Neural Identifier Training with Particle Filters

Dariana Gomez-Alvarez[1][0009−0002−7286−2083], Michel Lopez-Franco[1][0000−0002−4245−9683], Alma Y. Alanis[1][0000−0001−9600−779X], Carlos Lopez-Franco[1][0000−0001−8122−3799], Jesus Hernandez-Barragan[1][0000−0001−7518−1668], and Edgar N. Sanchez[0000−0002−8695−7879]

University of Guadalajara, Guadalajara, Jal. 44430, Mexico
maria.gomez2866@alumnos.udg.mx

Abstract. Nonlinear system identification is essential for modeling complex dynamics where analytical approaches fail. Recurrent high-order neural networks (RHONNs) excel in this task but require robust training methods. Traditional approaches like the Extended Kalman Filter (EKF) struggle with non-Gaussian noise and local linearization constraints. This paper proposes Particle Filter (PF) based training for RHONNs, enabling flexible and accurate state estimation. Simulations on a pendulum and a Duffing oscillator show that PF significantly reduces identification errors compared to EKF, especially under non-Gaussian noise. The results confirm PF's potential for robust and adaptive neural training, enhancing nonlinear system identification for applications in control and robotics.

Keywords: Neural identification · Nonlinear systems · Recurrent high-order neural networks · Particle filter · Online training.

1 Introduction

In control systems, a robust understanding of the target system is paramount for designing effective control strategies [10, 14, 15, 18]. Without this understanding, the resulting control inputs may lack robustness or fail entirely. Nonlinear system identification addresses this need by replicating the dynamic behavior of the original system in a mathematical model, enabling precise control design. Among the various approaches, neural networks have emerged as a powerful resource for nonlinear system identification due to their exceptional capacity to model complex nonlinearities [9, 13].

Recurrent Neural Networks (RNNs) and their advanced variants, RHONNs, are particularly effective [2, 17]. These architectures surpass traditional multilayer perceptrons by leveraging recurrent connections, which introduce memory into the model, enabling superior dynamic representation. High-order connections further enhance efficiency by reducing the number of required neurons, making RHONNs highly suitable for real-time applications [1].

The EKF has been one of the most widely used methods for training neural networks [16]. It extends the Kalman Filter's (KF) capability to manage nonlinear systems through local linearization. However, EKF's reliance on linearization

https://doi.org/10.61728/AE20255213

can result in the loss of significant nonlinear behaviors. Additionally, both the KF and EKF assume Gaussian noise distributions, a limitation in scenarios involving robotics or mobile systems where noise characteristics often deviate from Gaussian assumptions [4].

To address these limitations, this work explores the use of PFs as an alternative training method for RHONNs. PFs have demonstrated superior performance in highly nonlinear systems, often surpassing the KF and its variants [3, 7]. Unlike EKF, PFs are not constrained by Gaussian noise assumptions and can approximate arbitrary probability distributions, making them a versatile choice for nonlinear system identification [6].

This paper introduces the foundational concepts of neural network-based identifiers and provides the mathematical preliminaries of the Particle Filter. Furthermore, simulation results are presented to demonstrate the identification of nonlinear systems. The structure of this work is as follows: First, Recurrent High-Order Neural Networks are introduced as one of the most effective methodologies for addressing the system identification problem. Next, the generic Particle Filter algorithm is explained, along with its application to state estimation. The discussion then shifts to the use of the Particle Filter for training Recurrent High-Order Neural Networks, highlighting its advantages in handling non-Gaussian noise and strong nonlinearities. Finally, the results of two simulations are presented, comparing the performance of the Extended Kalman Filter and the Particle Filter in terms of their identification Mean Squared Error. The results are analyzed, and the superiority of the Particle Filter over the Extended Kalman Filter is established.

2 Recurrent High-order Neural Networks

For control tasks, RHONNs present numerous advantages, including their exceptional approximation capabilities and robustness to noise [1].

Consider a nonlinear system in discrete-time:

$$\chi(k+1) = F(\chi(k), u(k)) \tag{1}$$

where:

- $\chi \in \mathbb{R}^n$ is the state vector of the plant.
- $u \in \mathbb{R}^m$ is the external input vector.
- $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a nonlinear function.
- $k \in \mathbb{Z}^+$ represents discrete-time steps.

To identify the system shown in Equation (1), we consider the following discrete-time RHONN:

$$x_i(k+1) = w_i^\intercal \Lambda_i(\chi(k), u(k)), \quad i = 1, \ldots, n \tag{2}$$

where:

- x_i is the state of the i-th neuron.
- w_i is the vector of online-adjusted weights.
- n is the state dimension.
- Λ_i is described by Equation (3).

$$\Lambda_i(\chi(k), u(k)) = \begin{bmatrix} \prod_{j \in I_1} \lambda_{i_j}^{d_{i_j}(1)} \\ \prod_{j \in I_2} \lambda_{i_j}^{d_{i_j}(2)} \\ \vdots \\ \prod_{j \in I_L} \lambda_{i_j}^{d_{i_j}(L)} \end{bmatrix} \tag{3}$$

with:

$$\lambda_i = \begin{bmatrix} \lambda_{i_1} \\ \vdots \\ \lambda_{i_n} \\ \lambda_{i_{n+1}} \\ \vdots \\ \lambda_{i_{n+m}} \end{bmatrix} = \begin{bmatrix} S(\chi_1) \\ \vdots \\ S(\chi_n) \\ u_1 \\ \vdots \\ u_m \end{bmatrix} \tag{4}$$

where:

- m is the number of external inputs.
- L is the number of higher-order connections.
- $\{I_1, I_2, \ldots, I_n\}$ is a collection of non-ordered sets of dimensions $\{1, 2, \ldots, n+m\}$.
- d_{i_j} are nonnegative integers.

The function $S(\cdot)$ is a monotonic increasing, differentiable sigmoidal function defined as

$$S(\varsigma) = \alpha \frac{1}{1 + e^{-\beta\varsigma}} - \gamma, \tag{5}$$

where α and β are positive real numbers, and γ is a real number. A special case arises when $\alpha = \beta = 1$ and $\gamma = 0$, which results in the logistic function. Similarly, setting $\alpha = \beta = 2$ and $\gamma = 1$ yields the hyperbolic tangent function [2, 13, 17].

3 Particle Filter

The PF is a *Sequential Monte Carlo (SMC)* method used to estimate the posterior distribution of a state given noisy measurements by representing it as a set of weighted particles [4]. This approach is particularly effective in highly nonlinear and non-Gaussian state estimation problems, where traditional filtering

techniques such as the KF and its extended version struggle due to their reliance on Gaussian noise assumptions and local linearization [3].

In a PF, the posterior distribution of the state given the sequence of measurements $z(1:k)$ is approximated as:

$$p(\rho(k)|z(1:k)) \approx \sum_{s=1}^{N} \omega_s(k)\delta(\rho(k) - \rho_s(k)) \tag{6}$$

where:

- $\rho_s(k)$ are the sampled particles, which represent possible realizations of the state.
- $\omega_s(k)$ are the importance weights, indicating the likelihood of each particle given the observations.
- $\delta(\cdot)$ is the Dirac delta function, which expresses the fact that the posterior is approximated by a set of discrete samples rather than a continuous function [7].

This formulation allows PF to approximate arbitrary probability distributions, making it more robust for real-world applications where sensor noise and environmental disturbances often exhibit multimodal or heavy-tailed distributions [8, 12].

3.1 PF Algorithm Steps

PF operates sequentially as follows:

1. **Initialization** Particles are randomly sampled from a uniform distribution [4]:

$$\rho_s(0) \sim U(\rho_{\min}, \rho_{\max}) \tag{7}$$

and initialized with equal weights:

$$\omega_s(0) = \frac{1}{N}, \quad p = 1, \ldots, N \tag{8}$$

2. **Prediction** Each particle evolves according to an importance density function [3]:

$$\rho_s(k) \sim q(\rho(k)|\rho_s(k-1), z(k)) \tag{9}$$

In practice, this is often chosen as the system's transition model:

$$\rho_s(k) = f(\rho_s(k-1), u(k), v(k-1)) \tag{10}$$

3. **Weight Update** Each particle weight is adjusted based on the measurement likelihood [7]:

$$\omega_s(k) = \omega_s(k-1)p(z(k)|\rho_s(k)) \tag{11}$$

To maintain a valid probability distribution, weights are normalized [8]:

$$\omega_s(k) = \frac{\omega_s(k)}{\sum_{j=1}^{N} \omega_j(k)} \tag{12}$$

The weights are normalized to ensure they form a valid probability distribution, satisfying the condition that their sum equals unity. This normalization is essential for maintaining the probabilistic interpretation of the particle weights in the filtering process.

4. **Resampling** To address the weight degeneracy problem, resampling is performed when the effective sample size falls below a predefined threshold [5]:

$$N_{\text{eff}} = \frac{1}{\sum_{s=1}^{N} (\omega_s(k))^2} \tag{13}$$

Particles with low weights are discarded, while those with higher weights are duplicated:

$$\rho_s(k) = \rho_\tau^*(k) \tag{14}$$

Resampling prevents particle impoverishment, where a small subset of particles dominates the estimation, reducing diversity. Common resampling techniques include systematic resampling, multinomial resampling, and stratified resampling each balancing trade-offs between computational efficiency and variance reduction [3].

5. **State Estimation** The estimated state is obtained as the weighted sum of the particles:

$$\hat{\rho}(k) = \sum_{s=1}^{N} \omega_s(k)\rho_s(k) \tag{15}$$

This expectation-based estimation ensures a statistically representative approximation of the state [8].

4 The PF Training Algorithm

Each particle in the PF represents a potential configuration of the RHONN weights. The training process consists of iteratively updating these particles based on system observations, ensuring an accurate approximation of the nonlinear system dynamics. The PF training algorithm follows a sequential Bayesian estimation approach, as outlined in Algorithm 1.

Algorithm 1 Particle Filter-based Online Training for RHONN

Require: Number of particles N, initial RHONN weights $\{w_s(0)\}_{s=1}^{N}$, number of time steps K, system measurements $\{\chi(k)\}_{k=1}^{K}$, process noise variance σ_v^2, observation noise variance σ_n^2

Initialization:
for $s = 1$ to N **do**
  Sample initial RHONN weights: $w_s(0) \sim p(w_0)$
  Assign equal particle weights: $\omega_s(0) = \frac{1}{N}$
end for
for $k = 1$ to K **do**
  Prediction Step:
  for $s = 1$ to N **do**
    Sample predicted RHONN weights:

$$w_s(k) \sim \mathcal{N}(w_s(k-1), \sigma_q^2)$$

    Compute RHONN output:

$$x_s(k) = f_{\text{RHONN}}(w_s(k), \chi(k-1))$$

  end for
  Weight Update:
  for $s = 1$ to N **do**
    Compute measurement likelihood:

$$p(\chi(k)|w_s(k)) = \mathcal{N}(\chi(k); x_s(k), \sigma_n^2)$$

    Update particle weights:

$$\omega_s(k) = \omega_s(k-1)p(\chi(k)|w_s(k))$$

  end for
  Normalize Particle Weights:
  for $s = 1$ to N **do**
    Normalize:

$$\omega_s(k) = \frac{\omega_s(k)}{\sum_{j=1}^{N} \omega_j(k)}$$

  end for
  Resampling (if needed):
  Compute effective sample size:

$$N_{\text{eff}} = \frac{1}{\sum_{s=1}^{N} \omega_s(k)^2}$$

  if $N_{\text{eff}} < N/2$ **then**
    Resample particles $\{w_s(k)\}_{s=1}^{N}$ using systematic resampling
    Reset particle weights (if using systematic resampling):

$$\omega_s(k) = \frac{1}{N}, \quad \forall s$$

  end if
  Weight Estimation:
  Compute the estimated RHONN weights:

$$\hat{w}(k) = \sum_{s=1}^{N} \omega_s(k) w_s(k)$$

end for
Return final estimated RHONN weights: $\hat{w}(K)$

4.1 Initialization

The training process begins by initializing a population of N particles, where each particle represents a candidate set of RHONN weights. These initial weights are sampled from a prior distribution to ensure a diverse search space, following common practices in Particle Filters for system identification [3, 4]:

$$w_s(0) \sim p(w_0), \quad s = 1, \ldots, N \tag{16}$$

where $p(w_0)$ is typically chosen as a uniform or Gaussian distribution, ensuring that the search space is sufficiently explored. Each particle is assigned an initial importance weight:

$$\omega_s(0) = \frac{1}{N} \tag{17}$$

This uniform weighting represents an equal prior belief in all candidate solutions before any observations are made, a common initialization approach in Sequential Monte Carlo methods [5, 8].

Using the sampled weights, the RHONN generates an initial estimate of the system state:

$$x_s(0) = f_{\text{RHONN}}(w_s(0), x(0)) \tag{18}$$

where $x_s(0)$ is the predicted system state corresponding to the s-th particle. This initialization provides the foundation for the iterative refinement of RHONN weights through the Particle Filter process. The diversity of initial particles is essential to avoid premature convergence and ensure an adequate exploration of the weight space [12].

4.2 Prediction

Once the particles are initialized, the next step is the prediction phase, where each particle is propagated according to an importance density function. A common choice for this function is the system's transition model [3]:

$$w_s(k) \sim q(w(k)|w_s(k-1), x(k)) \tag{19}$$

This step introduces diversity by perturbing the particles, ensuring that the weight distribution can adapt as new observations are received. Using the predicted weights, the RHONN generates a state estimate for each particle:

$$x_s(k) = f_{\text{RHONN}}(w_s(k), x(k-1)) \tag{20}$$

where $x_s(k)$ is the predicted system state obtained using the RHONN with the sampled weights $w_s(k)$. This prediction will be evaluated against the true system state in the next step.

4.3 Weight Update

The weight update step refines the probability distribution over the RHONN weights by assigning higher weights to particles that better match the observed system behavior. The measurement likelihood is computed using a Gaussian distribution [5, 8]:

$$p(x(k)|w_s(k)) = \mathcal{N}(x(k); x_s(k), \sigma_n^2) \tag{21}$$

where σ_n^2 represents the variance of the measurement noise. The weights are then updated using Bayes' rule:

$$\omega_s(k) = \omega_s(k-1)p(x(k)|w_s(k)) \tag{22}$$

Finally, the weights are normalized to ensure they sum to unity:

$$\omega_s(k) = \frac{\omega_s(k)}{\sum_{j=1}^{N} \omega_j(k)} \tag{23}$$

This normalization step maintains a valid probabilistic interpretation of the particle weights [7].

4.4 Resampling

Resampling is performed to prevent degeneracy, a common issue in particle filters where most particles end up with negligible weights [3, 4]. The effective sample size is computed as:

$$N_{\text{eff}} = \frac{1}{\sum_{s=1}^{N} \omega_s(k)^2} \tag{24}$$

If N_{eff} falls below a predefined threshold (e.g., $N/2$), resampling is triggered. This step removes particles with low weights and duplicates those with higher weights, ensuring that the weight distribution remains well-represented:

$$w_s(k) = w_{\tau(s)}(k) \tag{25}$$

where $\tau(s)$ is an index mapping function that selects particles based on their weights. Systematic resampling is commonly used due to its computational efficiency and reduced variance [6].

4.5 Weight Estimation

After resampling, the RHONN weights are estimated by computing the weighted average over all particles:

$$\hat{w}(k) = \sum_{s=1}^{N} \omega_s(k) w_s(k) \tag{26}$$

This estimation represents the most probable RHONN weight configuration at time k given the observations up to that point. By iterating through the Particle Filter process, the RHONN gradually adapts its weights to accurately model the nonlinear system dynamics [12].

5 Simulation Results

To assess the performance of the PF as an online training method for the RHONN in system identification, multiple simulations were conducted. These simulations covered a range of models, from simple mathematical systems to highly nonlinear dynamics, incorporating both Gaussian and non-Gaussian noise distributions to evaluate robustness. However, the results presented focus on a simple pendulum and a Duffing oscillator. Both systems were identified using the same RHONN architecture and experimental conditions, with training performed via the EKF and, in contrast, the proposed PF-based approach. The Mean Squared Error (MSE) was used as the primary performance metric to quantify identification accuracy.

To demonstrate the superiority of the PF over the EKF in training RHONNs, Laplacian noise was added in both the process dynamics and measurement outputs. Unlike Gaussian noise, which is commonly assumed in classical estimation techniques, Laplacian noise has a sharp peak and heavy tails, making it more representative of real-world disturbances such as sensor faults, sudden perturbations, and outliers in robotic and engineering applications [3, 7, 11]. While the EKF relies on local linearization and assumes Gaussian noise [14, 19], the PF estimates the full posterior distribution using a set of weighted particles, making it well-suited for handling non-Gaussian noise and strong nonlinearities [4, 8, 12].

5.1 Pendulum System

To evaluate the effectiveness of the PF in training the RHONN, we first consider the identification of a simple pendulum system. The pendulum exhibits nonlinear dynamics due to the presence of the sine function, making it an ideal test case for assessing the robustness of different identification techniques under noisy conditions.

The system dynamics are governed by the following differential equation:

$$\frac{d^2\theta}{dt^2} + \frac{b}{\mu}\frac{d\theta}{dt} + \frac{g}{l}\sin\theta + \phi = 0 \tag{27}$$

where:

- θ represents the angular displacement of the pendulum.
- μ is the mass of the pendulum.
- l is the length of the pendulum.
- g denotes the acceleration due to gravity.

- b is the damping coefficient, representing energy loss due to friction or air resistance.
- ϕ represents process noise, which introduces stochastic perturbations to the system dynamics.

For system identification, we employ a RHONN model, which approximates the system's behavior through adaptive weight updates. The discrete-time RHONN model is defined as:

$$\begin{aligned} x_1(k+1) &= w_{11}S(\chi_1(k)) + w_{12}S(\chi_2(k))^2 \\ x_2(k+1) &= w_{21}S(\chi_1(k))^2 + w_{22}S(\chi_2(k)) \end{aligned} \tag{28}$$

where $S(\cdot)$ was chosen as the hyperbolic tangent function, and w_{ij} are the online trained weights.

The initial conditions for both the pendulum system and the RHONN models trained with the PF and EKF are identical:

$$\chi_1(0) = x_{1_{PF}}(0) = x_{1_{EKF}}(0) = \frac{\pi}{4} \tag{29}$$

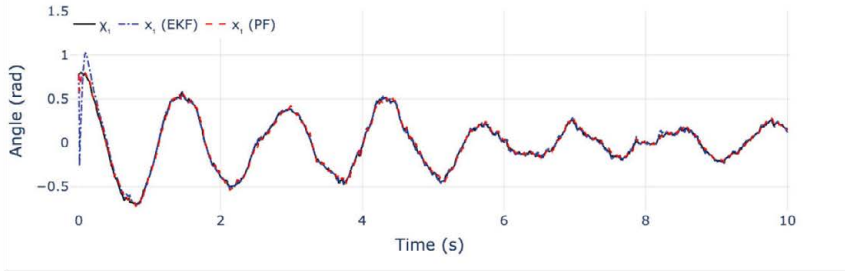
$$\chi_2(0) = x_{2_{PF}}(0) = x_{2_{EKF}}(0) = 0 \tag{30}$$

This ensures a fair comparison between the two training approaches.

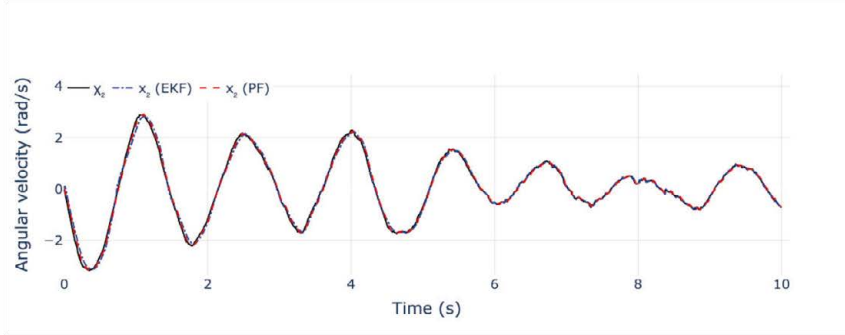
The training was performed using:

- EKF: This method assumes Gaussian noise and relies on local linearization, potentially limiting its effectiveness in highly nonlinear settings.
- PF: This method estimates the posterior distribution of the RHONN weights using a set of particles, allowing it to handle non-Gaussian noise and strong nonlinearities.

Results and Performance Analysis The system response, estimated by both RHONN models, is shown in Figure 1. The PF-trained RHONN demonstrates superior accuracy in tracking the pendulum's motion, particularly in the presence of Laplacian noise. This is further validated through quantitative analysis of the MSE, presented in Table 1.



(a) Pendulum's Angular Position



(b) Pendulum's Angular Velocity

Fig. 1: Comparison of PF-trained RHONN and EKF-based training for pendulum system identification. The PF method achieves more accurate tracking in both position and velocity, particularly under non-Gaussian noise conditions.

Table 1: Mean Squared Errors (MSE) for Different Training Methods - Pendulum System

Training Method	**MSE for x_1 (Position)**	**MSE for x_2 (Velocity)**
Particle Filter	**0.00065**	**0.00602**
Extended Kalman Filter	0.00291	0.01298

KeyObservations:

- The PF-based RHONN achieves significantly lower MSE values compared to the EKF-based RHONN, with a 77.8% reduction in position estimation error and a 53.6% reduction in velocity estimation error.
- The difference in performance is more pronounced in velocity estimation, where EKF struggles due to its reliance on local linearization.
- The presence of Laplacian noise degrades EKF performance more severely than PF, reinforcing the latter's advantage in handling non-Gaussian uncertainties.

5.2 Duffing Oscillator System

The Duffing oscillator is a well-known nonlinear system used to model oscillatory phenomena in physics and engineering. It is commonly employed to study chaotic behavior, bistable dynamics, and resonance effects. The Duffing equation describes the motion of a damped, periodically driven oscillator with a nonlinear restoring force and is given by:

$$\frac{d^2x}{dt^2} + \delta\frac{dx}{dt} + \alpha x + \beta x^3 = \gamma\cos(\omega t) + \phi \tag{31}$$

where:

- x represents the system's displacement.
- δ is the damping coefficient, governing energy dissipation.
- α and β define the nonlinear restoring force, with βx^3 introducing nonlinearity.
- γ represents the external forcing amplitude.
- ω is the forcing frequency.
- ϕ is the process noise, modeling external perturbations.

This system exhibits both periodic and chaotic motion, depending on parameter values, making it a challenging benchmark for system identification.

To approximate the Duffing oscillator's dynamics, it is employed a RHONN, following the discrete-time representation:

$$\begin{aligned} x_1(k+1) &= w_{11}S(\chi_1(k))^3 + w_{12}S(\chi_2(k)) \\ x_2(k+1) &= w_{21}S(\chi_1(k))^2 + w_{22}S(\chi_2(k))^3 \end{aligned} \tag{32}$$

where:

- $S(\cdot)$ is a sigmoidal activation function ensuring smooth, bounded outputs.
- w_{ij} are the adaptive weights, updated during training.

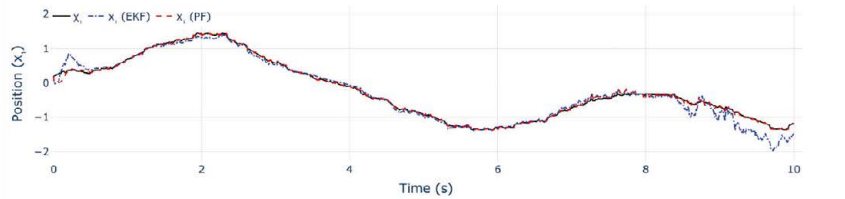
The initial conditions for both the Duffing system and the RHONN models trained with PF and EKF are set to:

$$\chi_1(0) = x_{1_{PF}}(0) = x_{1_{EKF}}(0) = 0.2 \tag{33}$$

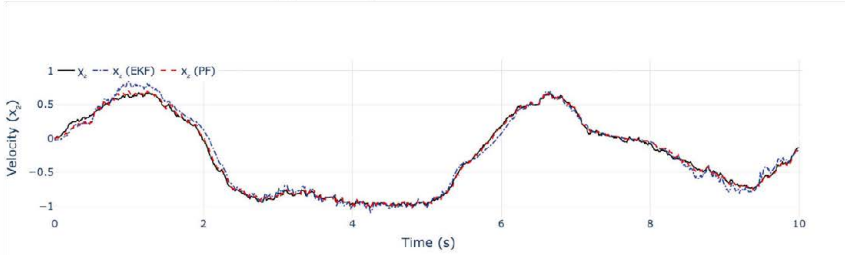
$$\chi_2(0) = x_{2_{PF}}(0) = x_{2_{EKF}}(0) = 0 \tag{34}$$

This ensures consistency in evaluating both training approaches under identical conditions.

Results and Performance Comparison The system response estimated by the RHONNs trained via both methods is shown in Figure 2. The PF-trained RHONN achieves noticeably improved accuracy in tracking both position and velocity, particularly in regions where nonlinearities dominate the system dynamics.



(a) Duffing Oscillator's Position



(b) Duffing Oscillator's Velocity

Fig. 2: Comparison of PF-trained RHONN and EKF-based training for Duffing oscillator identification. The PF method provides more accurate state estimation, particularly in capturing the system's nonlinear behavior.

Table 2: Mean Squared Errors (MSE) for Different Training Methods - Duffing Oscillator

Training Method	**MSE for x_1 (Position)**	**MSE for x_2 (Velocity)**
Particle Filter	**0.00184**	**0.00098**
Extended Kalman Filter	0.02284	0.00477

Key Observations:

– The PF-based RHONN significantly outperforms EKF, yielding lower MSE values for both position and velocity.

- The PF achieves a 91.93% reduction in position estimation error and a 79.40% reduction in velocity estimation error compared to the EKF.
- The EKF struggles with velocity estimation, likely due to its reliance on local linearization, which may not sufficiently capture the oscillator's chaotic behavior.
- The PF training method better handles Laplacian noise, leading to more robust identification results.

The superior performance of the PF-trained RHONN can be attributed to its ability to better approximate complex probability distributions, enabling more precise weight adaptation even in the presence of strong nonlinearities and stochastic perturbations. These results further support the use of PF-based training for nonlinear system identification where conventional approaches like EKF may be insufficient.

6 Conclusion

This study investigated the use of PFs as an alternative training method for RHONNs in nonlinear system identification. The proposed approach was tested with comparisons against the traditional EKF training method.

The results demonstrate that the PF-trained RHONN outperforms the EKF approach in terms of accuracy, particularly when handling non-Gaussian noise. The MSEs indicate that the PF-based method achieves lower identification errors, making it a more robust and reliable training strategy for highly nonlinear and uncertain systems. These findings align with previous studies highlighting the advantages of particle filters in Bayesian state estimation for nonlinear systems [3, 4, 7].

One of the main advantages of PF-based training is its ability to approximate arbitrary probability distributions, unlike EKF, which relies on Gaussian noise assumptions and local linearization [8, 16]. This makes PF particularly effective for real-world applications, where sensor noise and environmental disturbances often follow heavy-tailed distributions [7].

Despite these advantages, computational complexity remains a challenge, as PFs require resampling steps and a large number of particles to maintain accuracy [5, 12]. Future research will focus on optimizing particle resampling techniques and improving computational efficiency to enable real-time implementations in embedded control systems while exploring hybrid approaches like PF-EKF fusion.

Overall, the results suggest that Particle Filter-based training is a promising alternative to conventional methods, offering improved robustness, accuracy, and adaptability for training RHONNs in nonlinear system identification.

References

1. Alanis, A.Y., Rivera, J., Rangel, E.: Real-time discrete neural control applied to a linear induction motor. Neurocomputing **167**, 522–528 (2015). https://doi.org/10.1016/j.neucom.2015.02.065

2. Alanis, A.Y., Sanchez, E.N., Loukianov, A.G., Pérez-Cisneros, M.A.: Real-time discrete nonlinear identification via recurrent high order neural networks. Computación y Sistemas **14**(1), 63–72 (2010)
3. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online Nonlinear/Non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing **50**(2), 174–188 (2002). https://doi.org/10.1109/78.978374
4. Doucet, A., de Freitas, N., Gordon, N.: Sequential Monte Carlo Methods in Practice. Springer (2001). https://doi.org/10.1007/978-1-4757-3437-9
5. Doucet, A., Godsill, S., Andrieu, C.: On sequential monte carlo sampling methods for bayesian filtering. Statistics and Computing **10**, 197–208 (2000)
6. Doucet, A., Johansen, A.M.: A tutorial on particle filtering and smoothing: Fifteen years later. Handbook of Nonlinear Filtering **12**, 656–704 (2009)
7. Elfring, J., Stramigioli, S., Persis, C.D.: Particle filters: A hands-on tutorial. Sensors **21**(2), 438 (2021). https://doi.org/10.3390/s21020438
8. Gordon, N.J., Salmond, D.J., Smith, A.F.M.: Novel approach to Nonlinear/Non-Gaussian Bayesian state estimation. In: IEE Proceedings F - Radar and Signal Processing. vol. 140, pp. 107–113 (1993). https://doi.org/10.1049/ip-f-2.1993.0015
9. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall (1999)
10. Ioannou, P., Sun, J.: Robust Adaptive Control. Prentice Hall (1996)
11. Jaffer, A., Dudek, G.: Noise-adaptive soft sensors in mobile robots through particle filtering. Robotics and Autonomous Systems **57**(1), 79–92 (2009). https://doi.org/10.1016/j.robot.2008.05.007
12. Kitagawa, G.: Monte carlo filter and smoother for non-Gaussian nonlinear state space models. Journal of Computational and Graphical Statistics **5**(1), 1–25 (1996). https://doi.org/10.2307/1390750
13. Kosmatopoulos, E.B., Polycarpou, M.M., Christodoulou, M.A., Ioannou, P.A.: High-order neural network structures for identification of dynamical systems. IEEE Transactions on Neural Networks **6**(2), 422–431 (1995). https://doi.org/10.1109/72.363477
14. Ljung, L.: System Identification: Theory for the User. Prentice Hall, 2 edn. (1999)
15. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks **1**(1), 4–27 (1990). https://doi.org/10.1109/72.80202
16. Puskorius, G.V., Feldkamp, L.A.: Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. IEEE Transactions on Neural Networks **5**(2), 279–297 (1994)
17. Rios, J.D., Alanis, A.Y., Arana-Daniel, N., Lopez-Franco, C.: RHONN identifier for unknown nonlinear discrete-time delay systems. In: 2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC) (2015). https://doi.org/10.1109/ROPEC.2015.7395583
18. Slotine, J.J.E., Li, W.: Applied Nonlinear Control. Prentice Hall (1991)
19. Söderström, T., Stoica, P.: System Identification. Prentice Hall, Upper Saddle River, NJ (1989)