World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:6, 2008

Increasing The Speed of Convergence of an Artificial Neural Network based ARMA Coefficients Determination Technique

Abiodun M. Aibinu, Momoh J. E. Salami, Amir A. Shafie and Athaur Rahman Najeeb

***Abstract*—**In this paper, novel techniques in increasing the accuracy and speed of convergence of a Feed forward Back propagation Artificial Neural Network (FFBPNN) with polynomial activation function reported in literature is presented. These technique was subsequently used to determine the coefficients of Autoregressive Moving Average (ARMA) and Autoregressive (AR) system. The results obtained by introducing sequential and batch method of weight initialization, batch method of weight and coefficient update, adaptive momentum and learning rate technique gives more accurate result and significant reduction in convergence time when compared t the traditional method of back propagation algorithm, thereby making FFBPNN an appropriate technique for online ARMA coefficient determination.

***Keywords*—**Adaptive Learning rate, Adaptive momentum, Autoregressive, Modeling, Neural Network.

I. Introduction

The use of parametric modeling technique to predict or reconstruct a data sequence is concerned with the representation of data in an efficient technique [1]–[5], [11], [13]. This method have been used extensively in radar application, geophysical application, Medical signal processing, ultrasonic tissue backscatter coefficient estimation, speech processing, music understanding and more recently in the field of Magnetic Resonance Imaging (MRI) [2], [5], [11]–[13], [18], [19].

The use of parametric modeling technique involve two steps, namely Model selection and Model parameter determination [2], [8], [9]. Model selection is primarily concerned with the selection of appropriate modeling technique to represent the system or the signals under consideration [2], [4], [8]. some of the known and widely used models include Autoregressive Model (AR), Moving Average Model (MA), Auto Regressive Moving Average with External Input (ARX), Moving Average (MA) [2], [8], [9]. The use of different types of model may give similar result for the same system but one of the models may involve the determination of fewer model parameters, such a model is said to be more efficient in its representation than other technique. Furthermore, there exist a relationship between MA, AR and ARMA modeling system, this relationship is as captured in wold decomposition theorem [10]. The second step in parametric modeling is the determination of model parameters. This involve model order and model coefficients determination. Model Order determination involve determining an appropriate and best model order for the system by avoiding the use of too high model order which may lead to over fitting and the use of too low model which normally results in under-fitting, thereby making the system insensitive to noise. Therefore the need to accurately determining the appropriate model order for the system is of high importance in parametric modeling technique [2], [13]. Determination of model coefficients involve the use of methods optimal and sub-optimal technique to determine the ARMA model coefficients. Some of the known techniques include Prony, Pade Approximation method, Shank, etc and more recently the use of Neural Network technique [6]–[8].

This report is an improvement to the work reported in [6], [7]. It involves the introduction of different weight initialization techniques, introduction of adaptive learning rate and momentum and the introduction of batch method of weight and coefficient update to the proposed ARMA based FFBPNN.

The organization of this paper is as follows, In section II, the detail of using Neural network reported in [6], [7] will be discussed with all necessary equations and calculations given. Section III discusses method of increasing the speed of convergence and accuracy of a FFBPNN while the result obtained will be discussed in section IV, the conclusion is as contained in section V.

II. AR/ARMA Coefficients determination using Artificial Neural Network Technique

An Artificial Neural network (ANN) may consist of three (3) main types of layers, namely the input, hidden layers (One or more than one) and the output layer. Each of this layers may contain one or more nodes or neurons connected together by neuron weights. A typical neuron contains a summer unit and an activation function. There exists various activation units among which are; Bipolar; Sigmoid, Tangent, Polynomial, etc. A typical ANN neuron is as shown in Fig.1.

The use of FFBPNN in determining the coefficients of ARMA and NARMA reported in [6], [7] involve the use of a Three (3) layer network, namely the Input layer, 1-Hidden layer and 1-Output layer. The hidden layer contains neurons with adaptive second order polynomial activation function while the output layer contains a linear activation. The total number of input nodes is equivalent to the sum of the order of AR and MA parts. The diagrammatic representation of this is as shown in Fig. 2

Athaur Rahman Najeeb is with the Kulliyah of Engineering, International Islamic University Malaysia (IIUM), email: athaur@iiu.edu.my
Momoh. J. E Salami is with the Kulliyah of Engineering, International Islamic University Malaysia (IIUM), email: momoh@iiu.edu.my
Amir A. Shafie is with the Kulliyah of Engineering, International Islamic University Malaysia (IIUM), email: aashafie@iiu.edu.my
Abiodun. M. Aibinu, Malaysia, email: maibinu@gmail.com

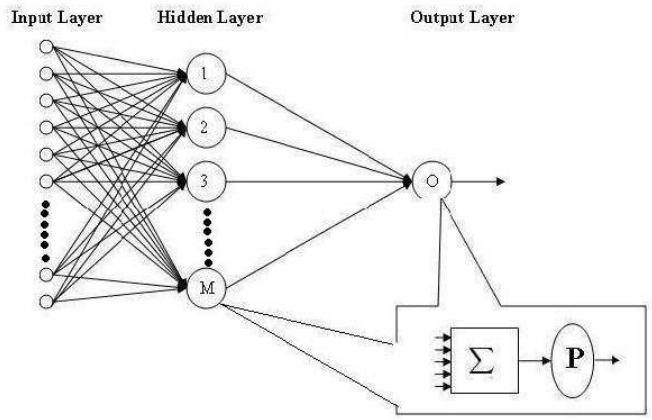

Fig. 1. FFBPNN showing various Layers and the activation functions

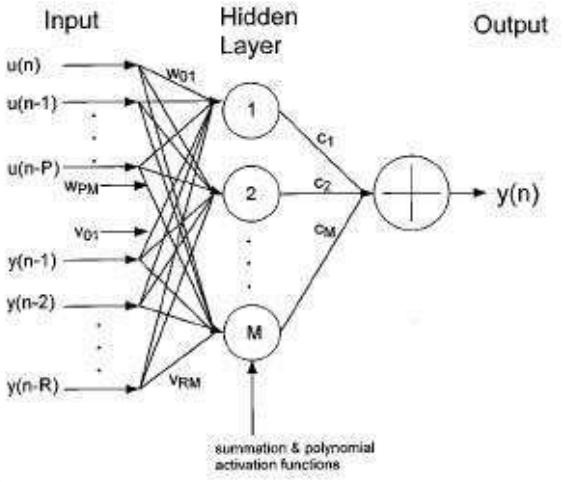

Fig. 2. Neural Network Technique for obtaining ARMA/NARMA coefficients. [6], [7]

The general ARMA equation is given by,

$$y(n) = -\sum_{k=1}^{p} a_k y(n-k) + \sum_{k=0}^{q} b_k x(n-k) \quad (1)$$

taking z-transform of eq. 1, we have

$$Y(z) = -\sum_{k=1}^{p} a_k Y(z) z^{-k} + \sum_{k=0}^{q} b_k X(z) z^{-k} \quad (2)$$

The output of the FFBPNN [6], [7] network is given by

$$y(n) = \sum_{k=1}^{M} w_{k1} P_i(x_i) + e(n) \quad (3)$$

Where M is the number of neurons with polynomial activation function in the hidden layer. The Polynomial activation function for the hidden neurons are given by

$$p_i(t) = \theta_{0i} + \theta_{1i} t + \theta_{2i} t^2 \quad (4)$$

substituting 4 into 8 gives

$$y(n) = w_{11} p_1(t) + w_{21} p_2(t) + \ldots + w_{M1} p_M(t) e(n) \quad (5)$$

$$\begin{aligned} y(n) &= w_{11}(\theta_{01} + \theta_{11} t + \theta_{21} t^2) + w_{21}(\theta_{02} + \theta_{12} t + \theta_{22} t^2) \\ &\quad + \ldots + w_{M1}(\theta_{0M} + \theta_{1M} t + \theta_{2M} t^2) \end{aligned}$$

where the input (t) to the activation part of the neuron for the AR section with order p is given by

$$t_i = \sum_{j=1}^{p} v_{ij} y(n-j) \quad (6)$$

that is

$$t_r = v_{1r} y(n-1) + v_{2r} y(n-2) + \ldots + v_{pr} y(n-p) \quad (7)$$

therefore,

$$\begin{aligned} y(n) &= w_{11}(\theta_{01} + \theta_{11}[v_{11} y(n-1) + v_{21} y(n-2)] \\ &\quad + \theta_{12}[v_{11} y(n-1) + v_{21} y(n-2)]^2) \\ &\quad w_{21}(\theta_{02} + \theta_{12}[v_{12} y(n-1) + v_{22} y(n-2)] \\ &\quad + \theta_{22}[v_{12} y(n-1) + v_{22} y(n-2)]^2) \end{aligned}$$

combining like terms and comparing coefficients we have

$$a_i = \sum_{j=1}^{M} w_{j1} \theta_{1j} v_{ij} y(n-i) \quad (8)$$

$$b_i = \sum_{j=1}^{M} w_{j1} \theta_{1j} v_{ij} y(n-i) \quad (9)$$

The weights (w_{j1}, v_{ij}) and the polynomial coefficients (θ_{1j}) are all obtained by the use of back propagation algorithm [**?**], [6], [7].

III. Increasing FFBPNN Accuracy and Convergence

The traditional Back propagation algorithm suffers from slow convergence thereby taking longer time in minimizing the objective Mean Square error (MSE) function. The MSE is given by

$$E(n) = \frac{1}{2} \sum_{j=1}^{N} (T - Y)^2 \quad (10)$$

where
T= Target data
Y=Output of the Output node
N= Number of output nodes or Neurons

The weight update equation in traditional back propagation algorithm is given by

$$Weight_{New} = Weight_{Old} + \Delta Weight \quad (11)$$

This is often express mathematically as

$$W(k+1) = W(k) + \Delta W(K+1) \quad (12)$$

and

$$\Delta W(K) = \mu \frac{\delta E}{\delta W}$$

therefore, eq. 12 becomes

$$W(k+1) = W(k) + \mu \frac{\delta E}{\delta W} \tag{13}$$

where μ is the learning rate , also

$$\frac{\delta E}{\delta W} = \delta_o Y^T$$

δ_o is the error signal for an output unit and is calculated as

$$\delta_o = (T - Y) f^1(net_{Input_{Hi}})$$

Eq. 13 becomes

$$W(k+1) = W(k) + \mu \delta_o Y^T \tag{14}$$

Which is the weight update equation for the output layer. Similarly the weight update equation for the hidden layers is given as

$$W(k+1) = W(k) + \mu \delta_h X^T \tag{15}$$

where

$$\delta_h = W_k \delta_o f^1(Input_{In}) \tag{16}$$

By the addition of momentum eq. 14 and eq. 15 becomes

$$W(k+1) = W(k) + \mu \delta_o Y^T + \alpha \Delta W(K-1) \tag{17}$$

and

$$W(k+1) = W(k) + \mu \delta_h X^T + \alpha \Delta W(K-1) \tag{18}$$

where α is the momentum term.

Several methods have been suggested in literatures [14]–[17] as method of increasing the speed and convergence capability of FFBPNN these include: Choice of initial weight, Method of weight update, variable learning rate and Momentum rate. In order to increase the speed of convergence and accuracy of this FFBPNN, an improved algorithm with the following characteristics is hereby proposed

✓Weight initialization using Nguyen and Widrow Method
✓Batch Weight and coefficients Update
✓Introduction of Adaptive Learning Rate
✓Introduction of Adaptive Momentum

A. Algorithm for calculating Back Propagation with Adaptive Weight and Adaptive Momentum

In order to simplify the derivation, the algorithm is as summarized below

1) **STEP 1:**
Initialize all weights (W_{ij}, V_{jk}), biases (ϕ_{ij}, ϕ_k) and Polynomial coefficients $(\theta_{0i}, \theta_{1i}, \theta_{2i})$
2) **STEP 2:**
Apply all the input vectors X to the input nodes
3) **STEP 3:**
Calculate the net inputs to the hidden nodes

$$NetInout = \sum W_{ij} X^T + \phi_{ij}$$

4) **STEP 4:**
Evaluate the output of the hidden nodes and their derivatives

$$Hidden = p_{i-hidden}(NetInout)$$

where

$$p_{i-hidden}(t) = \theta_{0i} + \theta_{1i} t + \theta_{2i} t^2$$

5) **STEP 5:**
Calculate the net inputs to the Output node

$$NetInout = \sum V_{ij} Hidden^T + \phi_k$$

6) **STEP 6:**
Evaluate the output of the Output node and its derivative

$$Hidden = p_{i-hidden}(NetInout)$$

where,

$$p_{i-hidden}(t) = t$$

7) **STEP 7:**
Evaluate the Error, $E(n)$
8) **STEP 8:**
Calculate

$$\Delta E = E(k) - E(k-1)$$

9) **STEP 9:**
Implement eq. 22 and eq. 21.
The difference in error between the present iteration and immediate past iteration determines if the learning rate and momentum should be increased or decrease. If the difference in error is greater than ψ , where $0 < \psi < 1$, then the learning rate and momentum are reduced otherwise they are increased by a factor β, where $0 < \beta < 1$.

$$\Delta E = E(k) - E(k-1) \tag{19}$$

$$W(k+1) = W(k) + \mu(k) \delta_h X^T + \alpha(k) \Delta W(K-1) \tag{20}$$

with

$$\mu(k+1) = \left\{ \begin{array}{ll} \Phi\mu(k) \ldots if \ldots \Delta E > 0; & 0 \le \phi \le 1 \\ \Phi\mu(k) \ldots if \ldots \Delta E < 0; & 1 \le \phi \le 1.9 \end{array} \right\} \tag{21}$$

also for the adaptive momentum $\alpha(k)$, we have

$$\alpha(k+1) = \left\{ \begin{array}{ll} \beta\alpha(k) \ldots if \ldots \Delta E > 0; & 0 \le \beta \le 1 \\ \beta\alpha(k) \ldots if \ldots \Delta E < 0; & 1 \le \beta \le 1.9 \end{array} \right\} \tag{22}$$

10) **STEP 10:**
Back Propagate the error from the output node to the hidden nodes
11) **STEP 11:**
Calculate new weights, biases and polynomial coefficients
12) **STEP 12:**
Stop if stopping criteria satisfied else repeat all except step 1

IV. Result and Discussion

In this paper, the effect of: weight initialization method; batch and incremental weight and coefficients update; adaptive learning rate; fixed and adaptive momentum rate for parametric AR or ARMA define by will be evaluated. The variance (σ^2) of the white noise input is 0.8.

1) Autoregressive Equation (AR)

$$y(n) = y(n-1) - 0.24y(n-2) + w(n) \qquad (23)$$

2) Autoregressive Moving Average (ARMA)

$$\begin{aligned} y(n) &= 0.13y(n-1) - 0.234y(n-2) \\ &\quad +0.67x(n) + 0.23x(n-1) \end{aligned}$$

A. Effect of Weight initialization methods on FFBPNN-ARMA technique

The result obtained from evaluating the effects of weight initialization on the convergence of FFBPNN-ARMA stated in Eq. 23 and 24 is as given in table I and II respectively. The plots of the Mean Square Error (MSE) against epoch are as shown in Fig. 3 and Fig. 4 for Eq. 23 and 24 respectively. The three methods of weight initialization considered are (a) Weight initialization by random number (b) Weight initialization by Normalized random number (c) weight initialization by the use of Nguyen and Widrow technique [17]. Results obtained shows that the use of Nguyen and Widrow technique converge to the expected MSE value faster than the two other techniques evaluated in this paper. Furthermore, it was also observe that the values of the coefficients obtained by the use of Nguyen and Widrow technique is more accurate than the other two method.

TABLE I
Effect of Initial Weight on FFBPNN-ARMA (AR, Eq. 23)

Methods	**MSE Value**	**No. Epoch**	$a(1)$	$a(2)$
Actual Value			1.0	-0.24
Random Number (RN)	0.001	66	0.9742	-0.2185
N. Random Number (NRN)	0.001	64	0.9756	-0.2205
Nguyem-Widrow (NW)	0.001	21	0.9777	-0.2265
Random Number (RN)	0.0001	75	0.9917	-0.2329
N. Random Number (NRN)	0.0001	52	0.9922	-0.2340
Nguyem-Widrow (NW)	0.0001	29	0.9941	-0.2367

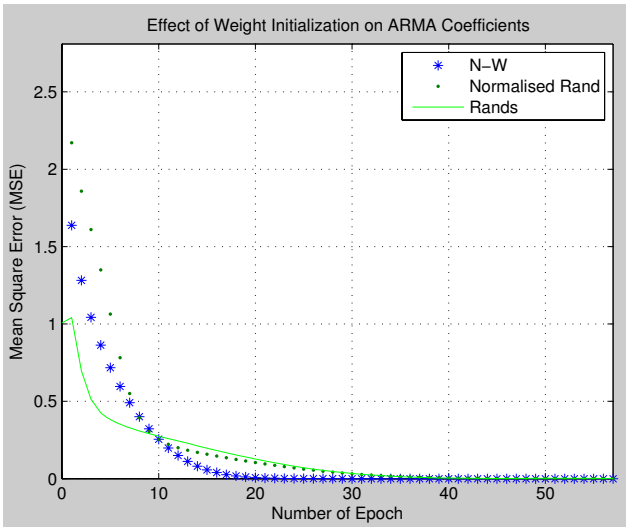

Fig. 3. Effect of Weight Initialization Techniques on FFBPNN-ARMA (100 Epoch, AR, Eq. 23)

TABLE II
Effect of Initial Weight on FFBPNN-ARMA (ARMA, Eq. 24)

Methods	**MSE Value**	$a(1)$	$a(2)$	$b(0)$	$b(1)$
Actual Value		0.13	-0.234	0.67	0.23
RN	0.001	0.1100	-0.1988	0.5991	0.2383
NRN	0.001	0.1219	-0.2221	0.6470	0.2403
NW	0.001	0.1197	-0.2323	0.6643	0.2385
RN	0.0001	0.1211	-0.2148	0.6199	0.2315
NRN	0.0001	0.1212	-0.2216	0.6347	0.2367
NW	0.0001	0.1315	-0.2362	0.6649	0.2301

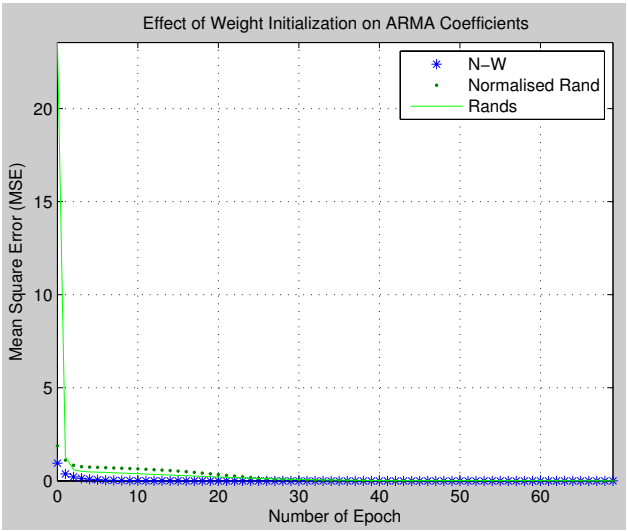

Fig. 4. Effect of Weight Initialization Techniques on FFBPNN-ARMA (ARMA, Eq. 24)

B. Effect of Batch Weight and coefficients updates on accuracy and convergence

In determining AR or ARMA coefficients using FFBPNN-ARMA, the ANN weights and the coefficients of the polynomial activation function must be updated at the same time. The effects of incremental and batch weight and coefficients updates are as shown in Table III and Table IV. Result obtained from series of experiments shows that the use of incremental or sequential weight and coefficients update converge to the required MSE value faster than the use of batch weight and coefficients update. Comparing result of the incremental weight and coefficients update obtained in Table III to Table I reveals that increasing the expected MSE value of FFBPNN-

ARMA syetm, the values of the coefficients obtained will almost be same with the actual value of the coefficients.

TABLE III
EFFECT OF BATCH WEIGHT AND COEFFICIENTS UPDATES ON FFBPNN-ARMA (AR, EQ. 23)

Methods	**Weight Type**	**MSE Value**	**No. Epoch**	$a(1)$	$a(2)$
Actual Value				1.0	0.24
Incremental	RN	10^{-5}	89	0.999	-0.240
Incremental	NRN	10^{-5}	63	0.996	-0.239
Incremental	NW	10^{-5}	41	0.998	-0.240
Batch	RN	10^{-5}	91	0.993	-0.234
Batch	NRN	10^{-5}	60	0.996	-0.238
Batch	NW	10^{-5}	53	0.999	-0.240

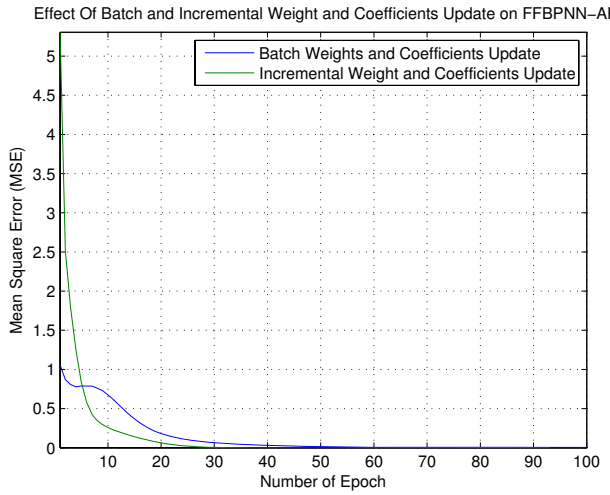

Fig. 5. Effect of Batch and Incremental Weight Update on AR

TABLE IV
EFFECT OF BATCH WEIGHT AND COEFFICIENTS UPDATES ON FFBPNN-ARMA (ARMA, EQ. 24)

Methods	**Weight Type**	**MSE Target**	$a(1)$	$a(2)$	$b(0)$	$b(1)$
Actual Value			0.13	-0.234	0.67	0.23
Incremental	RN	10^{-5}	0.112	-0.227	0.619	0.256
Incremental	NRN	10^{-5}	0.125	-0.245	0.635	0.239
Incremental	NW	10^{-5}	0.127	-0.234	0.653	0.245
Batch	RN	10^{-5}	0.125	-0.263	0.799	0.353
Batch	NRN	10^{-5}	0.128	-0.241	0.604	0.322
Batch	NW	10^{-5}	0.133	-0.230	0.647	0.285

C. Effect of addition of Momentum term on FFBPNN-ARMA

The introduction of momentum term to FFBPNN-ARMA as stated in Eq. (20), (22) and (21) are evaluated in this section. The results obtained (Fig. 7) reveals that the addition of the term greatly reduce the speed of convergence of the system. The percentage reduction in number of epoch varies from 10% to 20%, though the accuracy of the system was not significantly affected.

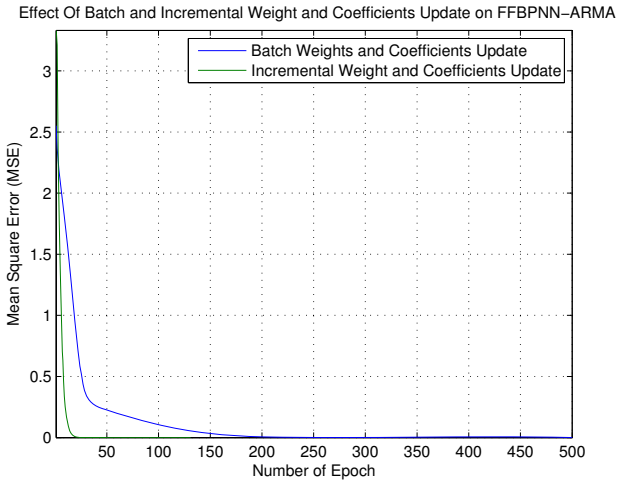

Fig. 6. Effect of Batch and Incremental Weight Update on ARMA

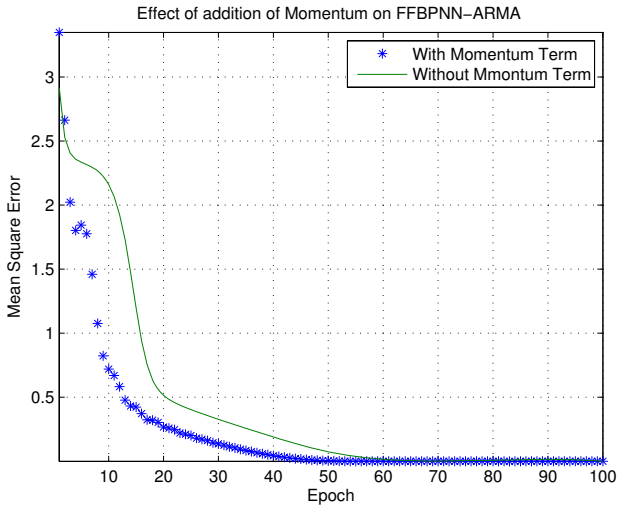

Fig. 7. Effect of addition of Momentum term

D. Effect of Fixed and Adaptive Momentum term on accuracy and convergence on FFBPNN-ARMA

The introduction of momentum term to FFBPNN-ARMA as discussed in subsection IV-C is evaluated in this section. Result obtained shows that the speed of convergence of FFBPNN-ARMA is greatly increased by the inclusion of this term in the back propagation algorithm. The percentage reduction in the number of epoch obtained by implementing adaptive momentum with batch weight and coefficients update ranges from 15% to 30%. Table V and Table VI shows the result obtained by comparing the effect of adaptive momentum (AM) term to fixed momentum (FM) term for FFBPNN system.

E. Effect of Fixed and Adaptive Learning rate and Momentum on accuracy and convergence on FFBPNN-ARMA

In this section, the effect of implementing both adaptive learning rate and adaptive momentum on FFBPNN-ARMA will be presented. Also the result obtained will be compared with the result obtained by the use of Incremental and batch

TABLE V
COMPARING THE EFFECT OF FIXED (F) AND ADAPTIVE (A) MOMENTUM ON FFBPNN-ARMA (AR, EQ. 23)

Methods	Weight Type	MSE Value	No. Epoch	$a(1)$	$a(2)$
Actual Value				1.0	0.24
Fixed Momentum (FM)	RN	10^{-5}	63	0.999	-0.240
FM	NRN	10^{-5}	47	0.999	-0.239
FM	NW	10^{-5}	38	0.999	-0.240
Adaptive Momentum (AM)	RN	10^{-5}	57	0.980	-0.34
AM	NRN	10^{-5}	39	0.969	-0.230
AM	NW	10^{-5}	23	1.001	-0.234

TABLE VI
COMPARING THE EFFECT OF FIXED (F) AND ADAPTIVE (A) MOMENTUM ON FFBPNN-ARMA (ARMA, EQ. 24)

Methods	Type	Epoch	a(1)	a(2)	b(0)	b(1)
Actual Value			0.13	-0.234	0.67	0.23
FM	RN	83	0.144	-0.228	0.656	0.225
FM	NRN	79	0.126	-0.229	0.581	0.235
FM	NW	63	0.170	-0.238	0.612	0.260
AM	RN	62	0.140	-0.234	0.651	0.348
AM	NRN	41	0.138	-0.214	0.652	0.310
AM	NW	36	0.133	-0.237	0.641	0.256

update of both the weights and polynomial coefficients. Result obtained by the addition of adaptive learning rate and adaptive momentum shows a significant reduction in the number of epoch required for convergence to the expected MSE value. The percentage reduction achieved varied from 20% to 50% of the result obtained when using the method reported in subsection IV-A. Furthermore, the accuracy of the coefficients is highly improved when compared with any of the techniques earlier discussed.

TABLE VII
COMPARING THE EFFECT OF FIXED (F) AND ADAPTIVE (A) LEARNING RATE AND MOMENTUM ON FFBPNN-ARMA (AR, EQ. 23)

Methods	Weight Type	MSE Value	Epoch	a(1)	a(2)
Actual Value				1.00	-0.24
Incremental	RN	10^{-5}	34	0.997	-0.229
Incremental	NRN	10^{-5}	17	0.999	-0.233
Incremental	NW	10^{-5}	16	1.010	-0.237
Batch	RN	10^{-5}	29	0.999	-0.241
Batch	NRN	10^{-5}	23	1.009	-0.241
Batch	NW	10^{-5}	18	1.001	-0.240

V. CONCLUSION

In this work, we present novel techniques of improving the accuracy and speed of convergence of a FFBPNN-ARMA system reported in [6], [7]. This system involves obtaining the coefficients of an ARMA system from the weights of the input and hidden layer neurons and the coefficients of a polynomial based activation function in a feed forward back propagation

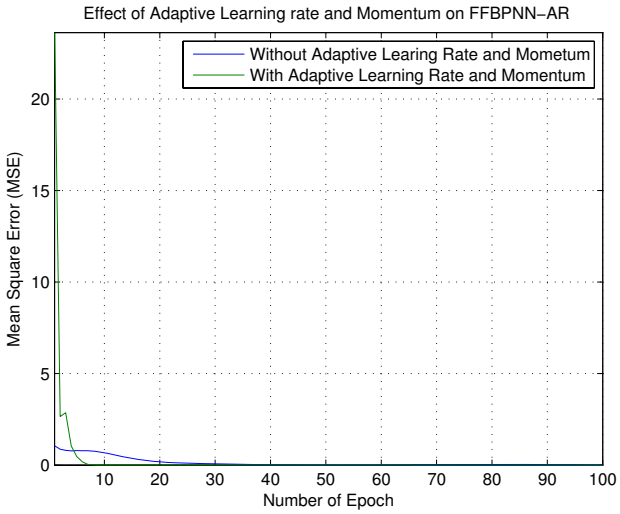

Fig. 8. Effect of adaptive Learning rate [α] and adaptive momentum μon FFBPNN-AR

TABLE VIII
COMPARING THE EFFECT OF FIXED (F) AND ADAPTIVE (A) LEARNING RATE AND MOMENTUM ON FFBPNN-ARMA (AR, EQ. 24)

Methods	Type	Epoch	a(1)	a(2)	b(0)	b(1)
Actual Value			0.13	-0.234	0.67	0.23
Incremental	RN	46	0.134	-0.238	0.597	0.240
Incremental	NRN	38	0.130	-0.228	0.581	0.211
Incremental	NW	31	0.170	-0.218	0.642	0.250
Batch	RN	16	0.156	-0.202	0.615	0.241
Batch	NRN	14	0.145	-0.224	0.600	0.245
Batch	NW	11	0.126	-0.231	0.641	0.258

artificial neural network (FFBPNN) system [6], [7]. Results obtained shows a reduction of 10% to 20% in number of epoch for a FFBPNN with the addition of momentum term to the traditional back propagation method of weight update in a feed forward artificial neural network system. In addition, a reduction of 15% to 30% in number of epoch was achieved by the introduction of adaptive momentum to the traditional back propagation system of equation while a significant 20% to 50% percentage reduction in number of epoch required to meet a specified MSE was accomplished by implementing batch weight and polynomial activation coefficients updates, adaptive learning rate and adaptive momentum to the system of traditional back propagation system of weight and coefficient update for a feed forward neural network system of equation. Furthermore, initializing the FFBPNN weight with the method of Nguyen and Widrow and normalized random number to the use of random value leads to great reduction in convergence time and number of epoch for a given minimum MSE. It was also observed that the use of random value for weight initialization sometimes leads to the FFBPNN converging at local mimina instead of global minima thereby giving less accurate value of the coefficients. The areas of application of this proposed algorithm include Magnetic Resonance Imaging reconstruction using parametric technique [2], [4], [12], signal and system modeling [9], [18], [19], Adaptive control, system and PID tunning.

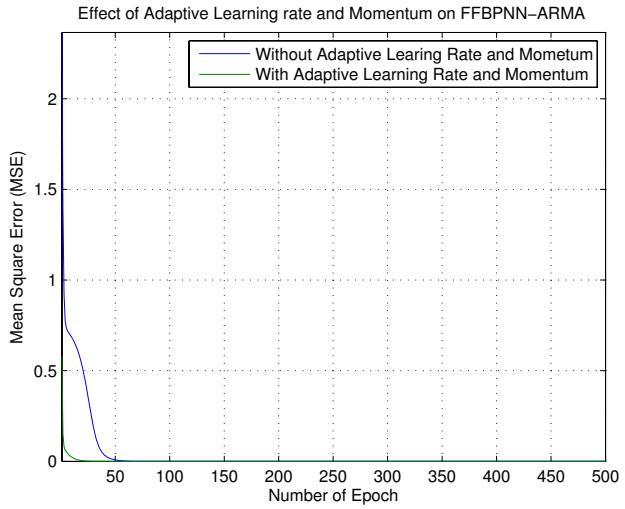

Fig. 9. Effect of adaptive Learning rate [α] and Adaptive Momentum μ on FFBPNN-ARMA

Acknowledgement: We shall like to express our appreciation to Prof. K. H. Chon for all his suggestion, advice and contribution to the success of this work.

REFERENCES

[1] Z. P. Liang, P. C. Lauterbur, "Principles of Magnetic Resonance Imaging, A signal processing perspective", IEEE Press, New York, 2000.

[2] A. M. Aibinu, M. J.E. Salami, A. A. Shafie and A. R. Najeeb, " Model Order Determination for MRI Signal", *accepted for publication , International Conference on Medical system Engineering (ICMSE)*, 2008, Singapore, August - September 2008.

[3] D. G. Nishimura, "Principles of Magnetic Resonance Imaging", April 1996.

[4] M. R. Smith, S. T. Nichols, R. M. Henkelman and M. L. Wood, "Application of Autoregressive Moving Average Parametric Modeling in Magnetic Resonance Image Reconstruction", *IEEE Transactions on Medical Imaging*, Vol. M1-5:3, pp 257 - 261, 1986.

[5] M. R. Smith, S. T. Nichols, R. Constable and R. Henkelman, "A quantitative comparison of the TERA modeling and DFT magnetic resonance image reconstruction techniques", *Magn. Reson. Med.*, Vol. 19 pp. 1-19, 1991.

[6] K. H. Chon, R. J. Cohen, "Linear and Non-Linear ARMA Model Parameter Estimation Using Artificial Neural Network", *IEEE Transactions on BioMedical Engineering*, Vol. 44, No 3, pp 168 - 174, 1997.

[7] K. H. Chon, D. Hoyer, A. A Armoundas, N-H Holstein-Rathlou and D. J Marsh, " Robust Nonlinear Autoregressive Moving Average Model Parameter Estimation Using Recurrent Artificial Neural Network", *Annals of BioMedical Engineering*, Vol. 27, pp 538-547, 1999.

[8] A. M. Aibinu, M. J. E. Salami, A. A. Shafie and A. R. Najeeb "Performance Evaluation of Autoregressive Moving Average (ARMA) coefficients determination methods", *accepted for publication , International Conference on Computer System (ICCS)* , 2008, Singapore, August- September 2008.

[9] M. H. Hayes, "Staitical Digital Signal processing and Modelling", John Wiley & Sons, Canada, 1996.

[10] E. C. Whitman, "The spectral analysis of discrete time series in terms of linear regressive models", *Naval Ordinance Labs Rep.*, NOLTR-070-109, White Oak, MD, June 23, 1974.

[11] Z. P. Liang, F. E. Boada, R. T. Constable, E. M. Haacke, P. C. Lauterbur, and M. R. Smith, "Constrained Reconstruction Methods in MR Imaging", Reviews of MRM, vol. 4, pp.67 - 185, 1992.

[12] E. Hackle and Z. Liang, "Superresolution Reconstruction Through Object Modeling and Estimation", *IEEE transactions in A.S.S.P*, 37: 592 - 595, 1989.

[13] R. Palaniappan, "Towards Optimal Model Oreder Selection for Autoregressive Spectral Analysis of Mental Tasks Using Genetic Algorithm", *IJCSNS International Journal of Computer Science and Network Security*, Vol. 6 No. 1A, January 2006.

[14] C. C. Yu, Bin-Da Liu "A Simple Procedure in Back Propagation Trainning", IEEE Trans. Autom. Control, vol. AC-19, pp. 529-535, 2001.

[15] C. C. Yu, Bin-Da Liu "A Backpropapgation Algorithm With Adaptive Learning Rate and Momentum coefficients", IEEE Trans. Autom. Control, pp. 1218-1223, 2002.

[16] S. Haykin, " Neural Networks: A comprehensive foundation, 2nd ed.", Eaglewood, Cliffs, NJ: Prentice Hall.

[17] D. Nguyen, B. Widrow, " Improving the learning speed of 2- Layer Neural Network by Choosing initial values of the adaptive weights " *Proc. Int. Joint Conference on Neural Networks*, Vol. 3, pp.21-26, July, 1990.

[18] J. Rissanen, "Modelling by shortest data description", Automatica, vol.14, pp.465-471, 1978.

[19] M.J. Salami, A. R. Najeeb, O. Khalifa, K. Arrifin, "MR Reconsturction with Autoregressive Moving Average", *International Conference on Biotechnology Engineering*, Kuala Lumpur, pp 676 - 704, May, 2007.