Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

# State Estimation and Tracking using Recurrent Neural Networks

Raja Gopal Reddy B[1]
Associate Professor,
Vardhaman College of Engineering
Hyderabad, India.

Bhanu Prasad C[2]
Assistant Professor
Vardhaman College of Engineering
Hyderabad, India

Padamati Harika[3]
Assistant Professor
Vardhaman College of Engineering
Hyderabad, India

Soma Keerthi Sonam[4]
Assistant Professor
Vardhaman College of Engineering
Hyderabad, India

***Abstract*—The aim of this paper is to demonstrate the suitability of recurrent neural networks (RNN) for state estimation and tracking problems that are traditionally solved using Kalman Filters (KF). This paper details a simulation study in which the performance of a basic discrete time KF is compared with that of an equivalent neural filter built using an RNN. Real time recurrent learning (RTRL) algorithm is used to train the RNN. The neural network is found to provide comparable performance to that of the KF in both the state estimation and tracking problems. The relative merits and demerits of KF vs RNN are discussed with respect to computational complexity ease of training and real time issues.**

***Keywords—Recurrent Neural Network, KF, Real time recurrent learning, Tracking, State estimation*.**

## I. INTRODUCTION

Traditionally, state estimation and tracking problems are solved using KFs (for example, see [1], [2]). Recurrent neural networks (RNN) have received much research attention because of their powerful capability to represent attractor dynamics and to preserve information through time [3]. KF is a well known recursive, linear technique that works optimally when the system equations are linear and the noises (system and measurement) are uncorrelated and white Gaussian [1]. Extended KF is formulated to deal with simple nonlinearities in the system equations. However, in general when the system and/or noise deviate from Kalman assumptions, the convergence and optimality results of KF are not guaranteed. Since neural networks of appropriate size are known to be capable of approximating a wider class of nonlinear functions [4], it is expected that neural networks, especially RNN, offer a better alternative for KF even when Kalman assumptions are violated. In this paper, we attempt to use RNNs to solve the estimation and tracking problems.

In recent years, a variety of approaches have been proposed for training the RNNs, such as the back propagation through time (BPTT) [5], real time recurrent learning algorithm (RTRL), extended Kalman filter (EKF), etc. In this paper, the RTRL algorithm is used for training the recurrent network. We present an example state estimation problem and a tracking problem to illustrate the application of RTRL trained RNN for these problems. Further, we also simulated KFs for solving the same problems to enable a comparison of the performance of RNN and KF. Relative performance is compared in terms of prediction capability and tracking error.

An introduction to KF and RNN are presented first and then the description of the problems is given next. Finally the results are presented and a discussion of relative merits and demerits of these two methods is given.

## II. DESCRIPTION OF KALMAN FILTER

The Kalman filter is a technique for estimating the unknown state of a dynamical system with additive noise. The KF has long been regarded as the optimal solution to many tracking and state prediction tasks [1]. The strength of KF algorithm is that it computes on-line. This implies that we don't have to consider all the previous data again to compute the current estimates; we only need to consider the estimates from the previous time step and the current measurement. Popular applications include, state estimation [6], navigation, guidance, radar tracking [2], sonar ranging, satellite orbit computation, etc. These applications can be summarized into various classes such as denoising, tracking and control problems. The basic KF is optimal in the mean square error sense (given certain assumptions), and is the best possible of all filters, if state and measurement inputs are Gaussian vectors and the additive noise is white and has zero mean [1].

We now begin the description of the KF. The block diagram of basic discrete time kalman filter is shown in Figure 1. We assume that the system can be modelled by the state transition equation,

$$X_{k+1} = AX_k + BU_k + W_k \quad (1)$$

where $X_k$ is the state at time $k$, $U_k$ and $W_k$ are an input control vector and additive noise from either the system or the process respectively. $B$ is the input transition matrix and $A$ is the state transition matrix.

The measurement system can be represented by a linear equation of the form,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

$$Z_k = HX_k + V_k \quad (2)$$

where $Z_k$ is the measurement prediction made at time *k*, $V_k$ is additive measurement noise and $H$ is the observation matrix. The KF uses a feed-back control for process estimation. The KF algorithm consists of two steps: a prediction step and an update step as described below.

Prediction (time-update): This predicts the state and process covariance at time *k+1* dependent on information at time *k*.

Update (measurement update): This updates the state, process covariance and Kalman gain at time *k+1* using a combination of the predicted state and the observation at time *k+1*.

Summary of Kalman Filter Equations are given below:

Step1: Predicted Sate

$$X_{k+1/k} = A_k X_{k/k}$$

Step2: Predicted Measurement

$$Z_{k+1/k} = H_k X_{k+1/k}$$

Step3: Predicted Sate Covariance

$$P_{k+1/k} = A_k P_{k/k} A_k^T + Q_k$$

Step4: Predicted Kalman Gain

$$K_{k+1} = P_{k+1/k} H_k^T \left[ H_k P_{k+1/k} H_k^T + R_k \right]^{-1}$$

Step5: Actual Measurement

$$Z_{k+1}$$

Step6: Updated (Estimated or Corrected) Sate

$$X_{k+1/k+1} = X_{k+1/k} + K_{k+1}(Z_{k+1} - Z_{k+1/k})$$

Step7: Updated (Estimated or Corrected) Sate Covariance

$$P_{k+1/k+1} = (1 - K_{k+1} H_k) P_{k+1/k}$$

The above seven equations constitute Kalman Filter Algorithm [1]. By knowing the initial conditions (State X and its Covariance P) and the noise covariance matrices (Process noise Q and Measurement noise R) the steps 1 to 4 can be executed. As soon as measurement is available steps 6 & 7 can be executed and cycle can be repeated for next measurement.

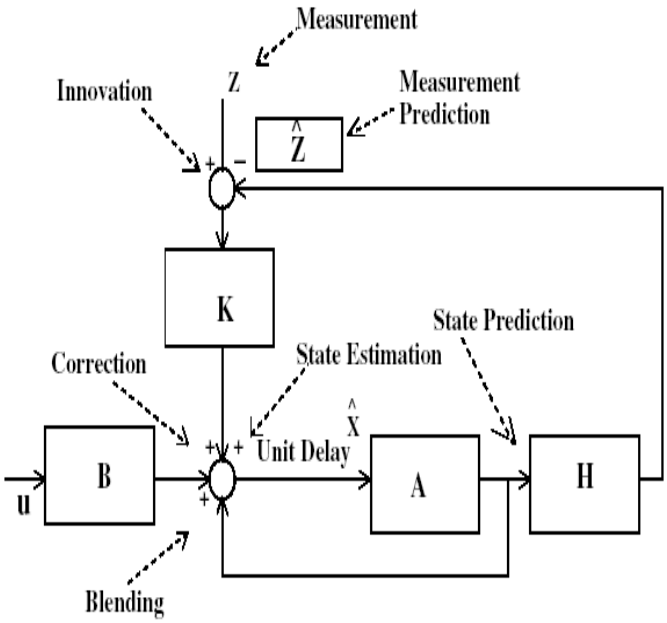


Fig.1 Block Diagram of Kalman Filter

## III. RECURRENT NEURAL NETWORK STRUCTURE

Recurrent Neural Networks (RNN) form a much wider class of neural networks, as they allow feedback connections between neurons, making them dynamical systems. The behavior of a recurrent network is dependent on all its previous inputs. Recurrent neural networks have been used in a number of identification and control scenarios [7].

The work reported in this paper differs from a previous attempt at comparison of KF with RNN in several ways [6].The feedback in our RNN is both from the output and hidden layers to the input layer unlike from only output to input layer in [6] and the learning method adopted here is RTRL as opposed to conjugate gradient method in earlier paper [6]. A simplified and more detailed representation of Recurrent Network is shown in Figure 2.

Units of the input layer $I$ and the recurrent layer $H$ and the output layer $O$ are fully connected through weights $W^{HI}$ and $W^{OH}$, respectively. The current output of the recurrent units at time *t* is fedback to the context units at time *t+1* through recurrent connections so that $C^{(t+1)} = H^{(t)}$. Hence, every recurrent unit can be viewed as an extension of input to the recurrent layer. As they hold contextual information from previous time steps, they represent the memory of the network.

Given the input pattern at time *t*, $I^{(t)} = (I_1^{(t)}, \cdots, I_i^{(t)}, \cdots, I_{|I|}^{(t)})$, and recurrent activates $H^{(t)} = (H_1^{(t)}, \cdots, H_j^{(t)}, \cdots, H_{|H|}^{(t)})$ the recurrent unit's net input $\hat{H}_i^{(t)}$ and output activity $net_i^{(t)}$ are calculated as

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

$$\hat{H}_i^{(t)} = \sum_j W_{ij}^{HI} I_j^{(t)} + \sum_j W_{ij}^{HC} H_j^{(t-1)} \quad (3)$$

$$net_i^{(t)} = f(\hat{H}_i^{(t)}) \quad (4)$$

where $|I|$, $|H|$ and $|O|$ are the number of input, hidden and output units, respectively, and $f$ is the activation function. In this work we are using the symmetrical transfer function $f(x) = \tanh x$.

*A. Learning Algorithm for RNN:*

There are several algorithms available to train recurrent networks based on streams of input-output data. Perhaps the most widely used are real-time recurrent learning (RTRL) and back propagation through time (BPTT) [5].

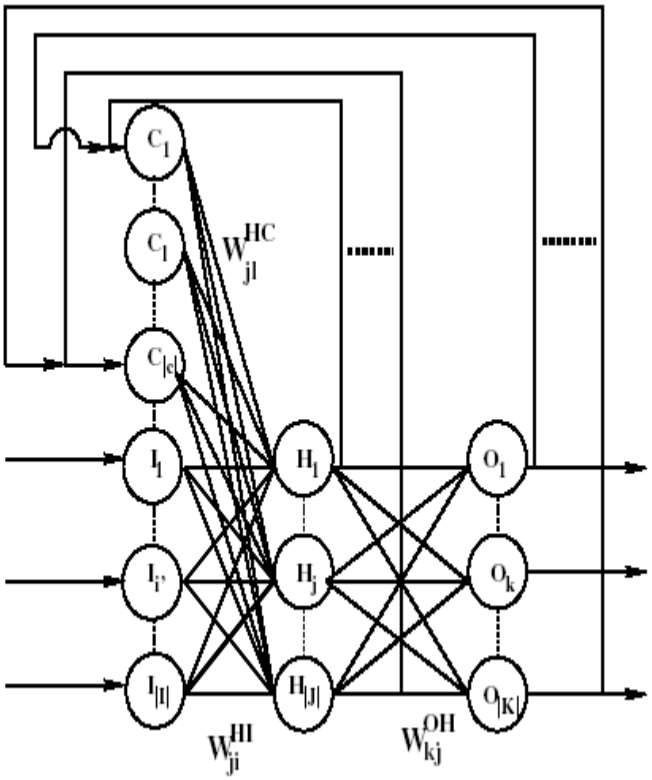


Fig. 2 RNN Architecture

In this paper RTRL algorithm is used for recurrent network training because of its good convergence property and its on-line nature [3]. Real-time recurrent learning (RTRL) has been independently derived by many authors, although the most commonly cited reference for it is Williams and Zipser [3]. This algorithm computes the derivatives of states and outputs with respect to all weights as the network processes the sequence, that is, during the forward step. The supervised learning process uses the 'Teacher Forcing' technique [3]. The advantage of using RTRL is the ease with which it may be derived and programmed for a new architecture as it does not involve any unfolding over time as in BPTT.

## IV. SIMULATION EXPERIMENTS

Simulation studies were performed for (A) state estimation problem [6] and (B) tracking problem. The configuration for the KF and RNN for each of these problems is described below.

*A. **System I:** State Estimation Problem [6]*

*Kalman Filter*: The state update equation and measurement equation are given by:

$$X(t) = 0.9X(t-1) + W(t)$$
$$Z(t) = X(t) + V(t) \quad (5)$$

and the noise sources are white Gaussian noise sequences with zero mean. The process noise covariance Q is 0.1997 and the measurement noise covariance R is 0.1. It can be observed that this is a simple scalar version of the state estimation problem where the state transition and measurement systems are scalar valued, with coefficients taken as 0.9 and 1, respectively.

*RTRL*: The architecture we took consisted of **1** input node, **3** hidden nodes and **1** output node.

*B. System II: Tracking Problem*

*Kalman Filter*: The state can be described as

$$X(t) = [x(t); \dot{x}(t)]^T .$$

The state update equation and measurement equation are given by:

$$X(t) = AX(t-1) + W(t)$$
$$Z(t) = HX(t) + V(t) \quad (6)$$

where $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, $H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,

$Q_k = \begin{bmatrix} 1/3 & 1/2 \\ 1/2 & 1 \end{bmatrix} \sigma_q^2$ and $R_k = \sigma_r^2$

where $Q_k$ and $R_k$ are the process and observation noise covariance matrices and $\sigma_q^2 = 0.01; \sigma_r^2 = 0.1$.

*RTRL*: The architecture consisted of 2 input nodes, 8 hidden nodes and 2 output nodes.
Simulation for each system is conducted as follows:

Training data sets as well as a separate test data set are produced by running the system equations. Each data set contains 100 sequences of 100 I/O pairs (Z(n),X(n)), for a total of 10000 I/O pairs. The data are scaled to the range [-1,1]. The RNN architecture comprises one hidden layer of nodes with nonlinear activation function (symmetrical transfer function tanh(x)), whereas the output nodes are linear. The state is initialized to some random value and the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

weights are also initialized to random values, uniformly distributed between -0.05 and +0.05. RTRL is used to train the net and it is trained until error criterion threshold is achieved (500 epochs). Testing is done with a separate data set and the results are reported in Figures 3 and 4. The KF parameters are computed using statistical estimation techniques over the same test data set.

## V. DISCUSSION

Figure 3 depicts the simulation of the state estimation problem. Figure 3(a) compares the resulting kalman and neural filter estimates of the true state. Figure 3(b) shows the error plots of KF and RNN. From Figure 3, it is evident that KF and RNN show comparable performance on the estimation problem. Figure 4 depicts the simulation of tracking problem. Figures 4(a) and 4(c) show the performance of KF versus RNN with respect to tracking of position and velocity of a vehicle, respectively. Figures 4(b) and 4(d) depict the corresponding error plots of position and velocity. It is evident from these figures that the difference between the desired and estimated values (tracking error) for RNN are almost zero whereas that with KF is not zero but appears to be a random value with zero-mean. The tracking error behaviour of KF is in expected lines as per the algorithm. Kalman filter is a simple, on-line, optimal algorithm but works only for linear systems with Gaussian noise. RNN is expensive in terms of space and time complexities. However, nonlinear approximation can be achieved and there is no restrictive Gaussian assumption with RNNs.

## VI. CONCLUSION

A recurrent neural network of the type described in this paper is capable of closely matching a basic KF in performance on state estimation and tracking problems. KF is less expensive computationally both in space and time complexities as compared to RNN trained via RTRL algorithm. However, the attractive feature of RNNs is that the technique works without any significant modifications for nonlinear and non-Gaussian cases also. Whereas in KF, whenever these assumptions are violated, the algorithm becomes differently structured and is more complex [4]. Thus, while the RTRL itself may not necessarily be the algorithm of choice for training recurrent networks, it may help provide a basis for both gaining a deeper understanding of existing recurrent network learning techniques and more importantly, creating more computationally attractive algorithms that allow one to optimize the trade-off between computational effort and learning speed. Other methods such as EKF are also available for training RNNs [8] and will be taken up in future, specifically for estimation and tracking problems.

## REFERENCES

[1] Brookner E., Tracking and kalman filtering made easy, John Willey and Sons, Inc, New York, USA, 1998.
[2] Pillai, S.K., Seshagiri Rao V., Balakrishnan S., "A New Method of Model Compensation in Kalman Filter – Applicaion to Launch Vehicles," in proceedings of The Fourteenth International Symposium on Space Technology and Science, Tokyo, 1984.
[3] Williams, R. J., and David Zipser,, "A learning algorithm for continually running fully recurrent neural networks", in Neural Computation vol. 1, pp. 270-280, 1989.
[4] Haykin, S., Neural Networks, Second Edition, Pearson Education, Inc, New Delhi, India, 1999.
[5] Werbos, P. J. "Backpropagation Through Time: What it does and how to do it?", in Proceedings of the ICNN, SanFrancisco, CA, USA, 1993.
[6] DeCruyenaere, J.P. and Hafez, H.M., "A comparison between Kalman filters and Recurrent Neural Networks," Proc. of the IJCNN, Baltimore, USA, IEEE Press, vol. **IV**, pp.247–251, 1992.
[7] Narendra, K. S., Parthasarathy, K., "Identification and control of dynamical systems using neural networks", in IEEE Transactions on Neural Networks, vol.1, No. l, pp.4– 27, 1990.
[8] Puskorius, G. V. and Feldkamp, L. A., "NeuroControl of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks", in IEEE Transactions on Neural Networks, vol. **5**, No. 2, pp.279–297, March 1994.

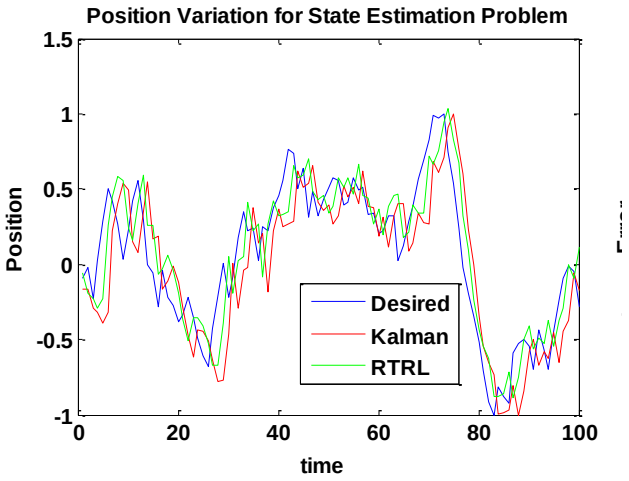


Fig. 3(a)

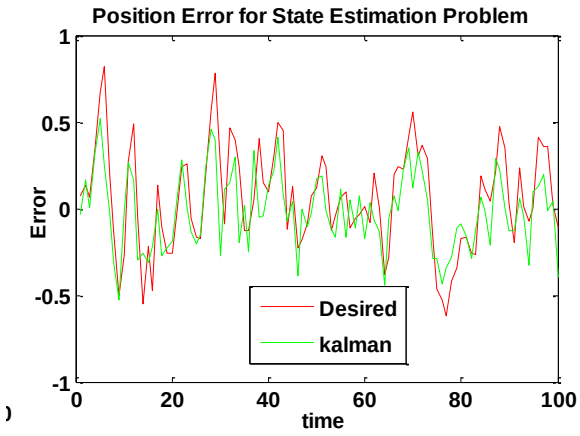


Fig. 3(b)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

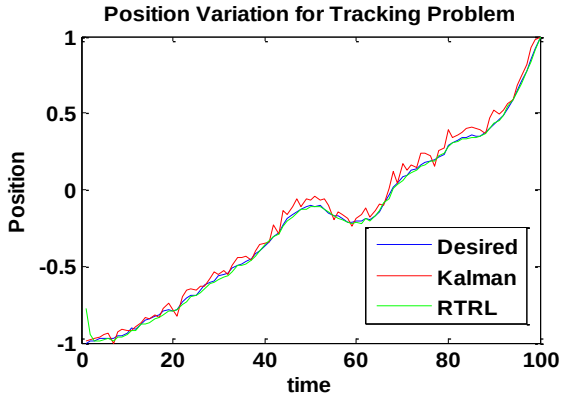


Fig. 4(a)

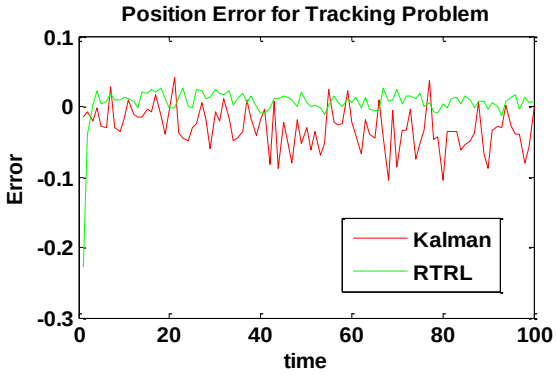


Fig. 4(b)