# Neural Network-Based Market Clearing Price Prediction and Confidence Interval Estimation With an Improved Extended Kalman Filter Method

Li Zhang, *Member, IEEE,* and Peter B. Luh, *Fellow, IEEE*

***Abstract*—Market clearing prices (MCPs) play an important role in a deregulated power market, and good MCP prediction and confidence interval (CI) estimation will help utilities and independent power producers submit effective bids with low risks. MCP prediction, however, is difficult, since MCP is a nonstationary process. Effective prediction, in principle, can be achieved by neural networks using extended Kalman filter (EKF) as an integrated adaptive learning and CI estimation method. EKF learning, however, is computationally expensive because it involves high dimensional matrix manipulations. This paper presents a modified U-D factorization method within the decoupled EKF (DEKF) framework. The computational speed and numerical stability of this resulting DEKF-UD method are significantly improved as compared to standard EKF. Testing results for a classroom problem and New England MCP predictions show that this new method provides smaller CIs than what provided by the BP-Bayesian method developed by the authors. Testing also shows that our new method has faster convergence, provides more accurate predictions as compared to BP-Bayesian, and our DEKF-UD MCP predictions are comparable in quality to ISO New England's predictions.**

***Index Terms*—Confidence interval, deregulated power market, extended Kalman filter, market clearing price, neural networks, prediction.**

## I. Introduction

THE MARKET clearing prices (MCPs) in a deregulated power market are volatile. For an independent system operator (ISO), the energy MCP is cleared by solving a unit commitment and economic dispatch problem with the bids and system conditions. High-quality MCP prediction and its confidence interval (CI) estimation would help utilities and independent power producers submit effective bids with low risks, and make good bilateral transaction decisions. What is a good MCP prediction and CI estimation method for a utility company who only has limited information? This is a difficult question because MCPs are heavily affected by load, which can go through rapid changes due to weather swings or seasonal changes causing MCP to be nonstationary [1]. Different estimation methods will have different CIs, while small CIs are preferred, and this topic has not been adequately investigated. How to develop an adaptive MCP prediction method with fast convergence and small CIs is the major issue addressed in this paper.

Among the variety of prediction methods, neural networks have been widely used because of their strong learning capability [2]. They can approximate any continuous multivariate function to a desired degree of accuracy [3], [4], or predict a nonstationary process if the weights are adaptively adjusted during on-line update [1]. The multilayer perceptron (MLP) network is one of the popular networks and will be used in this paper, with back-propagation (BP) and the Newton's method as its learning algorithms. Traditional BP is a first-order steepest decent method and suffers from slow convergence, and may not be effective for predicting nonstationary processes [4], [5]. Although the Newton's method requires less number of iterations to converge, it suffers from excessive computational requirements for large problems [6], including the MCP prediction under consideration.

The Kalman filter is a well-known method for recursive state estimation of linear dynamic systems, and is a minimum mean-square-error estimator. Through linearization, the extended Kalman filter (EKF) has been widely adopted for state estimation of nonlinear systems [7], [8], and can be used for state estimation of nonstationary processes because of its tracking capability [9]. EKF has been used to train MLP networks by treating weights of a network as the state of an unforced nonlinear dynamic system [10]–[13]. Since it is a second-order learning algorithm, fast convergence is expected. In addition to providing predictions, EKF can also estimate CIs based on its innovation covariance matrix. For nonlinear systems, EKF has been shown to preserve most of the properties of Kalman filter if nonlinearities are not severe [8]. Small CI estimation can thus be integrated with learning, as will be briefly reviewed in Section II, and a modified EKF considering the input uncertainty will then be developed in Section III.

Using EKF as a learning method for MCP prediction, however, may not be easy in view of the high dimensionality of the weights involved (e.g., 500), causing excessive computational requirements. Efficient implementation of EKF is therefore critical. A simplified EKF algorithm called the "decoupled EKF" (DEKF) was developed in [12], ignoring the interdependency of weights across neurons. Specifically, the input weights to a

Manuscript received January 24, 2003. This work was supported in part by Select Energy Inc., Berlin, CT, a company of the Northeast Utilities System, and by the National Science Foundation under Grant ECS 9726577. This paper was presented in part at the PES 2002 Winter Meeting, New York, January 2002. Paper no. TPWRS-00229-2002.

L. Zhang was with the University of Connecticut, Storrs, CT 06269-2157 USA. He is now with ISO New England, Inc., Holyoke, MA 01040 USA (e-mail: Lzhang@iso-ne.com).

P. B. Luh is with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-2157 USA (e-mail: Peter.Luh@uconn.edu).

Digital Object Identifier 10.1109/TPWRS.2004.840416

neuron are grouped, and the weight covariance matrix is assumed to be block diagonal. The innovation covariance for the entire network is first computed, and is then used within the decoupled groups to update Kalman gains, weight covariance matrices, and weights. Testing results showed that the degradation in prediction accuracy is minor.

Due to truncation and round-off errors, DEKF may lead to the loss of symmetry and positive definiteness of the weight covariance, causing difficulties when the number of data samples is large [13], [14]. What method could be used to improve the numerical stability and accuracy for DEKF? U-D factorization is a generic method to improve matrix computation. It factorizes matrices into unit upper triangular and diagonal matrices [7], [13]. When used in EKF, standard U-D factorization needs to update the innovation covariance matrix, Kalman gain, weight covariance matrix, and weights at the same level, while the standard DEKF updates the innovation covariance at the high level for the entire network and other matrices at the low level within individual groups. To overcome this difficulty, our idea is to update the innovation covariance at the high level for the entire network, while implementing U-D factorization at the low level within individual groups by modifying the standard U-D factorization. The new DEKF-UD method will be presented in Section IV with significant reduction in computation and improvement in numerical stability.

Numerical testing results for a classroom problem and for New England MCP prediction presented in Section V demonstrate that neural network learning with DEKF-UD is much faster than that with standard EKF or Joseph form covariance update EKF implementation. Testing also shows that DEKF-UD has faster convergence, and provides more accurate predictions as compared to BP. DEKF-UD also provides smaller CIs as compared to a BP-Bayesian method (a combined BP learning and Bayesian-based CI method) developed by the authors [15], with 50% reduction for a classroom problem and 7% reduction for New England MCP prediction.

## II. Literature Review

Neural network prediction methods have been briefly described in Section I, and more detailed coverage can be found in [3] and [16]. The methods to estimate CIs can roughly be classified into three categories: resampling [17], [18], perturbation model [19]–[23], and Bayesian inference [3], [15], [24], [25] to be briefly described below.

The resampling method derives CIs by randomly selecting data points with replacement from an original output data set to form multiple sample data sets. The CI of the mean can then be calculated from means of the sample data sets. This method resamples output data, and cannot effectively consider input uncertainties. The perturbation model examines the effect on output if some parameters are perturbed. It uses Taylor series expansion to relate changes in the output to perturbed parameters. This method, however, is difficult for a neural network with a large number of weights because it requires high dimensional weight covariance matrices. Both the resampling and perturbation model have difficulty to address the small CI issue.

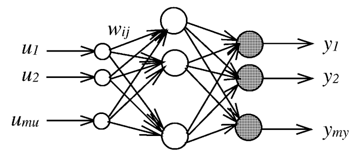

Fig. 1. A MLP network.

Bayesian learning for neural networks has attracted much attention recently. Starting with a prior distribution of weights for a neural network, the method develops a posterior distribution of the weights $w$ from historical data $D$. Optimized weight vector $w_{\mathrm{MP}}$ is obtained by maximizing a posterior distribution $p(w \mid D)$, or equivalently minimizing a cost function with sum-of-squares error plus an additional "weight-decay" regularization term [3]. The Bayesian method is therefore a minimum mean square error estimator by using, for example, BP to minimize the cost function. BP implements gradient descent in the weight space, where partial derivatives of the cost function with respect to weights are computed backward in layers. It is a first-order algorithm, and suffers from slow convergence. By linearizing the neural network, the prediction distribution conditioned on a new input and weights can be derived and approximated as Gaussian [15], [25]. Markov Chain Monte Carlo methods [3], [25] or an efficient implementation method [15] based on Gaussian approximation have been used to calculate the CIs.

EKF has been used for neural network learning. It is also developed under the Bayesian framework, and CI estimation is integrated with the whole learning procedure. Updating is based on each new observation, and the recursive implementation facilitates on-line operation.

## III. EKF-Based Learning With Input Uncertainty

With the MCP prediction in mind, a modified EKF learning method considering input uncertainty will be derived for multi-layer perceptron networks in this section.

A multi-layer perceptron (MLP) network [4] with a single hidden-layer is shown in Fig. 1. The MLP is trained by adjusting its weights using a set of input–output observations $\{u(t), y(t) : t = 1, \ldots, T\}$, where $T$ is the number of data samples, $u(t)$ is an $m_u \times 1$ input vector, and $y(t)$ is the corresponding $m_y \times 1$ output vector. When training is finished, the neural network predicts the output for a new input. Weights are then updated when actual output becomes available. The learning procedure includes training and update, and can be considered as a nonlinear estimation problem where the weights are to be estimated.

Applying EKF to neural network training, the first step is to organize all the network weights as a state vector $w(t)$. The training can then be described as a state estimation problem with the following unforced dynamic and observation equations [13]:

$$w(t+1) = w(t) \tag{1}$$

$$y(t) = f[w(t), u(t)] + \varepsilon(t) \tag{2}$$

where $w(t)$ is an $m_w \times 1$ state vector and $f[\cdot]$ is the MLP input-output relationship. The measurement uncertainties in output

$\varepsilon(t)$ are assumed to be zero-mean white Gaussian with covariance matrix $R(t)$.

In practice, certain key input factors (e.g., load) for MCP prediction may not be available in real time and need to be predicted, with associated uncertainties in predicted values. Assume that the predicted input $\hat{u}(t)$ is the true input $u(t)$ plus the input prediction uncertainty

$$\hat{u} = u + \tilde{u} \tag{3}$$

where $\tilde{u}(t)$ is zero-mean white Gaussian with covariance matrix $\Sigma_u(t)$. The state estimation is then to determine $w(t)$ to minimize the following sum of squared error cost function:

$$J = \frac{1}{2}\sum_{t=1}^{T} \|y(t) - f[w(t), u(t)]\|^2. \tag{4}$$

EKF estimates the state based on feedback from measurements [7], [8], and has two key steps: time update (propagation) and measurement update. The time update is to project forward in time the current state to obtain an *a priori* estimate for the next step. The measurement update then incorporates a new measurement to obtain a posteriori estimate based on Bayesian techniques. To derive the EKF formula considering input uncertainty, a first-order Taylor series expansion of $f[w(t), u(t)]$ around the estimated weights $\hat{w}(t)$ and predicted input $\hat{u}(t)$ is performed

$$\begin{aligned} y(t) = f[\hat{w}(t), \hat{u}(t)] &+ H(t)(w(t) - \hat{w}(t)) \\ &+ G(t)(u(t) - \hat{u}(t)) + HOT + \varepsilon(t). \end{aligned} \tag{5}$$

In the above, $H(t)$ is the partial derivative of $f[\cdot]$ with respect to $w(t)$ at the estimated weights, i.e., a Jacobian matrix with dimension $m_y \times m_w$

$$H(t) = \left(\frac{\partial f[w(t), u(t)]}{\partial w(t)}\right)^T \Bigg|_{\substack{w(t)=\hat{w}(t) \\ u(t)=\hat{u}(t)}} \tag{6}$$

and $G(t)$ is the partial derivative of $f[\cdot]$ with respect to $u(t)$ at the estimated input, i.e., a Jacobian matrix with dimension $m_y \times m_u$

$$G(t) = \left(\frac{\partial f[w(t), u(t)]}{\partial u(t)}\right)^T \Bigg|_{\substack{w(t)=\hat{w}(t) \\ u(t)=\hat{u}(t)}} \tag{7}$$

and the higher order terms (HOT) may be neglected. Then the estimated output $\hat{y}(t)$ is given by

$$\hat{y}(t) = f[\hat{w}(t), \hat{u}(t)] \tag{8}$$

with the measurement residual calculated as

$$\begin{aligned} \tilde{y}(t) &= y(t) - \hat{y}(t) \\ &\cong H(t)(w(t) - \hat{w}(t)) + G(t)(u(t) - \hat{u}(t)) + \varepsilon(t). \end{aligned} \tag{9}$$

The innovation covariance $S(t+1)$ is the covariance of the measurement residual (9). The difference between the modified EKF and standard EKF [7], [14] is at the innovation covariance in view of the input uncertainty. The rest of the derivation is straightforward by following the standard EKF derivation. The key steps are summarized below.

Time-update equations

$$\hat{w}(t) = \hat{w}(t) \tag{10}$$

$$\hat{y}(t) = f[\hat{w}(t), \hat{u}(t)]. \tag{11}$$

Measurement-update equations

$$\begin{aligned} S(t+1) = H(t+1)P(t)H(t+1)^T \\ + G(t+1)\sum_u(t+1)G(t+1)^T + R(t+1) \end{aligned} \tag{12}$$

$$K(t+1) = P(t)H(t+1)^T S(t+1)^{-1} \tag{13}$$

$$P(t+1) = P(t) - K(t+1)H(t+1)P(t) \tag{14}$$

$$\hat{w}(t+1) = \hat{w}(t) + K(t+1)\{y(t) - \hat{y}(t)\} \tag{15}$$

where $K(t)$ is the Kalman gain and $P(t)$ is the weight covariance matrix. In the above, the term $G(t+1)\sum_u(t+1)G(t+1)^T$ in (12) is extra in view of input uncertainty, and input uncertainty has a direct impact only on the innovation covariance. The variance of each output can be obtained from the diagonal elements of the innovation covariance matrix $S(t+1)$, and the CIs can then be obtained by deviating a certain number of standard deviations from the prediction. CI estimation is thus integrated with the learning process.

The recursive formula (10)–(15) can be used both for training and update, and for update, only one new measurement is used to fine-tune the weights.

## IV. Modified U-D Factorization Within the DEKF Structure

Though a modified EKF learning method considering input uncertainty was presented in the last section, it is difficult to implement for MCP prediction in view of the computation with high dimensional weight space. An efficient implementation is then the focus of this section. The computational difficulty for MCP prediction will be first analyzed, followed by the standard DEKF and U-D factorization implementation. A novel DEKF-UD algorithm with the combined DEKF and U-D factorization will be developed to improve the computational efficiency and numerical stability. The CIs from DEKF-UD will then be compared to those from the Bayesian inference method [15], [25].

### *A. Decoupled EKF*

The primary deterrent to use original EKF for a neural network with high dimensional weight space is the necessity of computing a symmetric $m_w \times m_w$ weight covariance matrix $P(t)$. Assuming that intra-neuron weight correlation is high, and inter-neuron weight correlation is low, a simplified algorithm called the "decoupled EKF" (DEKF) was therefore developed which ignores the interdependencies of weights from different neurons, thereby leading to reduced computational complexity and storage [12]. In developing DEKF, the input weights to a neuron are grouped, and the weight covariance matrix is assumed to be block diagonal. An example was given in [12] showing that DEKF is about 40 times faster than the original EKF implementation.

Assuming that total number of neurons in the hidden layer and output layer for a network is $g$, the measurement update equations for DEKF are given by

$$S(t+1)=\sum_{k=1}^{g}\left(H_k(t+1)P_k(t)H_k^T(t+1)\right) + G(t+1)\Sigma(t+1)G^T(t+1)+R(t+1) \tag{16}$$

$$K_k(t+1)=P_k(t)H_k(t+1)^TS(t+1)^{-1} \tag{17}$$

$$P_k(t+1)=P_k(t)-K_k(t+1)H_k(t+1)P_k(t) \tag{18}$$

$$\hat{w}_k(t+1)=\hat{w}_k(t)+K_k(t+1)\{y(t)-f[\hat{w}(t),\hat{u}(t)]\} \tag{19}$$

while $k=1,\ldots,g$ for (17)–(19). In the above, $P_k$ is the $m_{wk}\times m_{wk}$ block weight covariance matrix, $K_k(t+1)$ is the $m_{wk}\times m_y$ Kalman gain, $H_k(t+1)$ is a $m_y\times m_{wk}$ matrix containing the partial derivatives of the output with respect to weights, $w_k$ is the $m_{wk}\times 1$ input weights, and all these are for neuron $k$ or decoupled group $k$. It can be seen from these above equations that DEKF has a two-level structure: the innovation covariance $S(t+1)$ for the entire network is first computed in (16) at the high level, and is used at a low level within the decoupled groups to update Kalman gains, weight covariance matrices, and weights in (17)–(19). The innovation covariance $\mathrm{S(t+1)}$ could not be decoupled into groups since there is no meaning for decoupled innovation covariance. Analysis of the above equations indicates a computational complexity of $O(m_y^2m_w+m_y\sum_{k=1}^{g}m_{wk}^2)$ for DEKF as compared to $O(m_y^2m_w+m_ym_w^2)$ for the original EKF [11].

## B. Novel DEKF-UD Algorithm

Due to truncation and round-off errors, DEKF may lead to the loss of symmetry and positive definiteness of the weight covariance, causing difficulties when the number of data samples is large. Possible methods to overcome such difficulties include singular value decomposition (SVD), U-D factorization, square root filtering, and Joseph form covariance update. Among them, U-D factorization is a way to implement "square root filtering" without actually computing square roots, and is considered to be efficient, stable, and accurate [7]. In the standard U-D factorization, the covariance matrix $P$ is decomposed into $P=UDU^T$, where $U$ is unit upper triangular and $D$ is diagonal. U-D factorization for EKF neural network learning has been reported in [13], however, it by itself is not sufficient for MCP prediction. The purpose of this paper is to integrate U-D factorization within DEKF to further improve its efficiency and stability. The combination of these two methods, however, is difficult since

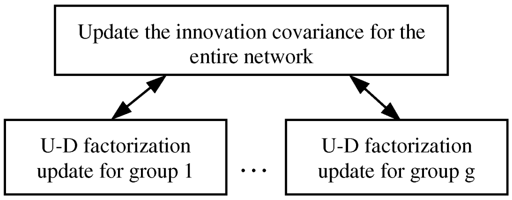
Update the innovation covariance for the entire network
U-D factorization update for group 1
...
U-D factorization update for group g

Fig. 2. Structure of the DEKF-UD algorithm.

the standard U-D factorization updates all the matrices at the same level, while the standard DEKF updates the innovation covariance at the high level for the entire network and other matrices at the low level within individual groups. To overcome this difficulty, our idea is to update the innovation covariance at the high level for the entire network, while implementing U-D factorization at the low level within individual groups. The overall structure of DEKF-UD algorithm is shown in Fig. 2.

A remaining difficulty for this new structure is the excessive computational requirements to compute the inverse innovation covariance $S^{-1}\ (t+1)$ (an $m_y\times m_y$ matrix) in (17) when the dimension of the output is high. To overcome this difficulty, our approach is to transform the problem into a sequence of single output problems [7], [8], [14]. The $m_y$-dimensional output is thus sequentially processed, and for each one, there is only a single output. Consequently, the computation of $S^{-1}\ (t+1)$ from $S(t+1)$ is trivial. In the following, the DEKF-UD algorithm will be illustrated for the single output case.

To develop the core of the DEKF-UD algorithm corresponding to the measurement update equations (16)–(19), the notation of [7] will mostly be followed. The factorized weight covariance $P_k(t)$ is represented as

$$P_k(t)=\tilde{U}\tilde{D}\tilde{U}^T \tag{20}$$

where $\tilde{U}$ is a unit upper triangular matrix with value 1 on the diagonal and $\tilde{D}$ is a diagonal matrix. The factorized $P_k\ (t+1)$ is represented as

$$P_k(t+1)=\hat{U}\hat{D}\hat{U}^T. \tag{21}$$

These matrices can be described as $\tilde{U}=[\tilde{u}_{1,\ldots,}\tilde{u}_{m_{wk}}]$ with $m_{wk}\times 1$ vector element $\tilde{u}_j, j=1,\ldots,m_{wk};\hat{U}=[\hat{u}_1,\ldots,\hat{u}_{m_{wk}}]$ with $m_{wk}\times 1$ vector element $\hat{u}_j, j=1,\ldots,m_{wk};\tilde{D}=\mathrm{Diag}(\tilde{d}_j)$ and $\hat{D}=\mathrm{Diag}(\hat{d}_j)$.

Assume that the neural network can be linearized as in (5) with $\mathrm{E}(\varepsilon)=0$ and $\mathrm{E}(\varepsilon^2)=R(t+1)$. The innovation variance $S(t+1)$ (a scalar) can be obtained from (16). The core of the DEKF-UD algorithm can then be derived by following the derivation of Theorem V.3.1 (U-D update algorithm, [7, p. 77]) and the Agee-Turner factorization update theorem (page 44, [7]). The steps of DEKF-UD corresponding to (16)–(19) are as follows.

Step 1) Calculate innovation variance for the entire network: $S(t+1)$ according to (16).

For group $k=1$ to $g$, cycle through Step 2 to Step 5.[1]

[1]For notational simplicity, steps 2 to 5 ignore the subscript k indicating the decoupled groups.

Step 2) Intermediate steps for (17) and (18)

$$f \equiv \tilde{U}^T H = (f_1, \ldots, f_{m_w})^T \tag{22}$$

$$v \equiv \tilde{D} f, \quad \text{with } v_i = \tilde{d}_i f_i, \quad i = 1, \ldots, m_w \tag{23}$$

$$K_2^T = \left( v_1, \ldots \overbrace{0 \ldots 0}^{n-1} \right), \quad c_{m_w} = -1/S(t+1). \tag{24}$$

Step 3) Intermediate steps for (17) and (18).
For $j = m_w$ down to $j = 2$, cycle through (25)–(27)

$$\hat{d}_j = \tilde{d}_j + c_j v_j^2 \tag{25}$$

$$c_{j-1} = c_j \tilde{d}_j / \hat{d}_j \tag{26}$$

$$\lambda_j = f_j c_{j-1}. \tag{27}$$

For $j = 2, \ldots m_w$, cycle through (28)–(29)

$$\hat{u}_j = \tilde{u}_j + \lambda_j K_j \tag{28}$$

$$K_{j+1} = K_j + v_j \tilde{u}_j. \tag{29}$$

Step 4) The Kalman gain is given by

$$K = K_{m_w+1}/S(t+1). \tag{30}$$

Step 5) Update weight estimate

$$w(t+1) = w(t) + K(y - \hat{y}). \tag{31}$$

In (25) and (26), $j$ recursively cycles from $m_w$ down to 2, instead of cycling from 2 up to $m_w$ as in the standard U-D factorization version in [7]. The innovation variance for the entire network $S(t+1)$ is thus used within decoupled groups through $c_{mw} = -1/S(t+1)$. This modification is the key for the combination of the standard U-D factorization and DEKF.

### C. Comparing EKF With the BP-Bayesian Method

Bayesian-based CIs using BP (BP-Bayesian) [15] have been described earlier in the Literature Review section. CIs by using EKF presented above in Section III are also based on the Bayesian framework. What are the similarities and differences between these two methods? From [25, eq. (31)], Bayesian-based prediction covariance considering input uncertainty is given by

$$\Sigma_y(t+1) = H(t+1)A^{-1}(t)H(t+1)^T + G(t+1)\sum_u(t+1)G(t+1)^T + R(t+1) \tag{32}$$

where $A$ is the Hessian of the cost function (4) with respect to the weights. This is very similar to formula (12) for EKF, and the only difference is that $P(t)$ (the weight covariance matrix) in (12) is replaced by $A^{-1}$. The weight covariance matrix has been shown to be statistically the same as the inverse Hessian $A^{-1}$ [22]. The appearances of the two formulas are therefore identical.

Though EKF and BP are based on the same Bayesian concept, they have major differences in performance. EKF is a second-order algorithm with a recursive implementation, and reaches a steady state much faster than BP for nonstationary processes. The CIs are obtained based on its innovation covariance matrix, which is embedded in the learning procedure to update the weights in EKF. BP is a first-order algorithm, and the CI, as second-order information, could not be used in its learning procedure. BP also needs to trade off between two implementation modes, batch mode or incremental mode. Batch mode could reach good weights in training but has difficulty for on-line implementation because of its computational requirements and the nonstationarity of the process. Incremental mode has the difficulty to select the number of update iterations for each new data point, since too few iterations may result in slow convergence, and too many iterations may lose the information obtained from earlier training. EKF therefore performs better in practice in terms of better prediction and smaller CIs that were preferable in a nonstationary environment. This will be demonstrated in Section V.

## V. Numerical Testing Results

The DEKF-UD algorithm for neural network learning and CI estimation has been implemented in C++ on a Pentium IV PC with 1.3-GHz CPU and 128-MB memory. Two examples are presented below. The first classroom problem shows that DEKF-UD has faster convergence and better prediction, and provides smaller CIs than what provided by the BP-Bayesian method. The second New England MCP prediction example demonstrates that DEKF-UD is practical for problems with high dimensional weight space in terms of computational efficiency and numerical stability. DEKF-UD also outperforms the BP-Bayesian method as in the first example, with prediction quality comparable to that of ISO New England. This example also shows the input factors for MCPs, and their partial correlation to the MCPs.

*Example 1:* An MLP network was used to approximate the following nonlinear function:

$$\bar{y} = \frac{2}{1 + e^{\left(\left(\frac{1}{1+e^{(-0.1-0.5u)}}\right)+\left(\frac{1}{1+e^{(0.5+0.4u)}}\right)-1\right)}} + 0.5\sin(0.5u) + 0.5\frac{u}{1+u^2} \tag{33}$$

which was composed of a sigmoidal activation function plus sinusoidal and rational terms. The function input and output $\{u, \bar{y}\}$ in (33) are noiseless, while noisy input and output will be used for neural network training and prediction as described in Section III. Specifically, 20 noisy data sets $\{\hat{u}, y\}$ were randomly generated for training with $\hat{u} = u + 0.0001\varepsilon_1, u = -11 + i$, and $y = \bar{y} + 0.01\varepsilon_2$, where $i$ is the data set index, $i \in [1, 2, \ldots, 20]$, and $\varepsilon_1, \varepsilon_2 \in \mathrm{N}(0,1)$. In prediction, another 20 random data sets were generated with the same noises as in training.

Results obtained by using the DEKF-UD and BP-Bayesian [15] methods were compared, where for the BP-Bayesian method, the incremental mode was used with one update iteration for each data sample in training and in update. Fig. 3 shows the training performance for these two methods. It demonstrates that DEKF-UD converges fast, and the mean absolute error (MAE) in training reduces below 0.05 within ten iterations, while the reduction of MAE for BP-Bayesian is very slow.

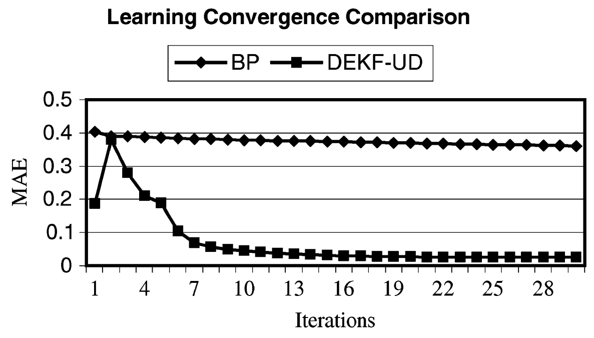

Fig. 3. Training comparison: DEKF-UD and BP-Bayesian.

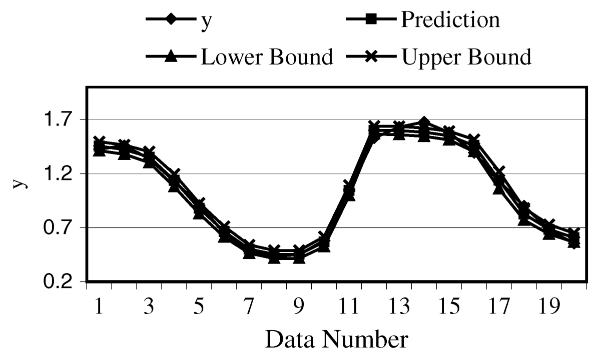

Fig. 4. DEKF-UD prediction and CIs.

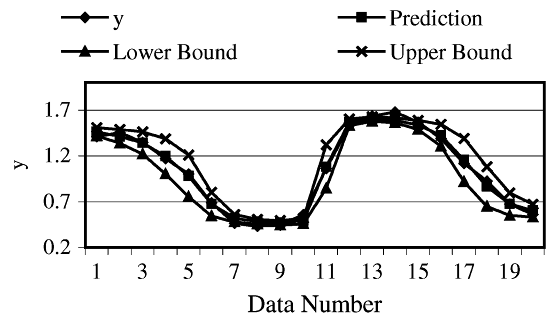

Fig. 5. BP-Bayesian prediction and CIs.

Fig. 4 shows the prediction and one-sigma CIs for DEKF-UD. The prediction curve matches the desired curve y very well, and the CIs are small. Fig. 5 shows the results obtained by using the BP-Bayesian method, with much larger CIs than those obtained by using DEKF-UD. Table I presents the statistics for these two methods based on 100 Monte Carlo runs, where for each run, a fixed number of training iterations are executed for a method to roughly reach a desired level of prediction accuracy. The table shows that the training time required by DEKF-UD is much smaller than that required by BP-Bayesian, and one-sigma CIs obtained by DEKF-UD are less than half of the size of those obtained by BP-Bayesian.

*Example 2:* In this example, MLP networks were used to predict day-head on-peak New England MCPs (average MCPs from 7 A.M. to 11 P.M.) and their CIs. The results from the DEKF-UD and BP-Bayesian methods, and ISO New England's (ISO-NE) predictions were compared.

ISO-NE's predictions were obtained by solving a unit commitment and economic dispatch problem based on bids submitted, and were available from its website. For the DEKF-UD and BP-Bayesian methods, ISO-NE's data were collected from May 1, 1999 to the end of May 2001 from ISO-NE's website (www.iso-ne.com). The input factors for MCP prediction include load, surplus, historical MCPs, gas and oil prices, and the aggregation of these data. The surplus is the total available capacity minus the required capacity at the peak hour. Qualitative variables, such as "day of the week" and holidays, are difficult to model. In this paper, there are seven extra zero-one input factors indicating day of the week, and holidays are treated as Saturday. The total number of the input factors is 50. The neural network thus had a total number of weights close to 500 and was trained on data from May 1, 1999 to June 30, 2000, then predicted from July 1, 2000 to May 31, 2001.

TABLE I
COMPARISON OF DEKF-UD, BP-BAYESIAN

| | DEKF-UD | BP-Bayesian |
|---|---|---|
| Training Iterations | 30 | 1500 |
| Training Time (Second) | 0.7 | 2.3 |
| Prediction Error (MAE) | 0.029 | 0.037 |
| Average Intervals | 0.046 | 0.10 |
| One-sigma CI Coverage (%) | 75% | 70% |

CI: Confidence intervals

TABLE II
PREDICTION RESULTS FOR THE THREE METHODS

| | DEKF-UD | BP-Bayesian | ISO-NE |
|---|---|---|---|
| MAE ($) | 5.7 | 6.3 | 5.5 |
| MAPE (%) | 11.1 | 12.1 | 10.2 |
| Intervals ($) | 5.8 | 6.2 | N/A |
| One-sigma CI Coverage (%) | 61 | 61 | N/A |

The overall prediction performance from July 1, 2000 to May 31, 2001 for these two methods and ISO-NE's predictions are summarized in Table II. It demonstrates that DEKF-UD has smaller MAE and mean absolute percentage error (MAPE) than what BP-Bayesian has, and DEKF-UD also provides smaller CIs than what BP-Bayesian provides. In view that the current New England market follows a single-settlement system and prices are not financially binding until real-time, the day-ahead MCP prediction by ISO-NE based on bids submitted has large errors. With much less information than what ISO-NE has, our neural network predictions are comparable in quality to the ISO's predictions.

To examine MCP predictions in hot summer days when price spikes are likely to happen, results for DEKF-UD, BP-Bayesian, and ISO-New England for July and August 2000 are summarized in Table III. It is clear that DEKF-UD outperforms BP-Bayesian for this critical period.

To graphically see the results, DEKF-UD predictions for November 2000 are shown in Fig. 6. The MAE for this month is $3.8, and MAPE is 7.3%. Fig. 7 demonstrates the CIs of DEKF-UD for the same month, with one-sigma CI coverage 73%, and this coverage is close to 68% Gaussian coverage. These CIs are significantly smaller than those obtained by BP-Bayesian as shown in Fig. 8, both cover one-sigma CIs. Tables I, II, and Fig. 7 also show that the CIs developed by our method are not always consistent with the 68% Gaussian coverage for each case. The inconsistency might be caused

TABLE III
2000 SUMMER PREDICTION FOR THE THREE METHODS

| Period | DEKF-UD | | BP-Bayesian | | ISO-NE | |
|---|---|---|---|---|---|---|
| | MAE ($) | MAPE (%) | MAE ($) | MAPE (%) | MAE ($) | MAPE (%) |
| July | 2.8 | 6.2% | 2.8 | 6.3% | 4.5 | 10.85% |
| Aug. | 5.5 | 11.2% | 7.01 | 13.4% | 4.4 | 9.7% |

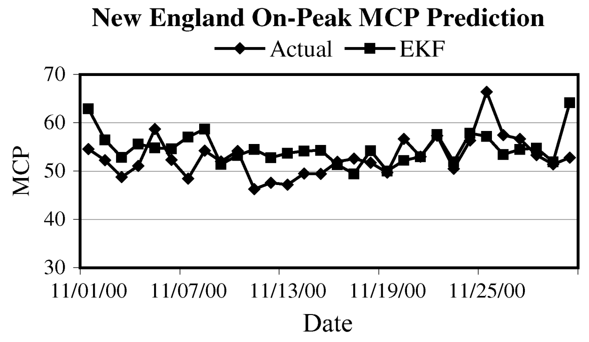

Fig. 6. New England on-peak MCP prediction (DEKF-UD).

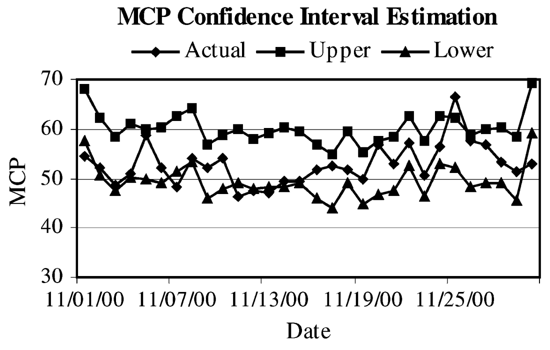

Fig. 7. New England on-peak MCP CI estimation (DEKF-UD).

by abnormal behaviors in the markets or by the inaccuracy of Gaussian approximation [22].

From the above results and many other tests not reported here, our DEKF-UD method is shown to be computationally efficient and numerically stable. Our method is also shown to outperform BP-Bayesian and the standard EKF method for problems with a high dimensional weight space.

To assess what inputs are most closely related to MCPs, a kind of "sensitivity method" was used. For each input factor, the sensitivity is calculated by using a kind of "central difference formula." With all other input factors at their means, the sensitivity for a particular input factor is obtained by the MCP with that input factor set to be its mean plus one standard deviation, minus the MCP with the input factor set to be its mean minus one standard deviation. Part of input factors is shown in Table IV. The left column of the table lists the input factors for MCP prediction. The input variables are ranked by the degree they affect the output, or the absolute values of the sensitivity.

The three input factors which are most closely related to the day ahead MCPs are the average day-ahead load, the maximum day-ahead load and the day-ahead surplus, as shown in Table IV. Following them are the day ahead gas price, historical MCPs

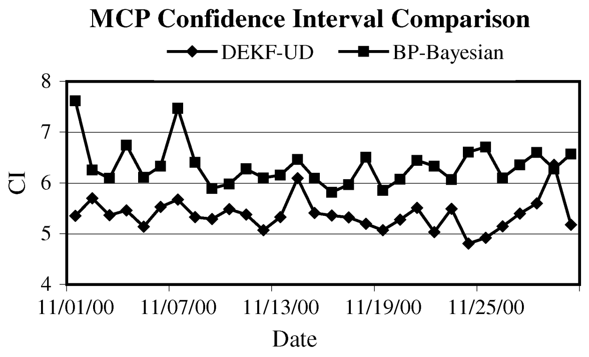

Fig. 8. MCP CIs comparison, one-sigma CI of DEKF-UD smaller than that of BP-Bayesian.

TABLE IV
PART OF THE SIGNIFICANT INPUT FACTORS

| Input Factors | The Sensitivity of MCP |
|---|---|
| AvgDayAheadLoad | 7.41 |
| MaxDayAheadLoad | 6.06 |
| DayAheadSurplus | -3.19 |
| AvgDailyMCP_TwoDaysBefore | -2.63 |
| DayAheadGasPrice | 2.13 |
| AvgMCP_TwoDaysBefore | -1.67 |
| MaxMCP_TwoDayBefore | 1.33 |
| MinDailyMCP_TwoDayBefore | 1.32 |
| MaxMCP_OneWeekBefore | -1.03 |
| MinMCP_TwoDaysBefore | 1.01 |

(Note: the MCPs and load are aggregated based on daily or on-peak period. If not specified, they are aggregated based on on-peak period. The day specified is based on the operating day (day ahead).)

(including two days before daily average MCP, on-peak average MCP, on-peak maximum MCP, daily minimum MCP, on-peak minimum MCP, and one week before maximum MCP). The day-ahead surplus has a negative correlation with MCPs since more surplus generally lowers MCPs. Other negative correlations are less easy to explain as they are related to historic MCPs with effects coupled in a complicated way.

## VI. CONCLUSION

A method with fast and accurate learning and small CIs is crucial for prediction in an uncertain and nonstationary environment. This paper investigated EKF-based neural network learning, and developed a novel method to significantly reduce computation and improve numerical stability. The resulting DEKF-UD learning is practical for neural networks with a high dimensional weight space such as MCP prediction, and provides smaller CIs than what provided by BP-Bayesian. Our DEKF-UD MCP predictions are better than the predictions obtained by using the BP-Bayesian method, and are comparable in quality to ISO New England's predictions. The method is currently being extended to predict locational marginal price using congestion information in deregulated power markets such as PJM.

## ACKNOWLEDGMENT

The authors would like to thank S. Rourke at Select Energy, D. Gionet at ISO New England, and Y. Ma at the University of

Connecticut. The authors would also like to thank the anonymous reviewers whose constructive comments contributed to improve the quality of this paper.

## References

[1] A. Khotanzad, R. Afkhami-Rohani, and D. Maratukulam, "ANNSTLF—Artificial neural network short-term load Forecaster—Generation three," *IEEE Trans. Power Syst.*, vol. 13, no. 4, pp. 1413–1422, Nov. 1998.

[2] D. W. Bunn, "Forecasting loads and prices in competitive power markets," *Proc. IEEE*, vol. 88, no. 2, pp. 163–169, Feb. 2000.

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford Univ., U.K.: Oxford Univ. Press, 1995.

[4] S. Haykin, *Neural Networks-A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[5] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 1, pp. 76–86, Jan. 1992.

[6] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[7] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*. New York: Academic, 1977.

[8] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1983.

[9] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[10] S. Singhal and L. Wu, "Training feedforward networks with the extended kalman algorithm," in *Proc. Int. Conf. ASSP*, 1989, pp. 1187–1190.

[11] G. V. Puskrius and L. A. Feldkamp, "Decoupled extended kalman filter training of feedforward layered networks," in *Proc. IEEE Int. Joint Conf. Neural Networks*, Seattle, WA, 1991, pp. 771–777.

[12] ——, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 279–297, Mar. 1994.

[13] Y. Zhang and X. Li, "A fast U-D factorization-based learning algorithm with applications to nonlinear system modeling and identification," *IEEE Trans. Neural Networks*, vol. 10, no. 4, pp. 930–938, Jul. 1999.

[14] Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques, and Software*. Norwood, MA: Artech House, 1993.

[15] L. Zhang and P. Luh, "Energy clearing price prediction and confidence interval estimation with cascaded neural networks," *IEEE Trans. Power Syst.*, vol. 18, no. 1, pp. 99–105, Feb. 2003.

[16] A. Atiya, "Bankruptcy prediction for credit risk using neural networks: A survey and new results," *IEEE Trans. Neural Networks*, vol. 12, no. 4, pp. 929–935, Jul. 2001.

[17] A. P. A. da Silva and L. S. Moulin, "Confidence intervals for neural network based short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 15, no. 4, pp. 1191–1196, Nov. 2000.

[18] C.-Y. Tsai and C.-N. Lu, "Bootstrap application in ATC estimation," *IEEE Power Eng. Rev.*, vol. 21, no. 2, pp. 40–42, Feb. 2001.

[19] G. Chryssolouris, M. Lee, and A. Ramsey, "Confidence interval prediction for neural network models," *IEEE Trans. Neural Networks*, vol. 7, no. 1, pp. 229–232, Jan. 1996.

[20] D. K. Ranaweera, G. G. Karady, and R. G. Farmer, "Effect of probabilistic inputs on neural network-based electric load forecasting," *IEEE Trans. Neural Networks*, vol. 7, no. 6, pp. 1528–1532, Nov. 1996.

[21] N. W. Townsend and L. Tarassenko, "Estimations of error bounds for neural-network function approximators," *IEEE Trans. Neural Networks*, vol. 10, no. 2, pp. 217–230, Mar. 1999.

[22] J. R. Donaldson and R. B. Schnabel, "Computational experience with confidence regions and confidence intervals for nonlinear least squares," *Technometrics*, vol. 29, no. 1, pp. 67–82, Feb. 1987.

[23] I. Rivals and L. Personnaz, "Construction of confidence intervals for neural networks based on least squares estimation," *Neural Networks*, vol. 13, pp. 463–484, 2000.

[24] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, 1992.

[25] W. A. Wright, "Bayesian approach to neural-network model with input uncertainty," *IEEE Trans. Neural Networks*, vol. 10, no. 6, pp. 1261–1270, Nov. 1999.

**Li Zhang** (S'99–M'02) received the B.S. degree in information and control engineering from Xi'an Jiaotong University, Xi'an, China, in 1992, the M.S. degree in automatic control from Shanghai Jiaotong University, Shanghai, China, in 1995, the M.Eng. degree in electrical engineering from the National University of Singapore, Singapore, in 1997, and the Ph.D. degree in electrical engineering from the University of Connecticut, Storrs, CT, in 2002.

He is currently a Market Administrator with ISO New England Inc., Holyoke, MA. He worked as an Engineer in Singapore during part of 1997, and has served as a Research and Teaching Assistant and Computer System Administrator at the University of Connecticut. His research interests include power systems, power markets, control systems and optimization, intelligent systems, and signal processing.

**Peter B. Luh** (M'80–SM'91–F'95) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1973, the M.S. degree in aeronautics and astronautics engineering from MIT, Cambridge, MA, in 1977, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, MA, in 1980.

Since 1980, he has been with the University of Connecticut, Storrs. He is currently the SNET Professor of Communications and Information Technologies with the Department of Electrical and Computer Engineering and the Director of the Taylor L. Booth Center for Computer Research and Applications. His major research interests include schedule generation and reconfiguration for manufacturing and power systems.

Dr. Luh is the Editor-in-Chief of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION. He was an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.