CHAPTER 4

IMPLEMENTATION OF ADAPTIVE SINGLE EWMA CONTROLLER

In this chapter, we first implement the proposed neural-based adaptive single EWMA controller. After that, an enhanced neural adaptive single EWMA controller will also be evaluated. Both proposed adaptive algorithms would be compared with the Patel-Jenkins adaptive algorithm that was mentioned in Section 2.2.3.

4.1 Implement Neural-Based Adaptive Single EWMA Controller

This section will implement the proposed neural-based adaptive single EWMA controller that was mentioned in Section 3.1. The proposed controller will be implemented by using the software of Matlab/Simulink. In this section, we first present the training result of the neural network. After that, we will make comparisons between the proposed NN-based and the Patel-Jenkins adaptive algorithms through implementing three examples.

4.1.1 Off-Line Training the Neural Network

The training data sets were generated by simulating different combinations of the disturbance and controller parameters. There were a total of 121 data sets (i.e. $\theta \in [0,1], \lambda \in [0,1]$) which implied that we had a total of 121 SACF patterns. We used 30 data sets to be the testing data, and the remainder to be the training data. A useful guide was provided by Box and Jenkins (1976), p. 33, who suggested that the size of time series (*t*) be at least 50 and lags (*h*) to analyze the series at most $t/4$. Thus, we simulated 50 runs at each simulation, and took 12 lags in each SACF and SPCAF pattern.

The learning rate we set to train the neural network was 0.15, and the momentum coefficient was 0.9. The summary of the training result was shown in table 4.1. The 12-17-1 network was the best network for the data sets, because of the lower training and testing RMSE (root mean square error). Thus, we utilized the 12-17-1 network structure to implement the NN-based EWMA controller on line in the following examples.

Table 4.1 Summary of the training result

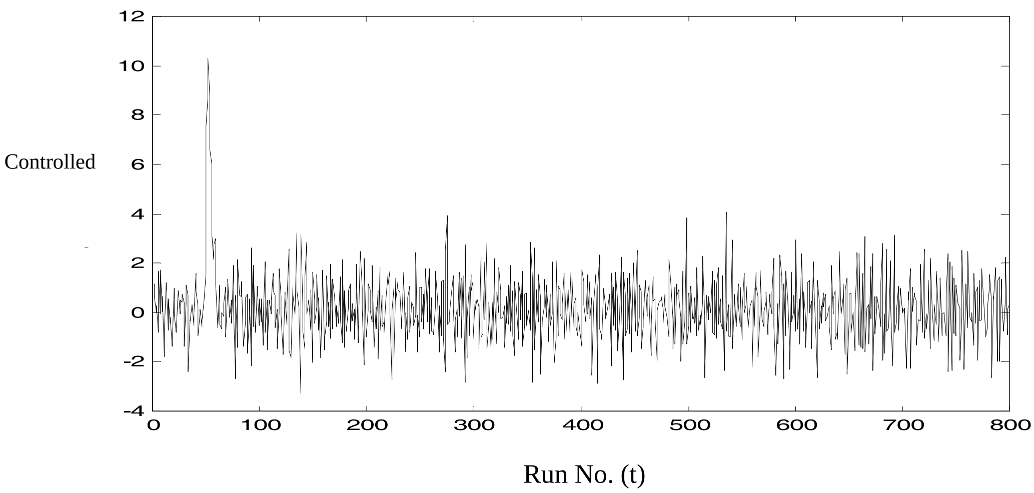
Structure	Training RMSE	Testing RMSE
12-15-1	0.0191	0.0213
12-16-1	0.0183	0.0210
12-17-1	0.0166	0.0198
12-18-1	0.0175	0.0220
12-19-1	0.0182	0.0232

4.1.2 Step Disturbance Model

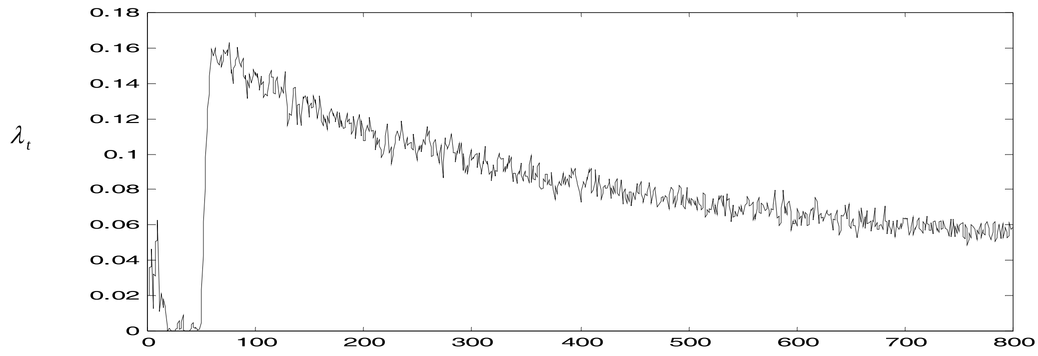
We first considered the example from Patel and Jenkins (2000). The step disturbance model can be expressed as follows:

$$N_t = \begin{cases} \Omega & t \geq t_s \\ 0 & t < t_s \end{cases} \tag{4.1}$$

where Ω is the level of the step change disturbance and t_s is the time of the disturbance introduced into the process. The tuner parameters in the Patel-Jenkins system [Equation (2.11)] were set to be the same as their simulation example. They were $\sigma_{\varepsilon}^{2}=1$, $\mu_{0}=0.1$, $\zeta_{0}=1$, $\delta=10^{-4}$, $\tau=0.005$, and the step disturbance is introduced at run 50 with $\Omega=10$. Figure 4.1(a) shows the controlled process output and Figure 4.1(b) plots the EWMA gain λ_{t} through 800 runs. On the other hand, the trained network structure 12-17-1 which was implemented on line to tune the EWMA controller gain under the step disturbance was also introduced at run 50 with magnitude 10. Figure 4.2(a) shows the NN-based controlled process output and Figure 4.2 (b) plots the NN-based EWMA gain λ_{t} through 800 runs. As expected, λ_{t} increased on a shift, and decreased to a small number. The uncontrolled inflation factor ($\hat{\sigma}_{y}^{2}/\hat{\sigma}_{\varepsilon}^{2}$) was 5.2795, and the controlled inflation factor under the Patel-Jenkins method was 1.8521, and 1.4401 in the NN-based EWMA controller. Thus, the performance of the NN-based controller was superior.

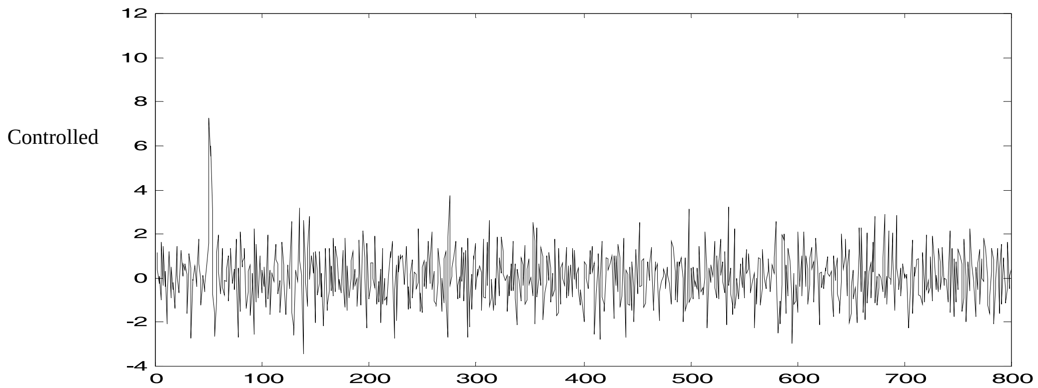


(a)

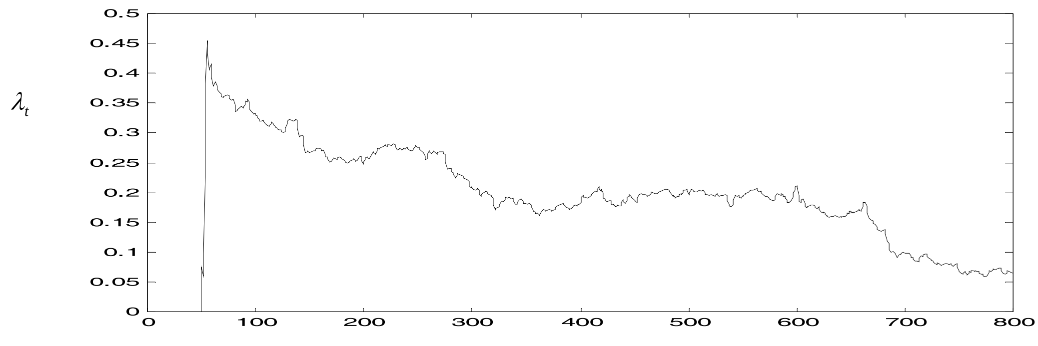


(b)

Figure 4.1 Patel-Jenkins adaptive method (a) controlled output (b) EWMA gain



(a)



(b)

Figure 4.2 NN-based adaptive method (a) controlled output (b) EWMA gain

4.1.3 IMA(1,1) Disturbance Model

In this example, we considered the IMA(1,1) disturbance model with moving average parameter $\theta = 0.2$ and $\sigma_{\varepsilon}^{2} = 1$. We knew that the optimal controller parameter was $\lambda^{*} = 1 - \theta$. Assume the IMA(1,1) disturbance model was introduced at run 50 over 800 runs. Figure 4.3 shows the EWMA gain under NN-based adaptive controller. The value of λ_t tended to 0.7846 (taking the sample mean of the last 100 runs). This was close to the optimal controller parameter of 0.8. The NN-based adaptive controlled inflation factor after 50 run was 1.025 which implies that the increased standard deviation (ISD) was 2.4672 %, and the ISD under Patel-Jenkins adaptive algorithm was 6.5762%. Thus, the NN-based adaptive algorithm produces a lower inflation in the controlled process.

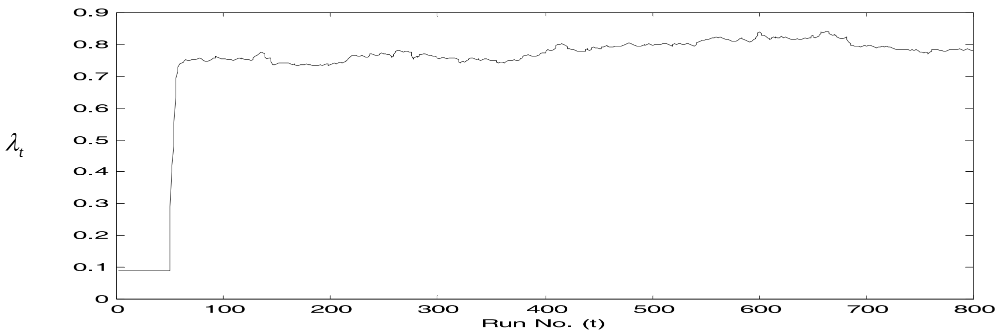


Figure 4.3 NN-based adaptive EWMA gain under IMA(1,1)

4.1.4 Trend Disturbance with Slow Ramp Rates

The Chemical Mechanical Planarization (CMP) is a very critical step for the Very Large Scale Integrated (VLSI) manufacturing. The objective of CMP is to obtain global within-wafer planarization. It is well known that the polish pad tends to wear-out with use, leading to a trend process in remove rate which needs to be compensated for. Therefore, this example will simulate the environment of the CMP process and apply the proposed approach to control it.

Consider the trend disturbance model which can be expressed as:

$$N_t = \begin{cases} S(t-t_s) & t \ge t_s \\ 0 & t < t_s \end{cases} \tag{4.2}$$

where S is the trend rate. The optimal EWMA controller parameter under trend disturbance can be solved by the following equation:

$$\sigma_\varepsilon^2 \lambda^3 - S^2\lambda^2 + 4S^2\lambda - 4S^2 = 0 \tag{4.3}$$

Assuming the trend disturbance with $S = 0.1$ and $\sigma_\varepsilon^2 = 1$ was introduced at run 50. Figure 4.4 shows the EWMA gain under NN-based adaptive method. Taking the sample mean of the last 100 runs, the value of λ_t tended to 0.3057 which was very close to the optimal value of 0.3061 which was obtained by solving Equation (4.3). The NN-based controlled inflation factor was 1.5919, and 2.0914 under Patel-Jenkins adaptive algorithm.

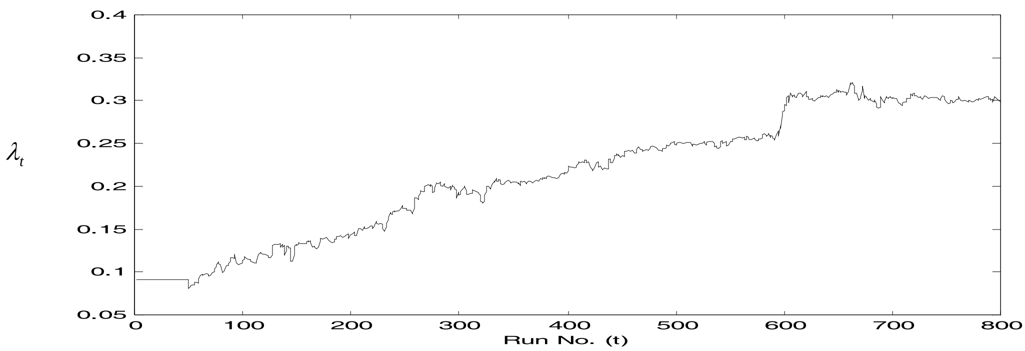


Figure 4.4 NN-based adaptive EWMA gain under trend disturbance

4.1.5 Discussion and Concluding Remarks

The effect of improperly setting the EWMA controller parameter would inflate the controlled process output variance has been demonstrated in this study. We have shown that the NN-based adaptive approach possesses better performance than the Patel-Jenkins adaptive algorithm on the controlled process output. Furthermore, the proposed system has been shown to be a stable system. From Section 4.1.2, as we expected the NN-based EWMA gain tended to a small value with time when a step disturbance model was introduced to the process. Sections 4.1.3 and 4.1.4 showed the EWMA gain behaving close to the optimal controller parameter when the IMA(1,1) and trend disturbance existed in the process. The proposed methodology could update the EWMA gain automatically, which would reduce the needs for operators to tune recipes in the process. Although the proposed methodology was implemented via simulation, nevertheless it is anticipated to improve the performance of the EWMA controller on an actual process.

4.2 Implement Enhanced Neural Adaptive Single EWMA Controller

4.2.1 Off-Line Training the Enhanced Neural Network

The training data sets were generated by simulating different combinations of the disturbance and controller parameters. There were a total of 121 data sets (i.e. $\theta \in [0,1], \lambda \in [0,1]$) which implied that we had a total of 121 SACF/SPACF patterns. We used 30 data sets to be the testing data, and the remainder to be the training data.

The considered case included 24 nodes at the input layer, and one node at the output layer. The problem in a full-connected neural network is to determine the number of neurons in the hidden layer. A trial and error approach was used to determine that a single hidden layer with 22 neurons formed the required structure for the considered problem. In order to improve the network performance, a 2^2 factorial design were used to find the learning rate and momentum constants (see Table 4.2). The factors involved in this design are the learning rate and the momentum constants, and the response variable was the number of epochs used to achieve the desired level

(0.01) of the root-mean-square (RMS) error. It can be observed that the best network performance (with smallest number of epochs) was achieved when the learning rate 0.15 and the momentum constant was 0.85. Figure 4.5 shows the learning behavior versus iterations of the selected network structure; it indicates that the network learns very fast and only required 3393 iterations (about 28 epochs). Thus, we will utilize the 24-22-1 network structure to implement the NN-based EWMA controller on line in the following examples.

Table 4.2 Experimental design results

Run	Learning Rate	Momentum Constant	Epoch
1	0.35	0.85	54
2	0.15	0.85	28
3	0.35	0.95	51
4	0.15	0.95	37

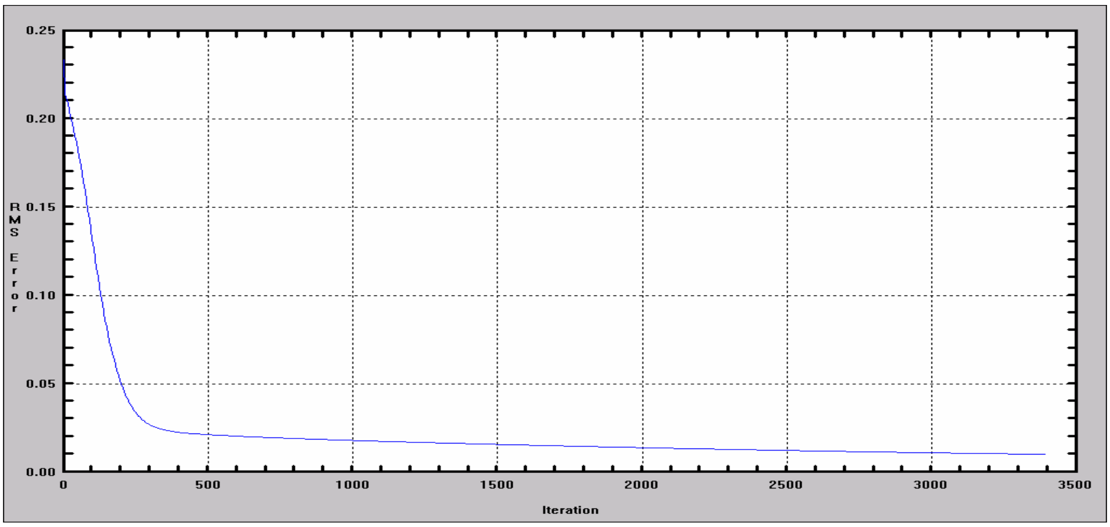
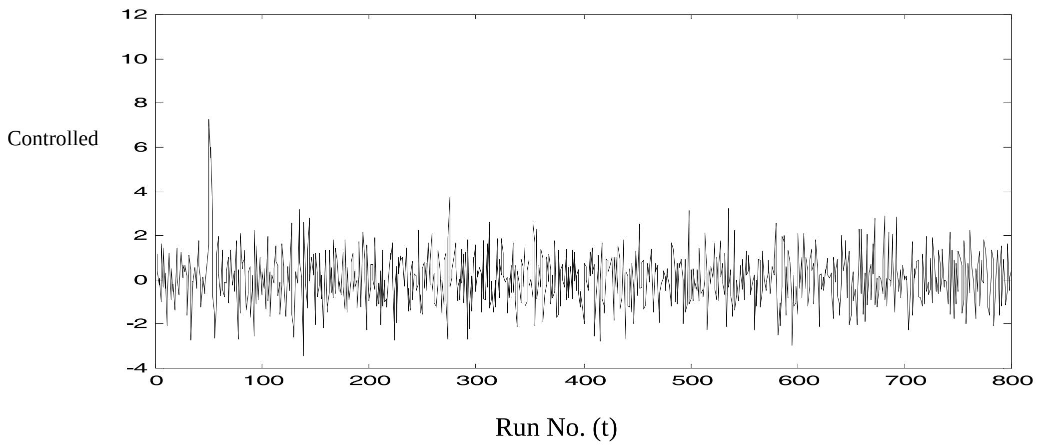


Figure 4.5 Enhanced neural network learning behavior

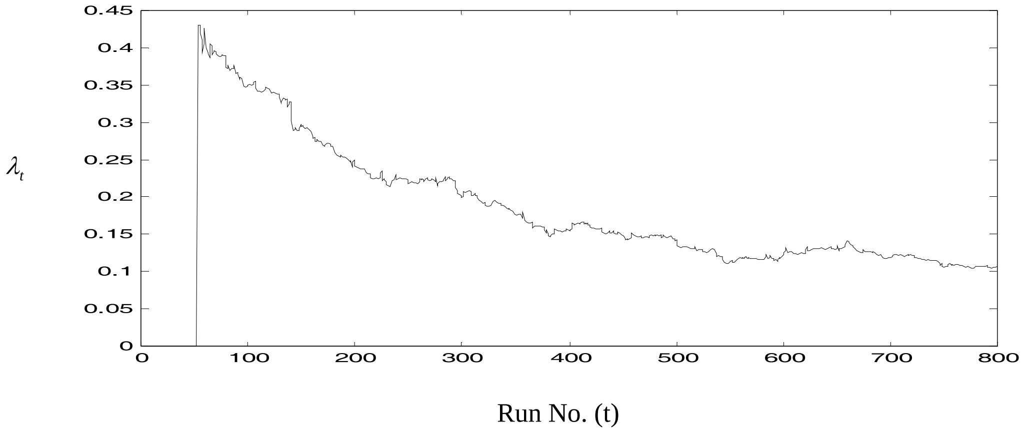
4.2.2 Step Disturbance Model

The step disturbance model was expressed in Equation (4.1). The tuner parameters in the Patel-Jenkins system were set to be the same as their simulation example. They were $\sigma_\varepsilon^2 = 1$, $\mu_0 = 0.1$, $\zeta_0 = 1$, $\delta = 10^{-4}$, $\tau = 0.005$, and the step disturbance was introduced at run 50 with $\Omega = 10$. Figure 4.1(a) shows the controlled process output under the Patel-Jenkins approach, and Figure 4.1(b) plots the EWMA gain λ_t through 800 runs.

The off-line trained network was implemented on line to tune the EWMA controller gain under the step disturbance which was also introduced at run 50 with magnitude 10. Figure 4.6(a) shows the NN-based controlled process output, and Figure 4.6(b) plots the enhanced NN EWMA gain λ_t through 800 runs. As expected, λ_t increased on a shift, and decreased to a small number with time. The performance of the uncontrolled process measured in the inflation factor ($\hat{\sigma}_y^2 / \hat{\sigma}_\varepsilon^2$) was 5.2795, the controlled inflation factor under the Patel-Jenkins method was 1.8521, and 1.3321 in the enhanced NN-based EWMA controller. Therefore, the enhanced NN adaptive EWMA controller possessed superior performance.



(a)



(b)

Figure 4.6 Enhanced NN adaptive method (a) controlled output (b) EWMA gain

4.2.3 IMA(1,1) Disturbance Model

The IMA(1,1) disturbance model will be considered in this example. We used the moving average parameter $\theta = 0.2$ and $\sigma_\varepsilon^2 = 1$ to simulate the problem. Assume that the disturbance was introduced at run 50 over 800 runs. Figure 4.7 shows the EWMA gain under the enhanced NN adaptive controller. We could see that the value of λ_t oscillated along with the optimal controller value (say 0.8). Taking the sample mean of the last 200 runs, it tended to 0.8078, which was close to the optimal controller parameter. The inflation factor of the controlled process output approximated to 1, which implied that the proposed adaptive controller has the ability to produce the minimum mean square error (MMSE) process output.

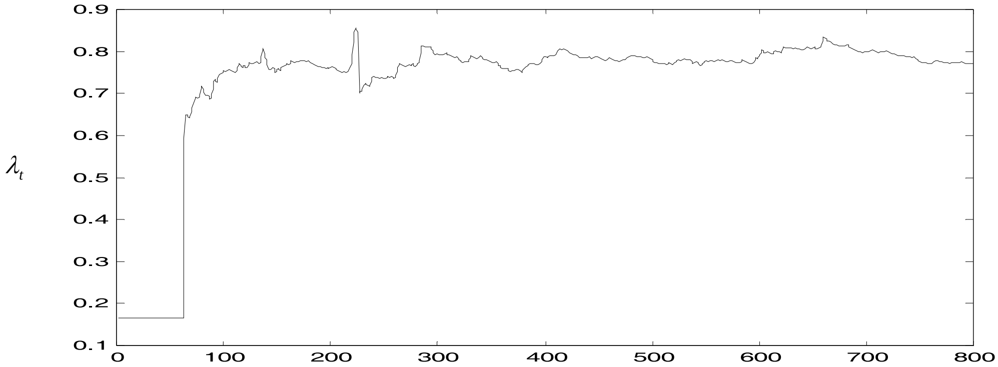


Figure 4.7 Enhanced NN adaptive EWMA gain under IMA(1,1)

4.2.4 Trend Disturbance Model with Slow Ramp Rates

This example considered the ramp disturbance model [see Equation (4.2)]. Following the example of Patel-Jenkins, we introduced the trend disturbance, with $S = 0.1, \sigma_\varepsilon^2 = 1$ at run 50. Figure 4.8 shows the EWMA gain under the enhanced NN adaptive method. We could see that the value of λ_t oscillated along with the optimal controller value (say 0.3061). The inflation factor under Patel-Jenkins was 2.0914, but only 1.3825 under the proposed method.

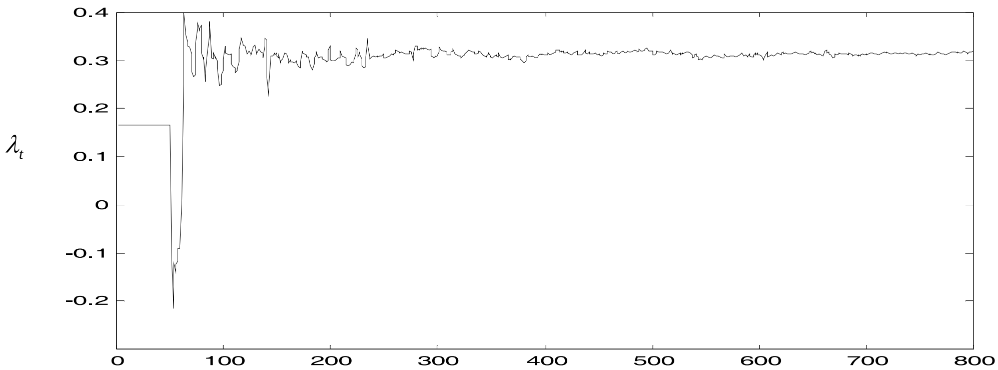


Figure 4.8 Enhanced NN adaptive EWMA gain under trend

4.2.5 Discussion and Concluding Remarks

The proposed enhanced NN adaptive algorithm was based on the pattern recognition of the SACF/SPACF patterns by training the neural network. The behavior of the off-line trained network showed that the network learns fast with input features being SACF/SPACF patterns. We have shown that the proposed approach possesses superior performance over the Patel-Jenkins adaptive algorithm through three implementations.