results in terms of time of convergence and less identification error*.**

Keywords---* *Differential Evolution, Conjugate Gradient, Neural Network, Nonlinear system identification

I. Introduction

Nonlinear system identification is a hard and complex task, there are several disciplines to construct and estimate models on nonlinear system: automatic control and system identification, [1] [2] ,mathematical sta-tistics [3], Physical modeling, [4] [5], learning theory and support vector machines [6] [7], neural network techniques [8] and several others. Recent results show that neural network technique seems to be very effective to solve the problem of identifying an implicit mathematical model from the observed input-output data when determine model information cannot be obtained. In fact, many authors have discussed the Neural Network (NN) modeling and application to system identification, prediction and control [9] [10] [11] [12] [13] .

There are different ways to train neural network learning algorithms:

- Gradient methods: the first derivative methods include gradient descent method and conjugate gradient method
- Hessian methods : the second derivative methods include steepest descent method,
- Newton-Raphson method, Gauss-Newton method, Pseudo-Newton method and Quasi-Newton method
- Genetic algorithm and evolutionary computing.

Many of the training algorithms are based on the conjugate gradient algorithm. It is one of the most widely used error minimization methods used to train neuron networks. Despite the general success of conjugate gradient method, the slight drawback of the above method is:

- The convergence rate is very low and hence it becomes unsuitable for large problems.
- Finding local minima, it is not guaranteed to find a global optimum.

Differential Evolution DE is a global search algorithm employ heuristics to allow escaping from local minima. DE is a simple stochastic optimization algorithm and evolves a population of potential solutions (individuals) in order to increase the convergence to optimal solutions.

In this paper, an alternative method based on Evolutionary algorithms proposed. The proposed method combines the differential evolution (DE), an evolutionary algorithm, with gradient descent (GD) which is local optimization algorithm to identify a nonlinear system .Our method is described in the following steps:
(1) Train Neural Network model by using the conjugate Gradient algorithm,
(2) Calculation of the conjugate gradient of error with respect to the weights values,
(3) Choice the weight by applying DE.

The remainder of this paper is organized as follows. Section 2 the problem formulation. Section 3 explains the concepts of optimization methods based on DE approaches. Section 4 describes the proposed algorithm. Simulations and comparisons are provided in Section 5. Last, Section 6 outlines the conclusion with a brief summary of results and future research

II. Problem Formulation

A majority class of nonlinear dynamic systems with an input u and an output y can be described in discrete time by the NARX (nonlinear autoregressive with exogenous input) input–output model:

$$y(k+1) = f(x(k)) \tag{1}$$

Where $y(k+1)$ denotes the output predicted at the future time instant $k+1$ and $x(k)$ is the regressor vector, consisting of a finite number of past inputs and outputs:

$$x(k) = \left[\, y(k) \dots y(k-n_y+1) \quad u(k) \dots u(k-n_u+1) \right]^T \tag{2}$$

Where n_y and n_u : the dynamic order of the system

The problem of nonlinear system identification is to deduce the unknown function f in (1) from some sampled data sequences $\{(u(k), y(k))\}$ where $k = 1, 2, \dots, N$.

In black-box modeling, these functions are approximated by some general function approximators such as neural networks, neuro-fuzzy systems, etc…

The neural network NARX model is described in the Figure 1.

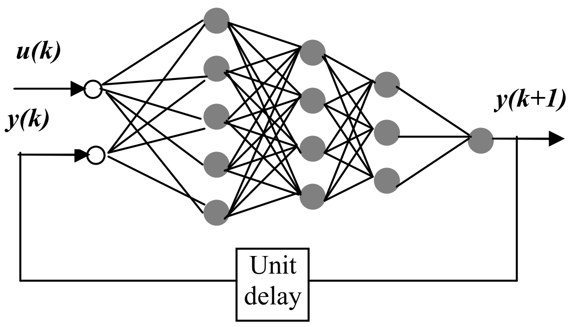

Figure 1: Neural Network NARX model

The objective of learning is to reduce the error signal $\varepsilon_k(n)$ so that the desired response will be close to the actual response $\varepsilon_k(n)$ at the output of neuron k at iteration n is defined by:

$$\varepsilon_k(n) = x_k(n) - y_k(n) \tag{3}$$

Where $x_k(n)$ refers to the desired response for neuron k and is used to compute $\varepsilon_k(n)$ and $y_k(n)$ refers to the function signal appearing at the output of neuron k at iteration n.

The total error is obtained by summing $\frac{1}{2}\varepsilon_j^2(n)$ over all neurons in the output layer. We may thus write

$$E(n) = \frac{1}{2} \sum_{k=1}^{N} \varepsilon_k^2(n) \tag{4}$$

Where N is the total number of all output nodes,

The net input to the j^{th} node of layer L is obtained by:

$$net_j^L = \sum_k w_{j,k}^L * y_k^{L-1} \tag{5}$$

Where $w_{j,k}^L$: the weight on the connection from the i^{th} node in layer *L-1* to the j^{th} node in layer *L*. The activation of a node y_k^L is defined by a function of its net input:

$$y_k^L = fct(net_j^L) \tag{6}$$

Where *fct* is any activation function

As mentioned above, there are different ways to train neural network learning algorithms. In our work, we used the local optimization algorithm of conjugate gradients (CG) and the differential evolution algorithm (DE), which is a population-based stochastic global search method, to yield a computationally efficient algorithm for training neuron network.

III. The Differential Evolution Method

Differential Evolution DE is a simple stochastic optimization algorithm and evolves a population of potential solutions (individuals) in order to increase the convergence to optimal solutions. DE is a global search algorithm employ heuristics to allow escaping from local minima.

The main concept of DE is generating trial parameter vectors by adding a weighted difference of two parameters to a third one.

This process is the appropriate genetic mutation operator of evolutionary algorithms. Output vectors are combined with another vector (called the target vector). This operation is called crossover. The result of crossover is a new vector, called the trial vector. The trial vector is accepted as an individual in the new generation if its fitness function value is less than the target vector fitness function value.

There are four processes in DE which are:

A. DE initialisation

All parameter vectors in a population are randomly initialized and evaluated using the fitness function. The initial NP D-dimensional vectors are chosen randomly and should cover the entire parameter space x_i^0 ; where $i = 1, 2 \dots NP$ and NP is size of population vectors, D is the number of decision variables.

B. DE Mutation operator

DE generates new vectors by adding the difference between two vectors to a third one. The mutant vector is generated according to:

$$\mathrm{v}_i^t = x_{Best}^t + F.(x_{r1}^t - x_{r2}^t) \tag{7}$$

Where v_i^t is a mutant vector; $r1$, $r2$ represent random integer mutually different from index i, F is a real and constant factor which controls the amplification of the Best vector (practically; $0 < F < 2$).

C. DE Crossover operator

Considering the each best vector x_{Best}^{t} in the current population, a trial vector u_i^t is generated by crossing the best vector with the corresponding mutant vector v_i^t under a pre-specified crossover rate $CR \in [0\ 1]$. The mutated vector's elements are then mixed with the elements of the predetermined Best vector to yield the so-called trial vector. Moreover, choosing a set of crossover points, like crossover operator in genetic algorithms or evolution strategies, the crossover is introduced to increase the diversity of the perturbed parameter vectors.

The trial vector is formulated by:

$$u_i^t = \begin{cases} v_i^t & \text{if (Jrand(j)} \leq \text{CR) or j=Rnbr(i)} \\ x_{Best}^t & \text{Otherwise} \end{cases} \quad (8)$$
$$j = 1, 2, ..., D$$

Where Jrand(j) is the jth evaluation of a uniform random number generator with outcome [0, 1], *CR* is the crossover constant [0 1], Rnbr(i) is randomly chosen index from 1to *D*, where *D* represents the number of dimension.

D. DE selection

If the trial vector u_i^t has equal or better fitness function value than that of its corresponding best vector x_{Best}^t , it replaces the Best vector. Otherwise, the Best vector remains in the next iteration.

Figure 2 describes the concept of DE+CG strategy for nonlinear system identification.

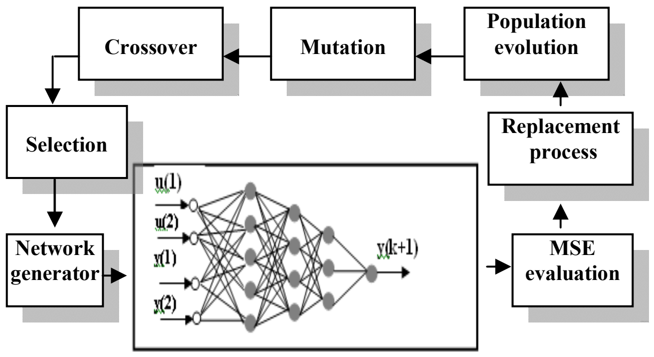

Figure2: Principe of CG+DE strategy for nonlinear system identification

In the training process, both the input vector u and the output vector *y* are known, and the synaptic weights in *W* are adapted to obtain appropriate functional mappings from the input u to the output *y*.

By using Matlab, there were several training algorithms in the Neural Netwok model, which have a variety of different computation but it does not exist an algorithm which is best suited. In our work, we try to implement our system by using the conjugate Gradient algorithm.

The optimization goal is to minimize the objective function E by optimizing the values of the network weights w, so a desired correspondence between inputs and outputs of the NN is achieved. Our objective here is to apply DE and CG to the weight optimization.

IV. The Proposed Algorithm

Step1. Train Neural Network model by using the conjugate Gradient algorithm to find a population of constant size that consists of NP, real- value vectors, $W_{i,G,}$ where I is index to the population and G is the generation to which the population belongs.

$$Pop_G = (W_{1,G}, \ldots\ldots, W_{NP,G}), \quad \text{Where } G=0,\ldots\ldots\ldots,G_{max}$$

In network training, each vector contains D network weights (chromosomes of individuals):

$$W_{i,G} = (w_{1,i,G}, \ldots\ldots\ldots w_{D,i,G}), \text{ Where } \quad i = 1,\ldots NP$$

Step2. After finding the first population by using the conjugate Gradient algorithm, the mutant vector is generated as follows:

$$v_{j,i,G+1} = w_{j,r1,G} + F(w_{j,r2,G} - w_{j,r3,G})$$

Where $i \neq r1 \neq r2 \neq r3 \in \{1,\ldots.NP\}$, $j = 1,\ldots\ldots D$;$r1, r2, r3 \in \{1\ldots\ldots.NP\}$, randomly selected , , $F \in [0, 1]$

Step3. Apply crossover to found trial vector

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

Where $CR \in [0, 1]$.

Step6. The selection between the trial vector and target vector is according to the following rule

$$w_{i,G+1} = \begin{cases} t_{j,i,G+1} \ if\ E(y, f(x, t_{j,i,G+1})) \leq E(y, f(x, w_{i,G})) \\ w_{i,G}\ otherwise \end{cases}$$

Step7. Assuming that the error E is to be minimized, the vector with the lower E value wins a place in the next generation's population. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation.

V. Results and Discussion

To validate the effectiveness of our proposed algorithm, we have conducted extensive computer simulations on NARX model identification problems. The simulation results are compared with the CG algorithm.

Example

The NARX (nonlinear autoregressive with exogenous input) input–output model to be identified is expressed by:

$$y(k+1)=\frac{y(k)}{1+[y(k)]^2}+u(k)^3$$

Where $y(k)$ is the output of the system at the k-th time step and $u(k)$ is the plant input

$$u(k)=\sin\left(\frac{2\pi k}{25}\right)+\sin\left(\frac{2\pi k}{10}\right)$$

Figure3 shows the actual output, $y(k)$ and identified plant output trained with conjugate gradient algorithm within the time step of 0 to 150. As indicated, we can conclude that the algorithm is unable to identify the nonlinear system.

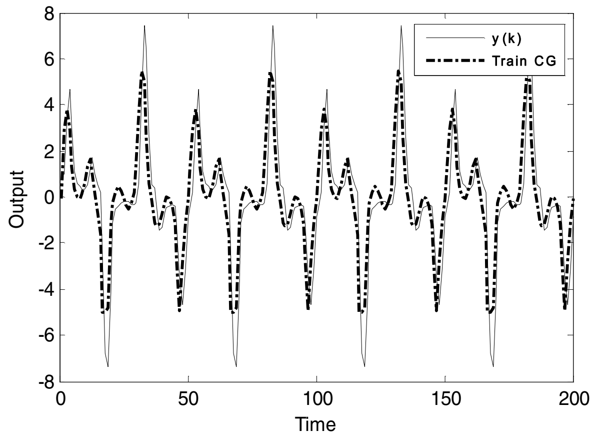

Figure 3: Output of the plant and of the identification model trained with only CG.

Figure4 shows the actual output, $y(k)$ and identified plant output trained with our proposed CG+DE algorithm. According to this figure, we can conclude that accurate identification of a nonlinear system of Example is achieved. From this it is clear that the CG+DE approach exhibits better identification ability compared to CG approach.

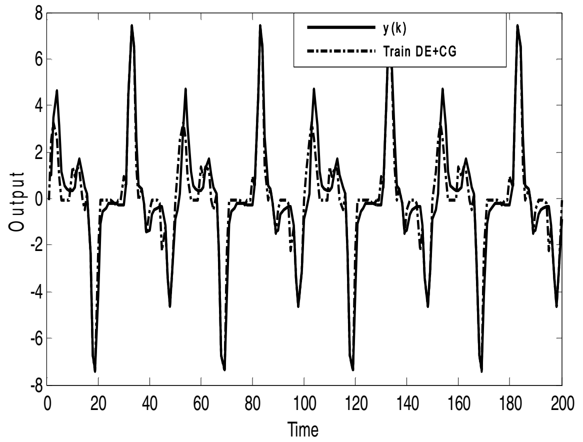

Figure 4: Output of the plant and of the identification model trained with DE+CG.

Figure5 shows the error between the actual and identified model for both the identification scheme.

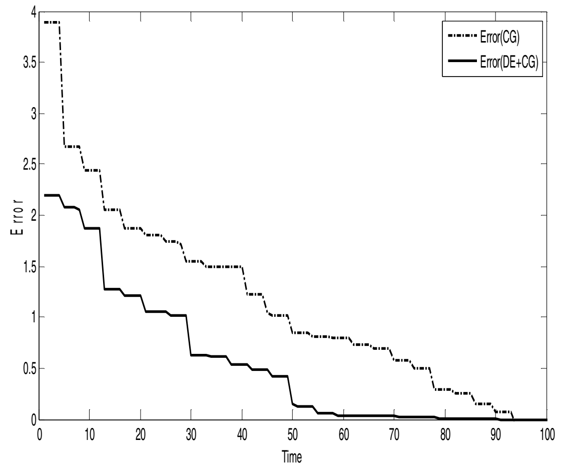

Figure 5: Comparison of error (CG) and error (CG+DE)

We can see from Figure5, the convergence of the proposed method is not only faster than other method but also much more stable.

Table1 summaries the performance of the proposed method of system identification.

Table1: Comparison of performance of two methods

	Method	Mean Squared Error (MSE)	Time of convergence
Example	CG CG +DE	2.2963 0.5337	94 70

The results have revealed that CG+DE algorithm has given quite promising results in terms of time of convergence and smaller mean squared error (MSE) compared to CG algorithm.

VI. Conclusion

In this paper, a new fast learning algorithm for neural networks based differential evolution and conjugate gradient training algorithm is introduced. Our proposed algorithm 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 proposed method improved the training efficiency of neural network algorithm to nonlinear system identification. The proposed algorithm is generic and easy to implement in all different plants. The simulation results showed that the proposed algorithm is robust and has given quite promising results in terms of convergence rate and smaller mean squared error (MSE) compared to CG algorithm.

REFERENCES

[1] J. Sjöberg, Q. Zhang, ,L. Ljung, A. Benveniste, B. Delyon, , P.Y. Glorennec, H. Hjalmarsson, and A. Juditsky, Nonlinear black-box modeling in system identification, A unified overview. Automatica, pp.1691–1724, 1995.

[2] L. Ljung, Q.Zhang, P. Lindskog, A. Juditsky and R. Singh, , An integrated system identification toolbox for linear and non-linear models. In: Proc. 14th IFAC Symposium on System Identification. Newcastle, Australia, 2006.

[3] T. Hastie, R.Tibshirani and J. Friedman,. The Elements of Statistical Learning, Springer, 2001.

[4] P. Fritzson, Object-oriented Modeling and Simulation with MODELICA 2.1. IEEE Press. Piscataway,NJ, 2004.

[5] T. Bohlin, Practical Grey-box Process Identification.Springer-Verlag. London , 2006.

[6] V. Vapnik, Statistical Learning Theory. Wiley, 1998.

[7] J.A.K. Suykens, T. Gestel, J De Brabanter, B. De Moor and J. Vandewalle, Least Squares Support Vector Machines. World Scientific. Singapore and often non-parametric models such as neural networks are used in place of intricate mathematics, 2002.

[8] A.R. Barron, Statistical properteis of artificial neural networks. In: Proceedings of the 28th IEEE Conference on Decision and Control. pp. 280–285, 1989.

[9] K.S. Narendra,.and K. Parthasarathy, Identification and Control of Dynamic Systems using Neural Networks, IEEE Transactions on NNs, pp. 4-27, 1990.

[10] J.T. Connor, R.Martin, and L. Atlas, Recurrent NN and Robust Time Series Prediction. IEEE Transactions on NNs, pp. 240-253, 1994.

[11] P.S. Sastry, G. Santharam, and K.P. Unnikrishnan, Memory Networks for Identification and Control of Dynamical Systems. IEEE Transactions on NNs, pp.306-320 , 1994.

[12] D.T. Pham and S. Yildirim, Robot Control using Jordan Neural Networks. Proc. of the Internat. Conf. on Recent Advances in Mechatronics, Istanbul, Turkey, 1995.

[13] A.C. Tsoi and A.D .Back, Locally Recurrent Globally Feedforward Networks, A Critical Review of Architectures. IEEE Transactions on NNs, pp. 229-239, 1994.