

# A recursive algorithm for nonlinear least-squares problems

A. Alessandri · M. Cuneo · S. Pagnan ·  
M. Sanguineti

Published online: 23 May 2007  
© Springer Science+Business Media, LLC 2007

**Abstract** The solution of nonlinear least-squares problems is investigated. The asymptotic behavior is studied and conditions for convergence are derived. To deal with such problems in a recursive and efficient way, it is proposed an algorithm that is based on a modified extended Kalman filter (MEKF). The error of the MEKF algorithm is proved to be exponentially bounded. Batch and iterated versions of the algorithm are given, too. As an application, the algorithm is used to optimize the parameters in certain nonlinear input–output mappings. Simulation results on interpolation of real data and prediction of chaotic time series are shown.

---

A. Alessandri and M. Cuneo were partially supported by the EU and the Regione Liguria through the Regional Programmes of Innovative Action (PRAI) of the European Regional Development Fund (ERDF). M. Sanguineti was partially supported by a grant from the PRIN project ‘New Techniques for the Identification and Adaptive Control of Industrial Systems’ of the Italian Ministry of University and Research.

---

A. Alessandri  
Department of Production Engineering, Thermoenergetics, and Mathematical Models (DIPTM), University of Genoa, P.le Kennedy Pad. D, 16129 Genova, Italy  
e-mail: alessandri@diptem.unige.it

M. Cuneo · S. Pagnan  
Institute of Intelligent Systems for Automation, ISSIA-CNR National Research Council of Italy,  
Via De Marini 6, 16149 Genova, Italy

M. Cuneo  
e-mail: marta@ge.issia.cnr.it

S. Pagnan  
e-mail: pagnan@ge.issia.cnr.it

M. Sanguineti (✉)  
Department of Communications, Computer and System Sciences (DIST), University of Genoa,  
Via Opera Pia 13, 16145 Genova, Italy  
e-mail: marcello@dist.unige.it

**Keywords** Nonlinear programming · Nonlinear least squares · Extended Kalman filter · Recursive optimization · Batch algorithms

## 1 Introduction

Recursive nonlinear least-squares algorithms have gained a lot of attention for their extensive use in a number of different research areas [1, 2]. The investigations were aimed at both deriving convergence results and improving the algorithmic efficiency. Connections were established between nonlinear least-squares methods and various techniques based on the Extended Kalman Filter (EKF) [3–5]. The difficulties in attaining convergence results in nonlinear least squares are well known in statistics, and most of the available results require strong assumptions on the distribution of regression errors (see, e.g., [6] and the references therein).

In this paper, we present a recursive algorithm for nonlinear least-squares problems. The algorithm is based on the EKF and, along the lines of results on EKF-based estimation for nonlinear systems [7], we prove that the algorithm's estimation error is exponentially bounded. In contrast to the stochastic analysis made in [7], we study convergence conditions in a deterministic context. This allows us to avoid assumptions on the statistics of the processes, which are difficult to verify in most nonlinear least-squares problems. As the proposed algorithm is a slight modification of the standard EKF algorithm, we have called it “modified extended Kalman filter (MEKF).” In [3] the focus is on the possibility of taking advantage of the EKF algorithm when processing data in blocks via a nonlinear least-squares approach. Taking the hint from this idea, we present also two extensions of the MEKF algorithm, aimed at dealing with large data sets in a batch way and by repeated iterations.

The batch and iterated extensions are crucial in applications such as machine learning, where large amounts of data have to be processed (see, e.g., [8] and the references therein). In neural networks learning, for example, after the appearance of backpropagation (BP) [9, 10], various methods have been proposed to optimize the neural network parameters by performing recursive optimization (i.e., using at each step only the data that become available at that step). Such approaches are well-suited to considering many data sets on line, but may suffer from poor performances. As compared with these techniques, the EKF provides a nice framework to perform batch optimization (i.e., using one data block at a time), with advantages over BP [3]. The good performances obtained by EKF-based learning algorithms may be ascribed to the information available from the covariance matrix, which, however, turns out to be quite demanding from the computational point of view. For this reason, we have devoted much attention to an efficient coding of our MEKF algorithm. Numerical results show that our algorithm performs very satisfactorily when applied to machine learning by neural networks.

The paper is organized as follows. The MEKF algorithm for solving nonlinear least-squares problems is presented in Sect. 2, together with the analysis of its properties. The batch and iterated versions, called BMEKF and IBMEKF, respectively, are described in Sect. 3. Section 4 is focused on the application of the proposed algorithms to the optimization of parameters in neural networks, and reports simulation results on two test-beds (namely, interpolation of real data and chaotic time-series prediction). Some conclusions are drawn in Sect. 5.

## 2 Statement and analysis of the algorithm

The following definitions and notations will be used throughout the paper. Let  $n, m$  be positive integers. For a real vector  $\underline{x} \in \mathbb{R}^n$ , we denote by  $\|\underline{x}\|$  the Euclidean norm of  $\underline{x}$ , i.e.,  $\|\underline{x}\| = \sqrt{x_1^2 + \dots + x_n^2}$ . For a symmetric matrix  $S \in \mathbb{R}^{n \times n}$ , we denote by  $\lambda_{\min}(S)$  and  $\lambda_{\max}(S)$  its minimum and maximum eigenvalue, respectively. Given a matrix  $M \in \mathbb{R}^{n \times m}$ , we denote by  $\|M\|$  the norm of  $M$  defined as  $\|M\| = \sqrt{\lambda_{\max}(MM^T)} = \sqrt{\lambda_{\max}(M^T M)}$ ; for a symmetric positive-definite matrix  $S$ ,  $\|S\| = \lambda_{\max}(S)$  (see [11]). Given two symmetric matrices  $S_1$  and  $S_2$ ,  $S_1 > S_2$  ( $S_1 \geq S_2$ ) means that the matrix  $S_1 - S_2$  is positive definite (semidefinite). Sequences (of vectors, matrices, etc.) are shortly denoted by using curly brackets, e.g.,  $\{s_i\}$ .

Consider a set  $\{(\underline{x}_t, \underline{y}_t), t = 0, 1, \dots\}$  of data, where  $\underline{x}_t \in X \subset \mathbb{R}^m$  and  $\underline{y}_t \in Y \subset \mathbb{R}^p$ , where  $X$  and  $Y$  are compact sets. Assume that each input–output pair is randomly generated via an unknown continuous function  $\underline{f} : \mathbb{R}^m \rightarrow \mathbb{R}^p$ , i.e.,  $\underline{y}_t = \underline{f}(\underline{x}_t)$ . A function  $\underline{\gamma}(\underline{w}, \underline{x})$ ,  $\underline{\gamma} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$ , where  $\underline{w} \in \mathbb{R}^n$  is a parameter vector, can be used to interpolate the input–output pairs by solving the following nonlinear least-squares (NLS) problem.

**Problem NLS<sub>t</sub>** Given a sequence  $\{R_t\}$  of positive definite matrices and a data set  $\{(\underline{x}_i, \underline{y}_i), i = 0, 1, \dots\}$ , find  $\underline{w}_t \in \mathbb{R}^n$  that minimizes  $J_t(\underline{w}) = \frac{1}{t+1} \sum_{i=0}^t \|\underline{y}_i - \underline{\gamma}(\underline{w}, \underline{x}_i)\|_{R_i}^2$ .

Problem NLS<sub>t</sub> is a nonlinear programming problem, and various techniques are available to solve it for each  $t = 0, 1, \dots$  [2].

It is quite common to have the need of solving Problem NLS<sub>t</sub> recursively (i.e., finding at each step  $t$  the solution  $\underline{w}_{t+1}^\circ$  of Problem NLS<sub>t+1</sub> using only  $\underline{w}_t^\circ$  and the new datum  $(u_{t+1}, y_{t+1})$ ). This led to a number of methods that obtained a wide diffusion in various applications. Among such approaches, we focus on recursive Extended Kalman Filter (EKF) methods [3, 5]. The use of a Kalman-like algorithm is suggested by the following observation: roughly speaking, the cost  $J_t(\underline{w})$  is related to the expected value of the regression error in Problem NLS<sub>t</sub>. Moreover, the linear version of Problem NLS<sub>t</sub> can be efficiently solved by using a Kalman filter [12]. Taking the hint from this, we consider the following recursive algorithm.

**Modified EKF (MEKF) Algorithm** For  $t = 0, 1, \dots$ ,

$$\hat{\underline{w}}_{t+1} = \hat{\underline{w}}_t + K_t[\underline{y}_t - \underline{\gamma}(\hat{\underline{w}}_t, \underline{x}_t)], \tag{1}$$

where

$$H_t = \left. \frac{\partial \underline{\gamma}(\underline{w}, \underline{x})}{\partial \underline{w}} \right|_{\underline{w}=\hat{\underline{w}}_t, \underline{x}=\underline{x}_t}, \tag{2}$$

$$K_t = P_t H_t^T (H_t P_t H_t^T + R_t)^{-1}, \tag{3}$$

$$P_{t+1} = (\alpha + 1)(P_t - K_t H_t P_t + \epsilon I), \tag{4}$$

$\epsilon > 0, \alpha > 0, P_0$  and  $R_t$  are symmetric positive definite matrices, and (1) is initialized with a given  $\hat{w}_0$ .

For each  $t = 0, 1, \dots, \hat{w}_t$  plays the role of an estimate of the (unknown) solution  $\hat{w}_t^\circ$  to Problem NLS<sub>t</sub>.

The main difference between the standard EKF algorithm and the MEKF algorithm consists in Eq. 4, which for the EKF is a Riccati equation

$$P_{t+1} = P_t - K_t H_t P_t + Q_t,$$

where each matrix of the sequence  $\{Q_t\}$  is positive semidefinite. In our case, this choice is not admissible because of technical reasons arising in the proof of the results given later on (see Lemma 3 in Appendix). For simplicity, we take  $Q_t = \epsilon I$ , but the only requirement is the choice of a positive-definite matrix  $Q_t$ . In a stochastic interpretation of the regression problem, the role of the factor  $\alpha + 1$  consists in increasing the value of the covariance.

In order to investigate the convergence properties of the MEKF algorithm, we first study properties of the solution of Problem NLS<sub>t</sub>. Toward this end, we make the following assumptions.

**Assumption 1** For  $t = 0, 1, \dots$ , Problem NLS<sub>t</sub> has a solution  $\underline{w}_t^\circ \in W \subset \mathbb{R}^n$ .

By  $C(X, Y)$  we denote the normed linear space of continuous functions defined on  $X \subset \mathbb{R}^m$  and with values in  $Y \subset \mathbb{R}^p$ , equipped with the supremum norm.

**Assumption 2** The set of functions  $\Gamma = \{\underline{\gamma}(\underline{w}, \cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^p, \underline{w} \in W\}$  is dense in  $C(X, Y)$ .

Assumption 2 corresponds to assuming that for every desired accuracy  $\epsilon$ , there exists a vector  $\underline{w}^* \in W$  such that the function  $\underline{\gamma}(\underline{w}^*, \cdot)$  approximates within  $\epsilon$  in the supremum norm the unknown continuous mapping  $\underline{f}$  generating the data, i.e., for every  $(\underline{x}, \underline{y}) \in X \times Y$  one has  $\|\underline{\gamma}(\underline{w}^*, \underline{x}) - \underline{y}\| \leq \epsilon$ . In the neural-network parlance this is called the *universal approximation property*, which is satisfied by a large variety of neural mappings  $\underline{\gamma}$  [13, 14].

**Assumption 3** The mapping  $\underline{\gamma}$  satisfies the following conditions:

- (i) for every  $\underline{w} \in W$ , the function  $\underline{\gamma}(\underline{w}, \cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^p$  is continuously differentiable;
- (ii) for every  $\underline{x} \in X$ , the function  $\underline{\gamma}(\cdot, \underline{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^p$  is Lipschitz, i.e., there exists  $L > 0$  such that for all  $\underline{w}, \underline{w}' \in W$ ,  $\|\underline{\gamma}(\underline{w}, \underline{x}) - \underline{\gamma}(\underline{w}', \underline{x})\| \leq L\|\underline{w} - \underline{w}'\|$ .

**Assumption 4** The cost function  $J_t : W \rightarrow \mathbb{R}$  is twice continuously differentiable and there exists  $l_{\min} > 0$  such that for  $t = 0, 1, \dots$  one has

$$\lambda_{\min} \left( \int_0^1 H_{J_t}(s\underline{w}_t^\circ + (1-s)\underline{w}_{t+1}^\circ) ds \right) \geq l_{\min},$$

where  $H_{J_t} : W \rightarrow \mathbb{R}^n \times \mathbb{R}^n$  is the Hessian of  $J_t$ .

Now we are ready to prove the following theorem.

**Theorem 1** *Suppose that Assumptions 1, 2, 3, and 4 hold. Then*

$$\underline{w}_{t+1}^\circ = \underline{w}_t^\circ + \underline{\varphi}_t, \quad t = 0, 1, \dots, \tag{5}$$

where  $\underline{\varphi}_t \triangleq (\int_0^1 H_{J_t}(s\underline{w}_{t+1}^\circ + (1-s)\underline{w}_t^\circ)ds)^{-1} \nabla J_t(\underline{w}_{t+1}^\circ)$ ,  $\lim_{t \rightarrow +\infty} \|\underline{\varphi}_t\| = 0$ ,  $\sup_{t \geq 0} \|\underline{\varphi}_t\| < \infty$ , and there exists  $\underline{w}^\circ \in \mathbb{R}^n$  such that  $\lim_{t \rightarrow +\infty} \underline{w}_t^\circ = \underline{w}^\circ$ .

*Proof* By the definition of the cost, we have

$$J_{t+1}(\underline{w}) = \frac{t+1}{t+2} J_t(\underline{w}) + \frac{1}{t+2} \|\underline{y}_{t+1} - \underline{\gamma}(\underline{w}, \underline{x}_{t+1})\|_{R_{t+1}}^2.$$

Hence

$$\nabla J_{t+1}(\underline{w}) = \frac{t+1}{t+2} \nabla J_t(\underline{w}) + \frac{2}{t+2} R_{t+1} \nabla_{\underline{w}} \underline{\gamma}(\underline{w}, \underline{x}_{t+1}) [\underline{y}_{t+1} - \underline{\gamma}(\underline{w}, \underline{x}_{t+1})],$$

where the gradients  $\nabla J_t(\underline{w})$  and  $\nabla_{\underline{w}} \gamma_i(\underline{w}, \underline{x})$ ,  $i = 1, 2, \dots, p$ , are considered as column vectors. For  $\underline{w} = \underline{w}_{t+1}^\circ$ , we have  $\nabla J_{t+1}(\underline{w}_{t+1}^\circ) = \underline{0}$  as a necessary optimality condition, so the previous equation gives

$$\nabla J_t(\underline{w}_{t+1}^\circ) = -\frac{2}{t+1} R_{t+1} \nabla_{\underline{w}} \underline{\gamma}(\underline{w}_{t+1}^\circ, \underline{x}_{t+1}) [\underline{y}_{t+1} - \underline{\gamma}(\underline{w}_{t+1}^\circ, \underline{x}_{t+1})].$$

By the properties of the matrix norms [11], we obtain

$$\|\nabla J_t(\underline{w}_{t+1}^\circ)\| \leq \frac{2}{t+1} \|R_{t+1}\| \|\nabla_{\underline{w}} \underline{\gamma}(\underline{w}_{t+1}^\circ, \underline{x}_{t+1})\| \|\underline{y}_{t+1} - \underline{\gamma}(\underline{w}_{t+1}^\circ, \underline{x}_{t+1})\|.$$

Take  $\varepsilon > 0$ . By Assumption 2, there exists  $\underline{w}^* \in W$  such that  $\|\underline{y}_{t+1} - \underline{\gamma}(\underline{w}^*, \underline{x}_{t+1})\| = \|\underline{f}(\underline{x}_{t+1}) - \underline{\gamma}(\underline{w}^*, \underline{x}_{t+1})\| \leq \varepsilon$ . Thus, by the triangle inequality and Assumption 3(i), we obtain  $\|\underline{y}_{t+1} - \underline{\gamma}(\underline{w}^*, \underline{x}_{t+1})\| = \varepsilon + L \|\underline{w}^* - \underline{w}_{t+1}^\circ\| \leq \varepsilon + Lr_W$ , where  $r_W$  is the radius of the set  $W$  and  $L$  is the Lipschitz constant of  $\underline{\gamma}(\cdot, \underline{x}_t)$ . Hence,

$$\|\nabla J_t(\underline{w}_{t+1}^\circ)\| \leq \frac{2}{t+1} \|R_{t+1}\| \|\nabla_{\underline{w}} \underline{\gamma}(\underline{w}_{t+1}^\circ, \underline{x}_{t+1})\| (\varepsilon + r_W). \tag{6}$$

Since each matrix of the sequence  $\{R_t\}$  is positive definite and  $\|\nabla_{\underline{w}} \underline{\gamma}(\underline{w}_{t+1}^\circ, \underline{x}_{t+1})\|$  admits a maximum over the compact set  $W$  by Assumption 3(ii), inequality Eq. 6 implies

$$\lim_{t \rightarrow +\infty} \|\nabla J_t(\underline{w}_{t+1}^\circ)\| = 0. \tag{7}$$

Let  $k : \mathbb{R} \rightarrow \mathbb{R}^n$  be defined as  $k(s) \triangleq \nabla J_t[s\underline{w}_{t+1}^\circ + (1-s)\underline{w}_t^\circ]$ , with derivative  $k'$ . By the Mean-Value Theorem, we have  $k(1) - k(0) = \int_0^1 k'(s)ds$  and so

$$\nabla J_t(\underline{w}_{t+1}^\circ) - \nabla J_t(\underline{w}_t^\circ) = \int_0^1 H_{J_t}[s\underline{w}_{t+1}^\circ + (1-s)\underline{w}_t^\circ](\underline{w}_{t+1}^\circ - \underline{w}_t^\circ)ds. \tag{8}$$

As  $\nabla J_t(\underline{w}_t^\circ) = 0$ , by Assumption 4 the equality Eq. 8 gives

$$\underline{w}_{t+1}^\circ = \underline{w}_t^\circ + \left[ \int_0^1 H_{J_t}[s\underline{w}_{t+1}^\circ + (1-s)\underline{w}_t^\circ] ds \right]^{-1} \nabla J_t(\underline{w}_{t+1}^\circ).$$

By Assumption 4, let  $\varphi_t \triangleq (\int_0^1 H_{J_t}[s\underline{w}_{t+1}^\circ + (1-s)\underline{w}_t^\circ] ds)^{-1} \nabla J_t(\underline{w}_{t+1}^\circ)$ , so  $\underline{w}_{t+1}^\circ = \underline{w}_t^\circ + \varphi_t$ . By Eq. 7 and Assumption 4,  $\lim_{t \rightarrow +\infty} \|\varphi_t\| = 0$ . So the sequence  $\{\underline{w}_t^\circ\}$  is Cauchy in  $\mathbb{R}^n$ , hence it converges to some  $\underline{w}^\circ \in \mathbb{R}^n$ .  $\square$

So far, we have investigated the asymptotic behavior of Problem NLS<sub>t</sub>. In the following, we shall study how close  $\hat{\underline{w}}_t$  is to  $\underline{w}_t^\circ$ . Toward this end, we need some additional assumptions.

We denote by  $\eta_t = \underline{y}_t - \underline{\gamma}(\underline{w}_t^\circ, \underline{x}_t)$  the error made in approximating with the value  $\underline{\gamma}(\underline{w}_t^\circ, \underline{x}_t)$  the output  $\underline{y}_t$  associated with the input  $\underline{x}_t$ . The vector  $\eta_t$  is bounded. Indeed, by Assumption 2, for every  $\varepsilon > 0$  there exists  $\underline{w}^* \in W$  such that  $\|\underline{y}_t - \underline{\gamma}(\underline{w}^*, \underline{x}_t)\| < \varepsilon$ . Hence  $\|\eta_t\| = \|\underline{y}_t - \underline{\gamma}(\underline{w}_t^\circ, \underline{x}_t)\| \leq \|\underline{y}_t - \underline{\gamma}(\underline{w}^*, \underline{x}_t)\| + \|\underline{\gamma}(\underline{w}^*, \underline{x}_t) - \underline{\gamma}(\underline{w}_t^\circ, \underline{x}_t)\| < \varepsilon + Lr_W$ , where  $L$  is the Lipschitz constant of  $\underline{\gamma}(\cdot, \underline{x}_t)$  and  $r_W$  is the radius of the set  $W$ .

**Assumption 5** There exist positive constants  $\eta_{\max}, h_{\max}, p_{\min}, p_{\max}$ , and  $r_{\min}$  such that for  $t = 0, 1, \dots$  one has:

- (i)  $\eta_t \eta_t^T \leq \eta_{\max} I$ ;
- (ii)  $\|H_t\| \leq h_{\max}$ ;
- (iii)  $p_{\min} I \leq P_t \leq p_{\max} I$ ;
- (iv)  $R_t \geq r_{\min} I$ .

**Assumption 6** For  $t = 0, 1, \dots$ , let  $\Lambda_t \triangleq \int_0^1 \frac{\partial \underline{\gamma}}{\partial \underline{w}}[s\underline{w}_t^\circ + (1-s)\hat{\underline{w}}_t, \underline{x}_t] ds$ ,  $F_t = (I - K_t \Lambda_t)^T P_{t+1}^{-1} (I - K_t \Lambda_t)$ , and  $G_t = (I - K_t H_t)^T P_{t+1}^{-1} (I - K_t H_t)$ . Then  $\lambda_{\max}(F_t) \leq \lambda_{\min}(G_t)$ .

Finally, we introduce one definition.

**Definition 1** A sequence  $\{v_t\}$  is exponentially bounded if there exist  $c_0, c_2 > 0$  and  $c_1 \in (0, 1)$  such that  $\|\underline{v}_t\|^2 \leq c_0 \|\underline{v}_0\|^2 c_1^t + c_2, t = 0, 1, \dots$

The next theorem states the boundedness of the error  $\underline{e}_t \triangleq \underline{w}_t^\circ - \hat{\underline{w}}_t$  between the solution  $\underline{w}_t^\circ$  of Problem NLS<sub>t</sub> and the vector  $\hat{\underline{w}}_t$  obtained by the MEKF algorithm.

**Theorem 2** Let  $\underline{w}_t^\circ$  be the solution of Problem NLS<sub>t</sub> and  $\hat{\underline{w}}_t$  be given by the MEKF algorithm (see Eqs. 1–4). If Assumptions 1–6 hold, then the sequence  $\{\underline{e}_t\}$  of the estimation errors is exponentially bounded.

*Proof* By Eqs. 1 and 5, the error sequence is given by

$$\underline{e}_{t+1} = \underline{e}_t - K_t(\underline{y}_t - \underline{\gamma}(\hat{\underline{w}}_t, \underline{x}_t)) + \varphi_t. \tag{9}$$

As  $\underline{y}_t = \underline{\gamma}(\underline{w}_t^\circ, \underline{x}_t) + \underline{\eta}_t$ , by the Mean-Value Theorem we obtain

$$\underline{y}_t = \underline{\gamma}(\hat{\underline{w}}_t, \underline{x}_t) + \Lambda_t(\underline{w}_t^\circ - \hat{\underline{w}}_t) + \underline{\eta}_t. \tag{10}$$

Therefore, Eq. 9 yields

$$\underline{e}_{t+1} = (I - K_t \Lambda_t) \underline{e}_t - K_t \underline{\eta}_t + \underline{\varphi}_t.$$

Let  $\Pi_t = P_t^{-1}$ ,  $t = 0, 1, \dots$ . Assumption 5(iii) enables us to introduce the Lyapunov function  $V_t(\underline{e}_t) = \underline{e}_t^T \Pi_t \underline{e}_t$ . By a little algebra, we obtain

$$\begin{aligned} V_{t+1}(\underline{e}_{t+1}) &= \underline{e}_t^T (I - K_t \Lambda_t)^T \Pi_{t+1} (I - K_t \Lambda_t) \underline{e}_t - 2(I - K_t \Lambda_t \underline{e}_t)^T \Pi_{t+1} K_t \underline{\eta}_t \\ &\quad + \underline{\eta}_t^T K_t^T \Pi_{t+1} K_t \underline{\eta}_t - \underline{\varphi}_t^T \Pi_{t+1} K_t \underline{\eta}_t \\ &\quad + 2(I - K_t \Lambda_t \underline{e}_t)^T \Pi_{t+1} \underline{\varphi}_t + \underline{\varphi}_t^T \Pi_{t+1} \underline{\varphi}_t. \end{aligned} \tag{11}$$

By using the upper bounds of Lemma 1 (see the Appendix) with  $a = b = \alpha/2$ , from Eq. 11 we have

$$\begin{aligned} V_{t+1}(\underline{e}_{t+1}) &\leq (1 + \alpha) \underline{e}_t^T (I - K_t \Lambda_t)^T \Pi_{t+1} (I - K_t \Lambda_t) \underline{e}_t \\ &\quad + \left(2 + \frac{2}{\alpha}\right) \underline{\eta}_t^T K_t^T \Pi_{t+1} K_t \underline{\eta}_t + \left(2 + \frac{2}{\alpha}\right) \underline{\varphi}_t^T \Pi_{t+1} \underline{\varphi}_t \end{aligned}$$

and, by Lemma 3 (see the Appendix), for  $\beta = \frac{k_2}{1+k_2}$  and  $k_2 = \epsilon [p_{\max}(1 + \frac{p_{\max} h_{\max}^2}{r_{\min}})^2]^{-1}$ , we get

$$\begin{aligned} V_{t+1}(\underline{e}_{t+1}) &\leq (1 - \beta) \underline{e}_t^T \Pi_t \underline{e}_t + \left(2 + \frac{2}{\alpha}\right) \underline{\eta}_t^T K_t^T \Pi_{t+1} K_t \underline{\eta}_t \\ &\quad + \left(2 + \frac{2}{\alpha}\right) \underline{\varphi}_t^T \Pi_{t+1} \underline{\varphi}_t. \end{aligned} \tag{12}$$

As  $\Pi_t = P_t^{-1}$ , by Lemma 2 (see the Appendix) and Assumption 5(iii) we have

$$\begin{aligned} \underline{\eta}_t^T K_t^T \Pi_{t+1} K_t \underline{\eta}_t &\leq \frac{1}{p_{\min}} \underline{\eta}_t^T K_t^T K_t \underline{\eta}_t \leq \frac{1}{p_{\min}} \left(\frac{p_{\max} h_{\max}}{\rho_{\min}}\right)^2 \underline{\eta}_t^T \underline{\eta}_t \\ &\leq \frac{\eta_{\max}}{p_{\min}} \left(\frac{p_{\max} h_{\max}}{r_{\min}}\right)^2 \end{aligned} \tag{13}$$

and

$$\underline{\varphi}_t^T \Pi_{t+1} \underline{\varphi}_t \leq \frac{\varphi_{\sup}^2}{p_{\min}}, \tag{14}$$

where  $\varphi_{\sup} \triangleq \sup_t \|\varphi_t\| < \infty$  as  $\|\varphi_t\| \leq \|[\int_0^1 H_{J_t}(s \underline{w}_{t+1}^\circ + (1-s) \underline{w}_t^\circ) ds]^{-1}\| \times \|\nabla J_t(\underline{w}_{t+1}^\circ)\|$  (the first term is bounded by Assumption 4, and the second one by the inequality Eq. 6).

To sum up, Eqs. 12, 13, and 14 give

$$V_{t+1}(\underline{e}_{t+1}) \leq (1 - \beta)V_t(\underline{e}_t) + k_4, \tag{15}$$

where

$$k_4 \triangleq \frac{\eta_{\max}}{p_{\min}} \left( 2 + \frac{2}{\alpha} \right) \left( \frac{p_{\max} h_{\max}}{r_{\min}} \right)^2 + \left( 2 + \frac{2}{\alpha} \right) \frac{\varphi_{\sup}^2}{p_{\min}}. \tag{16}$$

By applying  $t$  times the inequality Eq. 15, we get  $V_t(\underline{e}_t) \leq (1 - \beta)^t V_0(\underline{e}_0) + k_4 \sum_{i=0}^{t-1} (1 - \beta)^i$ . As  $V_t(\underline{e}_t) = \underline{e}_t^T \Pi_t \underline{e}_t$ , by Assumption 5(iii) we get

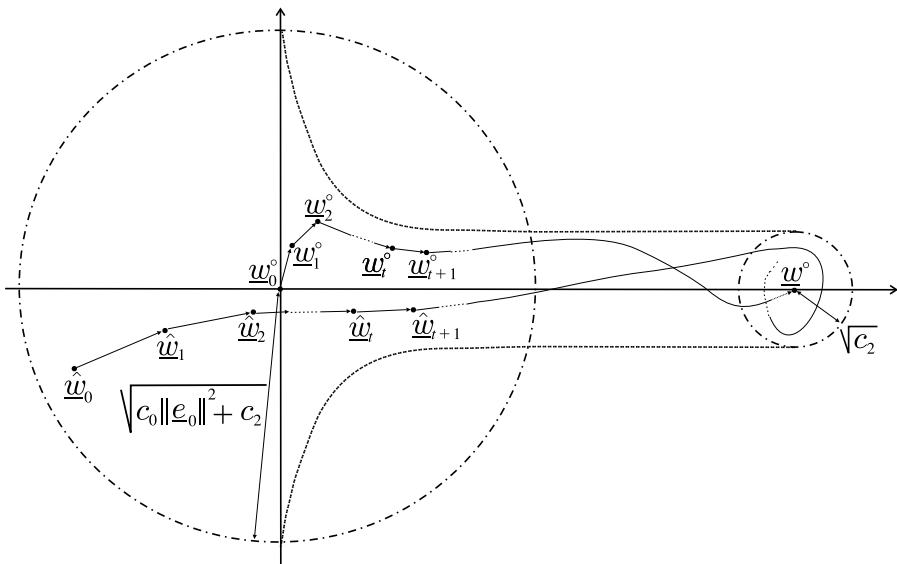
$$\|\underline{e}_t\|^2 \leq \frac{p_{\max}}{p_{\min}} \|\underline{e}_0\|^2 (1 - \beta)^t + \frac{k_4}{p_{\min}} \sum_{i=0}^{t-1} (1 - \beta)^i. \tag{17}$$

Since  $\beta \in (0, 1)$ ,  $\sum_{i=0}^{\infty} (1 - \beta)^i = 1/\beta$  and so, as all the terms of the series are positive,  $\sum_{i=0}^{t-1} (1 - \beta)^i \leq 1/\beta$  for all  $t$ . Thus, Eq. 17 implies

$$\|\underline{e}_t\|^2 \leq \frac{p_{\max}}{p_{\min}} \|\underline{e}_0\|^2 (1 - \beta)^t + \frac{k_4}{\beta p_{\min}},$$

which proves that the estimation error is exponentially bounded with  $c_0 = \frac{p_{\max}}{p_{\min}}$ ,  $c_1 = 1 - \beta$ , and  $c_2 = \frac{k_4}{\beta p_{\min}}$  (see the Definition 1). □

Theorems 1 and 2 allow one to figure out how the estimate  $\hat{w}_t$  provided by the MEKF algorithm behaves in comparison with the optimal solution; this is pictorially described in Fig. 1.



**Fig. 1** Qualitative behavior of the sequences  $\{w_t^o\}$  and  $\{\hat{w}_t\}$ , on the basis of Theorems 1 and 2

Some comparisons between Theorem 2 and [7, Theorem 3.1, p. 715] have to be made. A major point consists in the fact that we regard  $\{\underline{e}_t\}$  and  $\{\eta_t\}$  as sequences of deterministic variables instead of stochastic variables as in [7]. Note also that, in a stochastic context, Assumption 5(i) implies that the covariance of the regression error is finite.

### 3 Batch and iterated versions: the BMEKF and IBMEKF algorithms

Batch training enables one to deal more efficiently with large data sets, by dividing the patterns into data batches of fixed length  $N$  and by applying algorithms to one batch at a time [15]. In Fig. 2, the data batch is shifted by  $d$  data at each step, where  $d \leq N$  is a positive integer. The MEKF algorithm corresponds to  $d = N$ .

Given a fixed number of iterations, say  $t$ , if  $d = 1$ , then  $t = 0, 1, \dots, N + t$  input-output data are explored; if  $d = N$ , then  $N(t + 1)$  data are explored. In general, the amount of data processed at step  $t$  is equal to  $N + dt$  (see Fig. 3).

We also define

$$\underline{G}(\underline{w}, \underline{x}_{t-N}^{t-1}) \triangleq \begin{bmatrix} \gamma(\underline{w}, \underline{x}_{t-N}) \\ \gamma(\underline{w}, \underline{x}_{t-N+1}) \\ \vdots \\ \gamma(\underline{w}, \underline{x}_{t-1}) \end{bmatrix} \in \mathbb{R}^{PN}, \tag{18}$$

where  $\underline{x}_{t-N}^{t-1} \triangleq \text{col}(\underline{x}_{t-N}, \underline{x}_{t-N+1}, \dots, \underline{x}_{t-1}) \in X^N$  (recall that  $\underline{x}_t \in X \subset \mathbb{R}^m$ ). Similarly, let  $\underline{y}_{t-N}^{t-1} \triangleq \text{col}(\underline{y}_{t-N}, \underline{y}_{t-N+1}, \dots, \underline{y}_{t-1}) \in Y^N$  (recall that  $\underline{y}_t \in Y \subset \mathbb{R}^p$ ).

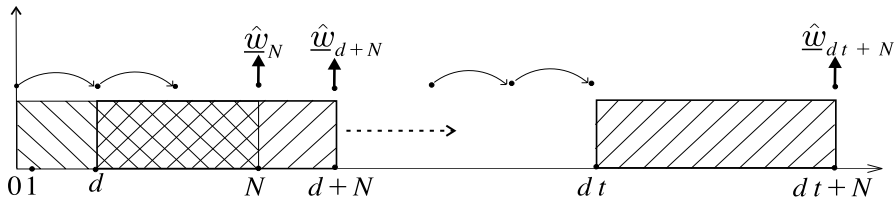


Fig. 2 The  $d$ -step batch-mode optimization

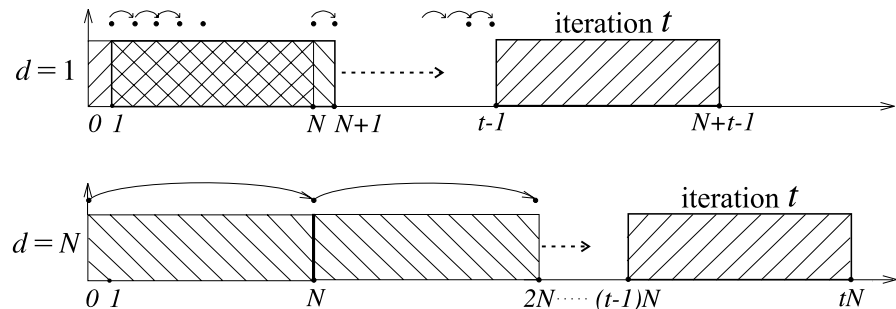


Fig. 3 Comparison between the one-step and the  $N$ -step batch-mode optimizations

The extension of the MEKF algorithm to the  $d$ -step framework can be expressed as follows.

**Batch-mode MEKF (BMEKF) Algorithm** For  $t = N - 1, N, \dots$ :

$$\hat{\underline{w}}_{t+1} = \hat{\underline{w}}_t + \mathcal{K}_t \left[ \underline{y}_{d(t-N+1)}^{d(t-N+1)+N} - \underline{G}(\hat{\underline{w}}_{t-1}, \underline{x}_{d(t-N+1)}^{d(t-N+1)+N}) \right], \tag{19}$$

where

$$\mathcal{H}_t = \left. \frac{\partial \underline{G}(\underline{w}, \underline{x})}{\partial \underline{w}} \right|_{\underline{w}=\hat{\underline{w}}_t, \underline{x}=\underline{x}_{d(t-N+1)}^{d(t-N+1)+N}}, \tag{20}$$

$$\mathcal{K}_t = \mathcal{P}_t \mathcal{H}_t^T (\mathcal{H}_t \mathcal{P}_t \mathcal{H}_t^T + \mathcal{R}_t)^{-1}, \tag{21}$$

$$\mathcal{P}_{t+1} = (\alpha + 1)(\mathcal{P}_t - \mathcal{K}_t \mathcal{H}_t \mathcal{P}_t + \epsilon I), \tag{22}$$

$\epsilon > 0$ ,  $\alpha > 0$ ,  $\mathcal{P}_0$  and  $\mathcal{R}_t$  are symmetric positive definite matrices, and Eq. 19 is initialized with a given  $\hat{\underline{w}}_{N-1}$ .

Note that if  $d$  and  $N$  are both taken equal to 1, then the BMEKF algorithm corresponds to the MEKF algorithm. It is important to remark that the convergence properties of MEKF (see Sect. 2) apply to BMEKF, too. In the latter, however, at each time  $t$ ,  $N$  input–output pairs are processed instead of one, as in the MEKF algorithm, thus more computations are involved: the matrix to be inverted is of dimension  $(pN)^2$  instead of  $p^2$  (see Eq. 21). Since in BMEKF one has to deal with larger matrices, efficient coding is crucial.

A generalization of the BMEKF algorithm consists in repeating the estimate and covariance updates by using the same batch of input–output patterns. By following the terminology used in [16], the repetitions are called *epochs* and the corresponding algorithm is called Iterated Batch-mode Modified EKF (IBMEKF) algorithm, which is described below.

**Iterated Batch-mode MEKF (IBMEKF) Algorithm** For  $t = N - 1, N, \dots$ :

$$\tilde{\underline{w}}_1 = \hat{\underline{w}}_t$$

for  $i = 1, 2, \dots, N_E$

$$\mathcal{H}_i = \left. \frac{\partial \underline{G}(\underline{w}, \underline{x})}{\partial \underline{w}} \right|_{\underline{w}=\tilde{\underline{w}}_i, \underline{x}=\underline{x}_{d(t-N+1)}^{d(t-N+1)+N}}$$

$$\mathcal{K}_i = \mathcal{P}_i \mathcal{H}_i^T (\mathcal{H}_i \mathcal{P}_i \mathcal{H}_i^T + \mathcal{R}_i)^{-1}$$

$$\mathcal{P}_{i+1} = (\alpha + 1)(\mathcal{P}_i - \mathcal{K}_i \mathcal{H}_i \mathcal{P}_i + \epsilon I)$$

$$\tilde{\underline{w}}_{i+1} = \tilde{\underline{w}}_i + \mathcal{K}_i \left[ \underline{y}_{d(t-N+1)}^{d(t-N+1)+N} - \underline{G}(\tilde{\underline{w}}_i, \underline{x}_{d(t-N+1)}^{d(t-N+1)+N}) \right]$$

end

$$\hat{\underline{w}}_{t+1} = \tilde{\underline{w}}_{N_E},$$

where  $\epsilon > 0$ ,  $\alpha > 0$ ,  $\mathcal{P}_0$ , and  $\mathcal{R}_i$  are symmetric positive definite matrices, and  $N_E$  is the number of epochs.

For  $N_E = 1$ , the IBMEKF algorithm reduces to BMEKF. Note that the convergence results of Sect. 2 apply to the IBMEKF algorithm, too. Note also that IBMEKF results from applying a slight modification of the so-called “iterated extended Kalman filter” (see, e.g., [17]) to Problem  $NLS_t$ . The possibility of using iterated Kalman filtering techniques to solve nonlinear programming problems is discussed in [3].

### 4 Application and numerical results

In this section, we apply the MEKF algorithm to the problem of optimizing the weights in nonlinear input–output mappings implemented by feedforward neural networks.

#### 4.1 Optimization of parameters in feedforward neural networks

*Feedforward neural networks* (in the following, for the sake of brevity, often called “neural networks” or simply “networks”) are nonlinear mappings composed of  $L$  layers, with  $v_s$  computational units in the layer  $s$  ( $s = 1, \dots, L$ ). The input–output mapping of the  $q$ -th unit of the  $s$ -th layer is given by

$$y_q(s) = g \left( \sum_{p=1}^{v_{s-1}} w_{pq}(s)y_p(s-1) + w_{0q}(s) \right), \quad s = 1, \dots, L; \quad q = 1, \dots, v_s, \quad (23)$$

where  $g : \mathbb{R} \rightarrow \mathbb{R}$  is called the *activation function*. The coefficients  $w_{pq}(s)$  and the so-called *biases*  $w_{0q}(s)$  are lumped together into the so-called *weights vector*  $\underline{w}^s$ . We let

$$\underline{w} \triangleq \text{col}(\underline{w}^1, \underline{w}^2, \dots, \underline{w}^L) \in W \subset \mathbb{R}^n,$$

where

$$n \triangleq \sum_{s=0}^L v_{s+1}(v_s + 1)$$

is the total number of weights,  $v_0 = m$ , and  $v_{L+1} = p$ . The function implemented by a feedforward neural network with the weights vector  $\underline{w}$  is denoted by  $\underline{\gamma}(\underline{w}, \underline{x})$ ,  $\underline{\gamma} : W \times X \rightarrow \mathbb{R}^p$ , where  $\underline{x} \in X \subset \mathbb{R}^m$  is the network input vector.

In applications, the elements of  $\underline{w}$  are optimized on the basis of a data set consisting of input–output pairs  $(\underline{x}_i, \underline{y}_i)$ ,  $i = 0, 1, \dots$ , where  $\underline{x}_i \in X \subset \mathbb{R}^m$  and  $\underline{y}_i \in Y \subset \mathbb{R}^p$  represent the input and the desired output of the network  $\underline{\gamma}$ , respectively. In the neurocomputing parlance, the process of parameters optimization is called “training” or “learning process.”

As recalled in the Introduction, BP is the most popular method for the optimization of parameters in feedforward neural networks [9]. Although BP has been successfully applied in a variety of areas, it suffers from slow convergence, which makes high-dimensional problems intractable. The slowness is to be ascribed both to the use of

the steepest-descent method, which performs poorly in terms of convergence in high-dimensional settings [18], and to the fixed, arbitrarily chosen step length. For these reasons, algorithms using also the second derivatives were developed [19] and modifications to BP were proposed (see, e.g., the acceleration technique presented in [20] and the approach described in [21], which is aimed at restoring the dependence of the learning rate on time). The determination of the search direction and of the step length by using methods of nonlinear optimization has been considered, for example, in [22].

Deeper insights can be gained by regarding the learning of feedforward neural networks as a problem of parameter estimation. If one assumes that the data are generated by a continuous function  $\underline{f} : \mathbb{R}^m \rightarrow \mathbb{R}^p$ , i.e.,  $y_t = \underline{f}(x_t)$ , then, for every  $X \subset \mathbb{R}^m$ , Assumption 2 holds, i.e., there exists a vector  $\underline{w}^* \in \bar{W} \subset \mathbb{R}^n$  such that

$$\underline{f}(x) = \underline{\gamma}(\underline{w}^*, x) + \underline{\eta}, \quad \forall x \in X, \quad (24)$$

where  $\underline{\eta} \in K \subset \mathbb{R}^p$  is the network approximation error (see, e.g., [23]). Let  $\underline{\eta}_t$  be the error made in approximating  $\underline{f}$  by the neural network implementing the mapping  $\underline{\gamma}$ , in correspondence of the input  $x_t$ . Then

$$y_t = \underline{\gamma}(\underline{w}^*, x_t) + \underline{\eta}_t, \quad t = 0, 1, \dots, \quad (25)$$

where  $\underline{\eta}_t$  is regarded as a disturbance associated with  $y_t$ . Equation 25 can be regarded as a nonlinear dynamic system; its state vector is given by the weights vector  $\underline{w}$ , which evolves according to a fictitious dynamics, i.e.,

$$\underline{w}_{t+1} = \underline{w}_t, \quad t = 0, 1, \dots \quad (26)$$

where  $\underline{w}_0 = \underline{w}^*$ . If the activation function in Eq. 23 is differentiable, then Eqs. 25 and 26 motivate using recursive state estimators for neural network training. Following this approach, training algorithms, based on the extended Kalman filter (EKF) and showing faster convergence than BP, were proposed (see, e.g., [24–29]). However, the advantages of EKF-based training are obtained at the cost of a notable computational burden (as matrix inversions are required) and of a large amount of memory. In [30] it was developed an optimization-based learning algorithm for feedforward neural networks, that copes with some limitations of BP and does not require matrix inversions.

When applying the MKEF algorithm to optimization of parameters in neural networks, the choice of  $R_t$  may be critical, as it is related to  $\underline{\eta}_t$  (the “approximation error”) and depends on the structure of the network and on the initial choice of the weights. Usually, the matrix  $R_t$  is chosen quite large and tuned on line (see, for example, [26, p. 962, formulas 34–36]). As to  $\alpha$ ,  $P_0$ , and  $\epsilon$ , too large values may cause slow convergence; by contrast, if they are taken too small, then the training process may suffer from inadequate generalization, i.e., poor capability of “approximating” new data.

In the following, we report the numerical results obtained on two benchmark least-squares problems: interpolation of real data and prediction of chaotic time series. The MEKF algorithm (*trainmekf*) was compared with nine widely-used training algorithms, available from the Matlab Neural Toolbox [16]: BFGS (Broyden-Fletcher-Goldfarb-Shanno) quasi-Newton backpropagation (*trainbfg*), Powell-Beale

conjugate gradient backpropagation (*traincgb*), Fletcher-Powell conjugate gradient backpropagation (*traincgf*), Polak-Ribière conjugate gradient (*traincgp*), gradient descent with momentum and adaptive backpropagation (*trainngdx*), Levenberg-Marquardt backpropagation (*trainlm*), one-step secant backpropagation (*trainoss*), resilient backpropagation (*trainrpf*), and scaled conjugate gradient backpropagation (*trainscg*).

The mean square error (MSE), its standard deviation, and the processing time (in s) obtained by using a “Pentium 4” 1.6 GHz computer were considered as performance indexes with the purpose of comparing the various learning algorithms. The numerical results show that the MEKF outperforms backpropagation and other widespread training algorithms.

### 4.2 Interpolation of real data: the “Building” PROBEN1 benchmark

The PROBEN1 benchmark collection [31] contains a large set of real data, which can be used for the prediction of energy consumption in a building (electrical energy, hot and cold water) on the basis of date, time of day, outside temperature, outside air humidity, solar radiation, and wind speed. The data set is composed of 2104 records (each consisting of 14 inputs and 3 outputs) for training and 1052 records for testing. The network to be trained is a feedforward neural network with a linear output layer and one hidden layer with four neurons and a sigmoidal activation function. The BMEKF algorithm was initialized with  $P_0 = 10^{-2}I$ . The parameters  $\alpha$  and  $\epsilon$  were chosen equal to  $10^{-2}$  and  $10^{-2}$ , respectively.

Table 1 summarizes the results for different choices of the window size  $N$  and  $d = N$ . The columns show the MSEs with the corresponding standard deviations and the processing times obtained by 100 trials. Different initial weights were used in each trial, uniformly randomly distributed between 0 and 1. As it can be seen from the table, the *trainmekf* algorithm performs quite well as regards the MSE evaluation of the training set, and outperforms all the other algorithms in terms of the test-set MSE. The processing time for BMEKF is much shorter than for all the other algorithms considered.

### 4.3 Prediction of chaotic time series

For an integer  $\tau \geq 1$ , the discrete-time Mackey-Glass series is given by the following delay-difference equation [32]:

$$x_{t+1} = (1 - c_1)x_t + c_2 \frac{x_{t-\tau}}{1 + (x_{t-\tau})^{10}}, \quad t = \tau, \tau + 1, \dots$$

The training data were generated using the values  $c_1 = 0.1$ ,  $c_2 = 0.2$ , and  $\tau = 36$ ; and the initial value  $x_0$  was randomly chosen. The data were arranged in sets made up of 100 series, each one with 2000 samples. The first 1000 time steps of each series were omitted. The successive 500 time steps were used for training and the remaining 500 for testing. The prediction was made by interpolating the values of  $x_{t+1}$  and  $x_{t+2}$  via the previous  $l + 1$  samples, that is,

$$(x_{t+2}, x_{t+1}) \leftarrow (x_t, x_{t-1}, x_{t-2}, \dots, x_{t-l}), \quad t = l, l + 1, \dots,$$

**Table 1** MSEs and processing times for the PROBEN1 data set, using a 4-neuron one-hidden-layer feedforward neural network

Algorithm	N = 30			N = 90			N = 150		
	Training set		Test set	Training set		Test set	Training set		Test set
	100× (error ± std)	time (s)	100× (error ± std)	100× (error ± std)	time (s)	100× (error ± std)	100× (error ± std)	time (s)	100× (error ± std)
<i>trainmekf</i>	0.82 ± 0.00	2.5	0.78 ± 0.00	0.79 ± 0.00	1.5	0.52 ± 0.00	0.75 ± 0.00	1.3	0.45 ± 0.00
<i>trainoss</i>	1.28 ± 0.01	68.9	2.40 ± 0.00	0.87 ± 0.00	24.3	1.65 ± 0.00	0.90 ± 0.00	17.8	1.82 ± 0.00
<i>trainidx</i>	1.51 ± 0.01	29.2	2.86 ± 0.00	1.14 ± 0.00	10.2	2.00 ± 0.00	1.18 ± 0.00	7.4	2.23 ± 0.00
<i>trainbfg</i>	1.75 ± 0.01	32.5	2.96 ± 0.01	3.28 ± 0.11	33.2	3.18 ± 0.08	1.36 ± 0.02	24.2	2.02 ± 0.00
<i>trainrp</i>	1.60 ± 0.01	29.1	2.98 ± 0.00	1.91 ± 0.03	10.3	2.82 ± 0.03	1.51 ± 0.01	7.4	2.38 ± 0.01
<i>trainlm</i>	2.35 ± 0.04	38.7	3.62 ± 0.03	3.54 ± 0.12	33.4	3.90 ± 0.11	5.18 ± 0.14	30.2	3.91 ± 0.11
<i>trainscg</i>	5.73 ± 0.09	55.8	8.31 ± 0.13	0.86 ± 0.00	19.6	1.53 ± 0.00	0.84 ± 0.00	14.2	1.65 ± 0.00
<i>traincgp</i>	6.98 ± 0.11	71.4	9.63 ± 0.14	0.82 ± 0.00	25.4	1.46 ± 0.00	0.84 ± 0.00	18.8	1.64 ± 0.00
<i>traincgb</i>	6.87 ± 0.12	58.4	11.30 ± 0.17	1.03 ± 0.01	26.1	1.55 ± 0.00	0.96 ± 0.01	19.0	1.69 ± 0.00
<i>traincgf</i>	13.54 ± 0.17	70.6	20.01 ± 0.21	0.98 ± 0.01	25.5	1.63 ± 0.00	0.88 ± 0.00	18.5	1.66 ± 0.00

**Table 2** MSEs and processing times for Mackey-Glass series prediction, using a 5-neuron one-hidden-layer feedforward neural network

Algorithm	N = 30			N = 90			N = 150		
	Training set	Test set		Training set	Test set		Training set	Test set	
	100× (error ± std)	100× (error ± std)	time (s)	100× (error ± std)	100× (error ± std)	time (s)	100× (error ± std)	100× (error ± std)	time (s)
<i>trainmekf</i>	0.03 ± 0.00	0.03 ± 0.00	0.3	0.01 ± 0.00	0.01 ± 0.00	0.2	0.01 ± 0.00	0.01 ± 0.00	0.2
<i>traincgf</i>	0.50 ± 0.03	0.51 ± 0.03	16.2	0.03 ± 0.00	0.04 ± 0.00	6.5	0.05 ± 0.00	0.05 ± 0.00	4.1
<i>traincgb</i>	0.15 ± 0.00	0.16 ± 0.00	30.9	0.01 ± 0.00	0.01 ± 0.00	12.8	0.01 ± 0.00	0.01 ± 0.00	8.0
<i>traincgp</i>	0.58 ± 0.01	0.60 ± 0.01	16.2	0.34 ± 0.00	0.34 ± 0.00	6.2	0.49 ± 0.00	0.49 ± 0.00	4.1
<i>trainscg</i>	0.10 ± 0.00	0.11 ± 0.00	31.1	0.01 ± 0.00	0.01 ± 0.00	13.4	0.01 ± 0.00	0.01 ± 0.00	12.3
<i>trainoss</i>	0.09 ± 0.00	0.09 ± 0.00	29.3	0.01 ± 0.00	0.01 ± 0.00	13.5	0.01 ± 0.00	0.01 ± 0.00	12.6
<i>trainrp</i>	0.11 ± 0.00	0.11 ± 0.00	29.4	0.01 ± 0.00	0.01 ± 0.00	13.7	0.01 ± 0.00	0.01 ± 0.00	12.2
<i>trainbfg</i>	0.12 ± 0.00	0.12 ± 0.00	41.2	0.02 ± 0.00	0.02 ± 0.00	15.2	0.03 ± 0.00	0.03 ± 0.00	12.2
<i>trainlm</i>	0.53 ± 0.02	0.56 ± 0.02	29.7	0.01 ± 0.00	0.01 ± 0.00	12.5	0.05 ± 0.00	0.05 ± 0.00	9.5
<i>traingdx</i>	0.77 ± 0.07	0.78 ± 0.07	49.2	0.00 ± 0.00	0.00 ± 0.00	17.8	0.00 ± 0.00	0.00 ± 0.00	14.7

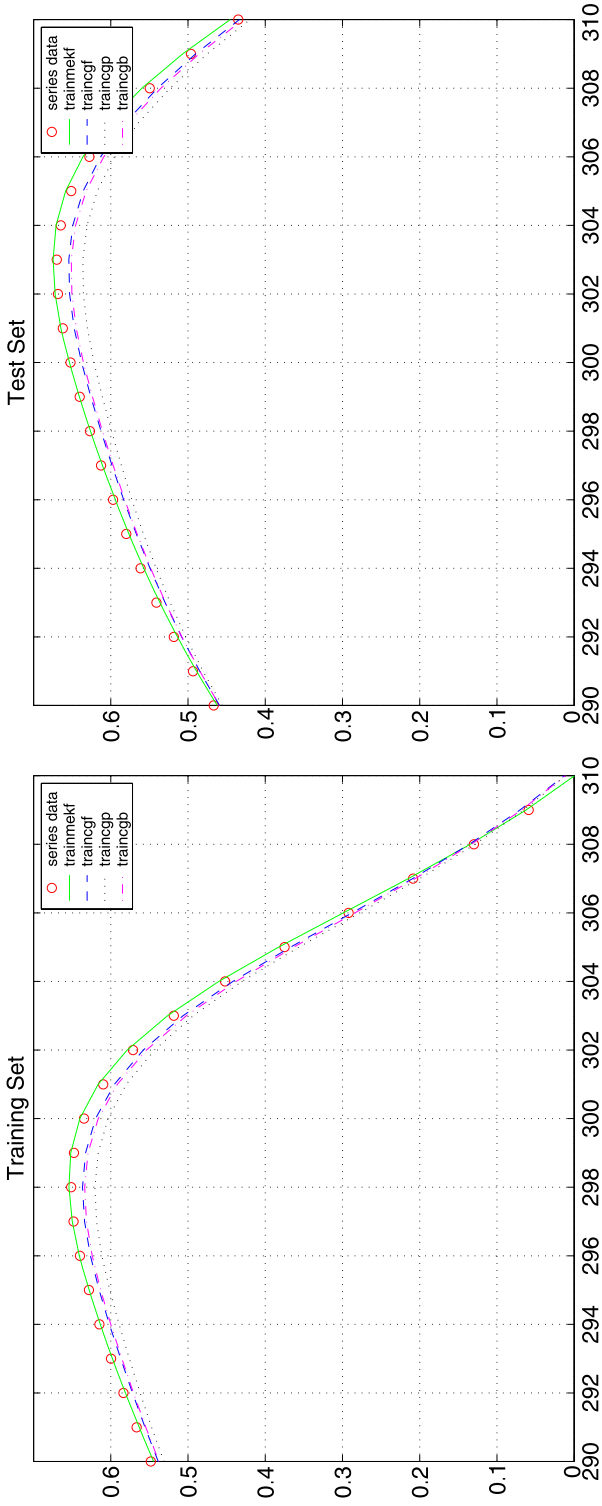


Fig. 4 Mackey-Glass series predictions of  $x_{t+1}$  by a 5-neuron one-hidden-layer neural network

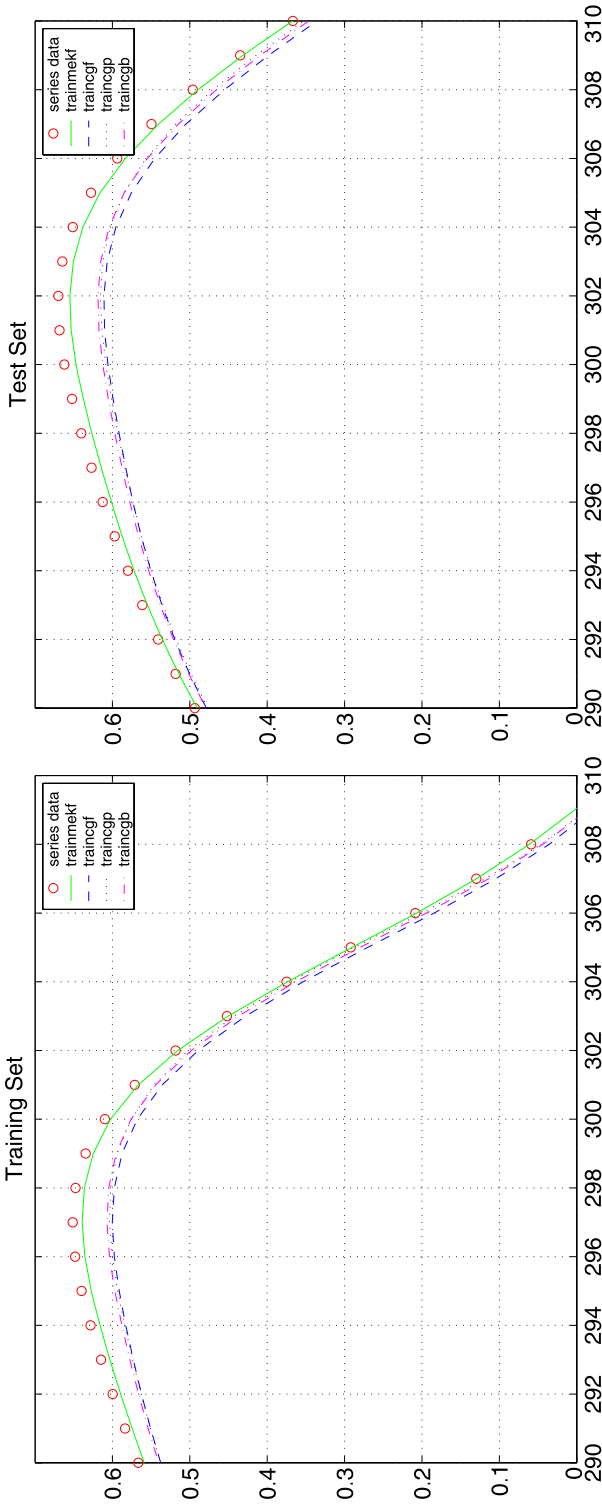


Fig. 5 Mackey-Glass series predictions of  $x_{t+2}$  by a 5-neuron one-hidden-layer neural network

where  $l$  was chosen equal to 4. Table 2 summarizes the MSE and processing time results averaged over 100 different trials with 5-neuron one-hidden-layer neural networks. In each trial, the time series were initialized with different initial conditions uniformly distributed between 0 and 0.4; the initial neural weights were randomly chosen between 0 and 1 according to a uniform distribution. The MEKF algorithm was initialized with  $P_0 = 10^{-2}I$ . The parameters  $\alpha$  and  $\epsilon$  were chosen equal to  $10^{-2}$  and  $10^{-2}$ , respectively.

As can be seen in Table 2, *trainmekf* outperforms all the other algorithms in terms of both the MSE and the computational load (“time” column). Figures 4 and 5 enable one to compare the prediction capabilities obtained by the four algorithms *trainmekf*, *traincgf*, *traincgb*, and *traincgp* in two simulation runs using the training and test sets, respectively.

## 5 Conclusions

The solution of nonlinear least-squares problems has been addressed via an algorithm based on the EKF. The proposed algorithm, called “modified extended Kalman filter” (MEKF) has been analyzed and its estimation error has been proved to be exponentially bounded. In addition, the algorithm is well-suited to being used in batch and iterated modes as well.

As an application, the problem of optimizing the parameters of nonlinear input–output mappings implemented by feedforward neural networks has been considered. Numerical results show interesting features and very good performances of MEKF, which outperforms many widespread algorithms for neural networks training. This may be ascribed to the computation of the covariance matrix, which provides a useful measure of the uncertainty associated with the estimate of the optimal weights, though it does not correspond to the real covariance of the underlying stochastic process. However, in a fair evaluation of the pros and cons, it has to be considered that the MEKF algorithm has to take into account the storage requirements of the covariance matrix.

## Appendix

In this appendix, the technical lemmas used for the proof of Theorem 2 are stated and proved.

**Lemma 1** *Let  $\Pi_t = P_t^{-1}$ . Under Assumption 5(iii), for every  $a, b > 0$ , the following inequalities hold for  $t = 0, 1, \dots$ :*

$$2[-(I - K_t \Lambda_t) \underline{e}_t]^T \Pi_{t+1} K_t \underline{\eta}_t \leq a \underline{e}_t^T (I - K_t \Lambda_t)^T \Pi_{t+1} (I - K_t \Lambda_t) \underline{e}_t + \frac{1}{a} \underline{\eta}_t^T K_t^T \Pi_{t+1} K_t \underline{\eta}_t, \quad (27)$$

$$2[(I - K_t \Lambda_t) \underline{e}_t]^T \Pi_{t+1} \underline{\varphi}_t \leq b \underline{e}_t^T (I - K_t \Lambda_t)^T \Pi_{t+1} (I - K_t \Lambda_t) \underline{e}_t + \frac{1}{b} \underline{\varphi}_t^T \Pi_{t+1} \underline{\varphi}_t, \quad (28)$$

$$-\underline{\varphi}_t^T \Pi_{t+1} K_t \underline{\eta}_t \leq \underline{\varphi}_t^T \Pi_{t+1} \underline{\varphi}_t + \underline{\eta}_t^T K_t^T \Pi_{t+1} K_t \underline{\eta}_t. \tag{29}$$

*Proof* Recall that, for every positive integer  $s$ , every symmetric positive definite matrix  $X \in \mathbb{R}^{s \times s}$ , and every  $\underline{v}_1, \underline{v}_2 \in \mathbb{R}^s$ , Young’s inequality [33] gives

$$2\underline{v}_1^T \underline{v}_2 \leq \underline{v}_1^T X \underline{v}_1 + \underline{v}_2^T X^{-1} \underline{v}_2. \tag{30}$$

By applying Eq. 30 to the first term of the left-hand side of Eq. 27 with  $\underline{v}_1 = -(I - K_t \Lambda_t) \underline{e}_t$ ,  $\underline{v}_2 = \Pi_{t+1} K_t \underline{\eta}_t$ , and  $X = a \Pi_{t+1}$ ,  $a > 0$ , we obtain

$$2[-(I - K_t \Lambda_t) \underline{e}_t]^T \Pi_{t+1} K_t \underline{\eta}_t \leq a \underline{e}_t^T (I - K_t \Lambda_t)^T \Pi_{t+1} (I - K_t \Lambda_t) \underline{e}_t + \frac{1}{a} \underline{\eta}_t^T K_t^T \Pi_{t+1} K_t \underline{\eta}_t.$$

Similarly one can prove Eq. 28, by using Eq. 30 with  $\underline{v}_1 = (I - K_t \Lambda_t) \underline{e}_t$ ,  $\underline{v}_2 = \Pi_{t+1} \underline{\varphi}_t$ , and  $X = b \Pi_{t+1}$ ,  $b > 0$ , and Eq. 29, by using Eq. 30 with  $\underline{v}_1 = \frac{1}{\sqrt{2}} \underline{\varphi}_t$ ,  $\underline{v}_2 = \frac{1}{\sqrt{2}} \Pi_{t+1} K_t \underline{\eta}_t$ , and  $X = 2 \Pi_{t+1}$ . □

**Lemma 2** *Under Assumptions 5(ii) and (iii), the following inequality holds for  $t = 0, 1, \dots$ :*

$$\|K_t\| \leq \frac{p_{\max} h_{\max}}{r_{\min}}. \tag{31}$$

*Proof* From Eq. 3 we have

$$\|K_t\| \leq \|P_t\| \|H_t^T (H_t P_t H_t^T + R_t)^{-1}\| \leq \|P_t\| \|H_t^T\| \|(H_t P_t H_t^T + R_t)^{-1}\|. \tag{32}$$

As  $H_t P_t H_t^T$  is positive semidefinite and  $R_t$  is positive definite, we obtain  $H_t P_t H_t^T + R_t \geq R_t \geq r_{\min} I$ , hence  $(H_t P_t H_t^T + R_t)^{-1} \leq R_t^{-1} \leq \frac{1}{r_{\min}} I$  and so

$$\|(H_t P_t H_t^T + R_t)^{-1}\| \leq \frac{1}{r_{\min}}.$$

The above inequality, Assumptions 5(ii) and (iii), and Eq. 32 give Eq. 31. □

**Lemma 3** *Under Assumptions 5(ii) and 6, for every  $\alpha > 0$  the following inequality holds for  $t = 0, 1, \dots$*

$$(\alpha + 1)(I - K_t \Lambda_t)^T \Pi_{t+1} (I - K_t \Lambda_t) \leq (1 - \beta) \Pi_t, \tag{33}$$

where  $\Pi_t = P_t^{-1}$ ,  $\beta = \frac{k_2}{1+k_2} < 1$ , and  $k_2 = \epsilon(p_{\max}(1 + \frac{p_{\max} h_{\max}^2}{r_{\min}})^2)^{-1}$ .

*Proof* From Eqs. 3 and 4 we have

$$\begin{aligned} P_{t+1} &= (\alpha + 1)(P_t - K_t H_t P_t + \epsilon I) \\ &= (\alpha + 1)(P_t - P_t H_t^T (H_t P_t H_t^T + R_t)^{-1} H_t P_t + \epsilon I) \end{aligned}$$

$$\begin{aligned}
 &= (\alpha + 1)(P_t - P_t H_t^T K_t^T + \epsilon I) \\
 &= (\alpha + 1)((I - K_t H_t)P_t(I - K_t H_t)^T + K_t H_t P_t(I - K_t H_t)^T + \epsilon I). \tag{34}
 \end{aligned}$$

Consider the term  $K_t H_t P_t(I - K_t H_t)^T$  in Eq. 34. By using Eq. 3 and the matrix equality (7B.5) in [12, p. 262], we obtain

$$\begin{aligned}
 (P_t^{-1} + H_t^T R_t^{-1} H_t)^{-1} &= P_t - P_t H_t^T (H_t P_t H_t^T + R_t)^{-1} H_t P_t \\
 &= (I - P_t H_t^T (H_t P_t H_t^T + R_t)^{-1} H_t) P_t = (I - K_t H_t) P_t.
 \end{aligned}$$

Thus,

$$I - K_t H_t = (P_t^{-1} + H_t^T R_t^{-1} H_t)^{-1} P_t^{-1} > 0. \tag{35}$$

As  $K_t H_t = P_t H_t^T (H_t P_t H_t^T + R_t)^{-1} H_t$  is positive semidefinite, by (35) the matrix  $K_t H_t P_t(I - K_t H_t)^T$  is positive semidefinite, too. Hence, from (34) we get

$$P_{t+1} \geq (\alpha + 1)((I - K_t H_t)P_t(I - K_t H_t)^T + \epsilon I)$$

and, by left and right multiplying for  $(I - K_t H_t)^{-1}$  and  $[(I - K_t H_t)^T]^{-1}$ , respectively, we have

$$\frac{1}{\alpha + 1} (I - K_t H_t)^{-1} P_{t+1} [(I - K_t H_t)^T]^{-1} \geq P_t + \epsilon (I - K_t H_t)^{-1} [(I - K_t H_t)^T]^{-1}. \tag{36}$$

Lemma 2 and Assumption 5(ii) enable us to obtain

$$\begin{aligned}
 I - K_t H_t &\leq \|I - K_t H_t\| I \leq (\|I\| + \|K_t H_t\|) I \\
 &\leq (1 + \|K_t\| \|H_t\|) I \leq \left(1 + \frac{p_{\max} h_{\max}^2}{r_{\min}}\right) I
 \end{aligned}$$

and so, by inverting both sides,

$$(I - K_t H_t)^{-1} \geq \frac{1}{1 + \frac{p_{\max} h_{\max}^2}{r_{\min}}} I.$$

By Eq. 36, since  $I = P_t P_t^{-1} \geq P_t / p_{\max}$ , by Assumption 5(iii) the previous inequality gives

$$\frac{1}{\alpha + 1} (I - K_t H_t)^{-1} P_{t+1} [(I - K_t H_t)^T]^{-1} \geq (1 + k_2) P_t,$$

where  $k_2 = \epsilon [p_{\max} (1 + \frac{p_{\max} h_{\max}^2}{r_{\min}})^2]^{-1} > 0$ . By inverting both sides of this inequality, we obtain

$$(\alpha + 1) (I - K_t H_t)^T P_{t+1}^{-1} (I - K_t H_t) \leq \frac{1}{1 + k_2} P_t^{-1}. \tag{37}$$

By Assumption 6 we can bound from above as follows

$$\begin{aligned} (I - K_t \Lambda_t)^T P_{t+1}^{-1} (I - K_t \Lambda_t) &\leq \lambda_{\max}[(I - K_t \Lambda_t)^T P_{t+1}^{-1} (I - K_t \Lambda_t)] I \\ &\leq \lambda_{\min}[(I - K_t H_t)^T P_{t+1}^{-1} (I - K_t H_t)] I \\ &\leq (I - K_t H_t)^T P_{t+1}^{-1} (I - K_t H_t). \end{aligned}$$

By Eq. 37 the latter inequality gives  $(I - K_t \Lambda_t)^T \Pi_{t+1} (I - K_t \Lambda_t) \leq \frac{1}{\alpha+1} \frac{1}{1+k_2} \Pi_t \leq \frac{1}{1+k_2} \Pi_t = (1 - \beta) \Pi_t$ .  $\square$

## References

1. Bates, D.M., Watts, D.G.: *Nonlinear Regression and Its Applications*. Wiley, New York (1988)
2. Bertsekas, D.P.: *Nonlinear Programming*, 2nd edn. Athena Scientific, Belmont (1999)
3. Bertsekas, D.P.: Incremental least-squares methods and the extended Kalman filter. *SIAM J. Optim.* **6**(3), 807–822 (1996)
4. Feldkamp, L.A., Prokhorov, D.V., Eagen, C.F., Yuan, F.: Enhanced multi-stream Kalman filter training for recurrent networks. In: Suykens, J., Vandewalle, J. (eds.) *Nonlinear Modeling: Advanced Black-Box Techniques*, pp. 29–53. Kluwer Academic, Dordrecht (1998)
5. Moriyama, H., Yamashita, N., Fukushima, M.: The incremental Gauss–Newton algorithm with adaptive stepsize rule. *Comput. Optim. Appl.* **26**(2), 107–141 (2003)
6. Shuhe, H.: Consistency for the least squares estimator in nonlinear regression model. *Stat. Probab. Lett.* **67**(2), 183–192 (2004)
7. Reif, K., Günter, S., Yaz, E., Unbehauen, R.: Stochastic stability of the discrete-time extended Kalman filter. *IEEE Trans. Autom. Control* **44**(4), 714–728 (1999)
8. Kůrková, V., Sanguineti, M.: Learning with generalization capability by kernel methods of bounded complexity. *J. Complex.* **21**(3), 350–367 (2005)
9. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representation by error propagation. In: Rumelhart, D.E., McClelland, J.L., PDP Research Group (eds.) *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol. I: Foundations, pp. 318–362. MIT, Cambridge (1986)
10. Widrow, B., Lehr, M.A.: 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proc. IEEE* **78**(9), 1415–1442 (1990)
11. Ortega, J.M.: *Numerical Analysis: A Second Course*. SIAM, Philadelphia (1990). Reprint of the 1972 edition by Academic, Now York
12. Jazwinski, A.H.: *Stochastic Processes and Filtering Theory*. Academic, New York (1970)
13. Pinkus, A.: Approximation theory of the MLP model in neural networks. *Acta Numer.* **8**, 143–196 (1999)
14. Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. *Neural Comput.* **3**(2), 246–257 (1991)
15. Heskes, T., Wiegierinck, W.: A theoretical comparison of batch-mode, on-line, cyclic, and almost-cyclic learning. *IEEE Trans. Neural Netw.* **7**(4), 919–925 (1996)
16. Demuth, H., Beale, M.: *Neural Network Toolbox—User’s Guide*. The Math Works, Natick (2000)
17. Bell, B.M., Cathey, F.W.: The iterated Kalman filter update as a Gauss–Newton method. *IEEE Trans. Autom. Control* **38**(2), 294–297 (1993)
18. Fletcher, R.: *Practical Methods of Optimization*. Wiley, Chichester (1987)
19. Battiti, R.: First- and second-order methods for learning: between steepest descent and Newton’s method. *Neural Comput.* **4**(2), 141–166 (1992)
20. Tollenaere, T.: SuperSAB: fast adaptive backpropagation with good scaling properties. *Neural Netw.* **3**(5), 561–573 (1990)
21. Jacobs, R.A.: Increased rates of convergence through learning rate adaption. *Neural Netw.* **1**(4), 295–307 (1988)
22. Denton, J.W., Hung, M.S.: A comparison of nonlinear optimization methods for supervised learning in multilayer feedforward neural networks. *Eur. J. Oper. Res.* **93**(2), 358–368 (1996)

23. Stinchcombe, M., White, H.: Approximation and learning unknown mappings using multilayer feed-forward networks with bounded weights. In: Proc. Int. Joint Conf. on Neural Networks IJCNN'90, pp. III7–III16 (1990)
24. Singhal, S., Wu, L.: Training multilayer perceptrons with the extended Kalman algorithm. In: Touretzky, D.S. (ed.) *Advances in Neural Information Processing Systems 1*, pp. 133–140. Morgan Kaufmann, San Mateo (1989)
25. Ruck, D.W., Rogers, S.K., Kabrisky, M., Maybeck, P.S., Oxley, M.E.: Comparative analysis of back-propagation and the extended Kalman filter for training multilayer perceptrons. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(6), 686–691 (1992)
26. Iiguni, Y., Sakai, H., Tokumaru, H.: A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter. *IEEE Trans. Signal Process.* **40**(4), 959–966 (1992)
27. Schottky, B., Saad, D.: Statistical mechanics of EKF learning in neural networks. *J. Phys. A: Math. Gen.* **32**(9), 1605–1621 (1999)
28. Nishiyama, K., Suzuki, K.:  $H_\infty$ -learning of layered neural networks. *IEEE Trans. Neural Netw.* **12**(6), 1265–1277 (2001)
29. Leung, C.-S., Tsoi, A.-C., Chan, L.W.: Two regularizers for recursive least squared algorithms in feedforward multilayered neural networks. *IEEE Trans. Neural Netw.* **12**(6), 1314–1332 (2001)
30. Alessandri, A., Sanguineti, M., Maggiore, M.: Optimization-based learning with bounded error for feedforward neural networks. *IEEE Trans. Neural Netw.* **13**(2), 261–273 (2002)
31. Prechelt, L.: PROBEN 1—A set of neural network benchmark problems and benchmarking rules. Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany, September 1994, Anonymous FTP: /pub/papers/techreports/1994/1994-21.ps.gzonftp.ira.uka.de
32. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. *Science* **197**, 287–289 (1977)
33. Hardy, G., Littlewood, J.E., Polya, G.: *Inequalities*. Cambridge University Press, Cambridge (1989)