Bayesian Estimation with Artificial Neural Network

Sehyun Yun and Renato Zanetti
The University of Texas at Austin, Austin, Texas
Email: shyun@utexas.edu, renato@utexas.edu

***Abstract*—A nonlinear filter based on an artificial neural network (ANN) is proposed to accurately estimate the state of a nonlinear dynamic system. The ANN is trained to learn the nonlinear mapping between the inputs and outputs of training data. The proposed filter is computationally efficient for online applications because estimation error can be directly estimated once the ANN is trained offline. The unscented transformation (UT) is employed in this filter to approximate the first two moments of the estimate. Under the scenarios considered in this paper, it is shown through numerical simulation that the proposed filter leads to better performance than the extended Kalman filter (EKF), unscented Kalman filter (UKF), and a state-of-the-art nonlinear filter.**

I. Introduction

State estimation refers to the process to estimate and predict the state of a dynamic system from noisy sensor data. Traditionally, model-based estimation techniques are employed which require a mathematical model of the dynamics and measurements. The objective of optimal state estimation is to minimize the error between the true and estimated state of a system. One of the most used criteria for optimal estimation is based on a minimum mean squared error (MMSE) performance index and the optimal solution of the MMSE estimator is the conditional mean [1]. For linear systems with additive Gaussian noise, the Kalman filter is the optimal MMSE estimator [2]. However, it is not typically possible to obtain the closed-form solution of the MMSE estimator for practical nonlinear estimation problems. Often, linear estimators for nonlinear systems such as extended Kalman filter (EKF) [3] and unscented Kalman filter (UKF) [4] are used for many practical applications due to their simplicity and efficiency.

Linear estimators for nonlinear systems are based on the linear MMSE (LMMSE) framework and can generally be grouped into two types according to the approximations used to estimate the first two moments (i.e., mean and covariance matrix) required for the LMMSE estimate. As extensions of the Kalman filter for nonlinear systems, the EKF and the Gaussian second-order filter (GSOF) [5] are defined with a local approximation (Taylor series) of the nonlinear dynamic and measurement functions. Another class of linear estimators for nonlinear systems rely on a set of deterministic regression points for statistical linearization of the nonlinear functions to obtain the first two moments. These estimators include the UKF, the quadrature Kalman filter (QKF) [6], and the cubature Kalman filter (CKF) [7]. These linear estimators for nonlinear systems are easy by the assumed form of the estimator function, i.e., the LMMSE estimator is an affine function of the observation.

Several methods have been developed to account for the nonlinearity of a system within the LMMSE framework [8]–[10]. Lan and Li. [8] demonstrated that the nonlinear conversion function of the measurement in a system can be included in an estimator to improve its estimation performance. The nonlinear conversion function in the estimator is designed to augment the linear measurement space of the system. Servadio and Zanetti. [9] proposed the Gaussian high-order polynomial update filter (HOPUFG) which approximates the non-Gaussian probability density function (PDF) using polynomial functions of Gaussian random vectors. Morimoto and Doya. [10] applied a machine learning (ML) algorithm to a nonlinear state estimation problem. The so-called reinforcement learning state estimator (RLSE) uses reinforcement learning (RL) to find an appropriate gain of the LMMSE estimator.

Besides, data-driven state estimation techniques based on variations of recurrent neural networks (RNNs) have recently been developed [11]–[13]. Krishnan et al. [11] proposed the deep Kalman filter (DKF) which replaces linear transformations on dynamic and measurement models into nonlinear ones parameterized by a stochastic RNN. Karl et al. [12] focused on improving the performance of the DKF by modeling state transitions with parametric models, yielding deep variational Bayes filters (DVBFs). Rangapuram et al. [13] presented a prediction model by parametrizing a linear state space model with a jointly-learned RNN. This study, however, focuses on a hybrid model-/data-driven estimation method to estimate state uncertainty.

Unlike LMMSE estimators that only require the mean and covariance matrix of the distribution (typically assuming it to be Gaussian), MMSE estimators typically approximate the complete distributions functions. These algorithms include the Gaussian sum filter (GSF) [14] and particle filter (PF) [15]. At the cost of more computations, these nonlinear estimators provide high accuracy for highly nonlinear systems. LMMSE algorithms only use mean and covariance matrix and assume a linear measurement update while MMSE algorithms approximate the entire distribution and do not make any specific assumption regarding the form of the estimator, they calculate the mean of the approximated posterior distribution.

Approximating distributions as Gaussian provides a conservative approximation since the Gaussian distribution has the maximum entropy (i.e. most uncertainty) out of any distribution with a given covariance matrix. Hence if the mean and the covariance matrix can be accurately approximated, using a Gaussian distribution with those characteristics provides a conservative approximation of the actual distribution.

The Gaussian particle filter [16] leverages this property to propose a nonlinear filter (a PF) that after the measurement incorporation resamples the particles by making the Gaussian approximation, hence loosing the knowledge of the actual shape of the distribution but gaining in particle diversity.

In this study, we propose a novel nonlinear estimator that approximates the posterior distribution as Gaussian. But rather than calculating the nonlinear estimator function directly from models via some approximation, e.g. sequential importance sampling like the PF, we use supervised learning to learn this nonlinear estimation function. An artificial neural network (ANN) is used to capture the nonlinear relationship between the inputs (i.e., prior state, covariance, and measurement innovation) and the corresponding outputs (i.e., posterior state). The ANN is trained offline and when deployed it provides an estimate efficiently calculated without having to carry the mathematical models and without extensive computations. We propose to use the unscented transformation (UT) to approximate the first two moments (i.e., mean and covariance matrix) of the estimate [4].

The remainder of the paper is organized as follows. Sections II and III describe the LMMSE and the ANN algorithm, respectively. Then, a new nonlinear algorithm with ANN is proposed in Section IV. Section V presents simulation results of the proposed algorithm, followed by some concluding remarks on the new methodology and the results.

II. Linear Minimum Mean Squared Error Estimation

The LMMSE estimator minimizes the mean squared error (MSE) among all linear estimators [17], that is, its estimate $\hat{\boldsymbol{x}}$ is a simple linear function of the measurement $\boldsymbol{y}$.

$$\hat{\boldsymbol{x}} = \boldsymbol{a} + K\boldsymbol{y} \tag{1}$$

where $\boldsymbol{a}$ and K are a vector and matrix, respectively. The theoretically optimal value of the LMMSE estimator is expressed as follow [17]:

$$\hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} + P_{xy}P_{yy}^{-1}(\boldsymbol{y} - \bar{\boldsymbol{y}}) \tag{2}$$

$$P_{ee} = P_{xx} - P_{xy}P_{yy}^{-1}P_{xy}^{\mathrm{T}} \tag{3}$$

where P_{ee} is the estimate error covariance matrix, and $\bar{\boldsymbol{x}}$, $\bar{\boldsymbol{y}}$, P_{xy}, P_{yy}, and P_{xx} are the first two moments of the state and measurement: $\mathbb{E}[x] = \bar{\boldsymbol{x}}$, $\mathbb{E}[y] = \bar{\boldsymbol{y}}$, $Cov(x,y) = P_{xy}$, $Cov(y) = P_{yy}$, and $Cov(x) = P_{xx}$. For linear measurements $\boldsymbol{y}_k = C_k\boldsymbol{x}_k + \boldsymbol{\eta}_k$ where the measurement noise $\boldsymbol{\eta}_k$ is a zero mean white sequence with covariance matrix R_k, the update of the sequential LMMSE estimator is the Kalman filter, which is given by:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + K_k(\boldsymbol{y}_k - C_k\hat{\boldsymbol{x}}_{k|k-1}) \tag{4}$$

$$P_{k|k}^{xx} = P_{k|k-1}^{xx} - K_k\left(C_kP_{k|k-1}^{xx}C_k^{\mathrm{T}} + R_k\right)K_k^{\mathrm{T}} \tag{5}$$

$$K_k = P_{k|k-1}^{xx}C_k^{\mathrm{T}}\left(C_kP_{k|k-1}^{xx}C_k^{\mathrm{T}} + R_k\right)^{-1} \tag{6}$$

where $P_{k|k}^{xx}$ is the posterior state estimation error covariance and K_k is the Kalman gain at time step k. Generally the values of $\bar{\boldsymbol{y}}$, P_{xy}, P_{yy} cannot be calculated exactly in the presence of nonlinear measurements, and various approximations of the optimal LMMSE estimator for nonlinear systems can be obtained by approximating these quantities in different ways. The UKF makes one such approximation and it is briefly reviewed in this section.

A. The Unscented Kalman Filter

Consider a general discrete-time nonlinear system given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}_k) + \boldsymbol{\nu}_k \tag{7}$$

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{\eta}_k \tag{8}$$

where k is the time step, $\boldsymbol{x}_k$ is an $n_{\boldsymbol{x}} \times 1$ vector, $\boldsymbol{y}_k$ is an $n_{\boldsymbol{y}} \times 1$ vector, $\boldsymbol{f}_k$ and $\boldsymbol{h}_k$ are some nonlinear functions, and the process noise $\boldsymbol{\nu}_k$ and measurement noise $\boldsymbol{\eta}_k$ are zero mean white sequence with covariance matrix Q_k and R_k, respectively, independent from all other random variables.

The UKF using UT approximates nonlinear functions with statistical linearization using a set of deterministic sigma points [18]. The most common schemes to effectively calculate sigma points are to assume that all distributions are Gaussian [19]. Suppose we have an augmented state and covariance matrix as follows:

$$\hat{\boldsymbol{x}}_{k|k}^{a} = \begin{bmatrix} \hat{\boldsymbol{x}}_{k|k} \\ 0 \\ 0 \end{bmatrix} \tag{9}$$

$$P_{k|k}^{a} = \begin{bmatrix} P_{k|k}^{xx} & 0 & 0 \\ 0 & Q_k & 0 \\ 0 & 0 & R_{k+1} \end{bmatrix} \tag{10}$$

Given an $(2n_{\boldsymbol{x}} + n_{\boldsymbol{y}}) \times (2n_{\boldsymbol{x}} + n_{\boldsymbol{y}})$ augmented covariance matrix $P_{k|k}^{a}$, we generate $2n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+1$ sigma points as follows:

$$\mathcal{X}_{k|k}^{a} = \left[\hat{\boldsymbol{x}}_{k|k}^{a},\ \hat{\boldsymbol{x}}_{k|k}^{a} \pm \sqrt{(L+\lambda)\,P_{k|k}^{a}}\right] \tag{11}$$

where L is the dimension of the augmented state, $2n_{\boldsymbol{x}} + n_{\boldsymbol{y}}$,

$$\mathcal{X}_{k|k}^{a} = \begin{bmatrix} \mathcal{X}_{k|k}^{x} \\ \mathcal{X}_{k|k}^{\nu} \\ \mathcal{X}_{k|k}^{\eta} \end{bmatrix} \tag{12}$$

and $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter [20]. The parameter α tunes the spread of the sigma points around $\hat{\boldsymbol{x}}_{k|k}$ and it is usually set to a small positive number ($10^{-4} \leq \alpha \leq 1$). κ is a secondary scaling parameter which is usually set to $3 - n_{\boldsymbol{x}}$. Based on the above sigma points, the corresponding weights are calculated as follows:

$$W_0^m = \frac{\lambda}{L+\lambda}, \quad W_0^c = \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \tag{13}$$

$$W_j^m = W_j^c = \frac{0.5}{L+\lambda}, \quad \text{for } j = 1, \cdots, 2L \tag{14}$$

where the parameter β is used to include prior knowledge of the distribution of $\boldsymbol{x}$. With the above sigma points and weights, the time update equations are expressed as follows:

$$\mathcal{X}^x_{j,k+1|k} = \boldsymbol{f}_k(\mathcal{X}^x_{j,k|k}) + \mathcal{X}^\nu_{j,k|k}, \quad j = 0, \cdots, 2L \tag{15}$$

$$\hat{\boldsymbol{x}}_{k+1|k} = \sum_{j=0}^{2L} W^m_j \mathcal{X}^x_{j,k+1|k} \tag{16}$$

$$P^{xx}_{k+1|k} = \sum_{j=0}^{2L} W^c_j \left[\mathcal{X}^x_{j,k+1|k} - \hat{\boldsymbol{x}}_{k+1|k}\right] \left[\mathcal{X}^x_{j,k+1|k} - \hat{\boldsymbol{x}}_{k+1|k}\right]^{\mathrm{T}} \tag{17}$$

The measurement update equations are then expressed as follows:

$$\mathcal{Y}_{j,k+1|k} = \boldsymbol{h}_{k+1}(\mathcal{X}^x_{j,k+1|k}) + \mathcal{X}^\eta_{j,k|k}, \quad j = 0, \cdots, 2L \tag{18}$$

$$\hat{\boldsymbol{y}}_{k+1} = \sum_{j=0}^{2L} W^m_j \mathcal{Y}_{j,k+1|k} \tag{19}$$

$$P^{yy}_{k+1|k} = \sum_{j=0}^{2L} W^c_j \left[\mathcal{Y}_{j,k+1|k} - \hat{\boldsymbol{y}}_{k+1}\right] \left[\mathcal{Y}_{j,k+1|k} - \hat{\boldsymbol{y}}_{k+1}\right]^{\mathrm{T}} \tag{20}$$

$$P^{xy}_{k+1|k} = \sum_{j=0}^{2L} W^c_j \left[\mathcal{X}_{j,k+1|k} - \hat{\boldsymbol{x}}_{k+1|k}\right] \left[\mathcal{Y}_{j,k+1|k} - \hat{\boldsymbol{y}}_{k+1}\right]^{\mathrm{T}} \tag{21}$$

$$\hat{\boldsymbol{x}}_{k+1|k+1} = \hat{\boldsymbol{x}}_{k+1|k} + P^{xy}_{k+1|k} \left(P^{yy}_{k+1|k}\right)^{-1} \left(\boldsymbol{y}_{k+1} - \hat{\boldsymbol{y}}_{k+1}\right) \tag{22}$$

$$P^{xx}_{k+1|k+1} = P^{xx}_{k+1|k} - P^{xy}_{k+1|k} \left(P^{yy}_{k+1|k}\right)^{-1} \left(P^{xy}_{k+1|k}\right)^{\mathrm{T}} \tag{23}$$

where $P^{yy}_{k+1|k}$ is the measurement innovation covariance, $P^{xy}_{k+1|k}$ is the cross covariance.

III. Artificial Neural Network

The ANN employed (also known as feedforward network and sometimes as back-propagation neural network) is a function approximation algorithm that aims to replicate the systematic relationship between the model input and the corresponding output by training the network based on a great number of data [21]. It is consisted of the information-processing units called artificial neurons and the neuron model is shown in Figure 1. The artificial neuron in the figure has N inputs, $x_1, x_2, \cdots, x_N$, and these inputs are fully connected to the neurons with different weights representing the strength of the connection between input and output data. The activation function f is a function that helps the network learn complex patterns in the data; it can add nonlinearity into a neuron. Popular types of activation functions include sigmoid, hyperbolic tangent, and rectified linear unit (ReLU). The bias b is used to adjust the output along with the weighted sum of the inputs to the artificial neuron.

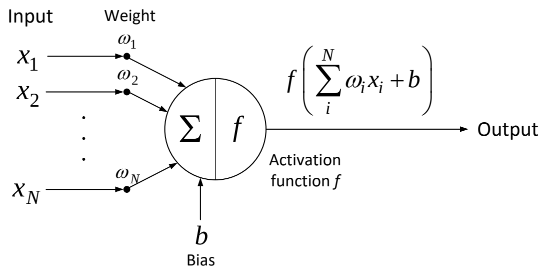


Fig. 1. Artificial neuron

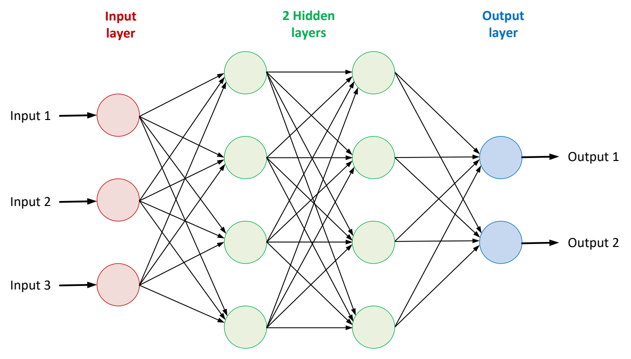


Fig. 2. Architecture of ANN

ANN is made up of neurons and layers by connecting the outputs of some neurons as input to the others and the architecture of ANN is shown in Figure 2. This figure describes a feedforward network with one input/output layer and two hidden layers. To find the optimal weights and biases in the ANN model, we need to first define a loss function that measures how well the network predicts outputs. The loss function is typically defined as MSE.

$$L = \frac{1}{N_T} \sum_{i=1}^{N_T} (\tau_i - o_i)^2 \tag{24}$$

where N_T is the number of training data, $\boldsymbol{\tau}$ is the vector of true labels and o is a vector of network predictions. To minimize the loss, the gradient descent algorithm is often used; it calculates the gradient of the loss function, then shifts the weights and biases in the opposite direction of the gradient to reduce the value of the performance index.

IV. Artificial Neural Network Based Filter

We propose an ANN-based filter (ANNF) to learn the functional relationship between estimator input (notably the measurement) and the estimator output (the estimated state). This function is not known and hence we cannot generate training data from it. However, we know the opposite relationship going from state to measurement, and we can use that known mathematical model to generate training data.

The MMSE estimate $\hat{\boldsymbol{x}} = \mathbb{E}\{\boldsymbol{x}|\boldsymbol{y}\}$ is a function of the measurement outcome $\boldsymbol{y}$ as well as the conditional distribution of the state given the measurement. We let the ANN learn this function, specifically:

- The distribution of $\boldsymbol{x}$ is approximated by a Gaussian with mean $\bar{\boldsymbol{x}}$ and covariance P_{xx}. The non-redundant elements of the square root of the covariance are provided as inputs to the ANN.
- The deviation between the measurement and its predicted value rather than the measurement alone are provided as inputs to the ANN.
- The output of the AANF is trained to predict the *a priori* error, $\boldsymbol{x}^{\text{True}} - \bar{\boldsymbol{x}}$ rather than the true state itself.

The ANNF estimate is given by:

$$\hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} + O^{\text{ANN}} \tag{25}$$

where O^{ANN} is the ANN output generated from the entered input.

A. *Offline Learning*

1) Static System: In the offline learning phase, given a large number of input/output training data, the ANN is trained to imitate the nonlinear relationship between the input (learning) and output (target) variables. In Eq. (25), the true *a priori* error, $\boldsymbol{x}^{True} - \bar{\boldsymbol{x}}$, is expected to be reduced to improve the accuracy of the estimator and hence chosen as the output variable. The state covariance P_{xx}, and the measurement innovation are chosen as the input variables in a static system. The larger the input covariance the more the filter will rely on the measurement innovation.

2) Dynamic System: The input variables of a static system cannot represent the characteristics of a dynamic system. Therefore, to include the state dynamics, the estimated states at the previous and current time are added as input variables. Figure 3 shows the concept of how to generate the input and output training data in a dynamic system. In the figure, red triangles represent the true states, green circles represent the state estimates, each big ellipse/circle (dashed-line) is the 3 sigma bound (i.e., 99.7% confidence interval) at each time step, t_0 is the initial time, and t_f is the final time of a simulation. First, at the time step t_0, each possible true state can be sampled from the given initial mean $\bar{\boldsymbol{x}}_0$ and covariance matrix P_0. Two possible true states ($\boldsymbol{x}_{0,0}^{True}$ and $\boldsymbol{x}_{0,1}^{True}$), for instance, are sampled as shown in Figure 3. These two true states then propagate according to Eq. (7), and each state estimate ($\bar{\boldsymbol{x}}_{1,0}$ or $\bar{\boldsymbol{x}}_{1,1}$) is sampled using the corresponding propagated true state ($\boldsymbol{x}_{1,0}^{True}$ or $\boldsymbol{x}_{1,1}^{True}$) and possible covariance matrix ($P_{1,0}$ or $P_{1,1}$). Various values are used for the covariance matrices to create a variety of data and the range of the covariance values for a dynamic system can be selected through try and error method. Lastly, the above process is repeated up to the final simulation time t_f.

We can now generate training input and output data using the above samples. The input variables at the time step t_{k+1} are the state $\bar{\boldsymbol{x}}_{t_k,i_N}$, the propagated state $\boldsymbol{f}_k(\bar{\boldsymbol{x}}_{t_k,i_N})$, the covariance matrix P_{t_k,i_N}, and the innovation $\boldsymbol{y}_{k+1,i_N} - \boldsymbol{h}_{k+1}\left(\boldsymbol{f}_k(\bar{\boldsymbol{x}}_{t_k,i_N})\right)$ where $i_N = 1, \cdots, N_T$ and N_T is the number of training dataset at each time step. The output variable at the time step t_{k+1} is the estimation error, $\boldsymbol{x}_{t_k+1,i_N}^{\text{True}} - \boldsymbol{f}_k(\bar{\boldsymbol{x}}_{t_k,i_N})$. For example, Figure 4 shows the input and output variables for a dynamic system at the time step t_{k+1}. Finally, the total training data set for a dynamic system consists of the generated data at the time series $t_1, t_2, \cdots, t_f$.

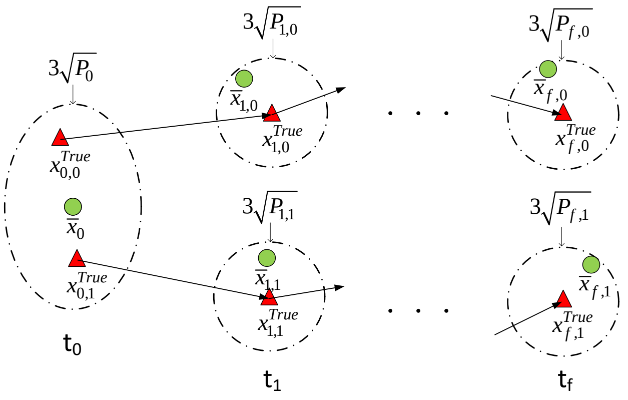
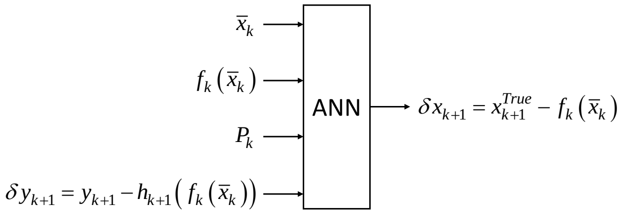


Fig. 3. The concept of generating training data in a dynamic system



$\bar{x}_k$
$f_k(\bar{x}_k)$
P_k
$\delta y_{k+1} = y_{k+1} - h_{k+1}(f_k(\bar{x}_k))$
ANN
$\delta x_{k+1} = x_{k+1}^{True} - f_k(\bar{x}_k)$

Fig. 4. Input and output variables in a dynamic system

In this study, two hidden layers with hyperbolic tangent activation function is used and Xavier initialization [22] is applied for the weights and biases initialization in the ANN model. Xavier initialization is designed to initialize the biases to zero, and the weights such that the variance of the inputs and outputs of the neurons stay the same across every layer, which mitigates the vanishing and exploding gradients problems [22]. Moreover, the Cholesky factor of the covariance matrix is used as an input variable instead of the covariance matrix itself.

B. *Online State Estimation*

The UT is used to calculate the first two moments of the estimate undergoing the nonlinear ANN transformation. A recursive algorithm is used and the distribution $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{k-1})$ at the prior time is assumed a Gaussian distribution. Given the estimate and covariance matrix, the sigma points and weights are calculated by Eq. (11) – Eq. (14). The sigma points are then propagated through Eq. (15). We now have input data for the trained ANN algorithm: $\mathcal{X}_{k|k}^{x}$, $\mathcal{X}_{k+1|k}^{x}$, $P_{k|k}^{xx}$, and $\mathcal{Y}_{k+1|k} - \hat{\boldsymbol{y}}_{k+1}\mathbb{1}_{1\times(2L+1)}$, where $\mathbb{1}_{1\times(2L+1)}$ is an indicator function. In the same way as in the offline learning phase, the Cholesky factor of the covariance matrix is used instead of the covariance matrix itself. Using the input data, the ANN provides $2L+1$ outputs, O_{k+1}^{ANN}, and the sigma points are updated with the outputs as follows:

$$\hat{\mathcal{X}}_{j,k+1}^{x} = \mathcal{X}_{j,k+1|k}^{x} + O_{j,k+1}^{\text{ANN}}, \quad j = 0, \cdots, 2L \tag{26}$$

Finally, the estimate and covariance matrix are calculated as follow:

$$\hat{\boldsymbol{x}}_{k+1|k+1} = \sum_{j=0}^{2L} W_j^m \hat{\mathcal{X}}_{j,k+1}^x \tag{27}$$

$$P_{k+1|k+1}^{xx} = \sum_{j=0}^{2L} W_j^c \left[\hat{\mathcal{X}}_{j,k+1}^x - \hat{\boldsymbol{x}}_{k+1|k}\right]\left[\hat{\mathcal{X}}_{j,k+1}^x - \hat{\boldsymbol{x}}_{k+1|k}\right]^{\mathrm{T}} \tag{28}$$

They are then used as a starting point for the next iteration.

V. Numerical Results

To evaluate the proposed algorithm in this paper, two different examples are considered: a simple scalar problem (used in Ref. [9]) and a Lorenz96 system (used in Refs. [9], [23], [24].

A. Scalar Problem

Consider the following simple scalar problem [9]. A univariate normal random vector x is distributed as

$$x \sim n(x;\ \mu,\ P) = n(x;\ 0,\ 0.1) \tag{29}$$

and a measurement is available and given by

$$y = \arctan(x) + \eta \tag{30}$$

where

$$\eta \sim n(\eta;\ 0,\ R) = n(\eta;\ 0,\ 0.0001) \tag{31}$$

The solutions of the EKF, UKF, optimal LMMSE estimator, proposed filter (ANNF), and the true joint distribution of x and y are shown in Figure 5 where 10^5 samples are used. The optimal solution of the MMSE estimator is the conditional mean which is the centerline of the joint distribution of x and y (blue points). Both the EKF and UKF are based on the LMMSE framework and use different approximation methods to calculate the estimates. Their solutions are therefore linear functions with different slopes as shown in Figure 5 where each slope of the lines is their Kalman gain. To calculate the Kalman gain, the local linearization of the nonlinear measurement function around the prior mean is used in the EKF, whereas the statistical linearization through a set of sigma points is used in the UKF. The LMMSE estimate is obtained by Eq. (2) and the first two moments of the state and measurement are calculated using the 10^5 samples of the true joint distribution of x and y. In the figure, it is shown that the solution of the UKF is closer to the LMMSE estimate than that of the EKF. On the other hand, the ANNF provides a curve solution since the ANN model can capture the nonlinear relationship between the inputs (the cholesky factor of the covariance matrix and the innovation) and the output (the estimation error). In this example, the ANNF uses the ANN model with 1 hidden layer with 3 neurons. The figure shows the ANNF provides the closest solution to the optimal MMSE estimator among the others.

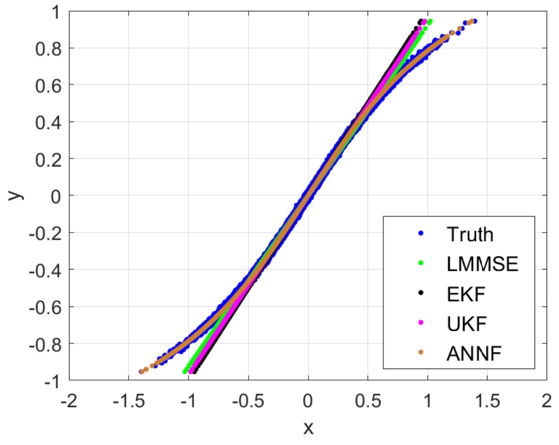


Fig. 5. Representation of the true joint distribution of the state and measurement and of the estimates from different estimators

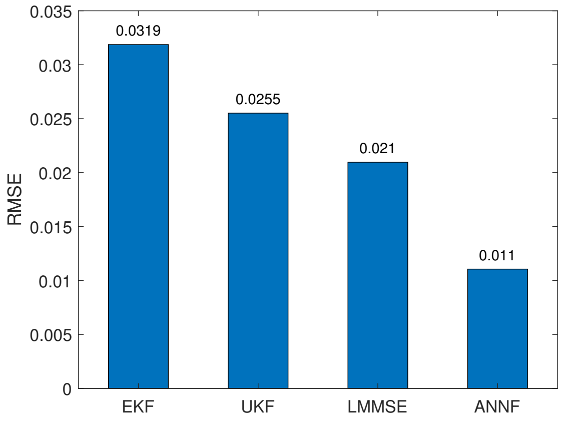


Fig. 6. RMSE for 10^5 samples

Moreover, the estimators are compared based on the root mean square error (RMSE) given by

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N_{\text{smaples}}} (x_i - \hat{x}_i)^2}{N_{\text{samples}}}} \tag{32}$$

Figure 6 shows the RMSE results of the estimators. The RMSEs of the EKF and UKF are higher than that of the LMMSE estimator and the best performance is obtained with the ANNF. From the results, it is shown that the ANNF is the best approximation of the MMSE for this example among the other estimators.

B. Lorenz96 System

In this example, a Lorenz96 system [9], [23], [24] is used to evaluate the performance of the proposed filter. The system models of the Lorenz96 system are expressed as follows:

$$\dot{x}_i(t) = x_{i-1}(t)\ (x_{i+1}(t) - x_{i-2}(t)) - x_i(t) + F + \nu_i(t) \tag{33}$$

$$\boldsymbol{y}_k = H\ \boldsymbol{X}(t_k) + \boldsymbol{\eta}_k, \quad H_{i,j} = \begin{cases} 1, & j = 2i - 1 \\ 0, & \text{otherwise} \end{cases}, \tag{34}$$
$$\text{for } i = 1, 2,\ j = 1, 2, 3, 4$$

where $x_i(t)$, $i = 1, 2, 3, 4$ are the elements of the 4th-dimensional vector $\boldsymbol{X}(t)$. In the dynamics equation, the following conventions for the state variables are used: $x_{-1} = x_{N-1}$, $x_0 = x_N$, and $x_1 = x_{N+1}$. The term F in the dynamics equation is a constant external force and it is set to 8 to cause a chaotic behavior in the system. The dynamics is propagated for 40 s at 2 Hz with a Runge-Kutta 7-8 integrator. The process noise remains constant over each 0.5 s interval with zero correlation between the intervals. The discrete measurements are available at 2 Hz. The measurements are linear and include only the elements with odd indices of the state vector. The process noise and measurement noise are assumed to be uncorrelated, white, zero mean, and covariance matrices are given by $Q = 10^{-6}$ and $R = 0.25\boldsymbol{I}_{2\times 2}$, respectively. The initial state of the system is a multivariate Gaussian distribution with $\boldsymbol{\mu}_0 = [F, F, F + 0.01, F]^{\mathrm{T}}$ and $P_0 = 10^{-6}\boldsymbol{I}_{4\times 4}$.

In this example, a Monte Carlo analysis is performed with 100 simulations and the performance results of the each state of the ANNF is shown in Figure 7. For training the ANN in the ANNF, the input and output data of the Lorenz96 system are generated by changing the possible true state and prior covariance matrix at each time step as shown in Figure 3. Diagonal covariance matrix is only used to generate the training data and the each diagonal term of the covariance matrix is a log uniform random variable with the range of 10^{-6} up to 10^{-1}. The number of training samples at each time step is 5×10^5 and the total number of training dataset is 4×10^7. The ANNF uses the following tuning parameters: $\alpha = 10^{-3}$, $\beta = 1$, and $\kappa = 0$, for its sigma points spread. In Figure 7, the gray lines are the estimation errors of 100 Monte Carlo simulations. The green line represents the filter's predicted error standard deviation of each state (3σ values) and the blue line represents the sample error standard deviation of each state (3σ values). The consistency of the ANNF is decided by the close values of the predicted (green line) and sample (blue line) standard deviation and the result shows the estimator is consistent. Finally, the black line represents the sample mean of the each estimation error and the ANNF is slightly biased over the simulation time. The reason is that the ANNF is based on Gaussian assumption since it uses UT to calculate the first two moments of the estimate.

In this example, the EKF and UKF diverge, which mean they fail to estimate the state of the system. Therefore, instead of linear estimators for nonlinear systems, the estimation performance of the HOPUFG-2-2 is compared to that of the ANNF. The HOPUFG-2-2 is a quadratic estimator with nonlinear functions approximated by the second-order Taylor series expansion [9]. Their estimation performances are compared based on the root sum squared (RSS) of the predicted and sample covariance matrix. For example, the RSS of the predicted covariance matrix is calculated as the square root of its trace: $\bar{\sigma} = \sqrt{\mathrm{tr}(P^{xx})}$, and of the sample covariance

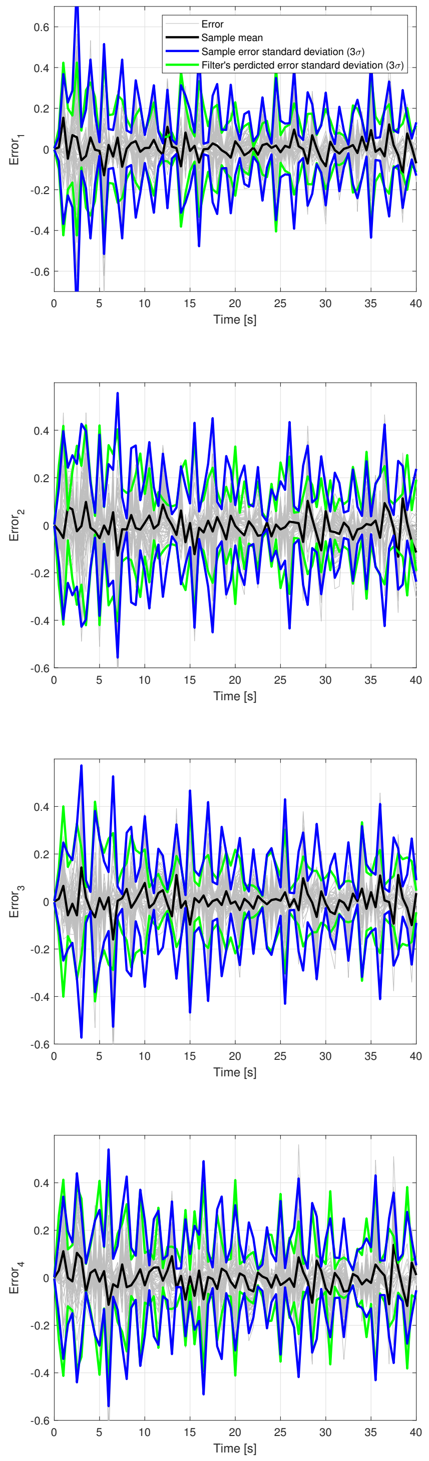


Fig. 7. 100 runs Monte Carlo performance test for the ANNF

matrix is derived from the Monte Carlo analysis. Figure 8 shows the comparison of the predicted and sample covariance matrix of the HOPUFG-2-2 and the ANNF. In the figure, the solid and dashed lines are respectively the RSS of the predicted and sample covariance matrix, and the red lines (ANNF) lie below the blue lines (HOPUFG-2-2), which means the ANNF

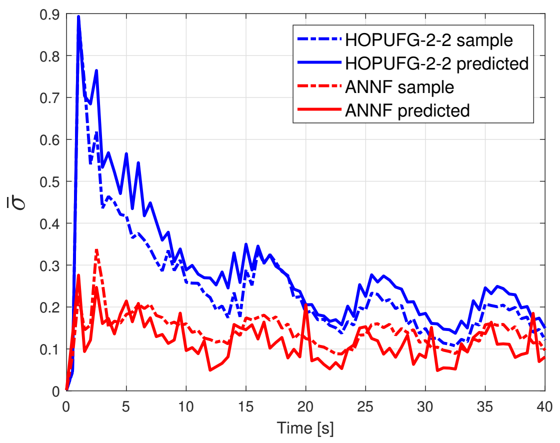


Fig. 8. Covariance comparison: HOPUFG-2-2 vs ANNF

provides the better performance than the HOPUFG-2-2 in terms of accuracy.

VI. Conclusion

In this paper, a novel estimator with the artificial neural network is proposed to account for the nonlinearity of a system. The neural network is used to capture the nonlinear relationship between the inputs and the corresponding outputs. The true *a priori* error is selected for the output variable based on the linear minimum mean squared error estimator. The *a priori* covariance matrix and the innovation are chosen as the input variables for a static system since the statics of the prior and the measurement are strongly related to the output variable. The states at the previous and current time are added to the input variables for a dynamic system to include the statistical properties of dynamics. Moreover, the estimator employs unscented transformation to approximate the first two moments of the estimate. Two numerical examples show that the proposed filter provides better performance than the linear estimators for nonlinear systems such as the extended Kalman filter and unscented Kalman filter, and a state-of-the-art nonlinear filter named HOPUFG in terms of accuracy.

References

[1] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley, 2001.

[2] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, March 1960, doi:10.1115/1.3662552.

[3] A. Gelb, Ed., *Applied Optimal Estimation*. Cambridge, MA: The MIT press, 1974.

[4] S. J. Julier and J. K. Uhlmann, “New extension of the kalman filter to nonlinear systems,” *SPIE Proceedings*, vol. 3068, 1997, doi:10.1117/12.280797.

[5] P. S. Maybeck, *Stochastic Models, Estimation, and Control, Volume 2*. New York, NY: Academic Press, 1982.

[6] I. Arasaratnam, S. Haykin, and R. J. Elliott, “Discrete-time nonlinear filtering algorithms using gauss-hermite quadrature,” *Proceedings of the IEEE*, vol. 95, no. 5, pp. 953–977, 2007, doi:10.1109/JPROC.2007.894705.

[7] I. Arasaratnam and S. Haykin, “Cubature kalman filters,” *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254 – 1269, June 2009, doi:10.1109/TAC.2009.2019800.

[8] J. Lan and X. R. Li, “Nonlinear estimation based on conversion-sample optimization,” *Automatica*, vol. 121, 2020, doi:10.1016/j.automatica.2020.109160.

[9] S. Servadio, R. Zanetti, and B. A. Jones, “Nonlinear filtering with a polynomial series of gaussian random variables,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 1, pp. 647–658, 2021, doi:10.1109/TAES.2020.3028487.

[10] J. Morimoto and K. Doya, “Reinforcement learning state estimator,” *Neural Computation*, vol. 19, no. 3, pp. 730–756, 2007, doi:10.1162/neco.2007.19.3.730.

[11] R. G. Krishnan, U. Shalit, and D. Sontag, “Deep kalman filters,” *arXiv preprint arXiv:1511.05121*, 2015.

[12] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, “Deep variational bayes filters: Unsupervised learning of state space models from raw data,” *In International Conference on Learning Representations (ICLR 2017)*, 2017.

[13] S. S. Rangapuram, M. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, “Deep state space models for time series forecasting,” *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS 2018)*, pp. 7796–7805, 2018.

[14] D. Alspach and H. Sorenson, “Nonlinear bayesian estimation using gaussian sum approximations,” *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, August 1972, doi:10.1109/SAP.1970.270017.

[15] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking,” *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002, doi:10.1109/78.978374.

[16] J. H. Kotecha and P. M. Djuric, “Gaussian particle filtering,” *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2592–2601, Oct 2003, doi:10.1109/TSP.2003.816758.

[17] X.-R. Li and V. P. Jilkov, “A survey of maneuvering target tracking: approximation techniques for nonlinear filtering,” vol. 5428, August 2004, pp. 537–550, doi:10.1117/12.553357.

[18] T. Lefebvre, H. Bruyninckx, and J. D. Schutter, “Comment on “A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators”,” *IEEE Transactions on Automatic Control*, vol. 47, no. 8, pp. 1406–1408, 2002, doi:10.1109/TAC.2002.800742.

[19] S. J. Julier and J. K. Uhlmann, “Unscented filtering and nonlinear estimation,” *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, March 2004, doi:10.1109/JPROC.2003.823141.

[20] E. A. Wan and R. V. D. Merwe, “The unscented kalman filter for nonlinear estimation,” *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000, doi:10.1109/ASSPCC.2000.882463.

[21] J. R. Rabuñal and J. Dorado, *Artificial Neural Networks in Real-Life Applications*. Idea Group Publishing, 2006, doi:10.4018/978-1-59140-902-1.

[22] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, pp. 249–256, 2010.

[23] S. Yun and R. Zanetti, “Sequential monte carlo filtering with gaussian mixture sampling,” *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 9, pp. 2069–2077, 2019, doi:10.2514/1.G004403.

[24] D. Raihan and S. Chakravorty, “Particle gaussian mixture filters-i,” *Automatica*, vol. 98, pp. 331–340, 2018, doi:10.1016/j.automatica.2018.07.023.