A Derivative-Free Kalman Filter for Parameter Estimation of Recurrent Neural Networks and Its Applications to Nonlinear Channel Equalization

Jongsoo Choi, Martin Bouchard, Tet Hin Yeap

School of Information Technology and Engineering
University of Ottawa, 800 King Edward Avenue
Ottawa, Ontario, K1N 6N5 Canada
E-mail: {jchoi,bouchard,tet}@site.uottawa.ca

Ohshin Kwon

School of Electronics and Information Engineering
Kunsan National University
San 68, Miryong-dong, Kunsan, 573-701 Korea
E-mail: kos@kunsan.ac.kr

ABSTRACT: Recurrent neural networks (RNNs) trained with gradient-based algorithms such as real-time recurrent learning or back-propagation through time have a drawback of slow convergence rate. These algorithms also need the derivative calculation through the error back-propagation process. In this paper, a derivative-free Kalman filter, so called the unscented Kalman filter (UKF), for training a fully connected RNN is presented in a state-space formulation of the system. The UKF algorithm makes the RNN have fast convergence speed and good tracking performance without the derivative computation. Through experiments of nonlinear channel equalization, the performance of the RNN with the UKF is evaluated.

1. INTRODUCTION

Recurrent neural networks (RNNs) are essentially dynamical systems where the states evolve according to certain nonlinear state equations. Due to their dynamic nature, they have been successfully applied to many problems including modeling and processing of temporal signals, such as prediction, adaptive control, system identification, and speech recognition [1]. In addition, the sequential nature of inputs and outputs in many fields makes RNNs attractive for the general task of sequence prediction, sequence generation, or sequence transduction. In digital communications, channel equalization is an example of sequential data processing, and an adaptive filter used as the equalizer in the communication receiver needs on-line learning to update its free parameters. Recently, RNNs have been successfully applied to channel equalization with a variety of network structures and learning algorithms [2],[3],[4],[5].

Many structures for RNNs have been developed, which are ranging from fully connected to partially (or locally) connected networks and ranging from single-layered to multi-layered networks. However, common problems still remained to be solved such as the analysis of the dynamical behavior of RNNs, and the capacity of learning algorithms to cope with the complexity induced by the network's dynamics. Hence intensive research works on dynamical network properties and the corresponding learning techniques have attained a good theoretical grounding for many papular algorithms [6]. We hereby focus on learning algorithms.

Typical learning algorithms for RNNs are real-time recurrent learning (RTRL) [7] for on-line learning and backpropagation through time (BPTT) [8] for off-line learning. These algorithms are totally based on the gradient method using first-order derivative information. The training problem is to update the free parameters of the network. Since weight updating affects the states at all times during the course of network state evolution, obtaining the error gradient is a complicated procedure. Moreover, due to the first-order derivative information, the RTRL and BPTT may exhibit slow convergence speed relative to learning techniques based on second-order derivative information. The extended Kalman filter (EKF) forms the basis of a second-order neural network training approach. The essence of the recursive EKF procedure is that an approximate covariance matrix that encodes second-order information about the training problem is maintained and evolved during training. However, the EKF is difficult to implement, difficult to tune, and only reliable for systems that are almost linear on the time scale of the update intervals [9]. In addition, the EKF provides first-order approximations to optimal nonlinear parameter estimation and needs the computation of derivative matrices (or Jacobians) in the linearization process of the nonlinear system.

In this paper, a derivative-free Kalman filter, called the unscented Kalman filter (UKF) [9],[10], is presented for training RNNs. The UKF can be an alternative to the EKF algorithm and may be easier to implement because it is not necessary to evaluate the Jacobians, which are needed in the EKF. We demonstrate the applicability of the UKF to RNN training. The performance of the UKF algorithm in nonlinear channel equalization applications is evaluated and compared with the RTRL.

2. FULLY CONNECTED RECURRENT NEURAL NETWORK

The formulation presented here is based on the standard fully connected RNN. The fully connected RNN consists

of q neurons with l external inputs, as shown in Fig. 1. Let the q-by-1 vector $\mathbf{x}(k)$ denotes the state of the network in the form of a nonlinear discrete-time system, the $(l+1)$-by-1 vector $\mathbf{u}(k)$ denotes the input (including bias) applied the network, and the p-by-1 vector $\mathbf{y}(k)$ denotes the output of the network. The dynamic behavior of the network, assumed to be *noise free*, is described by [11]

$$\begin{aligned} \mathbf{x}(k+1) &= \boldsymbol{\varphi}(\mathbf{W}_x(k)\mathbf{x}(k) + \mathbf{W}_u(k)\mathbf{u}(k)) \\ &= \boldsymbol{\varphi}(\mathbf{W}(k)\mathbf{z}(k)) \qquad (1) \\ \mathbf{y}(k) &= \mathbf{C}\,\mathbf{x}(k+1) \qquad (2) \end{aligned}$$

where $\mathbf{W}_x(k)$ is a q-by-q matrix, $\mathbf{W}_u(k)$ is a q-by-$(l+1)$ matrix, $\mathbf{C}$ is a p-by-q matrix; and $\boldsymbol{\varphi} : \mathbb{R}^q \rightarrow \mathbb{R}^q$ is a diagonal map. The two separate weight matrices can be merged into a whole weight matrix $\mathbf{W}(k)$ with q-by-$(q+l+1)$ dimension, that is,

$$\mathbf{W}(k) = [\mathbf{W}_x(k)\ \mathbf{W}_u(k)] \qquad (3)$$

and the $(q+l+1)$-by-1 vector $\mathbf{z}(k)$ can be defined as

$$\mathbf{z}(k) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix} \qquad (4)$$

where $\mathbf{x}(k)$ is the q-by-1 state vector and $\mathbf{u}(k)$ is $(l+1)$-by-1 input vector. The first element of $\mathbf{u}(k)$ is unity, which is the bias input, and in a corresponding way, the first column of $\mathbf{W}_u(k)$ is bias terms applied neurons. The dimensionality of the state space, namely q, is the *order* of the system. Therefore the state-space model of Fig. 1 is an l-input, q-output recurrent model of order q. Eq. (1) is the process equation of the model and Eq. (2) is the measurement equation. The process equation (Eq. (1)) in the state-space description of the network is rewritten in the following form:

$$\mathbf{x}(k+1) = \begin{bmatrix} \varphi(\mathbf{w}_1^T(k)\mathbf{z}(k)) \\ \varphi(\mathbf{w}_2^T(k)\mathbf{z}(k)) \\ \vdots \\ \varphi(\mathbf{w}_q^T(k)\mathbf{z}(k)) \end{bmatrix} \qquad (5)$$

where $\varphi(\cdot)$ is an activation function, and the $(q+l+1)$-by-1 weight vector $\mathbf{w}_i(k)$, which is connected to the ith neuron in the recurrent network, corresponds to the ith column of the transposed weight matrix $\mathbf{W}^T(k)$.

3. RTRL Algorithm for the RNN

The RTRL algorithm for training the RNN is briefly described in this section. To simplify the presentation of the RTRL, we define matrices as follows:

- The derivative matrix of the state vector $\mathbf{x}(k)$ with respect to the weight vector $\mathbf{w}_i$:

$$\boldsymbol{\Lambda}_i(k) = \frac{\partial \mathbf{x}(k)}{\partial \mathbf{w}_i(k-1)} \qquad (6)$$

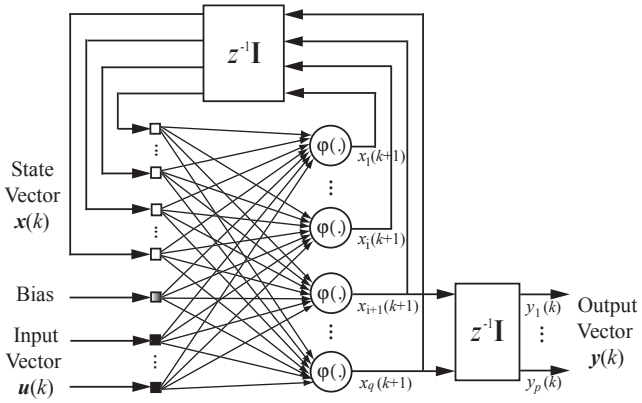


Figure 1. A layout of fully connected recurrent neural network.

- $\mathbf{Z}_i(k)$ is a q-by-$(q+l+1)$ matrix whose rows are all zero, except for the ith row that is equal to the transpose of vector $\mathbf{z}(k)$:

$$\mathbf{Z}_i(k) = \begin{bmatrix} \mathbf{0}^T \\ \mathbf{z}^T(k) \\ \mathbf{0}^T \end{bmatrix} \leftarrow i\text{th row} \quad i = 1, 2, \cdots, q. \qquad (7)$$

- $\boldsymbol{\Phi}(k)$ is a q-by-q diagonal matrix:

$$\begin{aligned} \boldsymbol{\Phi}(k+1) &= \operatorname{diag}\big[\varphi'(\mathbf{w}_1^T(k)\mathbf{z}(k)), \varphi'(\mathbf{w}_2^T(k)\mathbf{z}(k)), \\ &\qquad \cdots, \varphi'(\mathbf{w}_q^T(k)\mathbf{z}(k))\big]. \qquad (8) \end{aligned}$$

With these definitions, the following recursive equation $\boldsymbol{\Lambda}_i$ for the neuron i can be obtained by differentiating Eq. (5) with respect to $\mathbf{w}_i$ and using the chain rule of calculus:

$$\boldsymbol{\Lambda}_i(k+1) = \boldsymbol{\Phi}(k+1)\big[\mathbf{W}_x(k)\boldsymbol{\Lambda}_i(k) + \mathbf{Z}_i(k)\big] \qquad (9)$$

The objective of the learning process is to minimize a cost function obtained by the instantaneous sum of squared errors at time k, which is defined in terms of $\mathbf{e}(k)$ by

$$\mathcal{J}(k) = \frac{1}{2}\,\mathbf{e}^T(k)\,\mathbf{e}(k) \qquad (10)$$

where the p-by-1 error vector $\mathbf{e}(k)$ is defined by using the measurement equation (Eq. (2)):

$$\mathbf{e}(k) = \tilde{\mathbf{y}}(k) - \mathbf{y}(k) \qquad (11)$$

where $\tilde{\mathbf{y}}(k)$ denotes the desired output vector. The adjustment for the weight vector of the ith neuron, $\Delta\mathbf{w}_i(k)$, is:

$$\begin{aligned} \Delta\mathbf{w}_i(k) &= \eta\frac{\partial \mathcal{J}(k)}{\partial \mathbf{w}_i(k)} \\ &= \eta\mathbf{C}\boldsymbol{\Lambda}_i(k)\mathbf{e}(k), \quad i = 1, 2, \cdots, q. \qquad (12) \end{aligned}$$

4. A Derivative-Free Kalman Filter for Parameter Estimation

The Kalman filter has been a widely used filtering strategy, for over 30 years in the control and signal processing community. The EKF, which simply linearizes

all nonlinear models, could be a popular method, when the liner Kalman filter is applied to nonlinear systems. However, the EKF is difficult to implement, difficult to tune, and only reliable for systems that are almost linear on the time scale of the update intervals [9]. In parameter estimation of neural networks, the EKF provides first-order approximations to optimal nonlinear estimation through the linearization of the nonlinear system. These approximations can include large errors in the true *a posteriori* mean and covariance of the transformed Gaussian random variable, which may lead to suboptimal performance and sometimes filter divergence [12]. The UKF, first proposed by Julier and Uhlmann [10] and further extended by Wan and van der Merwe [12],[13], is an alternative to the EKF algorithm. The UKF provides third-order approximation of process and measurement errors for Gaussian distributions and at least second-order approximation for non-Gaussian distributions [14]. The UKF may have more accurate estimation features over the EKF in some applications [9],[12]. In addition, the UKF does not require the computation of Jacobians, for linearizing the process and measurement equations. This leads to a simpler implementation devoid of inverse matrix errors.

A. Unscented Transformation

Foundation to the UKF is the unscented transformation (UT). The UT is a method for calculating the statistics of a random variable which undergoes a nonlinear transformation [10]. Consider an L-by-1 random variable $\mathbf{x}$ that is nonlinearly transformed to yield a random variable $\mathbf{y}$ through a nonlinear function, $\mathbf{y} = f(\mathbf{x})$. In order to calculate the statistics of $\mathbf{y}$, a matrix $\boldsymbol{\chi}$ of $2L+1$ sigma vectors χ_i is formed as the followings:

$$\chi_i = \begin{cases} \bar{\mathbf{x}}, & i = 0 \\ \bar{\mathbf{x}} + (\sqrt{(L+\lambda)\mathbf{P_{xx}}})_i, & i = 1, \ldots, L \\ \bar{\mathbf{x}} - (\sqrt{(L+\lambda)\mathbf{P_{xx}}})_{i-L}, & i = L+1, \ldots, 2L \end{cases} \tag{13}$$

where $\bar{\mathbf{x}}$ and covariance $\mathbf{P_{xx}}$ are the mean and covariance of $\mathbf{x}$, respectively, and $\lambda = \alpha^2(L+\kappa) - L$ is a scaling factor. The constant α determines the spread of the sigma points around $\bar{\mathbf{x}}$; it is set to a small positive value, typically in the range $0.001 < \alpha < 1$. The constant κ is a secondary scaling factor that is usually set to $3-L$. The sigma points $\{\chi_i\}_{i=0}^{2L}$ are propagated through the nonlinear function

$$\mathcal{Y}_i = f(\chi_i), \qquad i = 0, \ldots, 2L. \tag{14}$$

This propagation produces a corresponding vector set that can be used to estimate the mean and covariance matrix of the nonlinear transformed vector $\mathbf{y}$. We can approximate the mean and covariance matrix of $\mathbf{y}$ using a weighted sample mean and covariance of the *a posteriori* sigma points [12],

$$\bar{\mathbf{y}} = \sum_{i=0}^{2L} W_i^m \mathcal{Y}_i \tag{15}$$

$$\mathbf{P_{yy}} = \sum_{i=0}^{2L} W_i^c (\mathcal{Y}_i - \bar{\mathbf{y}})(\mathcal{Y}_i - \bar{\mathbf{y}})^T \tag{16}$$

where the weighting factors are given by

$$\begin{aligned} W_0^m &= \frac{\lambda}{L+\lambda} \\ W_0^c &= \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) \\ W_i^m &= W_i^c = \frac{1}{2(L+\lambda)}, \quad i = 1, 2, \ldots, 2L. \end{aligned} \tag{17}$$

In the above equations, the superscripts m and c refer to the mean and covariance, respectively. β is used to take account of prior knowledge on the distribution of $\mathbf{x}$, and $\beta = 2$ is the optimal choice for Gaussian distributions.

B. On-Line Parameter Estimation by the UKF

To enable the Kalman filter for training the RNN, the network's behavior can be recast as the following nonlinear discrete-time system :

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \boldsymbol{\omega}(k) \tag{18}$$

$$\mathbf{y}(k) = \mathbf{h}\left(\mathbf{w}(k), \mathbf{z}(k)\right) + \boldsymbol{\nu}(k) \tag{19}$$

where the nonlinear function $\mathbf{h}(\cdot)$ is given by

$$\mathbf{h}\left(\mathbf{w}(k), \mathbf{z}(k)\right) = \mathbf{C}\,\boldsymbol{\varphi}\left(\mathbf{w}(k)\mathbf{z}(k)\right), \tag{20}$$

and the weight vector $\mathbf{w}(k)$ is defined by

$$\mathbf{w}(k) = \begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{w}_2(k) \\ \vdots \\ \mathbf{w}_q(k) \end{bmatrix} \tag{21}$$

where $\mathbf{w}_i(k)(i = 1, 2, \cdots, q)$ is the ith column of the transposed weight matrix $\mathbf{W}^T(k)$. Eq. (18), known as the process equation, specifies that the state of the system is given by the network's weight parameter values $\mathbf{w}(k)$ and is characterized as a stationary process corrupted by process noise $\boldsymbol{\omega}(k)$. The measurement equation, given in Eq. (19), represents the network's output vector $\mathbf{y}(k)$ as the nonlinear function $\boldsymbol{\varphi}(\cdot)$ of the weight vector $\mathbf{w}(k)$ and the vector $\mathbf{z}(k)$ which include both the input vector $\mathbf{u}(k)$ and the recurrent node activations $\mathbf{x}(k)$. This equation is buried by random measurement noise $\boldsymbol{\nu}(k)$. The process noise $\boldsymbol{\omega}(k)$ is typically characterized as zero-mean, white noise with covariance given by $E[\boldsymbol{\omega}_i \boldsymbol{\omega}_j^T] = \delta_{ij}\mathbf{Q}(k)$[1]. Similarly, the measurement noise $\boldsymbol{\nu}(k)$ is also characterized as zero-mean, white noise with covariance given by $E[\boldsymbol{\nu}_i \boldsymbol{\nu}_j^T] = \delta_{ij}\mathbf{R}(k)$.

[1] δ denotes the Kronecker delta.

From the state-space model of the RNN given in equations (18) and (19), the cost function to be minimized in the mean-squared error (MSE) sense is:

$$J(\mathbf{w}) = \mathbf{e}(k)^T \mathbf{R}^{-1}(k)\mathbf{e}(k) \tag{22}$$

where the error vector $\mathbf{e}(k)$ is given in Eq. (11). If the measurement-noise covariance $\mathbf{R}(k)$ is a constant diagonal matrix, it cancels out in the algorithm, and therefore can be set arbitrarily. The process-noise covariance $\mathbf{Q}(k) = E[\boldsymbol{\omega}(k)\boldsymbol{\omega}(k)^T]$ affects the convergence rate and the tracking performance. We define $\mathbf{R}(k)$ and $\mathbf{Q}(k)$ as

$$\begin{aligned} \mathbf{R}(k) &= \mu\mathbf{I} && (23)\\ \mathbf{Q}(k) &= (\lambda_{RLS}^{-1} - 1)\mathbf{P}(k) && (24) \end{aligned}$$

where μ can be set arbitrarily, and $\lambda_{RLS} \in (0, 1]$ is often referred to as the forgetting factor, in recursive least-squares (RLS) algorithms [14].

The UKF effectively evaluates the Jacobian through its sigma-point propagation, without the need to perform any analytical derivative calculation. Specific equations for the RNN using the UKF algorithm are summarized below. The weight vector in the network and the covariance matrix are initialized with

$$\begin{aligned} \hat{\mathbf{w}}(0) &= E[\mathbf{w}] && (25)\\ \mathbf{P}(0) &= E[(\mathbf{w} - \hat{\mathbf{w}}(0))(\mathbf{w} - \hat{\mathbf{w}}(0)^T]. && (26) \end{aligned}$$

The sigma-point calculation is given by

$$\boldsymbol{\Gamma}(k) = (L + \lambda)(\mathbf{P}(k) + \mathbf{Q}(k)) \tag{27}$$

$$\mathcal{W}(k) = [\hat{\mathbf{w}}(k), \hat{\mathbf{w}}(k) + \sqrt{\boldsymbol{\Gamma}(k)}, \hat{\mathbf{w}}(k) - \sqrt{\boldsymbol{\Gamma}(k)}] \tag{28}$$

$$\begin{aligned} \mathcal{D}(k) &= \mathbf{h}(\mathcal{W}(k),\, \mathbf{z}(k)) && (29)\\ \mathbf{y}(k) &= \mathbf{h}(\hat{\mathbf{w}}(k),\, \mathbf{z}(k)). && (30) \end{aligned}$$

The measurement-update equations are

$$\begin{aligned} \mathbf{P}_{\mathbf{yy}}(k) &= \sum_{i=0}^{2L} W_i^c (\mathcal{D}_i(k) - \mathbf{y}(k))(\mathcal{D}_i(k) - \mathbf{y}(k))^T \\ &\quad + \mathbf{R}(k) && (31) \end{aligned}$$

$$\mathbf{P}_{\mathbf{wy}}(k) = \sum_{i=0}^{2L} W_i^c (\mathcal{W}_i(k) - \hat{\mathbf{w}}(k))(\mathcal{W}_i(k) - \hat{\mathbf{w}}(k))^T \tag{32}$$

$$\begin{aligned} \boldsymbol{\Upsilon}(k) &= \mathbf{P}_{\mathbf{wy}}(k)\mathbf{P}_{\mathbf{yy}}^{-1}(k) && (33)\\ \hat{\mathbf{w}}(k+1) &= \hat{\mathbf{w}}(k) + \boldsymbol{\Upsilon}(k)\mathbf{e}(k) && (34)\\ \mathbf{P}(k+1) &= \mathbf{P}(k) - \boldsymbol{\Upsilon}(k)\mathbf{P}_{\mathbf{yy}}(k)\boldsymbol{\Upsilon}^T(k). && (35) \end{aligned}$$

The weight vector of the RNN is updated on-line with the above equations.

5. Nonlinear Channel Equalization

A. Communications System Model

A general model of a digital communications system with a decision feedback equalizer (DFE) is displayed in Fig. 2. It includes both linear and nonlinear distortions. A sequence, $\{s(k)\}$, extracted from a source of information is transmitted, and the transmitted symbols are then corrupted by channel distortion and buried in additive white Gaussian noise (AWGN). The channel with nonlinear distortion is modelled as

$$\begin{aligned} r(k) &= g(\hat{r}(k)) + \nu(k) \\ &= g\left(\sum_{i=0}^{N-1} h_i s(k-i)\right) + \nu(k) && (36) \end{aligned}$$

where $g(\cdot)$ is a nonlinear distortion, h_i is the linear finite impulse response of the channel with length N, $s(k)$ is the sequence of transmitted symbols, and $\nu(k)$ is the AWGN with zero mean and variance σ_0^2. The DFE is characterized by the three integers, m, n and d, known as the feedforward order, feedback order, and decision delay, respectively. The inputs to the DFE therefore consist of the feedforward inputs $\mathbf{r}(k) = [r(k), r(k-1), \cdots, r(k-m+1)]^T$ and feedback inputs $\mathbf{u}(k) = [u(k-1), \cdots, u(k-n)]^T$. The output of the DFE is $y(k)$ and it is passed through a decision device to determine the estimated symbol $\hat{s}(k-d)$. It is sufficient to use feedback order n [15],[16],

$$n = N + m - d - 2 \tag{37}$$

since the transmitted symbols contributing to decision of the equalizer at time k are given by $\mathbf{s}(k) = [s(k), s(k-1), \ldots, s(k-m-N+2)]^T$ for the feedforward order $m = d + 1$. The decision-feedback recurrent neural equalizer (DFRNE) using the fully connected RNN is used as the DFE in the following experiments. When the RNN is used as the DFRNE, the input vector $\mathbf{u}(k)$ includes the received signals from the channel and the decision feedback inputs, as well as the bias input.

B. Experiment 1: Convergence Rate

Channel Model 1: A linear channel model with a nonminimum phase has the transfer function:

$$H_1(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$$

which is generally used for channel equalization in the literature [2],[3],[17]. The nonlinear channel is modeled as

$$r(k) = \tanh(\hat{r}(k)) + \nu(k)$$

where a nonlinearity is applied to the output of the linear channel. This nonlinear distortion of the channel may take into account saturation effects due to transmission amplifiers. The learning rate of the RTRL is chosen empirically as $\eta = 0.1$ and this value ensures a stable convergence.

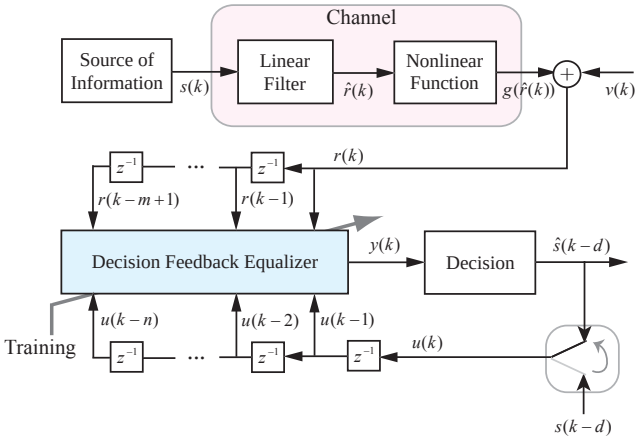


Figure 2. A communications system with decision feedback equalizer.

The parameters for the UKF are chosen empirically as $\alpha = 0.1$, $\mu = 0.5$ and $\lambda_{RLS} = 0.99$. The decision delay is $d = 2$.

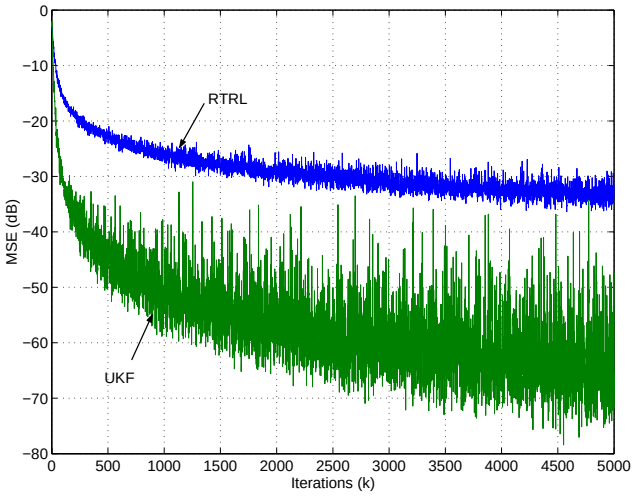
Convergence properties of the DFRNEs used in the simulations are depicted in Fig. 3, with log and linear scales of MSE values. These results are ensemble-averaged over 200 independent runs. Each run has a different BPSK random sequence and random initial weights for the DFRNEs, and is performed at a SNR of 14 dB. The UKF outperforms the RTRL in terms of convergence speed. For instance, MSE value of the UKF reaches around -50 dB after 10^3 training symbols, while MSE value of the RTRL reaches -27 dB. As shown in Fig. 3(b), the UKF reaches steady state after 100 iterations. These results confirm that the UKF algorithm provides an improvement with regard to both the convergence speed and the steady-state MSE. Fig. 4 shows the bit-error rate (BER) performance, averaged over 100 independent trials. In each trial, the first 100 symbols are used for training and the next 10^4 symbols are used for testing. The weight vectors of the DFRNEs are frozen after the training, and the transmission symbols are evaluated at the decision-directed mode. The UKF attains about 1.3 dB of improvement over the RTRL at 10^{-4} of BER. We have observed that the RTRL requires more than 200 training symbols to achieve the same BER performance of the UKF.

C. Experiment 2: Tracking Capability

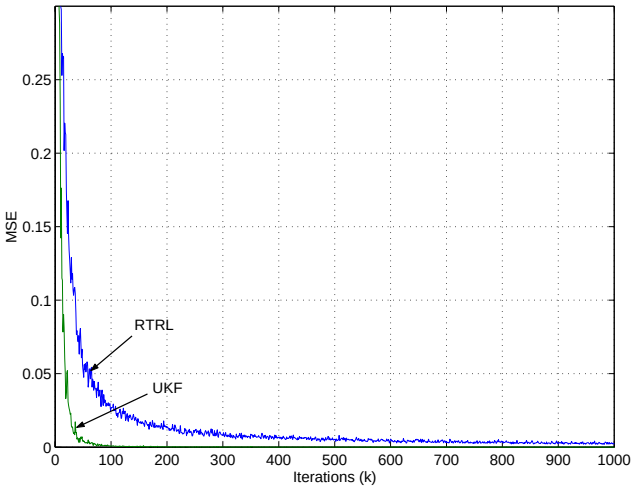
Channel tracking performance of the DFRNEs is tested for a time-varying channel, because tracking is a steady-state phenomenon, in contrast with convergence which is a transient phenomenon [14]. A nonlinear channel with time-varying coefficients is considered here.

Channel Model 2: A time-varying discrete-time channel is described by

$$H_2(z) = 1.0 + a_1(k)z^{-1} + a_2(k))z^{-2}.$$



(a) y-axis: Log scale (10 log_{10}(MSE))



(b) y-axis: Linear scale

Figure 3. Convergence properties for Channel Model 1 under SNR=14dB.

The nonlinear distortion employed in Channel Model 1 is applied to this channel. This channel model represents a nonlinear time-varying channel with $a_i(k)(i = 1, 2)$ varying with time k. These time-varying coefficients are generated by convolving white Gaussian noise and a Butterworth filter response. The bandwidth of the Butterworth filter determines the relative bandwidth (fading rate) of the channel. A nominal 2 kHz channel with a 2400 symbols/s sampling rate is assumed, and a second-order Butterworth filter having a 3 dB bandwidth of 0.5 Hz is used [18].

The parameters are set to the same values as those used in Channel Model 1. Fig. 5(a) shows the time-varying coefficients $a_1(k)$ and $a_2(k)$ drawn for a fading rate of 0.5 Hz. The DFRNEs are in training phase until $k = 2000$ and then they are switched to tracking phase at $k = 2001$. Unlike simulations for Channel Model 1, the DFRNEs still

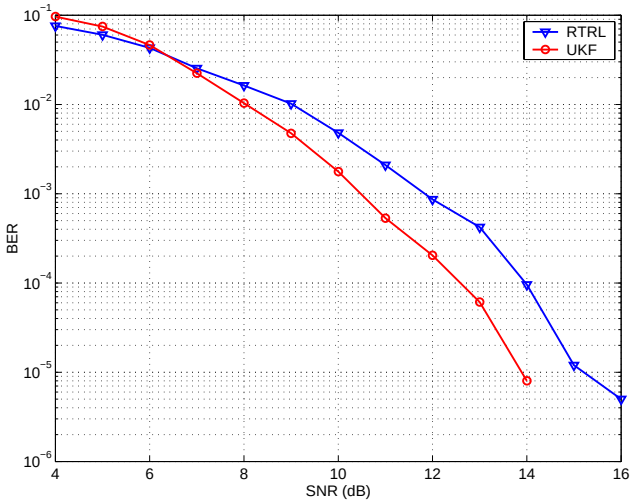


Figure 4. BER performance for Channel Model 1 using 100 training symbols.

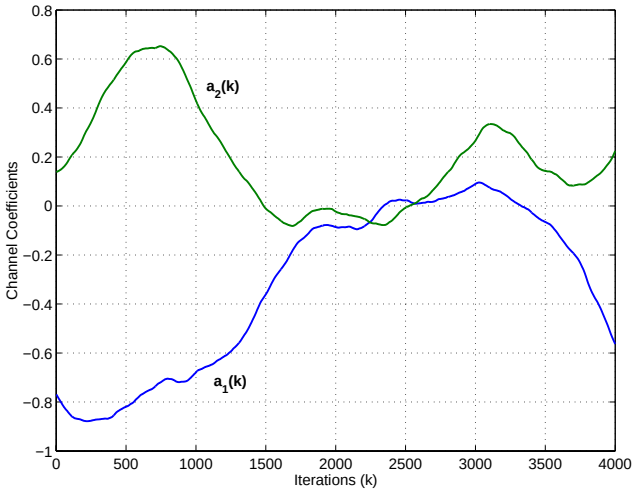
update their weight vectors during testing (tracking) phase in order to track fading characteristic of the channel. In Fig. 5(b), the channel tracking property is evaluated in both training phase and decision-directed tracking phase at a SNR of 15 dB. As expected, the UKF provides faster channel tracking capability than the corresponding RTRL. This result verifies that the MSE value of the UKF is much lower than that of the RTRL for both training and tracking phases. Fig. 6 shows eye diagrams during decision-directed tracking mode for 2×10^3 symbols. The equalized outputs of the UKF have no spots near the decision boundary. In contrast, some of the RTRL's equalized outputs are located in the decision boundary, which creates wrong symbol detections.

D. Comparison of Computational Complexity

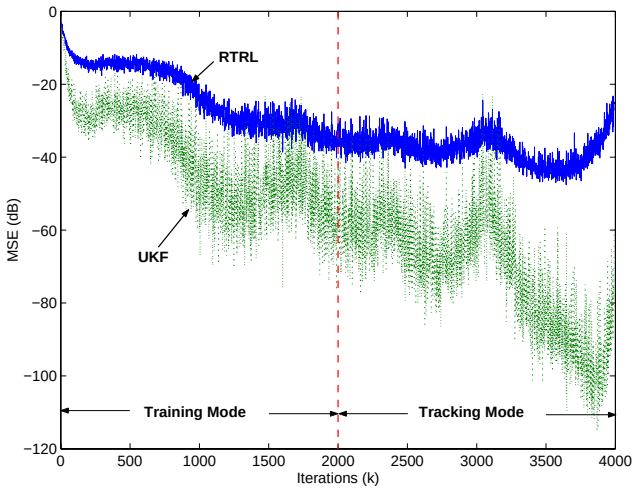
We represent the computational complexity in terms of the number of states (q) and weights (L). The computational time of the RTRL increases on the order $\mathcal{O}(L + q)$, and that of the UKF increases on the order $\mathcal{O}(L^3)$ [11],[12]. Although the UKF is more expensive than the RTRL in computational complexity, it leads to faster convergence rates, lower MSE levels, and smaller BERs, compared to the RTRL. There is an implementation versus complexity trade-off in using the UKF. As the network size grows, the computational expense required to train the transmitted symbols also increases. Fortunately, the DFRNE employing the UKF uses only a small number of neurons, and also needs relatively short training (or pilot) symbols.

6. Conclusions

A derivative-free Kalman filter, called the UKF, was derived for on-line parameter estimation of recurrent neural networks, and its performance was tested through channel



(a) Time-varying coefficients at 0.5Hz.

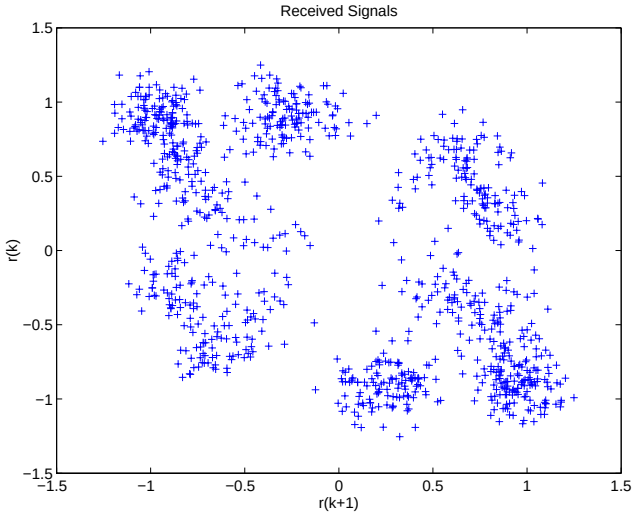


(b) Convergence at SNR=15dB.

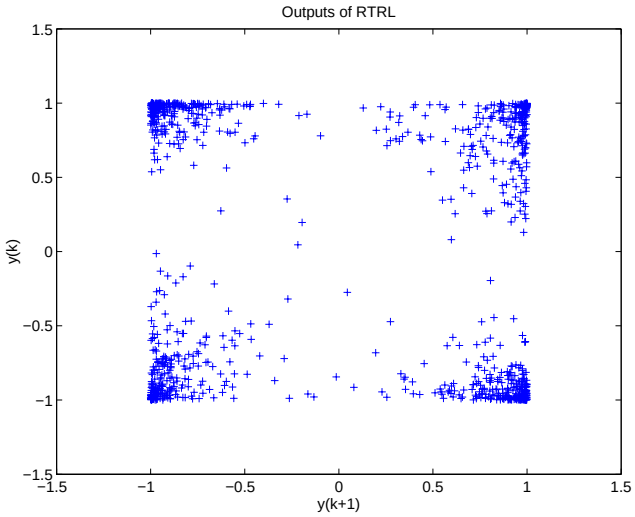
Figure 5. Channel tracking capability for Channel Model 3.

equalization experiments with ISI and nonlinear distortions. In regard of convergence rate, the UKF is superior to the RTRL, which only uses first-order information in the learning process. This means that the fast convergence rate of the UKF requires less training symbols and leads to better BER and pattern classification performance than the RTRL technique. In terms of channel tracking ability compared with the RTRL, the UKF algorithm has shown rapid tracking property for the channel with time-varying coefficients in both the training mode and the decision-directed tracking mode. The superiority of the UKF algorithm compared with the RTRL was consistent in convergence rate, channel tracking capability, as well as BER performance.

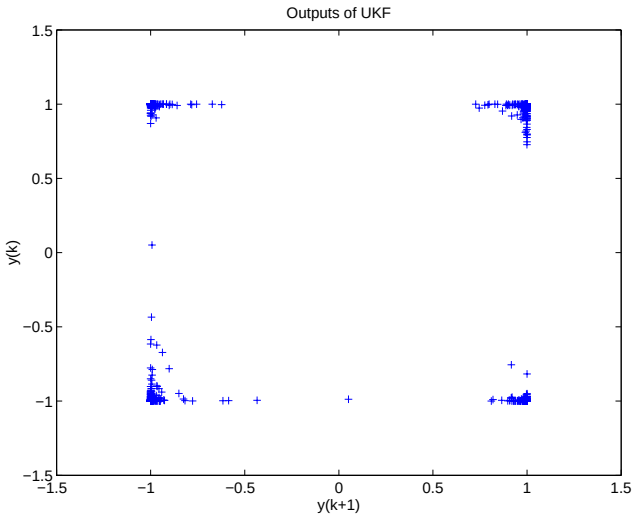
Regardless of the attractive performance of the UKF, there are some issues to be researched: the computational complexity and parameter selection. In terms of the com-



(a) Received Signals.



(b) Outputs of RTRL.



(c) Outputs of UKF.

Figure 6. Eye diagrams for Channel Model 2 during tracking mode (SNR=15dB).

putational cost, the UKF algorithm is relatively expensive. Reducing computational cost for the UKF would be a further research topic. The UKF is more sensitive to initial parameter values to be set than the RTRL. Developing a systematical method for parameter selection of the UKF is still an open question for researchers.

REFERENCES

[1] A. F. Atiya and A. G. Parlos, "New results on recurrent network training: Unifying the algorithms and accelerating convergence," *IEEE Transactions on Neural Networks*, vol. 11, pp. 697–709, May 2000.

[2] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalizations," *IEEE Transactions on Neural Networks*, vol. 5, pp. 267–278, March 1994.

[3] R. Parisi, E. D. Di Claudio, G. Orlandi, and B. D. Rao, "Fast adaptive digital equalization by recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2731–2739, November 1997.

[4] S. Ong, C. You, S. Choi, and D. Hong, "A decision feedback recurrent neural equalizer as an infinite impulse response filter," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2851–2858, November 1997.

[5] H. R. Jiang and K. S. Kwak, "On modified complex recurrent neural network adaptive equalizer," *Journal of Circuits, Systems, and Computers*, vol. 11, no. 1, pp. 93–101, 2002.

[6] B. Hammer and J. J. Steil, "Tutorial: Perspective on learning with RNNs," in *Proc. of the European Symposium on Artificial Neural Networks (ESANN)*, pp. 357–369, 2002.

[7] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.

[8] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, pp. 1550–1560, October 1990.

[9] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, pp. 477–482, March 2000.

[10] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proceedings of AeroSence: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

[11] S. Haykin, *Neural Networks: a Comprehensive Foundation, 2nd Ed.* Upper Saddle River, NJ: Prentice Hall, 1999.

[12] E. A. Wan and R. van der Merwe, "*The unscented Kalman filter,*" *in Kalman Filtering and Neural Networks, Edited by S. Haykin*. John Wiley and Sons, Inc., 2001.

[13] E. A. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications and Control Symposium (AS-SPCC)*, pp. 153–158, 2000.

[14] S. Haykin, *Adaptive Filter Theory, 4th Ed.* Upper Saddle River, NJ: Prentice Hall, 2002.

[15] S. Chen, B. Mulgrew, and S. McLaughlin, "Adaptive Bayesian equalizer with decision feedback," *IEEE Transactions on Signal Processing*, vol. 41, pp. 2918–2927, September 1993.

[16] M. Solazzi, A. Uncini, E. D. Di Claudio, and R. Parisi, "Complex discriminative learning Bayesian neural equalizer," *Signal Processing*, vol. 81, pp. 2493–2502, 2001.

[17] S. Chen, G. J. Gibson, B. Mulgrew, and S. McLaughlin, "Adaptive equalization of finite nonlilear channels using multilayer perceptrons," *Signal Processing*, vol. 20, pp. 107–119, 1990.

[18] C. Cowan and S. Semnani, "Time-variant equalization using a novel non-linear adaptive structure," *International Journal of Adaptive Control and Signal Processing*, vol. 12, no. 2, pp. 195–206, 1998.