International Journal of Electrical Power System and Technology
Vol. 1: Issue 1
www.journalspub.com

Dynamic System Identification for Looper Tension Control Using BPNN and Kalman Filtering

V.D. Bavdhane*
Department of Electrical Engineering, SKN Sinhgad College of Engineering, Pandharpur, Maharashtra, India

Abstract

System identification is an important area in control system. This paper discusses some of the reasons that cause the slow convergence of Back Propagation Neural Network (BPNN). This paper also comments on the effect of the number of neurons in the hidden layer when we apply BPNN with Kalman Filtering to dynamic system identification for a looper tension control. BPNN is based on LMS, and uses steepest descent method to find the optimum weights to the adjacent layer. It always consumes much time while training and it is not easy to get a global optimal value while applying to on-line training. The Levenberg–Marquardt algorithm (LMA) provides a numerical solution to the problem of minimizing a function, generally nonlinear, over a space of parameters of the function. Kalman Filtering is a better linear and discrete method for parameters estimation. By this way to solve a problem, it can involve the initial conditions, and also can be applied to stationary and non-stationary systems. So, applying BPNN and Kalman Filtering together to the dynamic system identification will give a better result both on convergent efficiency and stability.

Keywords: *BPNN, LMA, Kalman Filtering, system identification*

***Author for Correspondence** E-mail: vivek.bavdhane@gmail.com*

INTRODUCTION

When treating a control system, first of all, we have to infer the governing equation or transfer function based on some physical laws. However, it is very difficult to infer a mathematic model for a real system. Therefore, we need some experimental or numerical methods to establish the system's model. This paper describes a method that can approach to the real system. It yielded results which were very close to the real system.

Back Propagation Neural Network (BPNN) has a good performance in the function approximation [1]. Applying BPNN to the system identification can acquire the accuracy of requirement easily. The weighting between adjacent layer's neurons been adjusted by a systematic method to achieve an adaptive effect. BPNN tries to get the minimum mean square error between the expected value and the output value at the output layer by adjusting the weights. As input-output mapping may be highly non-linear it is difficult to get the optimum weighting function.

The traditional technology of optimization is based on a deterministic criterion function, usually, using the sum of the mean square error between the expected value and the output of the network for total samples. Therefore, when using the steepest descent method it should adopt the way of batch mode theoretically. But this is incompetent to real time weighting changing situation. Contrastively, the technology developed by the method of stochastic can adjust the weighting in real time according to the information of the past and the present. The Levenberg-Marquardt (LM) algorithm is an iterative

technique that locates the minimum of a function that is expressed as the sum of squares of nonlinear functions. Kalman Filtering is a famous method in linear system estimation based on the method of stochastic. So, combining BPNN and Kalman Filtering to system identification gives a satisfactory result in convergence efficiency and a consistency result [2, 3].

The plant model is a rolling mill that uses a looper for tension control. Tension is caused by the difference between the exit speed v1 in stand one and the entry speed v2 in stand two. Looper height H(t) is related to looper storage length L(t) by the mass conservation principle and can be empirically approximated by:

$$H(t)=1/(\alpha+\beta\sqrt{L(t)}) \quad (1)$$

α= 0.0001955567 and β = 0.028845145

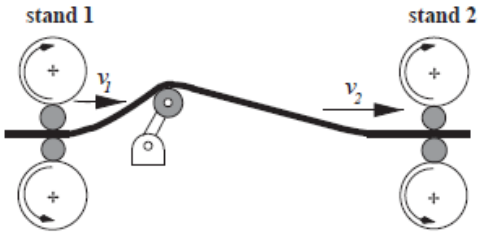


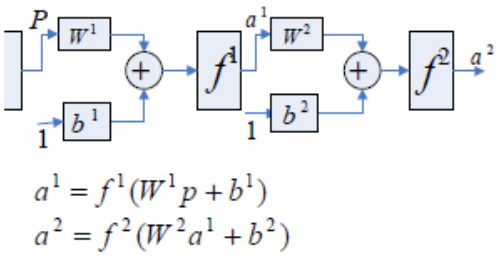
Fig. 1: *A Rolling Mill with Looper Control.*

NEURAL NETWORK APPROACH

Back propagation (BP) algorithm is the one of the extended algorithm of Least Mean Square (LMS) used to train the multilayer neural network. It uses Mean Square Error (MSE) as the index of performance function. To find the direction of the optimum weight adjustment, this method uses the steepest decent method during training process. The transfers function of neurons f^1 *and* f^2 can be a nonlinear differentiable function like 'logsig', 'tansig' etc. as shown in Figure 1. It has a good ability of function approach and pattern recognition [1, 4].
The weight updation rule is the function of error which is given as,

$$\frac{\partial F(w)}{\partial w} = e(k)\frac{\partial e(k)}{\partial w} \quad (1)$$

Where, *F(w)* is performance function.



$$a^1 = f^1(W^1 p + b^1)$$
$$a^2 = f^2(W^2 a^1 + b^2)$$

Fig. 2: *The Sketch of Two Layers NN.*

This algorithm uses the estimated value of error function at every iteration which causes slow convergence. For the multilayer network the performance curve may fall into local minima and cannot find the global optima. Kalman filter can help in these conditions.

LEVENBERG MARQUARDT ALGORITHM (LMA)

The Levenberg-Marquardt (LM) algorithm is an iterative technique that locates the minimum of a function that is expressed as the sum of squares of nonlinear functions. This algorithm appears to be the fastest method for training moderate-sized feed forward neural networks (up to several hundred weights). It also has a very efficient MATLAB implementation, because the solution of the matrix equation is a built-in function, so its attributes become even more pronounced in a MATLAB setting.

The main drawback of the Levenberg-Marquardt algorithm is that it requires the storage of some matrices that can be quite large for certain problems. The size of the Jacobian matrix is *Q* x *n*, where *Q* is the number of training sets and *n* is the number of weights and biases in the network.

If mem_reduc is set to 1, then the full Jacobian is computed, and no memory

reduction is achieved. If mem_reduc is set to 2, then only half of the Jacobian is computed at one time. This saves half the memory used by the calculation of the full Jacobian.

KALMAN FILTERING

Kalman Filtering is an optimum parameter estimator that can be used for linear and non-linear systems. The linear discrete model of Kalman Filtering is shown in Figure 3. The state space equations are introduced by Eq. (5) and (6) [2, 5]:

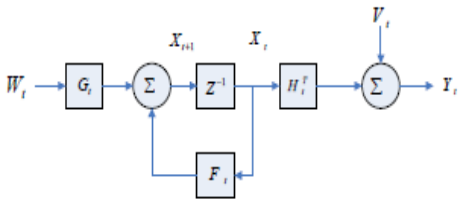


Fig. 3: *Structure of Kalman Filtering.*

$$X_{k+1} = F_k X_k + G_k W_k \quad (2)$$

$$Y_k = H_k^Y X_k + V_k \quad (3)$$

Where,

$\{X_k\}_{k\geq 1}$: State Variable

$\{Y_k\}_{k\geq 1}$: Observation or Output Value

$\{W_k\} = \{W_1, W_2, ...\}$: Input Noise

$\{V_k\} = \{V_0, V_1, ...\}$: Noise of Measurement

$E\{W_k W_i^T\} = Q_k \delta_{ki}$: Covariance of Input Noise

$E\{V_k V_i^T\} = R_k \delta_{ki}$: Covariance of Measurement Noise

$$K_k = F_k P_{k\backslash k-1} H_k \left(H_k^T P_{k\backslash k-1} H_k + R_k\right)^{-1} \quad (4)$$

$$X_{k\backslash k} = X_{k\backslash k-1} + P_{k\backslash k-1} K_k \left(Y_k - H_k^T X_{k\backslash k-1}\right) \quad (5)$$

$$P_{k\backslash k} = P_{k\backslash k-1} K_k H_k^T P_{k\backslash k-1} \quad (6)$$

$$X_{k+1\backslash k} = F_k X_{k\backslash k} \quad (7)$$

$$P_{k\backslash k+1} = F_k P_{k\backslash k} F_k^T + G_k Q_k G_k^T \quad (8)$$

Eq. (4) gives Kalman Gain; Eq. (5) and (6) are the measurement update equations; Eq. (7) and (8) are the time update equations.

Intuitively, there is a definite meaning in Eq. (5). It reveals that the estimated value of state filtering at k^{th} step is based on the estimated value that has been calculated by the data of the last step, and $(Y_k - H_k^T X_{k\backslash k-1})$ the correction to the measurement value at present. The amount of calibration been adjusted by the K_k. With this algorithm convergence is faster however, stability is not satisfactory. Hence, BPNN and Kalman filtering are together applied to the looper tension control problem to achieve both better convergence and stability.

BPNN AND KALMAN FILTERING ALGORITHM

Figure 4 shows the operation of a neuron which can be divided into two parts, one is the linear summation part and another one is the nonlinear functional part. This paper applies the algorithm of Kalman Filtering to get the optimal weighting and a better convergence efficiency of the network at the linear part.

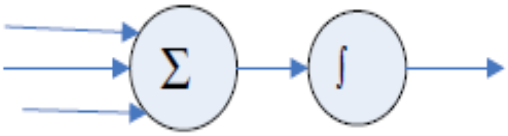


Fig. 4: *Operation inside a Neuron.*

At the nonlinear part, we still apply the steepest descent method to estimate the total output with a neuron. It means that we can use $-\Delta E/\Delta y^j_k$ as the adjustment direction of y^j_k, so, the estimation of the total summation with a neuron can be written as Eq. (9) and (10).

$$d_k^L = f^{-1}(t_k) \quad (9)$$

$$d_k^j = y_k + \mu\delta_k^j \quad j = 2,3\cdots L\text{-}1 \quad (10)$$

At the linear part, we can establish a Kalman filtering model for each neural shown as Figure 5.

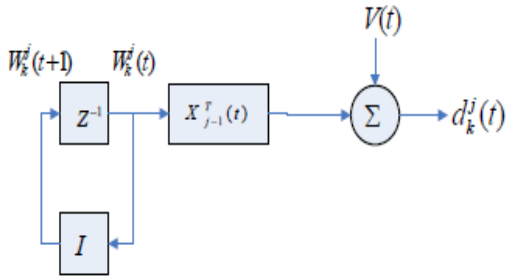


Fig. 5: *Kalman Filtering Model for Each Neuron.*

To compare the Figure 3 and 5, we can write down Eq. (14)~Eq. (16).

$$W_k^j(t+1) = W_k^j(t) \quad (11)$$

$$d_k^j(t) = y_k^T(t) + V(t) \quad (12)$$

$$y_k^T = X_{k-1}^T W_k^T + V(t) \quad (13)$$

Where j = 2,3,...,L ; k = 1,2,...,N_j , $W^j_k = [W^j_{k0}, W^j_{k0}, ..., W^j_{kNj-1}]^T$; $X_j = [W^j_0, W^j_1, ..., W^j_{kNj-1}]^T$; $X_j = [X^j_0, X^j_1, ..., X^j_{Nj}]^T$, $X^j_0 = 1$, and V(t) is the covariance matrix of Gaussian white noise with zero mean value. Kalman Filtering algorithm gives iteration learning equations as follows:

$$W_k^L(t) = W_k^L(t-1) + K_L(t)\left[f^{-1}(t) - y_k^L(t)\right] \quad (14)$$

$$W_k^j(t) = W_k^j(t-1) + K(t)\mu\delta_k^j \quad j=2,3..L-1 \quad (15)$$

L, j**:** express the last and hidden layer,
*k***:** express one of the neuron in the hidden layers.

Where,

$$K_j(t) = P_j(t-1)X_{-1}(t)\left[X_{j-1}^T(t)P_j(t-1)X_{j-1}(t) + R(t)\right]^{-1} \quad (16)$$

$$P_j(t) = P_j(t-1) - K(t)X_{j-1}^T(t)P_j(t-1) \quad (17)$$

SIMULATION RESULTS OF LOOPER TENSION CONTROL PLANT MODEL

Figure 6 indicates that the convergence of BPNN is very slow. At some point, it either rapidly converges, or jumps to a new level where it would again make little or no progress for quite a while.

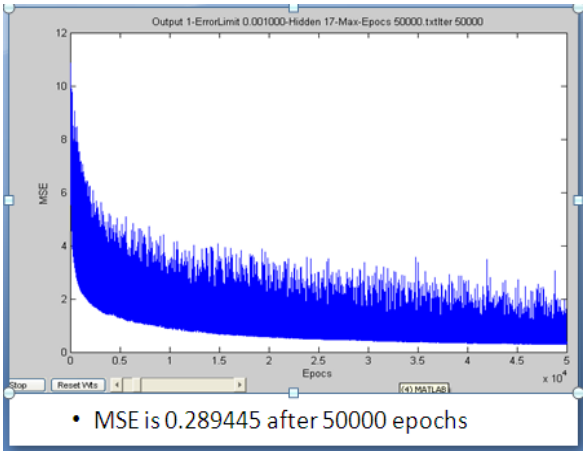


Fig. 6: *MSE v/s Epochs for BPNN Algorithm.*

Figure 7 indicates faster convergence than BPNN. Performance goal of 1e-005 MSE is obtained after 356 epochs. LMA is fastest converging algorithm than BPNN with steepest descent.

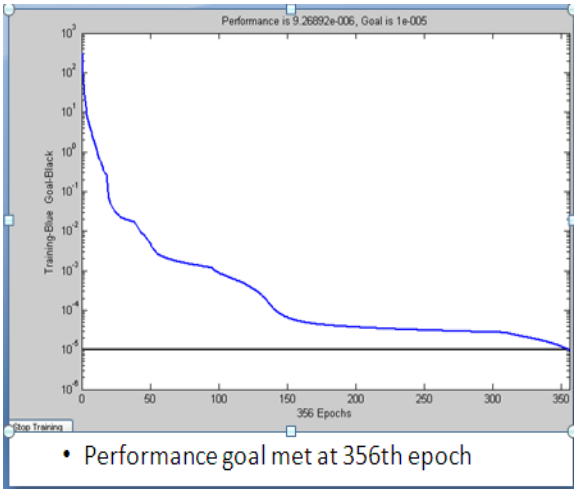


Fig. 7: *MSE v/s Epochs for LMA.*

Figure 8 indicates convergence of MSE within just 30 epochs for a goal of 1e-005. In this case extended Kalman filtering algorithm is used.

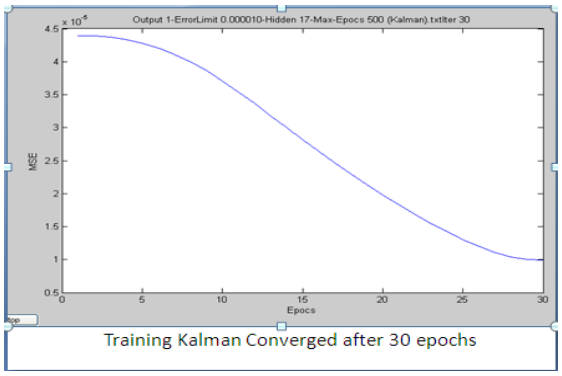


Fig. 8: *MSE v/s Epochs for EKF+BPNN.*

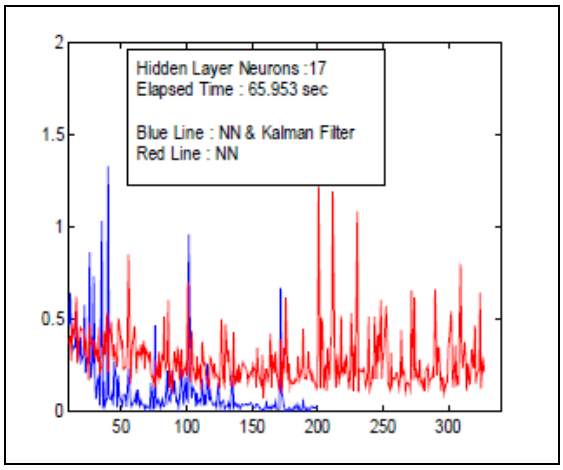


Fig. 9: *The MSE Variance of New Algorithm and BPNN While Training.*

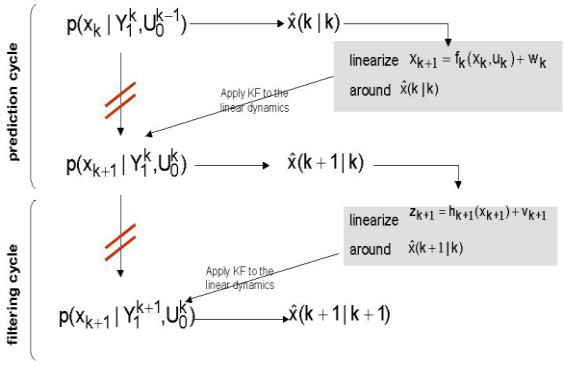
LIMITATIONS OF KALMAN FILTERING

1 The Kalman filter assumes that the noise properties are known. What if we do not know anything about the system noise?

2 The Kalman filter minimizes the "average" estimation error. What if we would prefer to minimize the worst-case estimation error? [6]

EXTENDED KALMAN FILTERING

The Extended Kalman filter (EKF) gives an approximation of the optimal estimate. The non-linearities of the systems' dynamics are approximated by a linearized version of the non-linear system model around the last state estimate. For this approximation to be valid, this linearization should be a good approximation of the non-linear model in the entire uncertainty domain associated with the state estimate.



Extended Kalman FILTER DYNAMIC CONCEPT

One iteration of the EKF is composed by the following consecutive steps:

1. Consider the last filtered state estimate ˆx(k|k),
2. Linearize the system dynamics, xk+1 = f(xk) + wk around ˆx(k|k),
3. Apply the prediction step of the Kalman filter to the linearized system dynamics just obtained, yielding ˆx(k + 1|k) and P(k + 1|k),
4. Linearize the observation dynamics, yk = h(xk) + vk around ˆx(k + 1|k),
5. Apply the filtering or update cycle of the Kalman filter to the linearized observation dynamics, yielding ˆx(k + 1|k + 1) and P(k + 1|k + 1).

Let F(k) and H(k) be the Jacobian matrices of f(.) and h(.), denoted by
F(k) = 5fk |ˆx(k|k)
H(k + 1) = 5h |ˆx(k+1|k)

The Extended Kalman filter algorithm is stated below:

Predict Cycle

ˆx(k + 1|k) = fk(ˆx(k|k))
P(k + 1|k) = F(k)P(k|k)FT (k) + Q(k)

Filtered Cycle

ˆx(k + 1|k + 1) = ˆx(k + 1|k) + K(k + 1)[yk+1 − hk+1(ˆx(k + 1|k))]

$$K(k+1) = P(k+1|k)H^T(k+1)[H(k+1)P(k+1|k)H^T(k+1) + R(k+1)]^{-1}$$
$$P(k+1|k+1) = [I - K(k+1)H(k+1)]P(k+1|k)$$

Contrary to the Kalman filter, the EKF may diverge, if the consecutive linearizations are not a good approximation of the linear model in the entire associated uncertainty domain [7–12].

CONCLUSIONS

The Kalman Filtering algorithm can give better performance to the dynamic system identification. From the example shown above, the following results should be highlighted:

1. The robustness of the new algorithm is better than the BPNN algorithm. It can always find the optimal weighting no matter what the initial weighting of the network is. By BPNN algorithm, the final weighting of the network is always affected by the initial weighting, so, we must run many times to find the optimal weighting.
2. It is not necessary to use too many neurons at the hidden layer to keep off the over fitting happen, and can save more memory storage.
3. The new algorithm is more predictable in its training. We notice that the backpropagation algorithm tends to reach a certain mean-squared error and remain there for quite a while making little or no progress.
4. The convergence of the back propagation algorithm depends heavily on the magnitude of the initial weights. If chosen incorrectly, the algorithm takes a long time to converge.

The new algorithm consumed much time for each iteration step, but it can save more numbers of iteration as Kalman filtering has good performance to the parameter estimation. Further performance can be improved by using Extended Kalman Filtering.

REFERENCES

1. Hagan Martin T. *Neural Network Design*. PWS. 1999.
2. Kung-Thin Lou. *On the Study of the Multilayer Neural Networks and Kalman Filtering for System Identification and Control.* Project of NSC. 1993.
3. Selero RS, Tepedelenliolu. A Fast Algorithm for Training Feedforward Neural Networks. *IEEE Trans. Signal Process.* 1992; 40(1): 202~210p.
4. Simon. Neural Networks-A Comprehensive Foundation. *PTR.* 1999.
5. Han Chen Thin. *Control Systems of Adaptive Technology.* Press. 2002.
6. www.innovatia.com/software/papers/minimax.htm
7. en.wikipedia.org/wiki/Extended_Kalman_filter
8. *www.cs.ucf.edu/~mikel/Research/.../kalman-filters-a-tutorial.pdf* http://users.isr.ist.utl.pt/~mir/pub/kalman.pdf
9. Narendra KS. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Trans. Neural Netw.* Mar 1991; 2(2): 252–262p.
10. Janabi-Sharifi F, A Neuro-Fuzzy System for Looper Tension Control in Rolling Mills.
11. Lary DJ, Mussa HY. Using an Extended Kalman Filter Learning Algorithm for Feed-Forward Neural Networks to Describe Tracer Correlations. *Atmos. Chem. Phys. Discuss.* 2004; 4: 3653–3667p.