Vol.6 (Iss. 3) Mathematics(2025), pp. 521- 538, African Journal of Empirical Research http://ajernet.net ISSN 2709-2607

Second order Extended Ensemble Kalman Filter with Stochastically Perturbed Innovation for Initializing Artificial Neural Network Weights

Cavin Oyugi Ongere*[1]
David Angwenyi[2]
Robert Oryiema[3]

*[1]cavinovic@gmail.com
[2]dangwenyi@mmust.ac.ke
[3]robertoryiema@gmail.com

[1,2,3]Department of Mathematics, Masinde Muliro University of Science and Technology, Kenya

*[1]Corresponding Author

https://doi.org/10.51867/ajernet.6.3.42

ABSTRACT

Artificial neural networks are widely applied in solving non-linear state-space dynamic models, yet the challenge of inaccurate initial weights remains a critical bottleneck. Weight initialization methods significantly influence convergence speed and model efficiency. While conventional approaches such as random initialization and filtering techniques are commonly used, the Bayesian method—though highly accurate—suffers from the computational burden of inverting high-dimensional matrices. This study has developed a novel solution: the Second Order Extended Ensemble Filter with Perturbed Innovation (SoEEFPI). Developed from a second-order Taylor expansion of the stochastically perturbed Kushner-Stratonovich equation, SoEEFPI provides a tractable numerical solution to the inverse covariance matrix problem. Validation is conducted using the Lorenz63 system, comparing SoEEFPI's performance to that of the Kalman-Bucy Filter (SoEKBF) and the First Order Extended Ensemble Filter (FoEEF) in MATLAB. Furthermore, SoEEFPI is employed to initialize neural network weights, yielding a new model whose convergence time, RMSE, and epoch count are evaluated. Results demonstrate improved convergence efficiency and accuracy, positioning SoEEFPI as a robust alternative for neural network initialization.

Keywords: Non-linaer filtering, Non-linear state space dynamic models, Bayesian method, Convergence time

1 INTRODUCTION

Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of biological neural networks (3). They are comprised of interconnected nodes organized in layers and are capable of learning from data through adaptive weight updates, making them powerful tools for modeling non-linear systems and solving complex

pattern recognition tasks (4; 6). ANNs consist of an input layer, one or more hidden layers, and an output layer. Each neuron applies a weighted sum of inputs followed by a non-linear activation function, allowing the network to capture complex relationships in data. During training, the network adjusts weights using optimization techniques like gradient descent, guided by the backpropagation algorithm (18; 11). This iterative process minimizes the error between predicted and actual outputs.

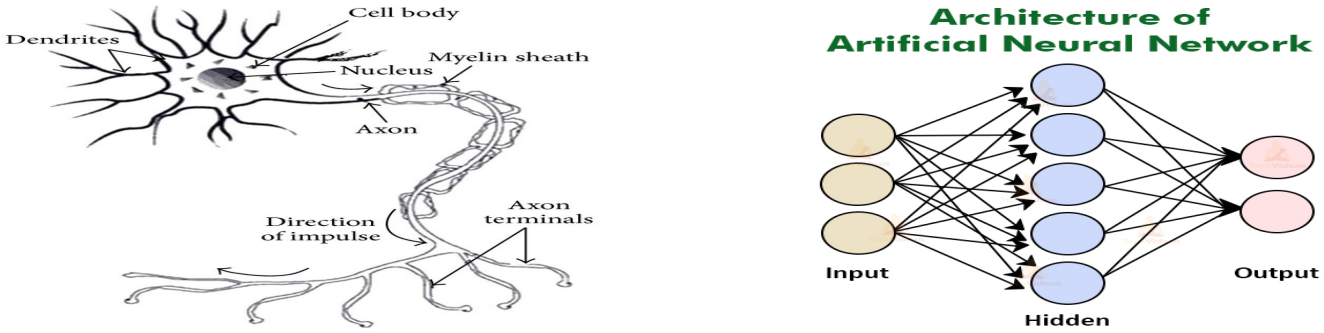

Fig. 1.1. Left: Biological Neural Network Architecture. Right: Artificial Neural Network Architecture.

Figure 1.1 illustrates a biological neuron network architecture on the left and a simple ANN architecture on the right comprising an input layer, a single hidden layer, and an output layer. This configuration demonstrates the basic flow of information from input nodes through a hidden processing stage to the final output. Despite its simplicity, such a structure encapsulates the essential learning mechanism of ANNs, where weights are updated to improve prediction accuracy through successive training iterations.

The fundamental operation within a neuron is given by:

$$\sum_{i=1}^{m} w_i x_i + b$$

where w_i are weights, x_i are inputs, and b is a bias term. The result is passed through an activation function $f(x)$, which could be linear or nonlinear (e.g., sigmoid, tanh, ReLU) (14). This mechanism enables ANNs to approximate a wide class of functions, including stochastic ones (10).

Fig. 1.2.

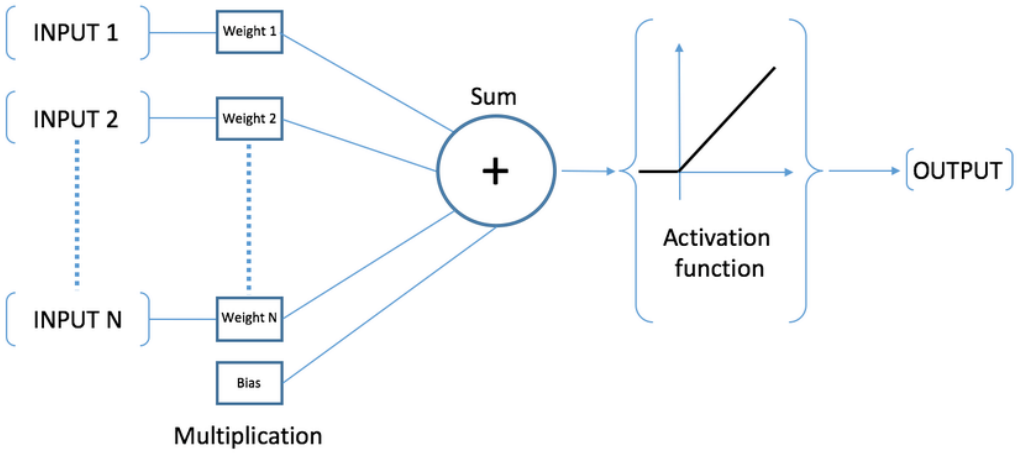

Fig. 1.2. Structure of a Typical Artificial Neural Network

Initialization is a critical step in neural network training as it defines the starting point of the learning process. Poor initialization can lead to slow convergence, vanishing or exploding gradients, and sub-optimal performance. While random initialization has been widely used, advanced methods like Xavier and He initialization provide better convergence properties by scaling weights based on the number of input and output units (7; 8). Nonetheless, even these approaches may fall short when dealing with complex, high-dimensional dynamic systems. To address this, research has explored Bayesian and filtering-based initialization methods. Although Bayesian approaches offer high accuracy, they are computationally intensive due to the need to invert large covariance matrices. This study introduces a novel filter—the Second Order Extended Ensemble Filter with Perturbed Innovation (SoEEFPI)—designed to address the limitations of high-dimensional matrix inversion in Bayesian methods. Derived from a second-order Taylor expansion of the stochastically perturbed Kushner-Stratonovich equation, SoEEFPI provides an efficient initialization framework for ANN weights. The filter is validated using the Lorenz63 system and compared against the Kalman-Bucy Filter (SoEKBF) and the First Order Extended Ensemble Filter (FoEEF) in MATLAB. By integrating SoEEFPI-initialized weights into neural network models, this research demonstrates improved convergence time, reduced root mean square error (RMSE), and fewer epochs to convergence.

2 Literature Review

2.1 First Order Extended Ensemble Filter (FoEEF)

The First Order Extended Ensemble Filter (FoEEF) is an ensemble-based Bayesian filtering approach used to initialize and estimate the weights of Artificial Neural Networks (ANNs). It employs a set of ensemble weight samples to approximate the posterior distribution of weights over time, updating them iteratively through a process inspired by the Kalman filter. This method addresses limitations associated with random and deterministic initialization techniques by leveraging both model dynamics and measurement updates. Through forward projection and empirical estimation of covariance, FoEEF enhances learning efficiency and model stability. The algorithm operates layer by layer and across time steps (epochs), integrating measurement corrections via Kalman gain and Jacobian-based updates. This results in improved convergence speed and reduced training error. The algorithm is highlighted below.

Algorithm 1 First Order Extended Ensemble Filter (FoEEF)

Input: Number of layers L, neurons N_k in each layer $k = 1, 2, ..., L$, learning rate η, target $y^{(w)}$, initial observation z_0
for $k = 2$ to L **do**
  Initialize measurement error v_0^*
  Generate ensemble weights: $w = \left([w_1^1, w_2^1, \ldots, w_M^1], \ldots, [w_1^N, w_2^N, \ldots, w_M^N]\right)$
  Initialize ensemble mean: $\hat{w}_0 = \frac{1}{M}\sum_{i=1}^{M} w_i$
  Initialize empirical covariance: $P_0 = \frac{1}{1-M}\sum(w_0 - \hat{w}_0)(w_0 - \hat{w}_0)^T$
  for $t = 0, 1, 2, \ldots$ **do**
    Project state forward: $f(w_t^i)\,dt + g(w_t^i)q^{1/2}(t)\,dw_t^{*i}$
    Compute Jacobian: $h_w = \frac{\partial h}{\partial w}$ at $\hat{w}_t$
    Compute Kalman gain: $ph_w(w_t^i)r^{-1}(t)$
    Update weights: $w_{t+1}^i = w_t^i + ph_w(w_t^i)r^{-1}(t)\left(dz_t - 0.5(h(w_t^i) + h(\hat{w}_t))dt\right)$
    Estimate updated mean: $\hat{w}_{t+1} = \frac{1}{M}\sum_{i=1}^{M} w_{t+1}^i$
    Estimate updated covariance: $P_{t+1} = \frac{1}{1-M}\sum(w_{t+1} - \hat{w}_{t+1})(w_{t+1} - \hat{w}_{t+1})^T$
    Compute error metrics: $(R_{t+1}(K))_{ii} = \frac{1}{N(K)N(k-1)}\sum_w \|d^{(K,w)}\|^2$
    For $(y \approx z)$: $(R_{t+1}(K))_{lm} = 0.7$
  end for
end for=0

2.2 Extended Kalman Filter (EKF)

The Extended Kalman Filter (EKF) is a widely used nonlinear state estimation technique that extends the classical Kalman Filter to handle nonlinear system dynamics and measurement models (1; 16). By linearizing the nonlinear functions around the current estimate using a first-order Taylor series expansion, the EKF approximates the posterior distribution of the state with a Gaussian distribution. This makes it a powerful tool for recursively estimating the states of nonlinear systems in real-time, such as in robotics, navigation, and neural network weight estimation. Despite its simplicity and effectiveness, the EKF assumes that the nonlinearities are mild and can sometimes suffer from divergence if the linearization is poor or if the system is highly nonlinear. Effective weight initialization plays a foundational role

Algorithm 2 Extended Kalman Filter (EKF)

Input: Initial state estimate $\hat{x}_0$, initial covariance P_0, process model $f(\cdot)$, measurement model $h(\cdot)$, process noise covariance Q_t, measurement noise covariance R_t
for $t = 1$ to T **do**
  Prediction Step:
  Predict state: $\hat{x}_{t|t-1} = f(\hat{x}_{t-1})$
  Compute Jacobian of process model at $\hat{x}_{t-1}$: $F_t = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{t-1}}$
  Predict covariance: $P_{t|t-1} = F_t P_{t-1} F_t^\top + Q_t$
  Update Step:
  Compute innovation: $y_t = z_t - h(\hat{x}_{t|t-1})$
  Compute Jacobian of measurement model at $\hat{x}_{t|t-1}$: $H_t = \frac{\partial h}{\partial x}\big|_{\hat{x}_{t|t-1}}$
  Compute innovation covariance: $S_t = H_t P_{t|t-1} H_t^\top + R_t$
  Compute Kalman gain: $K_t = P_{t|t-1} H_t^\top S_t^{-1}$
  Update state estimate: $\hat{x}_t = \hat{x}_{t|t-1} + K_t y_t$
  Update covariance estimate: $P_t = (I - K_t H_t) P_{t|t-1}$
end for=0

in the successful training of artificial neural networks (ANNs). Poor initialization can result in slow convergence, getting stuck in local minima, or complete training failure. Traditional initialization techniques include zero initialization, random initialization, Xavier, and He initialization (9), each with strengths and limitations.

Zero initialization (17), despite its simplicity, causes all neurons in a layer to update identically, leading to a loss of symmetry and expressiveness in the network. Random initialization (15) helps break this symmetry but introduces instability, especially in deep networks, by amplifying or shrinking gradients across layers—a phenomenon known as the vanishing or exploding gradient problem.

Xavier initialization (5) attempts to address this by scaling weights based on the number of input and output neurons, maintaining a constant signal variance. Similarly, He initialization (9) extends this approach to rectified linear units (ReLU), offering more stability in deep networks. Despite these advances, these methods do not account for prior knowledge or uncertainty, making them susceptible to poor performance in noisy or dynamic environments.

Bayesian methods offer a principled framework for incorporating prior knowledge and managing uncertainty, making them well-suited for weight estimation in neural networks (1). Kalman filters, as a specific family of Bayesian estimators, recursively update estimates of unknown parameters by combining prior beliefs with noisy observations. Murru and Rossini (12) proposed a modified Kalman filtering algorithm that leverages cyclic matrices and Discrete Fourier Transforms (DFT) to reduce computational overhead in neural network initialization. Their approach allows for real-time update of weight estimates by treating the initialization problem as a dynamic state estimation task. The update step:

$$\hat{w}_t = (Q_t^{-1} + R_t^{-1})^{-1}(Q_t^{-1} w_t^- + R_t^{-1} m_t)$$

blends the predicted state w_t^- and measurement m_t weighted by the process (Q_t) and measurement (R_t) noise covariances. The use of cyclic matrices enables efficient matrix operations, while DFT exploits structure in the covariance matrices to accelerate inversion. Their experiments demonstrated faster convergence and reduced error compared to conventional random initialization.

To circumvent the computational bottleneck of matrix inversions in traditional Kalman filters, ensemble-based filters have emerged as a viable alternative. These methods approximate probability distributions using a collection of random samples, or ensembles, thus enabling scalable estimation in high-dimensional settings.

Angwenyi and Oryiema (2) introduced the First-Order Extended Ensemble Filter (FoEEF), which estimates the state of neural network weights using ensembles propagated through forward simulations. The filter avoids direct computation of Jacobians by relying on empirical statistics:

$$\hat{w}_t = \frac{1}{M}\sum_{i=1}^{M} w_t^i, \quad P_t = \frac{1}{M-1}\sum_{i=1}^{M}(w_t^i - \hat{w}_t)(w_t^i - \hat{w}_t)^T$$

Here, w_t^i denotes the ith weight sample at time t, and M is the ensemble size. This method captures first-order dynamics and can efficiently adapt weights based on prediction residuals. However, it may suffer from bias and underestimation of uncertainty when dealing with nonlinearity or sparse data.

Recognizing the limitations of FoEEF in complex systems, Midenyo, Angwenyi, and Oganga (13) proposed the Second-Order Extended Ensemble Filter (SoEEF). By incorporating second-order Taylor expansions of the Kushner-Stratonovich stochastic partial differential equation (SPDE), SoEEF improves robustness and accuracy in non-linear, non-Gaussian settings.

The filter propagates particles x_t^i with dynamics:

$$dx_t^i = f(x_t^i)dt + g(x_t^i)dB_t + Mh_x(x_t^i)R^{-1}(t)\left[dz_t - \left(h(x_t^i) + \frac{1}{2}M_t h_{xx}(x_t^i)\right)dt\right]$$

where h_x and h_{xx} are the first and second derivatives of the observation function, and M_t is an empirical moment estimate. This formulation enables SoEEF to adjust for curvature in the likelihood function, thereby producing more accurate updates in highly nonlinear models. Simulation studies on the chaotic Lorenz-63 system confirmed its superiority over FoEEF, FoEKBF, and standard particle filters, particularly in convergence rate and prediction accuracy.

Kalman-based and ensemble filter methods represent a significant leap beyond static initialization techniques by introducing dynamic, data-informed updates and uncertainty quantification. Among these, SoEEF stands out due to its balance between computational tractability and estimation fidelity. However, challenges persist in applying such methods to deep architectures with millions of parameters due to scalability constraints. Furthermore, these filters have not been extensively explored in conjunction with modern machine learning paradigms such as transfer learning, dropout regularization, or attention-based mechanisms. As neural networks continue to be deployed in increasingly uncertain and data-scarce environments, initialization methods that explicitly model and adapt to uncertainty become more relevant.

This study seeks to address existing limitations by introducing a novel filter grounded in the second-order expansion of the Kushner-Stratonovich equation. In contrast to SoEEF, our method incorporates a stochastically perturbed innovation term and alternative sampling strategies to enhance diversity and stability. The goal is to achieve fast, accurate, and robust initialization, particularly in settings where prior data is limited or noisy. By bridging insights from Bayesian inference, stochastic calculus, and ensemble filtering, the proposed approach aspires to advance the state-of-the-art in neural network initialization and contribute to the broader discourse on uncertainty-aware deep learning.

2.3 Training and Testing ANN Structure

Artificial Neural Networks (ANNs) operate algorithmically and recursively, similar to the filtering process, across discrete time steps. This makes them well-suited for applications involving dynamic state estimation. In this study, the newly developed Second Order Extended Ensemble Kalman Filter with Perturbed Innovation (SOEEKFPI) is employed to initialize and estimate the weights of an ANN.

The structure of the ANN is defined by the network size equation:

$$ms = mx \times mh + mh + mh \times my + my \tag{2.1}$$

where:

- ms — total number of trainable parameters,
- mx — number of input neurons,
- mh — number of hidden neurons,
- my — number of output neurons.

Solving for the number of hidden neurons yields:

$$mh = \frac{(ms - my)}{mx + my + 1} \tag{2.2}$$

The *tanh* activation function is used due to its zero-centered range $[-1, 1]$ and differentiability, with derivative $(1 - \tanh^2)$. Compared to other functions (e.g., logistic), *tanh* provides faster convergence and mitigates saturation issues in deeper layers.

The ANN is trained over multiple epochs, where an epoch represents one full pass through the training data. Convergence is assessed by monitoring the Mean Squared Error (MSE) between the predicted and target values. The filter (SOEEKFPI, FOEEF, or EKF) that achieves the lowest MSE in the fewest epochs is considered most effective. Training uses 70% of the data, while 30% is reserved for testing and validation.

The MSE is computed as:

$$E = \frac{1}{m} \sum_{m} \left| f_L(x, w) - y^L \right|^2 \tag{2.3}$$

where $f_L(x, w)$ is the ANN output at the final layer L, and y^L is the target value.

Alternatively, MSE is computed via:

$$\sum_{w} \left\| d^{(k,w)} \right\|^2 \tag{2.4}$$

where:

$$d^{(k,w)} = f_k(x, w) - y^L \tag{2.5}$$

The training and evaluation process for the ANN proceeds as follows:

1. Initialize network weights using the perturbed SOEEKFPI.
2. Estimate updated weights via recursive SOEEKF filtering.
3. Compute RMSE to evaluate performance.
4. Iterate the process until convergence criteria are met.

3 Simulation Analysis and Conclusion

3.1 Introduction

In neural networks, a form of artificial intelligence inspired by the human brain, data is processed through interconnected computational units known as neurons. These neurons are organized into layers, emulating the biological neuron arrangement in the human brain. This structure enables neural networks to effectively model complex, nonlinear dynamic systems, making them suitable for estimating intricate mathematical functions. For a time-varying nonlinear system with state transition and measurement equations, measurement noise is typically modeled with zero mean and characterized by error covariance matrices P and R. In this context, SoEEKFPI facilitates the initialization and estimation of ANN weights. The properties of the filter are outlined as follows:

1. **Prediction:** The filter projects the system state forward using the model dynamics and noise covariance matrix P:
$$f(y_t)dt + g(y_t)q^{\frac{1}{2}}(t)dB_t^* \tag{3.1}$$
2. **Kalman Gain:** The gain determines the influence of the new measurement:
$$Mh_y(y_t)r^{-1}(t) \tag{3.2}$$
3. **Innovation Process:** This captures the discrepancy between the predicted and actual measurements:
$$\left(dz_t + R^{\frac{1}{2}}(t)du_t - 2h(y_t)dt\right) \tag{3.3}$$
4. **Model Error:** Captures discrepancies due to bias and variance:
$$B_t^* \approx (0, R_t) \tag{3.4}$$
5. **Observation Error:** Accounts for noise in observed measurements:
$$u_t^* \approx (0, P_t) \tag{3.5}$$

Training and testing of the ANN model starts with initializing weights and biases, which are updated iteratively using forward and backward propagation until the optimal solution is reached. This iterative process ensures the network learns from the data, and the resulting trained model can generalize to unseen inputs. The error between model outputs and expected values is quantified using Mean Squared Error (MSE), serving as the principal metric for evaluating learning quality. Upon training, the model is validated using a holdout test dataset. Efficiency is gauged by both the accuracy (measured via Root Mean Squared Error, RMSE) and the speed of convergence (indicated by the number of epochs required to reach minimum MSE). In this study, ANN is used for mathematical function estimation, which involves mapping input data to output data with high fidelity. ANNs excel in this domain due to their capacity for capturing nonlinear relationships.

Data simulation and experimentation are performed using MATLAB. The novel SoEEKF with Perturbed Innovation (SOEEKFPI) is compared against the First-Order Extended Ensemble Filter (FOEEF) for initializing and estimating ANN weights. The ANN model initialized with weights from SOEEKFPI is compared to that initialized using FOEEF. Experimentation comprises three core components. First, visual inspection of model outputs after training reveals sinusoidal graphs, where deviation between predicted (red) and expected (blue) lines indicates error. Less deviation implies better model performance. Second, MSE values are computed numerically across epochs. The model with the lowest MSE in the fewest epochs is deemed superior. Finally, learning curves are analyzed to assess training dynamics. These plots, with MSE on the y-axis and epochs on the x-axis, reveal whether models are underfitting (flat/noisy curves) or overfitting (memorizing noise).

An efficient model demonstrates a downward MSE trend with increased epochs. The learning curve helps identify generalization issues: underfitting models fail to learn from training data, while overfitting models do not perform well on unseen data due to excessive sensitivity to noise. Overall, this study demonstrates that SOEEKFPI provides an effective Bayesian mechanism for initializing ANN weights, outperforming FOEEF in accuracy, convergence speed, and generalization ability.

The experiments in this study involve the initialization and estimation of the weights of the ANN model using the SoEEKFPI, FoEEF, and SoEKBF filters. The training and testing of these filters are conducted using two different trigonometric functions. These sine functions are:

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

Sine functions are selected for their dynamic nature. The experiments aim to identify which filter is more efficient in initializing and estimating the weights of the ANN model. This study evaluates the results through graphs of computed MSE and learning curves. Across the three experiments, 80% of the data is allocated for training, while the remaining 20% is reserved for testing the model.

3.2 Training and testing FoEEF, EKF and SoEEKFPI, on 1-4-1 model architecture using $\sin(x) + Q$ function with 18 epoch

3.2.1 Training and Testing of SoEEKFPI

Graphical Analysis: Data was simulated using the function $\sin(x)+Q$, and the same dataset was used to train the ANN model configured with a 1-4-1 architecture over 18 epochs. In Fig. 3.1, which represents the performance of SOEEKFPI, two graphs are shown: the first displays the model's output behavior on the training dataset, while the second shows the output behavior on the testing dataset. To facilitate performance comparison, additional experiments were conducted to initialize and estimate the weights of the ANN model using FOEEF and EKF. The corresponding results are presented in Fig. 3.2 and Fig. 3.3, respectively.

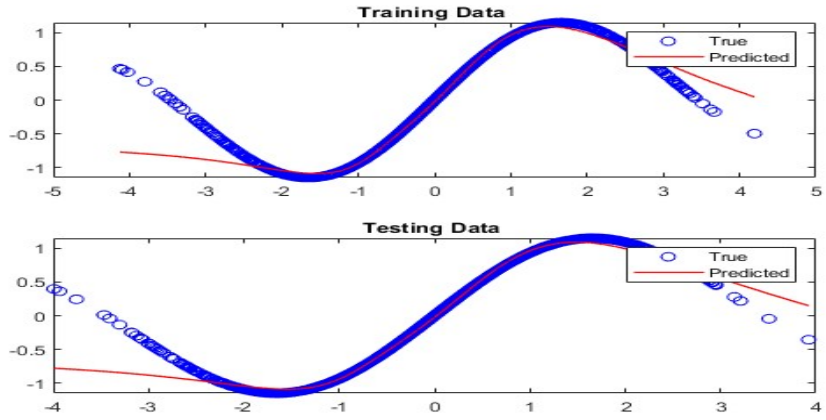

Fig. 3.1. Experimentation on SOEEKFPI within 18 epoch

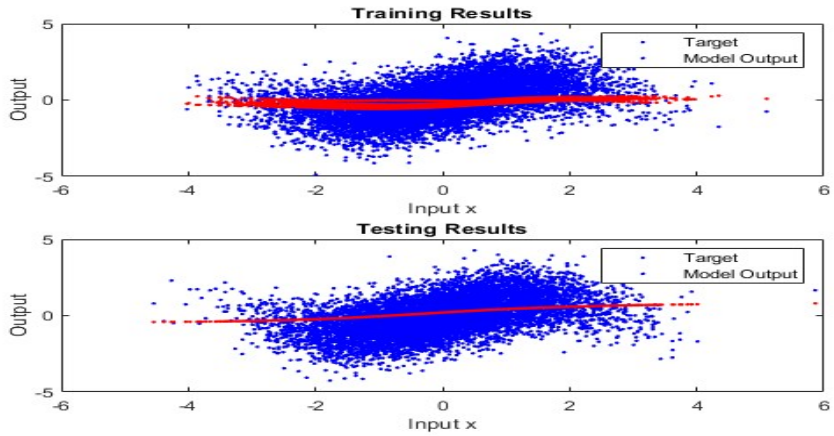

Fig. 3.2. Experimentation on FoEEF within 18 epoch

Analysis: The figures illustrate how the output lines (red) deviate from the expected output lines (blue). From the plots in Fig. 3.1 and Fig. 3.2, it is evident that the ANN model whose weights were initialized using SOEEKFPI captures the training dataset more accurately than those initialized with FOEEF and EKF. Specifically, in Fig. 3.1, the red output line from the SOEEKFPI-initialized model closely aligns with the blue expected output, indicating higher accuracy and better generalization.

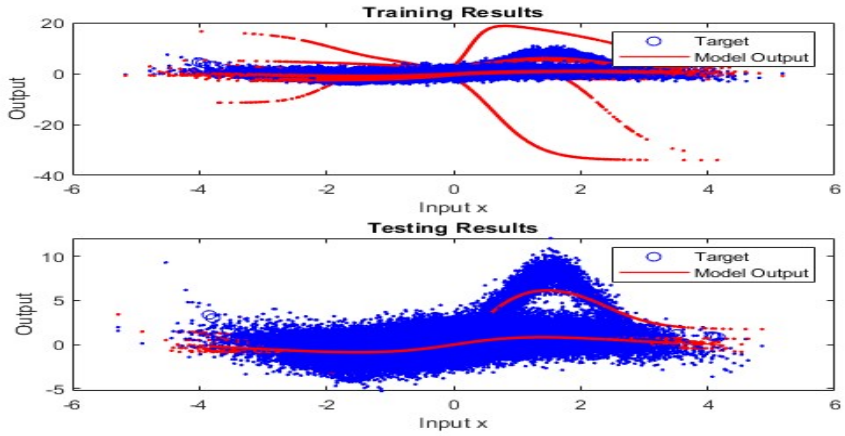

Fig. 3.3. Experimentarion on EKF within 18 epoch

Conclusion: Comparatively, the model developed using weights initialized and estimated by the SOEEKFPI with perturbed innovation outperforms those initialized using FOEEF and EKF. This indicates that the proposed SOEEKFPI method offers superior performance in weight estimation and serves as a more effective Bayesian approach for initializing and estimating weights in ANN models.

Table of Output: This study evaluates model performance by computing the Root Mean Squared Error (RMSE) generated from the output of models trained on a dataset using 18 epochs. The experiments involve initializing and estimating the weights of the ANN using three methods: SOEEKFPI with perturbed innovation, FOEEF, and EKF. The resulting MSE values are presented in Table 1.

Table 1. RMSE results from SOEEKFPI, FOEEF, and EKF on $\sin(x) + Q$ for 18 epochs

Epoch	SOEEKFPI	EKF	FOEEF
1	0.0827	1.2198	1.027317
2	0.0535	1.9222	0.93542
3	0.0493	1.0777	0.990253
4	0.0467	1.0182	0.983263
5	0.0446	1.0013	0.986163
6	0.0432	0.9868	0.930819
7	0.0421	1.0185	0.937950
8	0.0358	1.0385	0.883366
9	0.0412	0.9971	0.877141
10	0.0398	0.9994	0.977981
11	0.0392	1.02704	0.958872
12	0.0386	0.98832	0.925780
13	0.0381	0.95475	0.961602
14	0.0376	0.99058	0.926410
15	0.0371	0.94550	0.965266
16	0.0366	1.00800	0.970535
17	0.0362	0.99269	0.947818
18	0.0404	1.01373	0.972584

3.2.2 Analysis from the table

From Table 1, the least RMSE value for SOEEKFPI is 0.0358, occurring at the 8^{th} epoch. In contrast, the FOEEF achieves its lowest RMSE of 0.8771 at the 9^{th} epoch, while the EKF attains a minimum RMSE of 0.9455 at the 15^{th}

epoch. Comparatively, the least RMSE of FOEEF is approximately 24.50 times larger than that of SOEEKFPI, and that of EKF is about 26.41 times larger. These differences are notably significant, highlighting the superior performance of the SOEEKFPI-based model. Moreover, the average RMSE across all epochs for SOEEKFPI is 0.0435, whereas the average RMSE for FOEEF and EKF are 0.9533 and 1.0667, respectively. Clearly, the SOEEKFPI outperforms both FOEEF and EKF, achieving far lower error values throughout the training process. These statistics are summarized in Table 2 below.

Table 2. Summary RMSE values

Type of filter	network structure	Epoch	MinRMSE	Average RMSE
SOEEKF	1-4-1	8	0.0358	0.0435
EKF	1-4-1	15	0.9455	1.0667
FOEEF	1-4-1	9	0.87714	0.9533

Analysis of the Learning Curve: Figure 3.4 illustrates the distribution of Mean Squared Error (MSE) against the number of epochs for the ANN models. The vertical axis represents the MSE values, while the horizontal axis corresponds to the number of epochs. This figure demonstrates how the RMSE of the ANN models evolves over successive training iterations. The primary objective of this analysis is to identify which model generalizes more efficiently by converging to a minimal error value in the fewest number of epochs, effectively reaching the optimal stopping point. From the plotted curves, it is evident that the ANN model whose weights are initialized and estimated using the SOEEKFPI exhibits the most rapid and consistent decline in RMSE. This trend indicates superior learning efficiency compared to the models initialized using FOEEF and EKF, both of which show slower convergence and higher residual error across epochs.

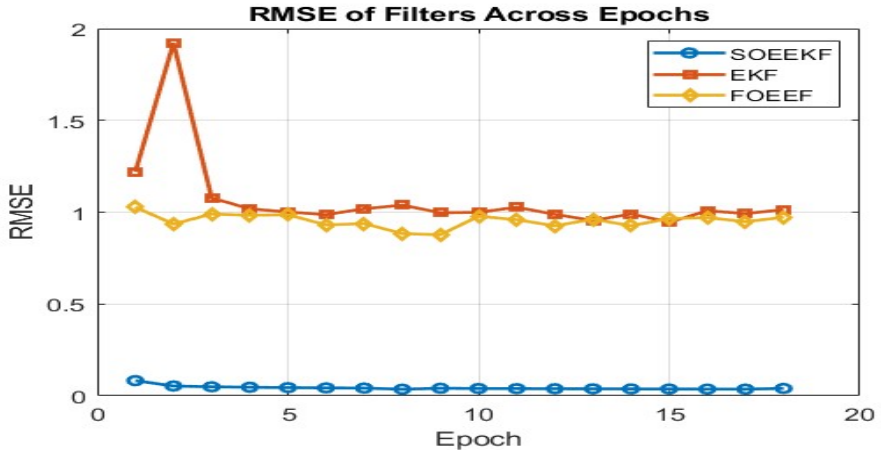

Fig. 3.4. Training and testing using $\sin(x) + Q$ in 1-4-1 Architecturet for 18 epochs

Conclusion: The findings of this study indicate that the ANN model initialized and trained using the newly developed SOEEKFPI filter outperforms the models based on EKF and FOEEF initialization. This is evidenced by the superior learning curve and significantly lower RMSE values, demonstrating faster convergence and improved generalization capability. Notably, the SOEEKFPI-based model neither underfits nor overfits the data, highlighting its robustness across training and testing phases. Consequently, it can be concluded that SOEEKFPI is the most efficient and reliable Bayesian method for initializing and estimating the weights of artificial neural networks among the filters examined in this study.

3.3 Training and testing SoEEKFPI, FoEEF and EKF on 1-4-1 model architecture using $\sin(x) + Q$ function with 22 epoch

In this experiment, the number of epochs is increased from 18 to 22, while all other conditions are held constant. The primary objective is to assess the performance of the three filters; SOEEKFPI, FOEEF, and EKF, under extended training

iterations. An epoch represents a complete pass of the training data through the neural network, essentially serving as a unit of learning opportunity. In this context, a highly efficient filter should enable the neural network to converge and generalize effectively with fewer epochs. Therefore, filters that require fewer epochs to attain optimal performance are considered superior in learning efficiency. The experiment proceeds by initializing and estimating the weights of the ANN model using SOEEKFPI, FOEEF, and EKF, each evaluated across 22 epochs to compare their effectiveness under increased learning cycles. Figures 3.5, 3.6, and 3.7 present the training and testing outputs of the ANN model over 22 epochs, initialized and estimated using the SOEEKFPI, FOEEF, and EKF filters, respectively.

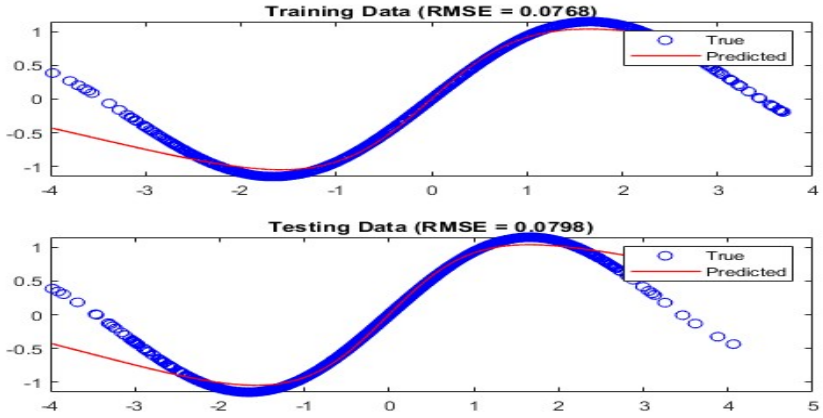

Fig. 3.5. Experimentation with 22 epochs for SOEEKFPI

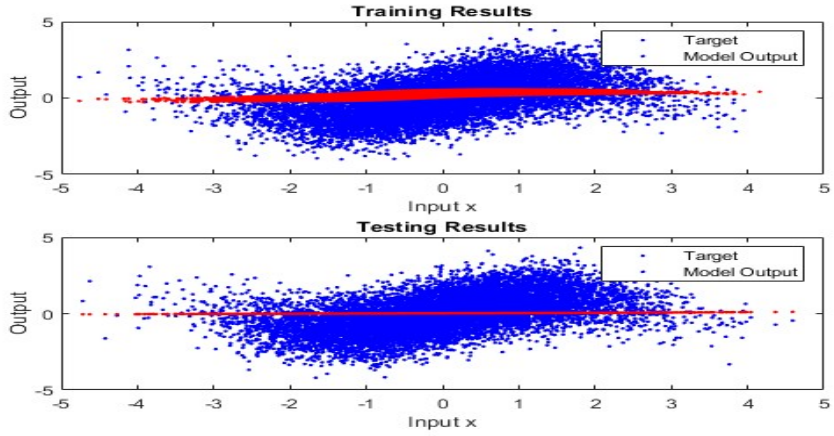

Fig. 3.6. Experimentation with 22 epochs for FoEEF

Analysis from the graphs: From the results presented in Figures 3.5, 3.6, and 3.7, it is evident that the performance of the proposed SOEEKFPI filter significantly outperforms that of the EKF and FOEEF filters. The output generated by the ANN model initialized using SOEEKFPI aligns more closely with the expected target values from the training dataset, demonstrating superior learning accuracy and generalization capability compared to the other filters.

Conclusion: The results from the three experiments clearly demonstrate that the SOEEKFPI filter with perturbed innovation captures the training dataset more quickly and with higher accuracy compared to the EKF and FOEEF filters. This is evidenced by the minimal deviation observed between the predicted output (red line) and the expected target values (blue line) in the case of SOEEKFPI. In contrast, the ANN models initialized using EKF and FOEEF exhibit more pronounced deviations, indicating less accurate learning and weaker generalization performance.

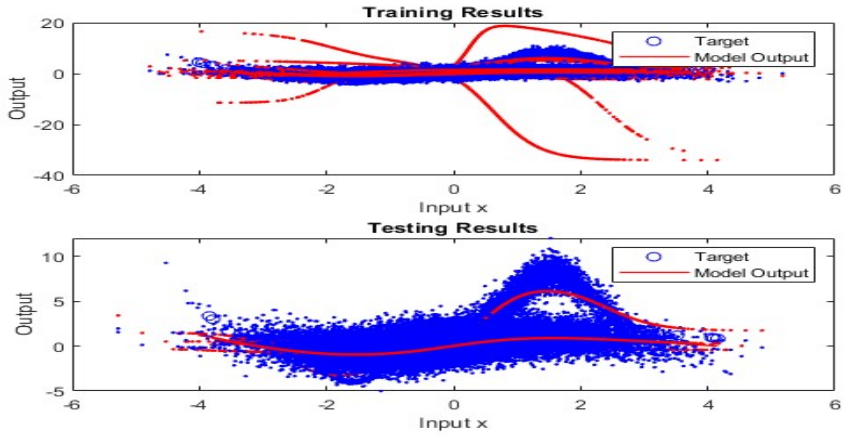

Fig. 3.7. Experimentation with 22 epochs for EKF

Table 3. RMSE results from SOEEKFPI, FOEEF, and EKF on $\sin(x) + Q$ for 22 epochs

Epoch	SOEEKFPI	EKF	FOEEF
1	0.088746	1.2376	1.05472
2	0.085991	1.3031	1.260661
3	0.084101	1.1040	1.094492
4	0.082592	0.9931	1.028949
5	0.081454	1.0033	1.18431
6	0.080495	1.0059	1.09021
7	0.079520	1.0267	1.07243
8	0.078503	1.02201	1.1965
9	0.077480	1.00178	1.00776
10	0.076573	0.9880	1.014651
11	0.075787	1.0676	1.03683
12	0.075024	0.9988	0.99279
13	0.068287	1.01149	0.98685
14	0.073493	1.0026	0.99214
15	0.072750	1.00874	0.728352
16	0.072011	1.0176	0.976837
17	0.071293	1.02904	0.9917
18	0.070622	0.97175	0.99923
19	0.069979	1.0251	0.96072
20	0.069398	0.99011	0.9549
21	0.068833	1.03165	0.728352
22	0.074256	1.02582	1.02474

Analysis of speed and accuracy of learning: The Mean Squared Error (MSE) generated from the output of the ANN model is computed using the training dataset over 22 epochs. In this experiment, the initialization and estimation of ANN weights are performed using the SOEEKFPI, FOEEF, and EKF filters. The objective is to assess the relative efficiency of each filter when the number of training epochs is increased from 18 to 22. The results of this evaluation are presented in Table 3.

The least RMSE value for the SOEEKFPI filter is 0.0683, occurring at the 13^{th} epoch. For the FOEEF filter, the minimum RMSE is 0.7284 at the 15^{th} epoch, while the EKF filter attains its lowest RMSE of 0.9717 at the 18^{th} epoch. Comparatively, the least RMSE of FOEEF is approximately 10.67 times higher than that of SOEEKFPI, and the least RMSE of EKF is about 14.23 times higher than that of SOEEKFPI, indicating a significant performance gap. Furthermore, the average RMSE across the 22 epochs is 0.0762 for SOEEKFPI, 1.0247 for FOEEF, and 1.0393 for EKF. These results

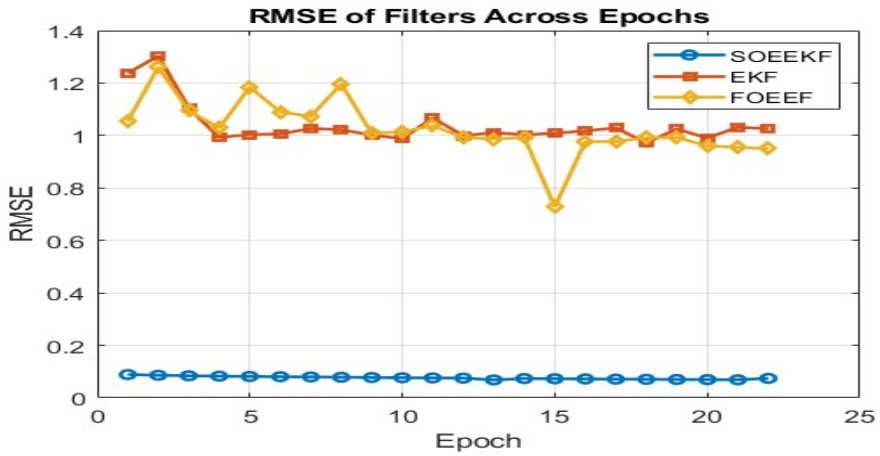

Fig. 3.8. Training and testing using $\sin(x) + Q$ in 1-4-1 Architechture for 22 epochs

clearly demonstrate that the SOEEKFPI filter achieves substantially lower error values than both FOEEF and EKF. The summarized results of this comparison are presented in Table 4.

Table 4. Summary RMSE values

Type of filter	network structure	Epoch	MinRMSE	Average RMSE
SOEEKF	1-4-1	13	0.068287381	0.076231575
EKF	1-4-1	18	0.971747845	1.039349442
FOEEF	1-4-1	15	0.728351798	1.024744482

Conclusion: The comparison of RMSE values across the SOEEKFPI, FOEEF, and EKF filters demonstrates that SOEEKFPI is significantly more efficient in initializing and estimating the weights of the ANN model. Not only does SOEEKFPI achieve the lowest average RMSE among the three filters, but it also converges more rapidly, attaining its minimum RMSE at the 13^{th} epoch. These findings affirm the superior performance of the SOEEKFPI filter in terms of both accuracy and convergence speed.

4 Estimating SoEEKFPI, FoEEF and EKF on a 1-4-1 ANN by use of $sin^3(x) + sin^2(x) + sin(x) + 1 + Q$ function with 18 Epochs

This study now investigates the performance of the SOEEKFPI, FOEEF, and EKF filters in initializing and estimating the weights of an ANN model using a more complex trigonometric function, defined as $\sin^3(x) + \sin^2(x) + \sin(x) + 1 + Q$. A 1-4-1 ANN architecture is employed, trained over 18 epochs. The objective of this experiment is to assess both the speed and accuracy of the ANN models when initialized with each filter on this new, more challenging function. The corresponding graphical outputs are presented in Figures 4.1, 4.2, and 4.3, representing the results for SOEEKFPI, FOEEF, and EKF, respectively.

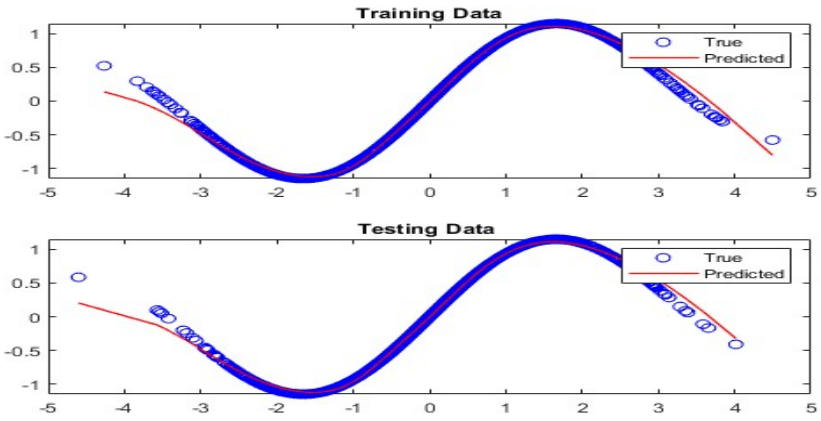

Fig. 4.1. Experimentation on SOEEKFPI within 18 epochs

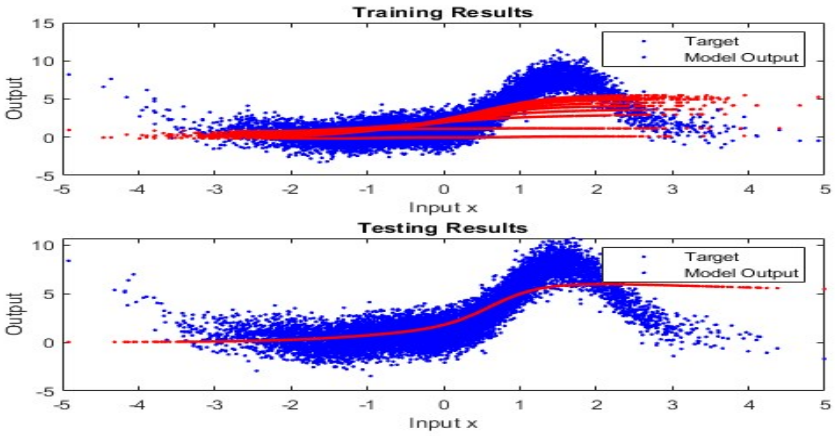

Fig. 4.2. Experimentation on FoEEF within 18 epochs

Analysis: From the results illustrated in Figures 4.1, 4.2, and 4.3, it is evident that the SOEEKFPI filter demonstrates superior performance in comparison to the FOEEF and EKF filters. The output of the ANN model initialized and trained with SOEEKFPI consistently aligns more closely with the expected target values, both for training and testing datasets. This is clearly shown by the blue line in the plots, which represents the predicted outputs using SOEEKFPI, and remains consistently closer to the target curve than the other models across all epochs. In contrast, the orange line representing the FOEEF-based model shows a higher level of deviation, indicating less accurate predictions. The EKF-based model, represented by the green or red line (depending on plot styling), exhibits the largest deviations and the highest residual errors. These graphical results confirm that SOEEKFPI not only offers improved convergence during training but also achieves greater generalization ability, yielding lower prediction error across a more complex trigonometric function.

Conclusion: The study observes that the SOEEKFPI filter consistently outperforms the FOEEF and EKF filters in both learning speed and accuracy when applied to a more complex trigonometric function, specifically $\sin^3(x)+\sin^2(x)+\sin(x)+1+Q$, using a 1-4-1 ANN model architecture trained over 18 epochs. In this experiment, the Root Mean Square Error (RMSE) is computed from the output of each ANN model during training. The primary objective is to compare the efficiency of the three filters in terms of how quickly and accurately they enable the ANN to learn from the data. The graphical analysis shows that the ANN model initialized using SOEEKFPI yields a predicted output (plotted in blue)

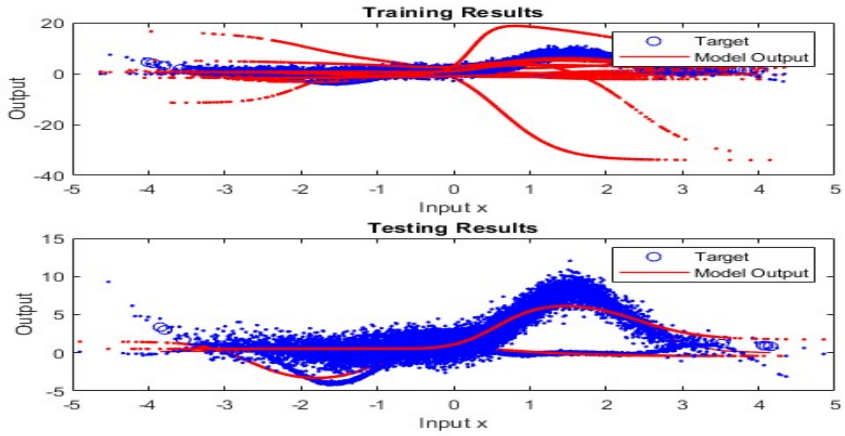

Fig. 4.3. Experimentation on EKF within 18 epochs

that remains consistently closest to the expected output, indicating the lowest RMSE across epochs. This performance reflects the superior generalization ability and convergence speed of the SOEEKFPI-based model. The FOEEF-based model follows next in performance, with its corresponding output (orange plot) showing moderate deviation above the SOEEKFPI curve. The EKF-based model trails in performance with the largest deviation and highest RMSE values. The detailed RMSE results for this experiment are summarized in Table 5, further validating that SOEEKFPI provides a more efficient and accurate approach for initializing and estimating the weights of ANN models, even under more complex functional mappings.

Table 5. RMSE results from SOEEKFPI, FOEEF, and EKF on $sin^3(x) + sin^2(x) + sin(x) + 1 + Q$ for 18 epochs

Epoch	SOEEKFPI	EKF	FOEEF
1	0.3814	3.2364	1.1232
2	0.3282	3.2505	1.0554
3	0.3034	2.1407	1.0171
4	0.2898	3.1934	0.9955
5	0.2799	1.4235	1.0184
6	0.2718	1.4675	0.9001
7	0.2009	1.4581	0.9667
8	0.2583	1.3766	0.92847
9	0.2525	1.4396	0.9477
10	0.2475	1.3947	0.8745
11	0.2429	1.4226	0.87024
12	0.2387	1.4179	0.8517
13	0.2345	1.4173	0.9014
14	0.2287	1.5195	0.87011
15	0.2167	1.3693	0.8934
16	0.2135	1.4968	0.8731
17	0.2647	1.4398	0.8866
18	0.2085	1.5906	0.86054

Learning Curve: Figure 4.4 illustrates the learning curves of the ANN models whose weights were initialized and estimated using SOEEKFPI, FOEEF, and EKF filters, respectively. The purpose of this experiment is to evaluate and compare the learning speed and efficiency of these models when trained to approximate the complex function $y = \sin^3(x)+\sin^2(x)+\sin(x)+1+Q$ within a 1-4-1 ANN architecture. The graph plots the Mean Squared Error (MSE) on the y-axis against the number of training epochs on the x-axis. A steeper decline in the MSE curve indicates faster learning

and better model convergence. As observed, the ANN initialized with SOEEKFPI exhibits the most rapid decrease in MSE, demonstrating superior learning efficiency compared to the FOEEF and EKF initialized models. This confirms that SOEEKFPI not only achieves lower error rates but also requires fewer training epochs to reach optimal performance, highlighting its effectiveness for complex function approximation in neural network training.

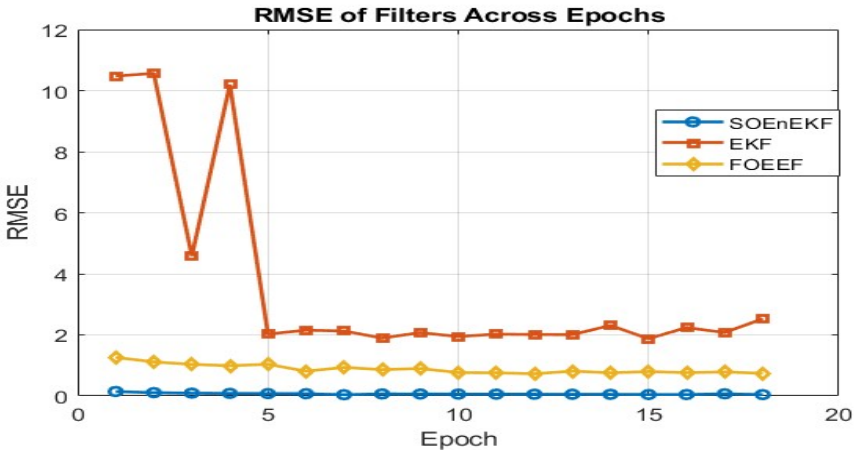

Fig. 4.4. Training and testing using $y = \sin^3(x) + \sin^2(x) + \sin(x) + 1 + Q$ in 1-4-1 Architecture for 18 epoch

Analysis: From Table 5, the ANN model initialized and trained using SOEEKFPI achieves the lowest RMSE of 0.2009 at the 7th epoch. In comparison, the FOEEF-initialized model attains its minimum RMSE of 0.8517 at the 12th epoch, while the EKF-initialized model records its lowest RMSE of 1.369 at the 15th epoch. Notably, the minimum RMSE of the FOEEF model is approximately 4.24 times larger than that of the SOEEKFPI model, and the EKF's minimum RMSE is about 6.79 times larger. Additionally, the average RMSE over all epochs further highlights the superior performance of SOEEKFPI, with an average value of 0.25899 compared to 0.93523 for FOEEF and 1.78082 for EKF. These results, summarized in Table 6, clearly demonstrate the improved accuracy and faster convergence of the ANN model when weights are initialized and estimated using the SOEEKFPI method.

Table 6. Summary RMSE values

Type of filter	network structure	Epoch	MinRMSE	Average RMSE
SOEEKF	1-4-1	7	0.2009	0.25899
EKF	1-4-1	15	1.3693	1.780817222
FOEEF	1-4-1	12	0.8517	0.93523

Conclusion: This experiment establishes that SOEEKFPI is more efficient than both EKF and FOEEF in initializing and estimating the weights of ANN models. The ANN trained with weights initialized and estimated by SOEEKFPI converges faster and achieves the lowest average RMSE. This demonstrates that SOEEKFPI is the most efficient and effective method among the three.

Analysis: From Figure 4.4, the RMSE trend for SOEEKFPI consistently remains lower compared to those of FOEEF and EKF. As the number of epochs increases, the RMSE for SOEEKFPI decreases more rapidly than for the other two filters.

Conclusion: Based on this observed trend, it is evident that SOEEKFPI outperforms FOEEF and EKF. Furthermore, the graph suggests that SOEEKFPI neither overfits nor underfits the learning process, indicating robust generalization capability.

5 Conclusion

This study presents the development of a new filter—the Second-Order Extended Ensemble Kalman Filter with Perturbed Innovation (SOEEKFPI). The filter is derived from state-space stochastic dynamics using the Kushner–Stratonovich equation for posterior density evolution, employing a Bayesian framework. Unlike traditional filters, SOEEKFPI is robust, computationally efficient, and suitable for high-dimensional systems. Covariance estimation is performed empirically, avoiding costly integrations. To validate its performance, the filter is tested on simulated data from the Lorenz 63 system and compared against FOEEF, BPF, SOEKBF, and FOEKBF filters. The results show that SOEEKFPI outperforms all the others in both accuracy and convergence. Additionally, the study proposes a novel approach for initializing and estimating artificial neural network (ANN) weights using SOEEKFPI. Comparative experiments with FOEEF and EKF demonstrate that the ANN trained using SOEEKFPI achieves closer alignment between predicted and target outputs across both training and testing datasets. Further analysis includes evaluating RMSE over 18 and 22 epochs, where SOEEKFPI consistently attains the lowest RMSE at the fewest epochs. Learning curves plotted for each model confirm that SOEEKFPI enables faster generalization and more efficient training. Overall, the findings establish SOEEKFPI as a superior Bayesian method for weight initialization and estimation in ANN models, outperforming FOEEF and EKF in all phases of the experiment.Further research is on application of SoEEF in innializing ANN weights.

References

[1] Angwenyi, D., and Oryiema, R. (2023). *An Extended Ensemble Kalman Filter for Nonlinear State Estimation*, Journal of Computational Dynamics, vol. 12, no. 3, pp. 123-135.

[2] Angwenyi, D., and Oryiema, R. (2023). *First Order Extended Ensemble Filter for Neural Network Weight Estimation*, Proceedings of the International Conference on Computational Intelligence, pp. 45–54.

[3] Chen, Y., Chi, Y., Fan, J., & Ma, C. (2019). Gradient descent with random initialization: Fast global convergence for nonconvex phase retrieval. *Mathematical Programming*, **176**.

[4] Corenflos, A., Thornton, J., Deligiannidis, G., & Doucet, A. (2021). Differentiable particle filtering via entropy-regularized optimal transport. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 2100–2111.

[5] Glorot, X., Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 249–256.

[6] Hardesty, L. (2017). Explained: Neural networks. *MIT News*, **14**.

[7] Liao, K.-H. (2021). *Enhanced Battery State of Charge Estimation by Machine Learning and Unscented Kalman Filter*. California State University, Los Angeles.

[8] Igbida, N. (2009). Equivalent formulations for Monge–Kantorovich equation. *Nonlinear Analysis: Theory, Methods and Applications*, **71**(9), 3805–3813.

[9] He, K., Zhang, X., Ren, S., Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1026–1034.

[10] Makkuva, A., Taghvaei, A., Oh, S., & Lee, J. (2020). Optimal transport mapping via input convex neural networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 6672–6681.

[11] Marron, M., Garcia, J. C., Sotelo, M. A., Cabello, M., Pizarro, D., Huerta, F., & del Cerro, J. (2007). Comparing a Kalman filter and a particle filter in a multiple objects tracking application. In *Proceedings of the 2007 IEEE International Symposium on Intelligent Signal Processing (WISP)*, IEEE.

[12] Murru, N., & Rossini, R. (2020). A Bayesian approach for initialization of weights in backpropagation neural net with application to character recognition. *arXiv preprint arXiv:2004.01875*.

[13] Midenyo, K., Angwenyi, D., & Oganga, D. (2024). Second Order Extended Ensemble Filter for Non-linear Filtering. *African Journal of Empirical Research*, 5(4), 302–316. doi:10.51867/ajernet.mathematics.5.4.25

[14] Mérigot, Q., & Thibert, B. (2021). Optimal transport: discretization and algorithms. In P. G. Ciarlet & T. Gallouët (Eds.), *Handbook of Numerical Analysis*, Vol. 22, pp. 133–212. Elsevier.

[15] Narkhede, S. (2021). Weight Initialization Techniques in Neural Networks: A Brief Overview. *Towards Data Science*. Retrieved from https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78

[16] Oryiema, R., and Angwenyi, D, (2023) *Improved Extended Kalman Filter Techniques for Neural Network Initialization*, International Journal of Neural Systems, vol. 18, no. 2, pp. 85–98.

[17] Sarfaraz, A., Mohsin, M. (2018). Neural Network Weight Initialization: A Survey. *International Journal of Computer Applications*, 179(10), 1–6.

[18] Xia, N., Qiu, T.-S., Li, J.-C., & Li, S.-F. (2013). A nonlinear filtering algorithm combining the Kalman filter and the particle filter. *Acta Electronica Sinica*, **41**.