World Applied Sciences Journal 8 (1): 01-09, 2010
ISSN 1818-4952

Standard Fuzzy Model Identification using Gradient Methods

Laiq Khan, S. Anjum and R. Badar

Department of Electrical Engineering, COMSATS Institute of Information Technology, Abbottabad, Pakistan

Abstract: Identification of unknown system by training the parameters adaptively using different fuzzy models has been proven an interesting research area over last few decades. The objective of this paper is to identify a standard fuzzy system using different gradient methods and discuss their characteristics. Approach of this work is to calculate the gradient of the appropriate cost function to minimize the error by updating the parameters and estimate the system perfectly. Results of these methods are compared on the basis of best approximation and fast convergence.

Key words: Standard fuzzy system . gaussian membership function . gradient methods

INTRODUCTION

The dynamics of unknown non-linear plant is often difficult to model. In literature, many techniques have been devised to cope with this issue. One of the most commonly used techniques is the adaptation of the plant response by training the parameters or dynamics of the nonlinear model to follow the required response.

Fuzzy modeling has been proven a powerful modeling tool for non-linear system identification [1, 2, 3]. It can identify the complex non-linear systems by minimizing the cost function using estimation of parameters and fuzzy rules. Hence fuzzy systems use the human made linguistic rules as compared to neural networks with same approximation results.

A fuzzy logic system converts the user supplied rules into their corresponding mathematical form. This not only simplifies the work but produces more accurate results. These systems are simple, flexible and they can model the complex non-linear function even if inexact and incomplete data is available [4]. Fuzzy system can be created to match any given set of input output data.

In fuzzy systems, parameters of membership functions are trained to approximate the objective function. There are several methods used for updating the parameters e.g., genetic algorithms [5, 6, 7], gradient methods [8] and Least mean square algorithms etc.

Gradient methods depend on the derivatives of the functions and derivatives simply show the slope of a function. So if the slope of a function is identified, then function can be minimized by converging towards the minimum point [9]. In identification process these methods are used to minimize the error function for best approximation. Researchers are revealing many applications of gradient methods to fuzzy system models and enormous work can be found in literature, for further reading the reader is referred to see [10, 11, 12, 13, 14, 15].

This article endeavors to present an introduction to various gradient techniques. In addition, these gradient methods have been used for identification of the fuzzy non-linear system [16, 17].

This paper is arranged as follows: Section 2 gives the introduction of a standard fuzzy system and the appropriate cost function used for its identification. In section 3 gradient methods and mathematical description for tuning of parameters has been discussed. Section 4 gives a discussion about the simulation results. Finally, section 5 concludes this research.

STANDARD FUZZY SYSTEM

The unknown function $f(x|\rho)$ can be approximated by the standard fuzzy model using a set of rules, which can be expressed as:

$$R_i: \text{IF } x_1 \text{ is } F_{i1} \text{ and } \ldots \text{ and } x_n \text{ is } F_{in}$$
$$\text{THEN } y_i = \beta_i,\ i = 1,2,\ldots, L$$

where R_i shows its *ith* rule, F_{ij} are fuzzy sets, L is the number of rules, $x = (x_1, x_2 \ldots, x_n)^T$ are the number of variable inputs with single output y (x) and β_i are the centers of output membership functions.

Suppose, standard fuzzy model for Gaussian membership function is given as:

Correspoding Author: Dr. Laiq Khan, Department of Electrical Engineering,
COMSATS Institute of Information Technology, Abbottabad, Pakistan

$$f(x|\rho_j)=\frac{\sum_{i=1}^{L}\beta_i\mu_i(x)}{\sum_{i=1}^{L}\mu_i(x)} \quad (1)$$

Where

$$\mu_i(x)=\prod_{j=1}^{l}\exp\left(-\frac{1}{2}\left(\frac{x_i-c_j^i}{\sigma_j^i}\right)^2\right) \quad (2)$$

j specifies the universe of discourse, j = 1, 2, …, l.

Parameters ρ of the input membership functions are spread σ_i^j and centres c_i^j, centre β_i is the parameter of output membership function. These are tuned for approximating the fuzzy system.

Suppose if there are m data pairs (x^m, y^m) for tuning than the learning error function will be defined as [18]:

$$e_m=-\frac{1}{2}\left(f(x^m|\rho)-\tau\right)^2 \quad (3)$$

Here, gradient methods are used to minimize the error e_m by adjusting the parameters ρ, which for fuzzy system are σ_i^j, c_i^j and β_i to adapt the response of the reference signal as illustrated in Fig. 1.

GRADIENT METHODS

These methods are used to find the minimum of the objective function by training the parameters of the system [19, 20, 21]. These parameters are tuned to produce the best approximation. These methods require the first or second derivative of the objective function with respect to the parameter which has to be tuned.

Gradient methods discussed in this section are as follows:

- Steepest descent method.
- Newton's method.
- Quasi Newton's method.
- Levenberg Marquardt for quasi Newton's method.
- Gauss Newton's method.
- Levenberg Marquardt for Gauss Newton's method.

Steepest decent method: Steepest decent is one of the initial and simple technique used for minimizing the cost function. All gradient methods iteratively update the parameters, i.e., values of parameters are set

initially and then updated accordingly in order to minimize the problem. Mathematically, steepest descent is represented as:

$$\rho(k+1)=\rho(k)-\gamma g_k \quad (4)$$

where, γ is the learning rate. It has small positive value adjusted for fast convergence. g_k is the gradient of cost function and k is the iteration index.

$$g_k=\frac{\partial e_m}{\partial\rho}=\left[f(x^m|\rho)-\tau^m\right]*\frac{\partial e_m}{\partial\rho} \quad (5)$$

$$g_k=\frac{\partial e_m}{\partial\rho}=-\varepsilon*\frac{\partial e_m}{\partial\rho} \quad (6)$$

Fuzzy system discussed in this article has three parameters, by using (4) and (6) all parameters will be updated as follows,

Update law for the output membership function centers (β):

$$\beta_i(k+1)=\beta_i(k)-\gamma\frac{\partial e_m}{\partial\beta_i} \quad (7)$$

$$\frac{\partial e_m}{\partial\beta_i}=\varepsilon_m*\frac{\partial}{\partial\beta_i}\frac{\sum_{i=1}^{L}\beta_i*\mu_i(x^m)}{\sum_{i=1}^{L}\mu_i(x^m)} \quad (8)$$

Substituting the value of μ from (2) in (8) and differentiating w.r.t. β_i the gradient becomes:

$$\frac{\partial e_m}{\partial\beta_i}=\varepsilon*\frac{\prod_{j=1}^{l}\exp\left(-\frac{1}{2}\left(\frac{x_j^m-c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{L}\prod_{j=1}^{l}\exp\left(-\frac{1}{2}\left(\frac{x_j^m-c_j^i}{\sigma_j^i}\right)^2\right)} \quad (9)$$

Using (8) in (7), the final update law for the output membership function becomes:

$$\beta_i(k+1)=\beta_i(k)-\gamma\varepsilon*\frac{\prod_{j=1}^{l}\exp\left(-\frac{1}{2}\left(\frac{x_j^m-c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{L}\prod_{j=1}^{l}\exp\left(-\frac{1}{2}\left(\frac{x_j^m-c_j^i}{\sigma_j^i}\right)^2\right)} \quad (10)$$

Update law for input membership function spreads (σ)

$$\sigma_i(k+1)=\sigma_i(k)-\gamma\frac{\partial e_m}{\partial\sigma_i} \quad (11)$$

$$\frac{\partial e_m}{\partial \sigma_i} = \varepsilon * \frac{\partial}{\partial \sigma_i} \frac{\sum_{i=1}^{L} \beta_i * \prod_{j=1}^{l} \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{L} \prod_{j=1}^{l} \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i}{\sigma_j^i}\right)^2\right)} \quad (12)$$

Differentiating and simplifying (12):

$$\frac{\partial e_m}{\partial \sigma_i} = \varepsilon * \left(\sum_{i=1}^{L} \beta_i - \frac{\sum_{i=1}^{L} \beta_i * \mu_i(x^m)}{\sum_{i=1}^{L} \mu_i(x^m)} \right) * \left(\frac{(x_j^m - c_j^i)^2}{(\sigma_j^i)^3} \right) * \mu_i(x^m) \quad (13)$$

Using (13) and (11), the update form for σ_i is given by:

$$\sigma_i(k+1) = \sigma_i(k) - \gamma\varepsilon * \left(\frac{\sum_{i=1}^{L} \beta_i - f(x^m | \rho)}{\sum_{i=1}^{L} \mu_i(x^m)} \right) * \left(\frac{(x_j^m - c_j^i)^2}{(\sigma_j^i)^3} \right) * \mu_i(x^m) \quad (14)$$

Update law for the input membership function centers (c):

$$c_i(k+1) = c_i(k) - \gamma \frac{\partial \varepsilon}{\partial c_i} \quad (15)$$

$$\frac{\partial e_m}{\partial c_i} = \varepsilon * \frac{\partial}{\partial c_i} \frac{\sum_{i=1}^{L} \beta_i * \prod_{j=1}^{l} \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{L} \prod_{j=1}^{l} \exp\left(-\frac{1}{2}\left(\frac{x_j^m - c_j^i}{\sigma_j^i}\right)^2\right)} \quad (16)$$

Differentiating and simplifying (16) yields:

$$\frac{\partial e_m}{\partial c_i} = \varepsilon * \left(\sum_{i=1}^{L} \beta_i - \frac{\sum_{i=1}^{L} \beta_i * \mu_i(x^m)}{\sum_{i=1}^{L} \mu_i(x^m)} \right) * \left(\frac{(x_j^m - c_j^i)}{(\sigma_j^i)^2} \right) * \mu_i(x^m) \quad (17)$$

Using (17) in (15) the update relation for input membership function centers becomes,

$$c_i(k+1) = c_i(k) - \gamma\varepsilon * \left(\frac{\sum_{i=1}^{L} \beta_i - f(x^{m|\rho})}{\sum_{i=1}^{L} \mu_i(x^{m,k})} \right) * \left(\frac{(x_j^m - c_j^i)}{(\sigma_j^i)^2} \right) * \mu_i(x^{m,k}) \quad (18)$$

Equations (10), (14) and (18) are the update equations for minimizing the error functions.

Newton's method: This method is based on the second order expansion of the Taylor series. The basic idea behind Newton's method is to approximate the quadratic functions. This method proceeds to find the minimum of the quadratic function in one step. Mathematically, it is represented as,

$$\rho(k+1) = \rho(k) - H^{-1} g_k \quad (19)$$

where, 'H' is the Hessian matrix, consists of the second order partial derivatives of the cost function. This method is less efficient if the function is not a quadratic and will not converge in one step. Since, this method completely depends on the function and initial guesses of the parameters, so there is no assurance that it will converge at all. Moreover, the size of Hessian matrix increases and computation becomes more complex as the number of parameters increase.

For tuning of parameters, firstly Hessian matrix H is calculated as:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial \rho_1^2} & \frac{\partial f}{\partial \rho_1 \partial \rho_2} & \cdots & \frac{\partial f}{\partial \rho_1 \partial \rho_2} \\ \frac{\partial f}{\partial \rho_2 \partial \rho_1} & \frac{\partial^2}{\partial \rho_2^2} & \cdots & \frac{\partial f}{\partial \rho_2 \partial \rho_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f}{\partial \rho_1 \partial \rho_1} & \frac{\partial f}{\partial \rho_1 \partial \rho_2} & \cdots & \frac{\partial^2}{\partial \rho_1^2} \end{bmatrix} \quad (20)$$

g_k is the gradient matrix and is given by,

$$g_k = \begin{bmatrix} \frac{\partial e_m}{\partial \rho_1} \\ \frac{\partial e_m}{\partial \rho_2} \\ \vdots \\ \frac{\partial e_m}{\partial \rho_l} \end{bmatrix}$$

Here update law for all the parameters are calculated separately as defined for the previous method to avoid complexity.

Update law for the output membership function centers (β)

$$\begin{bmatrix} \beta_1(k+1) \\ \beta_2(k+1) \\ \vdots \\ \beta_l(k+1) \end{bmatrix} = \begin{bmatrix} \beta_1(k) \\ \beta_2(k) \\ \vdots \\ \beta_l(k) \end{bmatrix} - \begin{bmatrix} \frac{\partial^2 f}{\partial \beta_1^2} & \frac{\partial f}{\partial \beta_1 \partial \beta_2} & \cdots & \frac{\partial f}{\partial \beta_1 \partial \beta_2} \\ \frac{\partial f}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2}{\partial \beta_2^2} & \cdots & \frac{\partial f}{\partial \beta_2 \partial \beta_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f}{\partial \beta_l \partial \beta_1} & \frac{\partial f}{\partial \beta_l \partial \beta_2} & \cdots & \frac{\partial^2}{\partial \beta_l^2} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial e_m}{\partial \beta_1} \\ \frac{\partial e_m}{\partial \beta_2} \\ \vdots \\ \frac{\partial e_m}{\partial \beta_l} \end{bmatrix} \quad (21)$$

Use (9) for calculation of g_k and H is calculated by taking the second derivative of the gradient equation.

Update law for the input membership function spreads (σ)

$$\begin{bmatrix} \sigma_1(k+1) \\ \sigma_2(k+1) \\ \vdots \\ \sigma_l(k+1) \end{bmatrix} = \begin{bmatrix} \sigma_1(k) \\ \sigma_2(k) \\ \vdots \\ \sigma_l(k) \end{bmatrix} - \begin{bmatrix} \frac{\partial^2 f}{\partial \sigma_1^2} & \frac{\partial f}{\partial \sigma_l \partial \sigma_2} & \cdots & \frac{\partial f}{\partial \sigma \rho \sigma_2} \\ \frac{\partial f}{\partial \sigma_2 \partial \sigma_1} & \frac{\partial^2}{\partial \sigma_2^2} & \cdots & \frac{\partial f}{\partial \sigma_2 \partial \sigma_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f}{\partial \sigma_l \partial \sigma_1} & \frac{\partial f}{\partial \sigma \rho \sigma_2} & \cdots & \frac{\partial^2}{\partial \sigma_1^2} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial e_m}{\partial \sigma_1} \\ \frac{\partial e_m}{\partial \sigma_2} \\ \vdots \\ \frac{\partial e_m}{\partial \sigma_1} \end{bmatrix} \quad (22)$$

By using (13) we can find out the gradient g_k w.r.t. σ_i.

Update law for the input membership function centers (c)

$$\begin{bmatrix} c_l(k+1) \\ c_2(k+1) \\ \vdots \\ c_l(k+1) \end{bmatrix} = \begin{bmatrix} c_l(k) \\ c_2(k) \\ \vdots \\ c_l(k) \end{bmatrix} - \begin{bmatrix} \frac{\partial^2 f}{\partial c_1^2} & \frac{\partial f}{\partial c_1 \partial c_2} & \cdots & \frac{\partial f}{\partial c_1 \partial c_2} \\ \frac{\partial f}{\partial c_2 \partial c_1} & \frac{\partial^2}{\partial c_2^2} & \cdots & \frac{\partial f}{\partial c_2 \partial c_1} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f}{\partial c_1 \partial c_1} & \frac{\partial f}{\partial c_1 \partial c_2} & \cdots & \frac{\partial^2}{\partial c_1^2} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial e_m}{\partial c_1} \\ \frac{\partial e_m}{\partial c_2} \\ \vdots \\ \frac{\partial e_m}{\partial c_1} \end{bmatrix} \quad (23)$$

g_k can be find using (17) and elements of Hessian matrix for input membership centers can be computed by taking the second derivative of the parameters as defined in the matrix.

Quasi Newton's method: The concept of Quasi Newton's emerged to make the Newton's method more effective and speedy. Appropriate selection of learning rate γ makes the approximation of the system faster. Although, there is not much difference among these two methods. Quasi Newton's method also converges in one step to the minimum

point in case of quadratic functions, but comparatively it tracks the desired response in less time. Mathematically, it is represented as:

$$\rho(k+1)=\rho(k)-\gamma(H)^{-1}g_k \tag{24}$$

Rest of the calculations can be carried out on the same lines as described in the previous section for each parameter update.

Levenberg marquart for quasi newton's method: Levenbereg Marquardt solves the serious problem in Newton's method and that is the singularity of matrix which makes it non invertible. Suitable value of λI when added with Hessian matrix makes it non-singular. Mathematically, it is written as,

$$\rho(k+1)=\rho(k)-\gamma(H+\lambda I)^{-1}g_k \tag{25}$$

Gauss Newton's method: Gauss Newton method is also called the linearization method. It expands through Taylor series to attain a linear model that estimates the original non-linear model [22].
Mathematically Gauss Newton method is expressed as:

$$\rho(k+1)=\rho(k)-\gamma(J^TJ)^{-1}J \tag{26}$$

J is the Jaccobian matrix also known as gradient matrix. Gradient of all the parameters are calculated and used in (26) for update.

Update law for the output membership function centers (β)

$$\begin{bmatrix}\beta_1(k+1)\\ \beta_2(k+1)\\ \vdots\\ \beta_1(k+1)\end{bmatrix}=\begin{bmatrix}\beta_1(k)\\ \beta_2(k)\\ \vdots\\ \beta_1(k)\end{bmatrix}-\gamma\left[\begin{bmatrix}\frac{\partial e_m}{\partial\beta_1}\\ \frac{\partial e_m}{\partial\beta_2}\\ \vdots\\ \frac{\partial e_m}{\partial\beta_l}\end{bmatrix}^T\begin{bmatrix}\frac{\partial e_m}{\partial\beta_1}\\ \frac{\partial e_m}{\partial\beta_2}\\ \vdots\\ \frac{\partial e_m}{\partial\beta_1}\end{bmatrix}\right]^{-1}\begin{bmatrix}\frac{\partial e_m}{\partial\beta_1}\\ \frac{\partial e_m}{\partial\beta_2}\\ \vdots\\ \frac{\partial e_m}{\partial\beta_l}\end{bmatrix} \tag{27}$$

Update law for the input membership function spreads (σ)

$$\begin{bmatrix}\sigma_1(k+1)\\ \sigma_2(k+1)\\ \vdots\\ \sigma_1(k+1)\end{bmatrix}=\begin{bmatrix}\sigma_1(k)\\ \sigma_2(k)\\ \vdots\\ \sigma_1(k)\end{bmatrix}-\gamma\left[\begin{bmatrix}\frac{\partial e_m}{\partial\sigma_1}\\ \frac{\partial e_m}{\partial\sigma_2}\\ \vdots\\ \frac{\partial e_m}{\partial\sigma_1}\end{bmatrix}^T\begin{bmatrix}\frac{\partial e_m}{\partial\sigma_1}\\ \frac{\partial e_m}{\partial\sigma_2}\\ \vdots\\ \frac{\partial e_m}{\partial\sigma_1}\end{bmatrix}\right]^{-1}\begin{bmatrix}\frac{\partial e_m}{\partial\sigma_1}\\ \frac{\partial e_m}{\partial\sigma_2}\\ \vdots\\ \frac{\partial e_m}{\partial\sigma_1}\end{bmatrix} \tag{28}$$

Update law for the input membership function centers (c)

$$\begin{bmatrix}c_1(k+1)\\ c_2(k+1)\\ \vdots\\ c_1(k+1)\end{bmatrix}=\begin{bmatrix}c_1(k)\\ c_2(k)\\ \vdots\\ c_1(k)\end{bmatrix}-\gamma\left[\begin{bmatrix}\frac{\partial e_m}{\partial c_1}\\ \frac{\partial e_m}{\partial c_2}\\ \vdots\\ \frac{\partial e_m}{\partial c_1}\end{bmatrix}^T\begin{bmatrix}\frac{\partial e_m}{\partial c_1}\\ \frac{\partial e_m}{\partial c_2}\\ \vdots\\ \frac{\partial e_m}{\partial c_1}\end{bmatrix}\right]^{-1}\begin{bmatrix}\frac{\partial e_m}{\partial c_1}\\ \frac{\partial e_m}{\partial c_2}\\ \vdots\\ \frac{\partial e_m}{\partial c_1}\end{bmatrix} \tag{29}$$

Levenberg marquart for quasi gauss newton's method: Levenberg Marquardt gives the modification for Gauss Newton's by adding the steering factor which overcomes the problem of singularity in the function. Since $|J^T J|$ is a singular matrix, so its inverse does not exist and approximation of the function will stop. Therefore, a pseudo Hessian matrix Q is introduced and written as,

$$Q = J\,J+\lambda I \tag{30}$$

Where, λI is the Levenberg Marquardt steering factor, λ is the small non negative value.

Mathematically, this method is represented as:

$$\rho(k+1)=\rho(k)-\gamma(J^TJ+\lambda I)^{-1}J \tag{31}$$

Replacing ρ by respective parameter and substituting the value of J, the update laws for each parameter can be calculated.

SIMULATION RESULTS AND DISCUSSION

Simulink with MATLAB R2009a has been used to generate the simulation results. The system consists of two Gaussian membership functions with two inputs and each has three parameters to be tuned. The mathematical relation used for simulation model is given by:

$$f(x\,|\,\rho_j)=\frac{\beta_1\mu_1(x)+\beta_2\mu_2(x)}{\mu_1+\mu_2} \tag{32}$$

where, x is the variable input, the simulation results have been shown for sinusoidal signal but it can be varied to other cases e.g., square and sawtooth etc. waveforms. A random signal has been selected as reference signal.

Learning rate γ is adjusted for the fast and perfect tracking, if the value of γ is increased from the certain

Table 1: Features of gradient methods

S. No.	Gradient method	Learning rate (γ) Current value	Max. limit	Min. limit	Convergence	Design
1.	Steepest Descent	0.1	3.50	0.05	Slow	Simple
2.	Newton's method	-	-	-	Very fast	Very complex
3.	Quasi Newton's method	2.0			Fast	Very complex
4.	Levenberg Marquardt for Newton's method	1.0	1.85	0.08	Fast	Very complex
5.	Gauss Newton's method	0.1	0.10	0.08	Medium	Complex
6.	Levenberg Marquardt for Gauss Newton's method	1.5	2.10	0.03	Medium	Complex

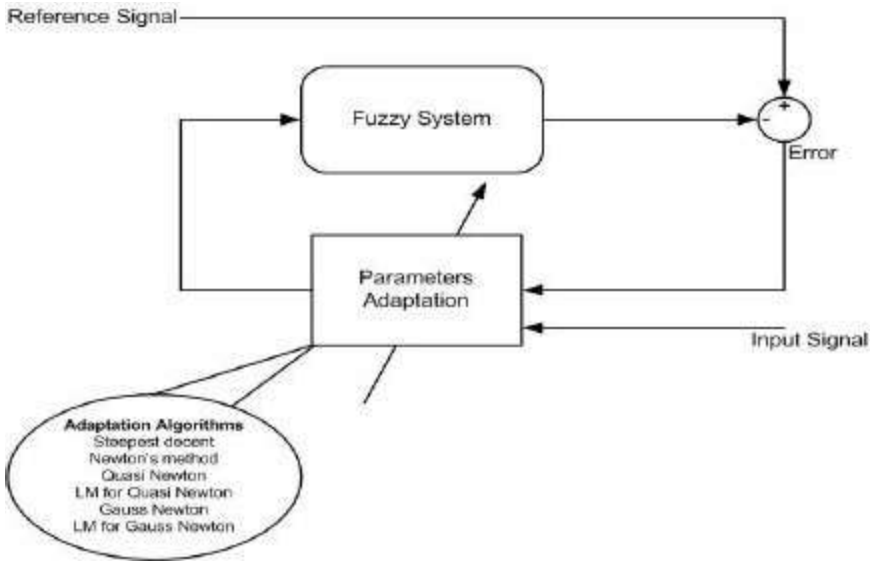

Fig. 1: Standard fuzzy model identification using parameters adaptation

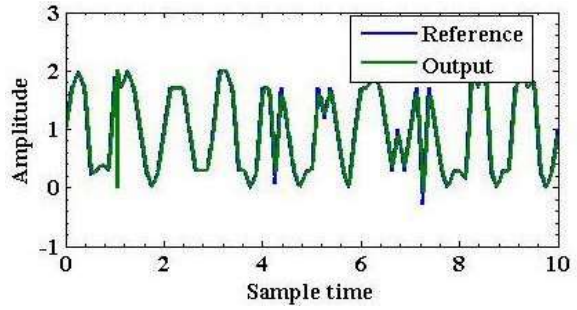

Fig. 2: Output of steepest descent method

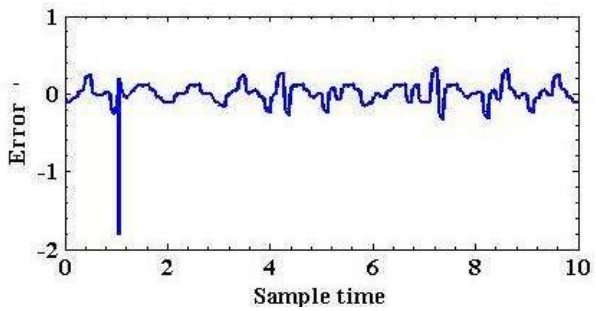

Fig. 3: Error in Steepest descent method

limit then the system goes to the unstable condition and if its value is decreased from certain value then system will stop tracking the reference input, as illustrated in Table 1.

Observations show that complex methodology makes the system very slow even if the method used for approximation is very effective. Newton's method converges in one step for quadratic function. Figure 4 shows that this method gives the best approximation, but highly complex calculation of Hessian matrix makes the system very slow.

The Quasi Newton's method main idea was to introduce the learning factor which makes the system faster. It also depends on the second derivative but due to γ it tracks faster than Newton's method. Fig. 5 and 7 show that the efficiency of both the techniques is almost the same but there is no abrupt error in case of Quasi Newton and tracking of reference signal is

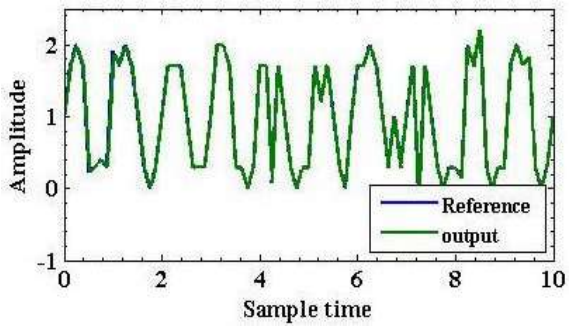

Fig. 4: Output of Newton's method

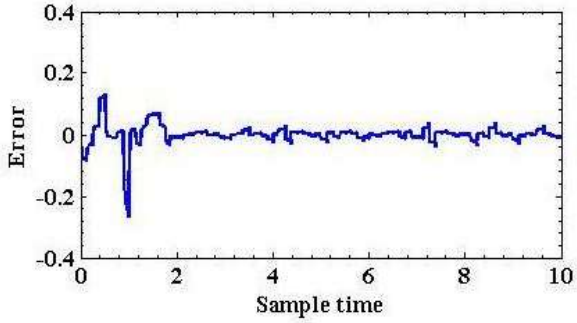

Fig. 5: Error in Newton's method

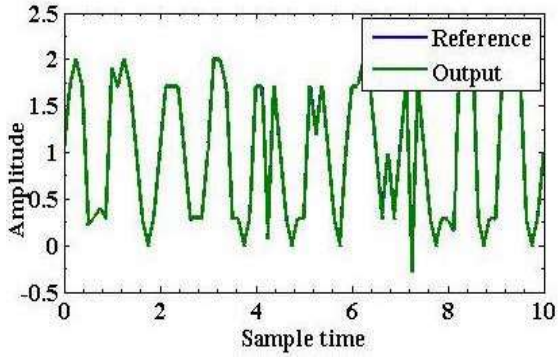

Fig. 6: Output of Quasi Newton's method

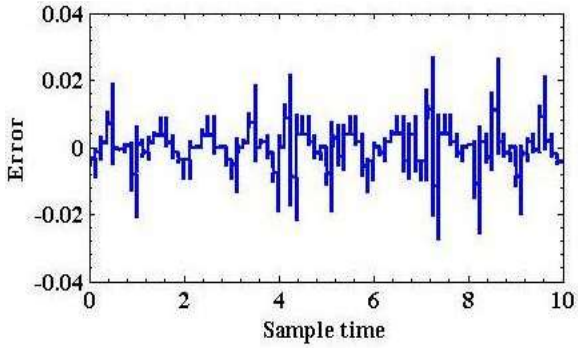

Fig. 7: Error in Quasi Newton's method

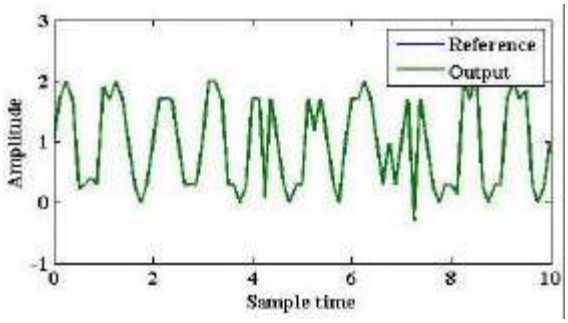

Fig. 8: Output of Levenberg Marquardt for Newton's method

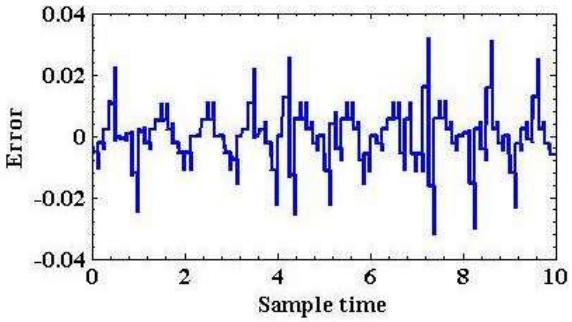

Fig. 9: Error in Levenberg Marquardt for Newton's method

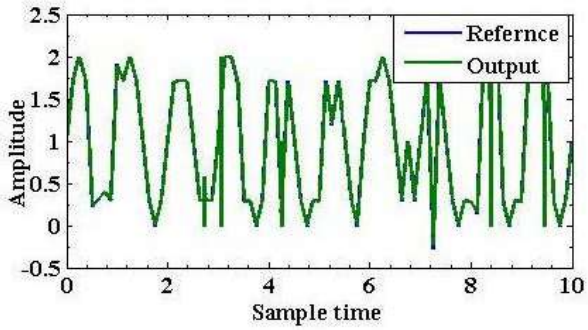

Fig. 10: Output of Gauss Newton's method

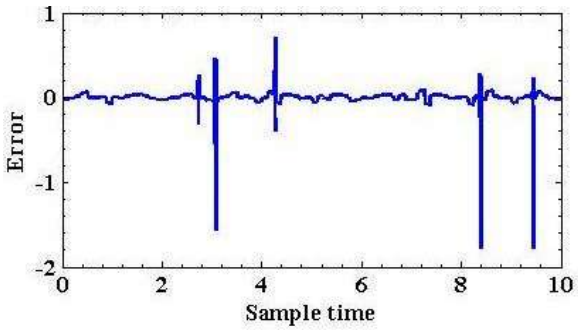

Fig. 11: Error in Gauss Newton's method

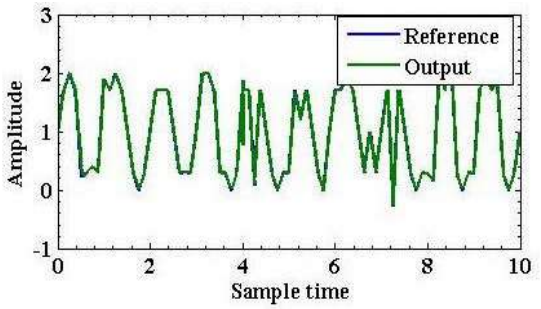

Fig. 12: Output of Levenberg Marquardt for Gauss Newton's method

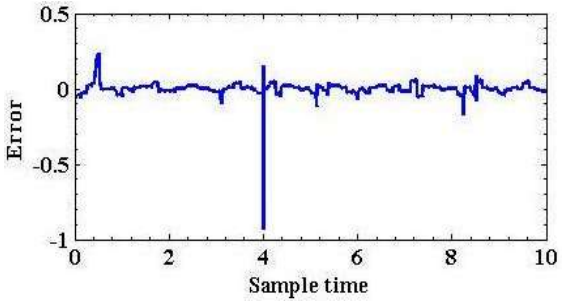

Fig. 13: Error of Levenberg Marquardt for Gauss Newton's method

smooth. Levenberg Marquardt for Newton's method is the modified form of above explained methods it assures that the matrix is non singular and its inverse exists. Otherwise, its performance is almost same as Quasi Newton's method, as it can be seen from Fig. 6 and 8.

Steepest descent slowly converges towards the optimal point. From Fig. 2 and Fig. 3 it is clear that its results are not as good as obtained from other methods but still it is preferred due to simplicity of calculations.

Gauss Newton is less complex than Newton's method as it depends on the first derivatives; they also converge quickly not in first step but still faster than steepest descent. From Fig. 10 and Fig. 11 it can be seen that its tracking is not as effective as Newton method but shows better results than steepest descent.

CONCLUSION

In this paper different gradient methods are used for tracking of a reference signal using standard fuzzy model. It has been observed that the efficiency of the techniques reduces with complexity. Higher the complexity of the algorithm better is the tracking performance at the cost of simulation speed. Suitable values of the parameters used in gradient methods, must be adjusted for the system to work properly.

REFERENCES

1. Cerrada, M., J. Atuilar, E. Colina and A. Titli, 2005. Dynamical membership functions: An approach for adaptive fuzzy modeling. Fuzzy Sets and Systems , pp: 513-533.
2. Kosko, B., 1994. Fuzzy systems as universal approximators. IEEE Trans. Comput., 43: 1329-1333.
3. Pedrycz, W. and M. Reformat, 2003. Evolutionary fuzzy modeling. IEEE Trans. Fuzzy Syst., 11 (5): 652-665.
4. Zadeh, L.A., 1984. Making computer think like people. IEEE Spectrum, pp: 26-32.
5. Cordon, O., M.J. Del Jesus and F. Herrera, 1998. Genetic learning of fuzzy rulebased classification systems cooperating with fuzzy reasoning methods. Int. Jr. Intelligent Syst., 13: 1025-1053.
6. Lee, C.W. and Y.C. Shin, 2003. Construction of fuzzy systems using least-squares method and genetic algorithm. Elsevier, 137 (3): 297-323.
7. Sanchez, L. and J. Otero, 2004. A fast genetic method for inducting descriptive fuzzy models. Fuzzy Sets and Systems , 141 (1): 33-46.
8. Shi, Y., M. Mizumoto, N. Yubazaki and M. Otani, 1996. A learning algorithm for tuning fuzzy rules based on gradient descent method. Proc. of 5th IEEE Int. Conf. Fuzzy Syst., New Orleans, Louisiana, pp: 55-61.
9. Dan, S., 2002. Sum normal optimization of fuzzy membership functions. Int. Jr. Uncertain Fuzziness Knowledge-Based Syst., 10 (4): 363-384.
10. Kim, E., M. Park and M. Park, 1997. A new approach to fuzzy modeling. IEEE Trans. Fuzzy Syst, 5 (3): 328-337.
11. Dan, S., 2005. H_8 estimation for fuzzy membership function optimization. Int. Jr. Approx. Reasoning, 40 (30): 224-242.
12. Hatanaka, T., Y. Kawaguchi and K. Uosaki, 2004. Nonlinear system identification based on evolutionary fuzzy modeling. IEEE Congr. Evolutionary Computation, 1: 646- 651.
13. Mohamed, I., 2003. Non-linear System Identification using Neural Networks Trained with Natural Gradient Decent. Eurasip Jr. Applied Signal Processing, pp: 1229-1237.
14. Oh, S., W. Pedrycz and H. Park, 2003. Hybrid identification in fuzzy neural networks. Fuzzy sets and systems , 138 (2): 399-426.
15. Toivoneni, H.T. and P.M. Makilaf, 1987. Newton's method for solving parametric linear quadratic control problems. Int. Jr. Control, 46 (3): 897-911.
16. Castellano, G. and M.A. Fanelli, 1997. An approach to structure identification of fuzzy models. Proc. 6th IEEE Int. Conf. Fuzzy Syst., Amendola, Italy, 1: 531-536.

17. Keming, X., T.Y. Lin and Z. Jianwei, 1998. The Takagi Sugeno fuzzy model identification method of parameter varying systems. Polkowski, L. and A. Skowron (Eds), Springer-Verlag, Berlin, Heidelberg, pp: 155-162.
18. Khan, Z., S. Chawala and D. Dave, 1990. A note on comparison of nonlinear least squares algorithms. ACM., NY, USA, 25 (2): 10-18.
19. Antoniou, A., 1999. Digital filters analysis, design and applications. Tata McGraw-Hill, 2nd Edn., New Delhi, India
20. Madsen, K., H.B. Nielsen and O. Tingleff, 2004. Methods for non-linear least squares problems. 2nd Edn., pp: 5-24.
21. Passino, K.M., 1997. Fuzzy Control. Addison Wesley Longman, California.
22. Guillaume, P. and R. Pintelon, 1996. A Gauss Newton like optimization algorithm for weighted nonlinear least squares problems. IEEE Trans. Signal Processing, 44 (9): 2222-2228.