Initialization and Estimation of Weights and Bias using Bayesian Technique

Abstract

Researches on artificial neural network models have shown that the method used to initialize and estimate weights and bias always determines the rate at which the network will converge and how efficient the model will perform. Although there are several methods that can be used to initialize and estimate network weights and bias, Bayesian approach is widely considered as a more efficient method for modelling artificial neural networks because it can easily compute the inverses of covariance matrices with high dimension that otherwise are computationally expensive. This study has developed a new filter, the First order Extended Ensemble Filter(FoEEF), that applies numerical solution in solving the inverses of high dimensional covariance matrices from I$\hat{to}'s$ stochastic state-space dynamical models. The research applies the new FoEEF Filter to initializes and estimates the weights and bias of artificial neural network. Comparison on the performance of FoEEF filter in estimatin weights and bias are done against the performance of EKF on function estimations by using two different functions, $sin(x)+Q$ function and $3sin(x)^3-3sin(x)^2-sin(x)+1$ function, within 18 and 22 epochs. Emphasis of the performance is placed on the rate at which the models converge .This study gauges the performance by determining the minimum value of the mean square error produced from each epoch and average mean square error minimum value. The outcome from the study showed that FoEEF filter had the lowest value of mean square error and average mean square error which was achieved with the least number of epochs compared to EKF filter. This study concluded that the new FoEEF performed better than EKF and is a more suitable filter for that can be used for initialization and estimation of weights and bias in artificial neural network.
.

Keywords: Bayesian technique; Filtering; Estimation; State Space Dynamical System; Extended Ensemble Kalman Filter

2020 Mathematics Subject Classification: 62F15; 62F40; 62M05

1 Introduction

Complex interaction of multiple factors simultaneously with non linear dynamics and computer science has allowed for the creation of intelligent agents like artificial neural network.[7] which can be used, among other things, to estimate mathematical functions.
Artificial neural network model is a model that is used to handle non linear data and approximate functions or other mathematical operators [7, 16, 9, 10].
Initialization and weight estimation in artificial neural network can be done by using different methods such as random process, maximum likelihood method, Bayesian technique, Cauchy initialization method etc. The major problem encountered when using any of this methods is computing the inverse of matrices. when such matrices are high dimensional, their computation becomes inefficient and expensive, i.e. the computation of their inverse takes a lot of time. To overcome this problem, this study developed FoEEF filter that uses empirical estimate to compute the inverse of high

dimension matrix. From the study, it can be seen that FoEEF method initializes and estimates the weights and bias in a more efficient and way.

The central theme in artificial neural network models is analogy borrowed from the structure and function of human nervous system cells.

The building blocks and functionality of human nervous system is a combination of millions of neurons that are interconnected with each neuron having three parts which are

- Dendrite: This is the input part. Dendrite receive information as signals from other neurons or the surroundings.
- Cell body: The information processing part. It processes all information it receives from dendrite.
- Axon: This is the output terminal. Axon link neuron to each other by sending signals from soma to other neurons.

Figure (1) shows three part structural components of a neuron in the human nervous system.

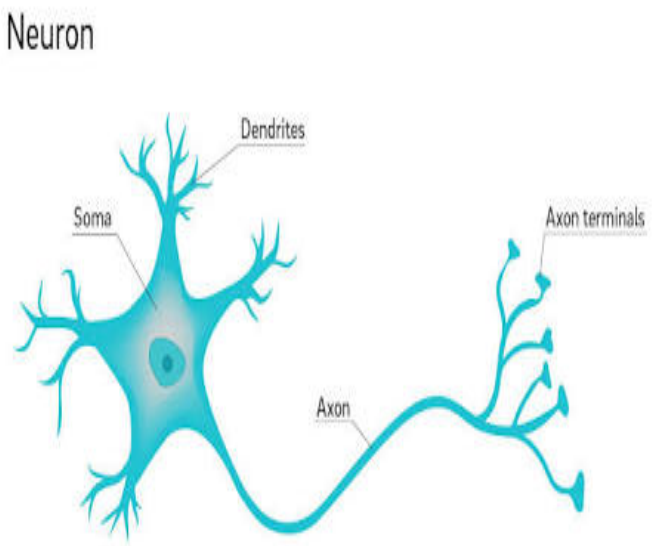

Figure 1: Biological Neural Network Architecture

Similar to human nervous system, artificial neural network is a combination of several neurons. Each neuron has three parts as shown in figure(2).

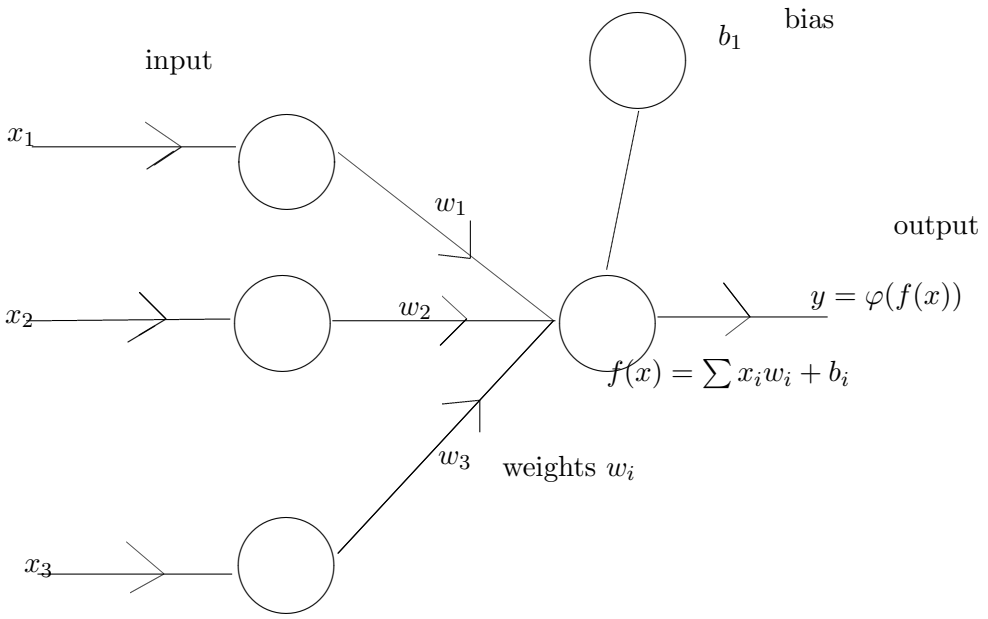

Figure 2: Simple Artificial Neural Network

These three parts can be described as

- Input node: The node receives input data x_i which are mathematical variables for $x_i|_{i=1,2.....n}$. Each x_i is assigned weight $w_{ji}|_{i=1,2.....n,j=1,2,.....m}$ according to its importance.
- Activation and Transfer function: The product of x_i and w_i are summed to form a function $f(x) = \sum_{i=1}^{n} x_i w_{ji} + b_j$ where b_j is a bias. The function $f(x)$ is then transformed by the transfer function $\varphi(f(x))$ to obtain output.
- Output: The value $y_j|_{j=1,2.....m}$ such that has $y_j = \varphi(f(x))|_{\forall j}$.

The representation in figure (2) is of a simple artificial neural network which can be referred to as a single layer perceptron composed of input nodes and output nodes only(no hidden layer). Artificial neural network generally are multi-layered with several complex layers in their architecture. Such networks are called multiple-layers perceptron.
Complex architecture in artificial neural network from input neuron, hidden neuron to the output neuron is represented in Fig (3). This is based on a feed forward MLP where x_i is the input variable, w_{ji} represent the weight, the function $f(x)$ represents summation of all the products of variable and the weight, φ represents the transfer function that act on $f(x)$ and h_j is the output value.
Several neurons are arranged in layers which are always three, input layer, hidden layer and output layer. The columns of neurons in this layers are called nodes or units and each node or unit in a layer is connected to every node or unit in the next layer. For easier understanding of multiple layer Perceptron, an architecture with input layer having two nodes, hidden layer with two nodes, output layer with two nodes and assumed bias of zero value to each node is represented by Figure (3).

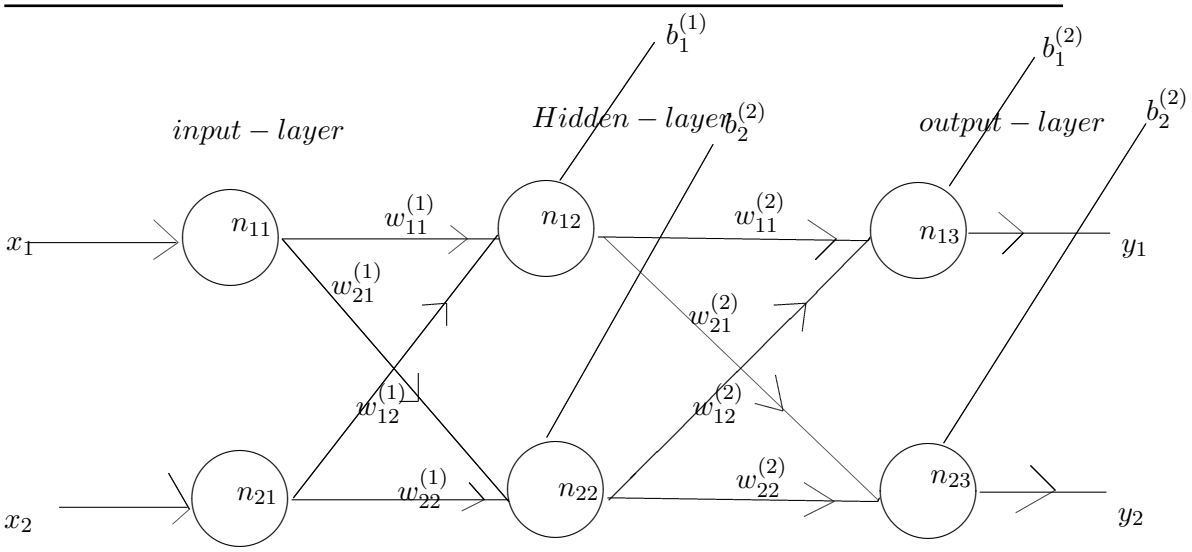

Figure 3: Simple three layered perceptron

Where n_{pk} represents the pth neuron in the kth layer, $w_{ji}^{(k)}$ represents the weight from the ith node in the k^{th} layer to the j^{th} node in the $(k+1)^{th}$ layer.
Neuron n_{11} and neuron n_{21} form the input layer, neuron n_{12} and neuron n_{22} form the hidden layer and neuron n_{13} and neuron n_{23} form the output layer.
x_1 and x_2 are input variables to neuron n_{11} and neuron n_{21} respectively.
There are two output from neuron n_{11}. These are $\varphi(x_1 w_{11}^{(1)})$ and $\varphi(x_1 w_{21}^{(1)})$ that are the value for neuron n_{12} and neuron n_{22} respectively .
Similarly, the two output values from neuron n_{21} are $\varphi(x_2 w_{12}^{(1)})$ that is the value for neuron n_{12} and $\varphi(x_2 w_{22}^{(1)})$ that is the value of neuron n_{22}.
If we take $v_1^{(1)}$ and $v_2^{(1)}$ as the weighted sums at the hidden layer given by

$$v_1^{(1)} = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 \tag{1.1}$$

and

$$v_2^{(1)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 \tag{1.2}$$

Then

$$\begin{bmatrix} v_1^{(1)} \\ v_2^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & ww_{22}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

1.1 First order Extended Ensemble Filter

The estimation of weight w_i and bias b_i can be done using different methods with different results. the best method is one which will make the neural network model to converge faster with a higher level of efficiency. In this research, a new method is developed, the FoEEF filter that estimates the weights and bias in a more efficient way.
The prediction state, kalman gain and innovation processes of the filter are

---

Prediction

$$f(w_t^i)dt + g(w_t^i)q^{\frac{1}{2}}(t)dw_t^{*i} \tag{1.3}$$

Gain

$$ph_w(w_i^i)r^{-1}(t) \tag{1.4}$$

Innovation

$$(dz_t - 0.5(h(w_t^i) + h(\hat{w}_t))dt) \tag{1.5}$$

Diffusion error

$$w_t^* \approx (0, Q_t) \tag{1.6}$$

Observation error

$$v_t^* \approx (0, R_t) \tag{1.7}$$

is applied to estimate weights and bias as a Bayesian method

$$f(w_t^i)dt + g(w_t^i)q^{\frac{1}{2}}(t)dw_t^{*i} \tag{1.8}$$

Gain

$$ph_w(w_i^i)r^{-1}(t) \tag{1.9}$$

Innovation

$$(dz_t - 0.5(h(w_t^i) + h(\hat{w}_t))dt) \tag{1.10}$$

Diffusion error

$$w_t^* \approx (0, Q_t) \tag{1.11}$$

Observation error

$$v_t^* \approx (0, R_t) \tag{1.12}$$

is applied to estimate weights and bias as a Bayesian method

1.2 Training and Testing of Artificial Neural Network

This study trains and tests the First order Extended Ensemble Kalman Filter on artificial neural network model is performed on two functions,$sin(x) + Q$ and $sin(x).^3 + sin(x).^2 + sin(x) + Q$ with emphasis on efficiency and rate of convergence of the model.
In the training and testing process, data is split into two sets. 70% for the data is used to train the neural network and the remaining 30% of the data is used for testing.
The two processes are guided by Algorithm (5.1.1).

Algorithm 1.1 Algorithm of First order Extended Ensemble Filter(FoEEF)

1: L is the set of layers in the neural network
2: $N(k)$ the number of neurons in G for $k = 1, 2, 3, 4.....L$
3: η is the learning rate
4: $y^{(w)}$ Target output
5: z_0 Initial observations/measurement
6: R_0 for $k = 1.....L$
7: $([w_1^1, w_2^1, ...w_M^1], ...[w_1^N, w_2^N, ...w_M^N])$ *Generate weights w randomly*
8: $\hat{w}_0 = E(w_0)$.
$= \frac{1}{M} \sum_{i=1}^{M} w_i$.
Initialize the filter object with initial values of the state.
9: $P_0 = E(w_0 - \hat{w}_0)(w_0 - \hat{w}_0)^T$
$= \frac{1}{1-M} \sum_{i=1}^{M} (w_0 - \hat{w}_0)(w_0 - \hat{w}_0)^T$
Initialize with empirical estimate of state covariance matrix P.
10: while $S(G) = 2$ to S^* do
11: for $t = 0, 1, 2$ do
12: z_{t+1} = random.sample *Random numbers sampling* (-h,h)
13: $h_w = \frac{\partial h}{\partial w}$ given $\hat{w}_t$ The Jacobian of h_w
14: $w_{t+1}^i = w_t^i + p h_w(w_t^i) r^{-1}(t)(dz_t - 0.5(h(w_t^i) + h(\hat{w}_t))dt)$ State propagation for each ensemble i.
15: $\hat{w}_{t+1} = E(w_{t+1})$.
$= \frac{1}{M} \sum_{i=1}^{M} w_{ji}$.
16: $P_{t+1} = E(w_t - \hat{w}_t)(w_t - \hat{w}_t)^T$
$= \frac{1}{1-M} \sum (w_{t+1} - \hat{w}_{t+1})(w_{t+1} - \hat{w}_{t+1})^T$
17: $(R_{t+1}(G))_{ii} = \frac{1}{N(G)N(G-1)} \sum_w ||d^{(G,w)}||^2$
18: $(R_{t+1}(K))_{lm} = 0.7$

2 Simulation, Analysis and conclusion

Training and testing of artificial neural network models is a process that first involves first assigning to the network initial values of weights and bias. This process is called initialisation. The second part is to estimate the value of weights and bias using a back propagation method by updating those weights and bias upto the point at which the neural network is capable of functioning independently. This weights and bias are updated by minimising the difference between the output from the network and the target.
The neural network is then tested to determine the level at which it has learned enough in terms of the speed of convergences and efficiency.
One of the major applications of artificial neural network are facial recognition, character recognition, classification and mathematical ***function estimation***.
Experimentation in this research is based on function estimation.
A function estimation can be described in this case as an underlying mathematical function or mapping function that maps any set of input data to a set of output data with consistency, efficiency and without bias. The neural network tries to best approximate this mapping function. Basically artificial neural network are function approximation algorithm
This study uses simulated data based on algorithm (5.1.1) to train the new FoEEF filter. In the research experiment, two functions are estimated,

- $sin(x) + Q$
- $3sin(x)^3 - 3sin(x)^2 - sin(x) + 1$

The experiments are simulated using MatLab software.
In the experimentation, two epochs, that is 18 and 22 epochs, are used. The two epochs are run in two different neural network structures, the 1-4-1 and 2-4-1 structures. The performance from the experiment generates by FoEEF filter is then compared with results generated by using EKF filter with special emphasis on the rate of converge and efficiency. The aim of the experiment is to gauge which between the two filters convergence faster and is more efficient.
The research performs and analyses the experiment in three parts.

1. Part one involves analysis of observation made from graphical output after training and testing the neural network.
   In the output, blue images represent the target values and red images represent the outputs from the filters.
   Analysis in this part is done after the filters are trained and tested by observing the extend at which the output values from the filters(red line) is mapped into target values(blue lines). An efficient filter is one whose output line is perfectly mapped into the target line.
2. In the second part, the study tabulates the output values for analysis. A table with the number of epochs, convergence points and computed mean square error is constructed. The experiment aims to compare which of the two filters have minimum value of the convergence, the filter with the smallest least square error and the average mean square error. In the analysis, the filter that gives the smallest value for both convergence and mean square error is the one that is more efficient.
3. In the third part, convergence values (x-axis) is plotted against computed mean square error(y-axis) to determine which of the two filters has low distribution of mean square error.

2.1 Training and testing of FoEEF and EKF on 1-4-1 network using sin(x)+Q function with 18 epoch

In this section, sin(x)+Q function is used to train FoEEF filter and EKF filter on a 1-4-1 neural network structure using 18 and 22 epochs.

Figure (4) and figure (5) represents training and testing with 18 epochs for FoEEF and EKF respectively.

Training and testing of FoEEF filter

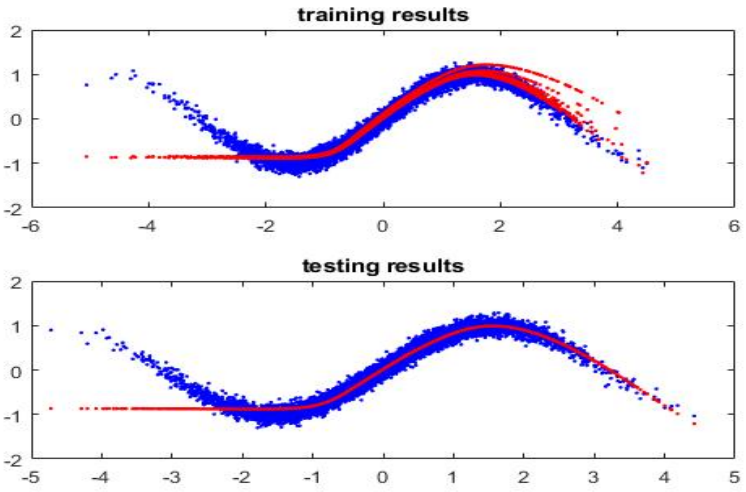

Figure 4: Experimentation on FoEEF filter within 18 epoch

Training and testing of EKF filter

Graphical Analysis

From figure (4) represent training and testing from FoEEF and figure (5) represents training and testing from EKF, it can be observed that FoEEF captures the training data faster than EKF. This is indicated by the fact that there is no significant deviations that can be seen from the graph in figure(4). The outcome from the graph of figure(5) represents the training and testing of EKF and has some significant deviations.

Figures (4) also shows a more efficient output from FoEEF test data set. This figure is represented by a marched line compared to the output from EKF that is represented by figure(5)

Conclusion

Initialisation of weights and bias of artificial neural network takes place in the very early stages of the training process. Different models initialise neural networks with different values giving different outcomes. Observations that are made from the figures by comparing performances of FoEEF and EKF shows that FoEEF performs better than EKF and therefore it is more efficient filter than EKF

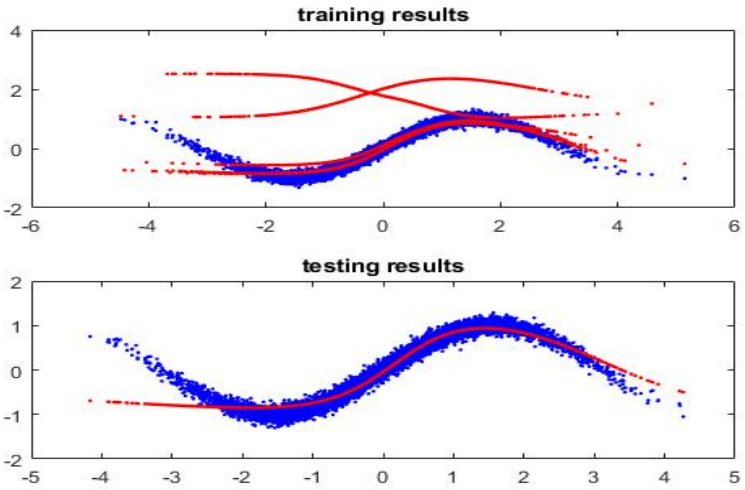

Figure 5: Experimentation on EKF within 18 epoch

in initialising and estimating the weights and bias of artificial neural network.

Table of output

Table (5.1) represent data from simulated results. The table shows the number of epoch for each FoEEF and EKF, the convergence value for each epoch and the corresponding mean square error measurement.

Epoch	EKF	MSE(EKF)	FoEEKF	MSE(FoEEF)
1	3.3945	0.41949	-0.8508	0.2861
2	2.9451	0.0393	0.6685	0.96914
3	3.9115	1.35649	-0.5747	0.06698
4	0.2775	6.09750	-0.8553	0.29096
5	4.6630	3.67177	0.25361	0.3243
6	5.9955	10.55396	0.9713	1.65685
7	6.3740	13.1564	-0.8796	0.3177
8	0.1157	6.9227	0.1044	0.17664
9	2.1579	0.3468	-0.2657	0.00251
10	-0.8320	12.8079	-0.8707	0.3078
11	1.6702	1.15909	-0.7729	0.20885
12	1.5647	1.39739	-0.8719	0.30914
13	5.6730	8.56256	0.2716	0.3451
14	-0.3333	9.48709	-0.7147	0.1590
15	-0.7894	12.5019	-0.8773	0.31518
16	-0.2454	8.95094	-0.8584	0.2943
17	3.4936	0.55769	-0.8391	0.27374
18	-0.3536	9.61256	0.7208	1.07472

Table 1: Result from FoEEF and EKF on sin(x)+Q for 18 epochs

Analysis from the table

Observation made from table (5.1) shows that the minimum value of the mean square error(MSE) for FoEEF is 0.002519036. This minimum value occurs at the *9th* epoch for FoEEF. Comparatively, a run of EKF gives the minimum value of mean square error(MSE) of 0.039317. This values occurs at the *2nd* epoch.

The average mean square error for FoEEF is 0.409945 while the average mean square error for EKF is 5.97785833. From the table, it can be observed that EKF has a high value of average mean square error when compared to the value of the mean square error of FoEEF.

The comparison are relayed on the array below.

$Algorithm$	$network$	$Epoch$	$MinMSE$	$AverageMSE$
$FoEEF$	$1-4-1$	9	0.00251	0.409945
EKF		2	0.0393	5.97785833

Conclusion from the table

The minimum value of mean square error for EKF is ***fifteen times*** higher than the minimum value of mean square error for FoEEF. This is a significant difference in efficiency measurement between the two filters.

The average values for the mean square error for the two filters shows that EKF has a ***fourteen times lager*** average mean square error than .

This research concludes that FoEEF convergence faster than EKF because of its very lower mean square error of 0.00251 when compared to a much higher mean square error for EKF of 0.0393. The research also concludes that FoEEF is more efficient than EKF given that the the average mean square error for FoEEf of 0.409945 is far much lower than the mean square error for EKF of

5.97785833.

Analysis from the graph on the distribution

A analysis done on the distribution of the mean square error for FoEEF and the mean square error for EKF is shown on figure (6) below. The figure represent a plot of mean square error(MSE) from the output measurement of FoEEF(blue) and the mean square error from output measurement of EKF(red).

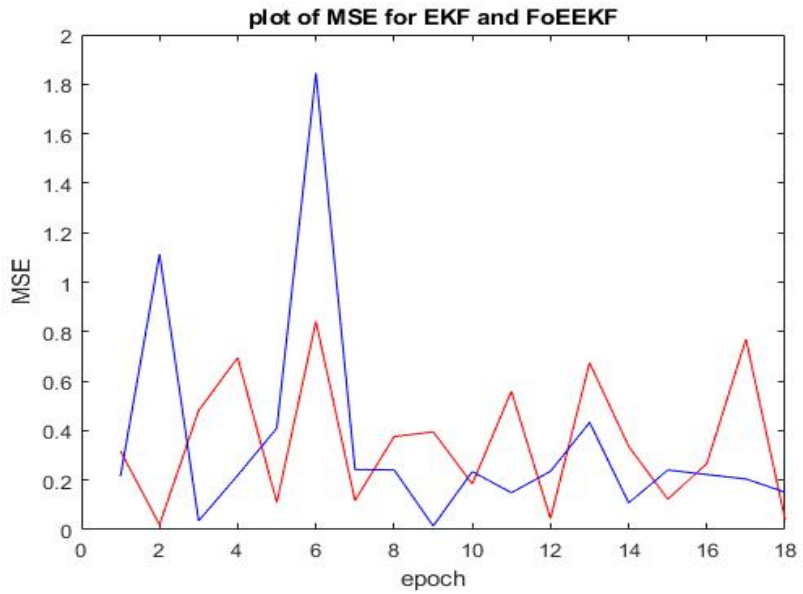

Figure 6: Training and Testing using sin(x)+Q in 1-4-1 Architecture for 18 epoch

The trajectory observed from figure(6) shows that the distribution of mean square error for FoEEF is lower than the distribution of mean square error for EKF.

Conclusion

FoEEF filter is more efficient than EKF filter in initialising the weights and bias of artificial neural network since the trend of the distribution of its mean square error is lower than the distribution of the mean square error for EKF.

2.2 Training and testing of FoEEF and EKF on 1-4-1 network using sin(x)+Q function with 22 epoch

After training FoEEF and EKF with 18 epochs, another experiment on training and testing is done for the same FoEEF and EKF filters using the same function estimation, that is sin(x)+Q function, on the same 1-4-1 network structure but this time the experiment is run within 22 epochs. The aim of the experiment is to gauge the performance of the two filters on a different number of epochs The figures (7) and (8) are measurements outputs from of training and testing of FoEEF and EKF experimented within 22 epochs

Training and testing FoEEF

Figure (7) below shows the graphical output from training and testing of FoEEF filter within 22 epochs.

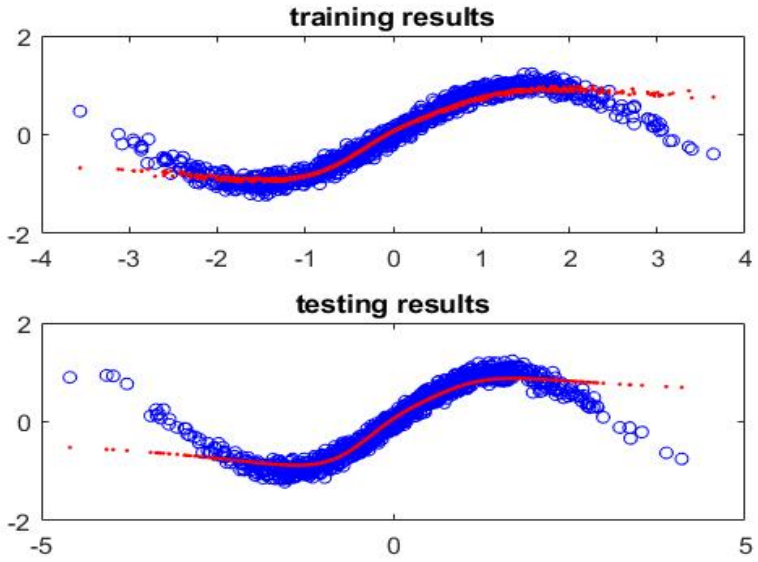

Figure 7: Estimation with 22 epoch for FoEEF

Training and testing of EKF filter within 22 epochs

Figure (8) below shows the graphical representation of the output from training and testing of EKF filter within 22 epochs.

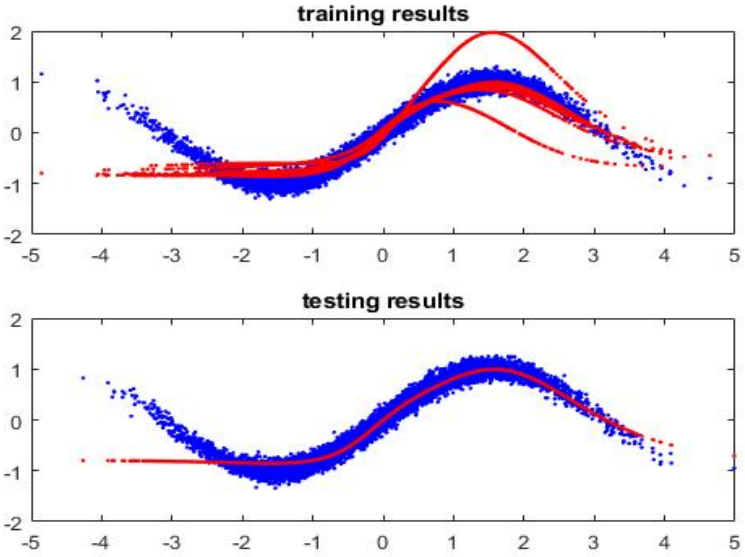

Figure 8: Estimation with 22 epoch for EKF

Analysis from the graph of figures 7 and 8

The observation made from training and testing of FoEEF filter in figure (5.4) shows that the output values represented by red lines from FoEEF filter is much closely mapped to the target values represented by the blue lines. The deviations of the output from the target is therefore noticeably small when compared to that of EKF filter in figure(5.5). This is because in figure(5.5), EKF shows some significant deviation between the output lines(red) and the target values represented by the blue lines. This large deviation is can be observed both in training and testing outputs data set.

Conclusion

FoEEF is a better filter than EKF and therefore can initialise weights and bias in artificial neural network in a more efficient way than EKF. This is based on observations made from the experimentation in which FoEEF has captured both training and testing data set in a better way than EKF

Analysis from the table

Table (5.2) represents output values for each of the 22 epochs, the values at which the network convergences, computed mean square error(MSE) for both FoEEF and EKF, average MSE for for both FoEEF and EKF.

Epoch	**EFK**	**MSE(EKF)**	TIME(S)	**EEFK**	**MSE(FoEEF)**
1	0.02224	7.42327	0.0032	-0.22581	0.00811
2	1.74412	1.00538	0.0025	0.56729	0.78001
3	-1.09165	14.73380	0.0019	-0.523918	0.04327
4	-1.14944	15.18078	0.0019	-0.59115	0.07577
5	0.11160	6.94430	0.0019	0.50770	0.67830
6	0.70720	4.16000	0.0017	0.77863	1.19797
7	0.90889	3.37794	0.0017	-0.65613	0.11576
8	-0.92212	13.46112	0.0017	-0.65416	0.11443
9	-0.82698	12.77205	0.0018	0.73897	1.11273
10	0.54916	4.82965	0.0019	-0.11160	0.20239
11	-0.68750	11.79454	0.0018	-0.55788	0.058562
12	0.70103	4.1852	0.0019	-0.36431	0.00234
13	0.11190	6.94273	0.0019	0.73301	1.10020
14	0.83977	3.63680	0.0019	-0.64186	0.28214
15	0.59241	4.64142	0.0016	0.85714	1.37601
16	-0.95617	13.71213	0.0018	0.96247	1.63422
17	0.02536	7.40626	0.0019	0.78799	1.21855
18	0.07577	7.13444	0.0017	-0.58947	0.074848
19	-0.9251	13.48361	0.0017	-0.67951	0.13222
20	-0.76719	12.34828	0.0017	-0.68205	0.13407
21	0.74899	3.99126	0.0017	0.59137	0.82312
22	-0.11072	8.16549	0.0019	0.4330	0.56086

Table 2: Result from FoEEF and EKF on sin(x)+Q for 22 epochs

From table(5.2), it is be observed that the minimum mean square error value for our new filter(FoEEF) is 0.008114226. This value occurs at the very first epoch in the 22 epoch training. The minimum mean square error value for EKF is 1.00538 which occurs at the second epoch. It is observed that the EKF mean square error value is ***one hundred and twenty four times*** lager than the mean square error of FoEEF. This difference can be considered as highly significant.
The table also shows the values obtained from EKF to be generally lager than values from FoEEF. The average mean square error for FoEEF is 0.53299 while that for EKF is 8.24229. The value from EKF is ***fifteen times lager*** than the value for FoEEF. The results form the table are summarised below

Algorithm	*network*	*Epoch*	*MinMSE*	*averageMSE*
FoEEF	$1-4-1$	1	0.008114226	0.53299
EKF		2	1.00538	8.24229

Conclusion
The difference between the minimum mean square error for FoEEF(0.00811) and and that of EKF(1.00538) is major. EKF gives ***one hundred and twenty four times lager*** values of mean square error than FoEEF.
The research computed the average mean square errors for both filters. The average mean square

error from EKF output(8.24229) is ***fifteen times lager*** when compared to the mean square error from FoEEF output(0.53299)
The analysis from the experiment shows how our new FoEEf filter converges faster than EKF filter. It also shows that the efficiency exhibited by our new FoEEF filter is better than EKF filter.
The results analysed from table(5.2) proves that FoEEF filter is more suitable at weight and bias initialisation and estimation in artificial neural network than EKF.

Distribution Analysis
A graphical analysis is done by plotting the values of MSE for each epoch.

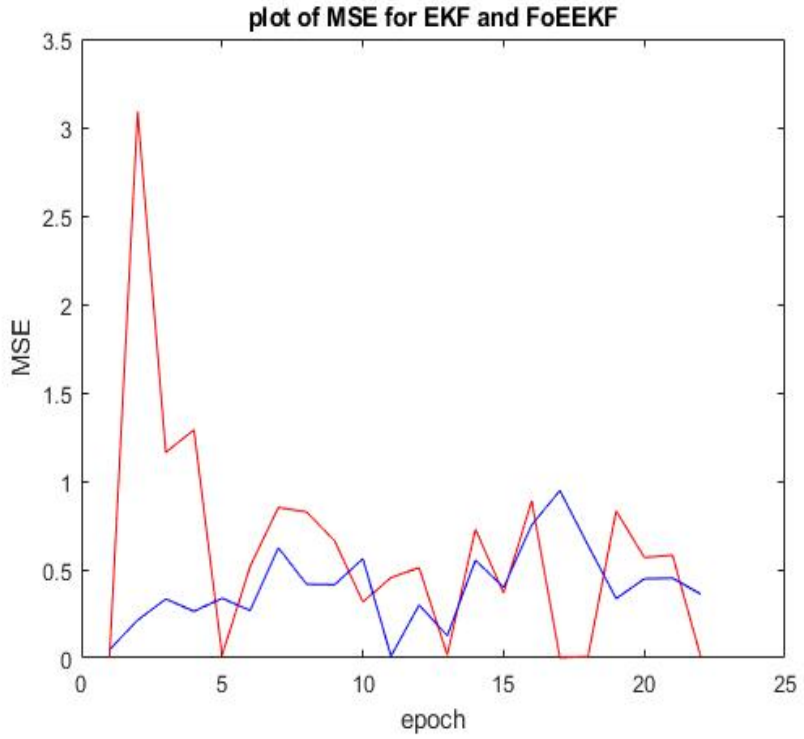

Figure 9: Distribution of MSE on 22 epoch

Observation from figure(9), a plot of the distribution of mean square error from FoEEF and EKF filters for each epoch shows that generally, the new FoEEF filter has lower distribution of mean square error for each number of epoch than EKF filter. This implies that our new FoEEF filter is much more efficient in most stages of epochs than the EKF filter.

3 Estimating FoEEF and EKF models using the function y=$3sin(x)^3-3sin(x)^2-sin(x)+1$ in 2-4-1 Neural Network with 18 and 22 Epochs

This study carried out further investigation of FoEEF and EKF filters on a more complex trigonometric function using a 2-4-1 neural network structure. The experiment was run on two epochs. The first was 18 epochs and the second was 22 epochs.

Training and testing of FoEEF filter using 18 epochs

Figure(5.7) below represent training and testing of FoEEF within 18 epochs on y=$3sin(x)^3 - 3sin(x)^2 - sin(x) + 1$ function.

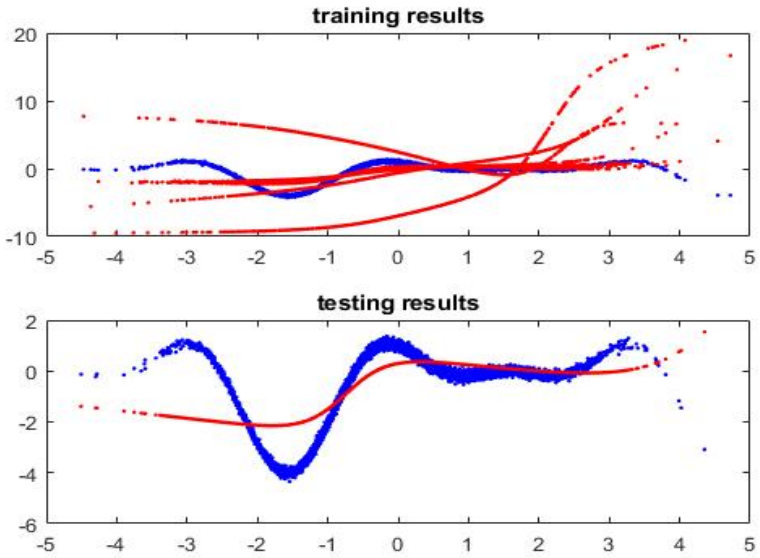

Figure 10: Estimation of FoEEF with 18 epoch

Training and testing of EKF filter using 18 epochs

Figure(10) below represent training and testing of EKF within 18 epochs on y=$3sin(x)^3-3sin(x)^2-sin(x)+1$ function.

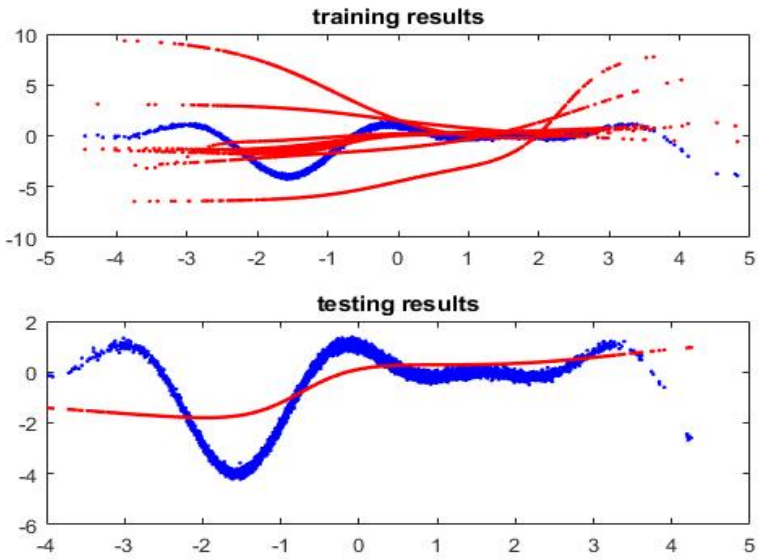

Figure 11: Estimation of EKF with 18 epoch

Graphical Analysis and conclusion

As the function to be estimated becomes more complex, it can be observed that EKF filter represented by figure (11) looses track and becomes less efficient in the training process. The lines(red) that represent the output values from the neural network shows five very large miss-match and significant deviations from the the lines(blue) that represent the target values. In comparison to figure(10) that represent output from training and testing of FoEEF filter, it can be observed that there are only two small deviation of output(red lines) from the target(blue line).

At the validation stage, EKF filter performance(red lines) represented by figure(12) is even less efficient since it can not estimate the target function(blue line) well. Output from EKF filter gives a near straight line that does not match the target Values at all. When this is compared with output from FoEEF filter, it can be seen that FoEEF filter curves and matches the target values in a in a better way than EKF filter.

Epoch	EKF	MSE(EKF)	FoEEF	MSE(FoEEF)
1	1.03893	2.91685	-5.52982	27.18506
2	-3.69554	41.50398	5.30047	31.54359
3	0.24000	6.28408	0.57006	0.78491
4	6.2074	11.97623	-0.79780	0.23224
5	-0.04161	7.77533	-0.24031	0.00571
6	0.05272	7.25809	-1.36830	1.10757
7	-0.14537	8.36473	0.20387	0.27015
8	0.2940	6.01600	0.17390	0.23989
9	-1.34882	16.77429	0.35022	0.4437
10	0.27221	6.12362	0.24281	0.31215
11	-0.00124	7.55184	0.13221	0.20079
12	-0.47969	10.41035	0.32776	0.41429
13	-0.37608	9.75250	-1.43684	1.25653
14	-1.76350	20.34296	-0.02634	0.08383
15	-1.62469	19.1101	0.36190	0.45940
16	-0.08141	7.99885	0.31150	0.39362
17	0.19908	6.49091	0.12774	0.19680
18	0.26072	6.18064	0.03616	0.12394

Table 3: Result from FoEEF and EKF for 18 epochs

Analysis from Table

Table (5.3) below represents output from simulation done the on the function y=$3sin(x)^3-3sin(x)^2-sin(x)+1$ that is run within 18 epochs. The table contains epochs numbers, the convergence value for each epoch and computed mean square error for each epoch.

Analysis and Conclusion

From the table(5.3) it can be seen that the minimum mean square error for the new filter(FoEEF) is 0.005711518 which occurs at the *5th* epoch. comparatively, the minimum value of mean square error for EKF filter is 6.01600 which occurs at the *8th* epoch.

This difference is major and significant since it shows that the mean square error for EKF filter is ***three times lager*** than that of FoEEF filter. In terms of efficiency, the new FoEEF filter is more efficient than EKF filter when functions become complex since the experiment shows that FoEEF filter has a smaller convergence and average mean square error than EKF filter.

The array below shows values of MSE square error and MSE.

$Algorithm$	$network$	$Epoch$	$MinMSE$	$averageMSE$
$FoEEF$	$2-4-1$	5	0.005711518	3.6252
EKF		8	6.01600	11.629

Distribution Analysis
Figure(5.9) is a plot of the mean square error(y-axis) vs the epoch in a 18 epoch simulation. the figure show the distribution of FoEEF filter(blue line) and EKF filter(red line). The aim of the graph in figure(5.9) is to determine which of the two filter has a lower distribution of the mean square error from the experimentation

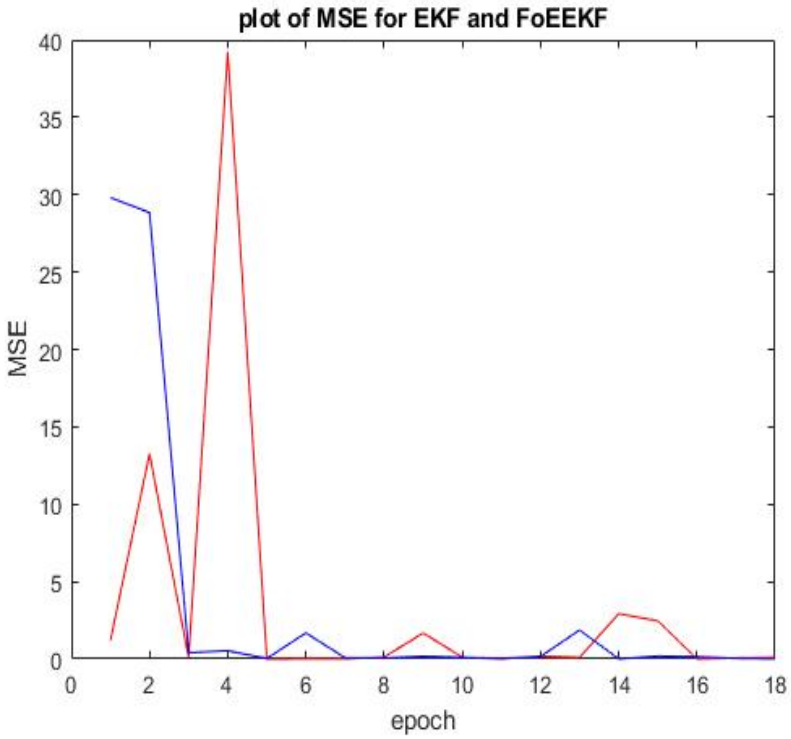

Figure 12: Distribution of MSE for 18 epoch

From figure(12), it be seen that the distribution of the mean square error of FoEEF filter is lower than than of EKF filter. Most of the values for the new filter as observed from figure(5.9) lies on the lower part of the axis just next to the x-axis that has a zero value.
The study concludes that FoEEF filter is more efficient than EKF filter when estimating complex function.

3.1 Training FoEEF and EKF using 22 epochs

Figure(5.10) below represent training and testing of FoEEF with 22 epochs on y=$3sin(x)^3 - 3sin(x)^2 - sin(x) + 1$ function.

Training and testing of FoEEF filter using 22 epochs

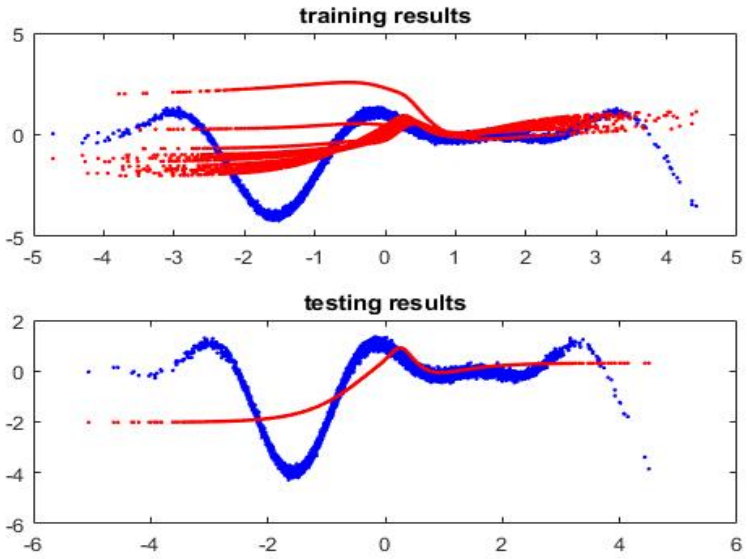

Figure 13: Estimation of FoEEF with 22 epoch

Training and testing EKF filter using 22

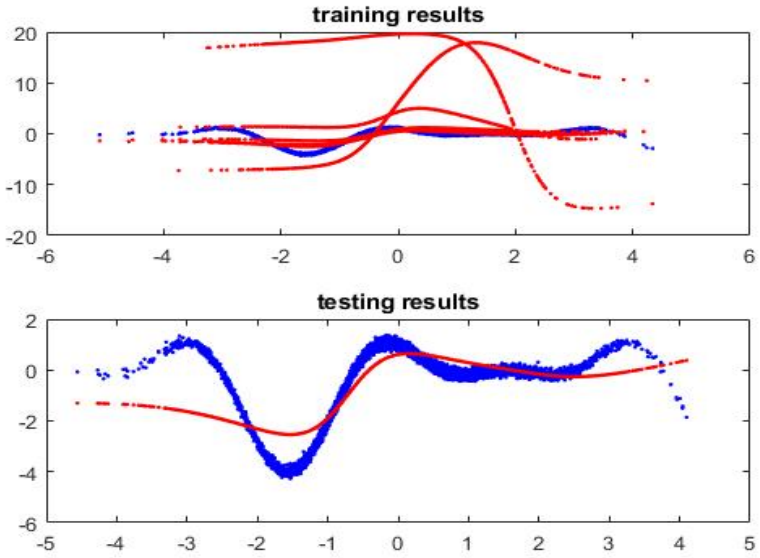

Figure 14: Estimation of EKF with 22 epoch

Graphical Analysis

Figures(13) and figure(14) shows the training and testing of the new FoEEF filter and EKF filter respectively. It can be seen from the two figures that the output from the new FoEEF filter represented by red lines in figure (13) is closely matched to the target values that is represented by the blue lines for both training and testing process. This study compared the output from figure(13) with that of figure(14). Training and testing of FoEEF in figure(13) matched the target values than output from EKF filter in figure(14) which can be seen to have three lines(red) that significantly deviates away from the target line(blue).

Conclusion

When the results from the two graphs of figure (13) and figure (14) are compared, this study concludes that the new FoEEF filter initialises and estimates the weights and bias of the training data in a more efficient and faster way than EKF filter. This is shown by how much the outputs represented by graphs from figure(13) and (14) matches with the lines that represent the values of the target. Output lines(red) from figure(14) matched the target value in a more efficient way than the graph in figure (13) and therefore FoEEF filter is a better filter than EKF filter and is more suitable for use in initialising and estimating weights and bias of artificial neural network.

This study did tabulated the result from simulation on FoEEF and EKF for the number of epochs, the convergence value for each epoch, the mean square error for each filter at each epoch and the average mean square error for each filter.

The main aim is to determine which filter has the smallest convergence value, mean square error and average mean square error which is are measures of efficiency of the filters. The results for the experiment are represented in table(5.4).

Epoch	FoEEF	MSE(FoEEF)	EKF	MSE(EKF)
1	0.9736	0.965	15.1323	1.1970
2	0.5432	0.251	17.95141	13.252
3	0.9984	0.0387	1.5642	0.08711
4	-0.4093	0.47	0.9690	39.22
5	-0.6336	0.0008	-2.0727	0.00181
6	0.5359	0.263	-0.2952	0.01163
7	0.1549	0.703	0.4780	0.0081
8	-1.1825	0.054	0.6004	0.1219
9	-1.0108	1.791	-1.4501	1.6735
10	0.2109	0.248	-0.8224	0.1071
11	0.6503	0.804	-0.1132	0.0029
12	-0.3222	1.883	0.6436	0.1802
13	0.2102	0.043	0.2525	0.1029
14	0.2453	0.086	0.5311	2.9184
15	-0.8497	1.275	0.3889	2.3624
16	0.0075	1.0053	0.9860	0.0068
17	-0.026	0.093	-1.56835	0.06463
18	0.09418	1.990	0.43938	0.09977
19	-0.4749	0.067	-0.6328	4.71
20	-1.3722	0.356	0.30358	1.52
21	-1.432	0.108	0.33715	1.44
22	0.399	0.7494	0.64559	0.799

Table 4: FoEEF with 22 epochs

Analysis from the table

Observations from tables (5.4) shows that the minimum mean square error(MSE) for FoEEF has a value of 0.0008 and occurs at the *5th* epoch. The minimum mean square error value for EKF is 0.0068 and occurs at the *16th* epoch.

The study computed the ratio of mean square error of FoEEf and EKF. The mean square error of EKF is ***eight times lager*** than the mean square error of FoEEF.

The average mean square error of EKF was found to be 3.1767 while the average mean square error for FoEEF is 0.602. The study also computed the ratio of the average mean square error of EKF and FoEEF. The average mean square error for EKF is ***five times lager*** than the average mean square error for FoEEF.

The array below represents the number of epochs, minimum mean square error for FoEEF filter and EKF filter and the corresponding average mean square error for both filter.

$Algorithm$	$Network$	$Epoch$	$MinMSE$	$AverageMSE$
$FoEEF$	$2-4-1$	5	0.0008	0.602
EKF		16	0.0068	3.1767

Conclusion

Values from the table shows that the mean square error of EKF is eight times lager than that of FoEEF. The average mean square error for both FoEEF and EKF was computed and the value for

EKF eas five times lager than that of of FoEEF.
This result from the experiment shows that the new FoEEF filter is more efficient that EKF filter hence the new FoEEF filter is more suitable for initialising and estimating weights and bias of artificial neural network model.

References

[1] Angwenyi, D. (2019). Time-Continuous State and Parameter Estimation with application to SPDEs. PhD Thesis, *Institute Fur Mathematik AG Numerische Mathematik,* University of Potsdam.

[2] Arulampalam S., Maskell S., Gordon N., & Clapp T., (2002) *A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking*, IEEE Transactions on Signal Processing, Vol. **50**, No. 2 174-188.

[3] Kalman R., Bucy R., (1961), *New Result in Linear Filtering and Prediction Theory*, Journal of Basic Engineering.

[4] Evensen G., (2006), Data Assimilation. The Ensemble Kalman Filter, Springer-Verlag, New York.

[5] Fischler, R-H. (1998). A Mathematical History of the Golden Number. Dover Publications.

[6] Girosi, F. and Poggio, T. (1990). Networks and the best approximation property. Biol. Cybern., 63: 169,176.

[7] Girosi, F., Jones, M. and Poggio, T. (1995). Regularization theory and neural network architectures. Neural Comput., 7: 219,269.

[8] Hateley James(2017). *The Lorenz system Formulation*, Semantic scholar.

[9] Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2: 359,366.

[10] Ito, Y. (1991). Representation of functions by superpositions of a step or sigmoidal function and their applications to neural network theory. Neural Networks, 4: 385,394.

[11] Jazwinski A., (2007), Stochastic Processes and Filtering Theory, Academic Press, Inc., New York, Dover edition.

[12] Kantas N., Doucet A.,Maciejowski J.(2017) An Overview of Sequential Monte Carlo Methods for Parameter Estimation in General State-Space Models

[13] Kiyosi It(1944). Stochastic Integral. Proc. Imperial Acad. Tokyo 20, 51952

[14] Kushner H., (1962), *On the Differential Equations Satisfied by Conditional Probability Densities of Markov Processes, with Applications*, J. SIAM Control, Ser. A, Vol. 2, No. 1.

[15] Poggio, T. and Girosi, F. (1990a). Networks for approximation and learning. Proc. IEEE, 78(9): 1481,1497.

[16] Poggio, T. and Girosi, F. (1990b). Regularization algorithms for learning that are equivalent to multilayer networks. Science, 247: 978,982.

[17] Reich S. and Cotter C., (2015), Probabilistic Forecasting and Bayesian Data Assimilation, Cambridge University Press.

[18] Rudolph Kalman (1960), Transactions of the ASME, Journal of Basic Engineering, 82 (D), 35-45, 1960).

[19] Youngjoo kim and Hyochoong Bang (2018). *Introduction to Kalman filter and its Application*, Intech Open Journal, DOI: 10.5772/intechopen.80600.

[20] Zheqi Shen, Yaoumin Tang (2014). *A modified ensemble Kalman particle filter for non-Gaussian systems with non-linear measurement functions*, Journal of Advances in Modelling Earth Sciences, https://doi.org/10.1002/2014MS000373.