A Hybrid Linear/Nonlinear Training Algorithm for Feedforward Neural Networks

Seán McLoone, Michael D. Brown, George Irwin, *Senior Member, IEEE,* and Gordon Lightbody

***Abstract*— This paper presents a new hybrid optimization strategy for training feedforward neural networks. The algorithm combines gradient-based optimization of nonlinear weights with singular value decomposition (SVD) computation of linear weights in one integrated routine. It is described for the multilayer perceptron (MLP) and radial basis function (RBF) networks and then extended to the local model network (LMN), a new feedforward structure in which a global nonlinear model is constructed from a set of locally valid submodels. Simulation results are presented demonstrating the superiority of the new hybrid training scheme compared to second-order gradient methods. It is particularly effective for the LMN architecture where the linear to nonlinear parameter ratio is large.**

***Index Terms*—Feedforward neural networks, off-line training algorithms, second-order methods.**

I. Introduction

THE training of feedforward neural networks like the multilayer perceptron (MLP) and radial basis function (RBF) network can be posed as an unconstrained optimization problem. The objective is to minimize a performance index, such as the sum squared error (SSE) in (1), with respect to the network parameters $(\boldsymbol{w})$

$$E(\boldsymbol{w}) = \sum_{i=1}^{N_V} (y_i - d_i)^2 \tag{1}$$

where N_V is the number of training vectors and y_i and d_i are the actual and desired output of the network for the ith training vector. The training strategy employed then depends on whether the index is minimized with respect to all the weights or just the linear ones and leads to two general classifications for the optimization problem:

- full nonlinear optimization;
- linear optimization of the output layer weights only, with the nonlinear (hidden layer) weights initialized before training commences.

Nonlinear and linear training of feedforward neural networks are now briefly reviewed in turn.

Manuscript received January 22, 1997; revised October 12, 1997 and March 22, 1998.

S. McLoone and G. Irwin are with the Advanced Control Engineering Research Centre, Department of Electrical and Electronic Engineering, The Queen's University of Belfast, Belfast BT9 5AH U.K.

M. D. Brown is with the School of Mechanical Engineering, University of Leeds, Leeds LST 9JT U.K.

G. Lightbody is with the Department of Electrical Engineering and Microelectronics, University College Cork, Cork, Ireland.

Publisher Item Identifier S 1045-9227(98)04453-1.

A. Nonlinear Training

The batch backpropagation (BBP) algorithm developed by Rumelhart *et al.* [24] in 1986 is the fundamental nonlinear training algorithm for MLP's. This is a simple gradient descent method in which the network weights are adjusted with respect to an overall cost function $E(\boldsymbol{w})$ using a learning rule of the form

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \eta_k \boldsymbol{g}_k \tag{2}$$

where

$$\boldsymbol{g} = \left[\frac{\partial E(w_1)}{\partial w_1} \quad \frac{\partial E(w_2)}{\partial w_2} \quad \cdots \quad \frac{\partial E(w_n)}{\partial w_n} \right]^T \tag{3}$$

and η_k is the step size at the kth iteration. While BBP is a very popular algorithm because of its relative simplicity and historical significance, it is very slow and subject to frequent failures. Consequently many researchers have developed heuristical extensions to BBP in an attempt to improve performance. These include the delta-bar-delta method [10], Quickprop [7], Rprop [25], and adaptive backpropagation [21].

Random search methods such as simulated annealing, Chemotaxis, diffusion, and genetic algorithms [9], [27], [29], [30], [33] are promising alternatives to gradient-based training algorithms. These methods are characterized by having a random search element in the training process which allows the algorithms to escape from local minima and converge to the global minimum of the cost function. They are generally used in conjunction with some form of gradient-based training algorithm as convergence of pure random search techniques tends to be very slow [15].

More significant advances in training have come from the use of powerful second-order optimization techniques from the field of unconstrained optimization ([2], [28], [29], and [16]). These methods, classed as second-order because they directly or indirectly exploit second derivative (curvature) information to generate their search directions, have a strong theoretical basis and are vastly superior to gradient descent (BBP). The algorithms can either be of matrix or nonmatrix (vector) type. The former are typically two orders of magnitude faster than BBP but require the storage of the Hessian matrix, its inverse or approximations thereof. The latter only require the storage of a few additional vectors but are then only one order of magnitude faster than BBP. To date researchers have tended to favor the vector-based algorithms because of memory restrictions and the main contributions include the limited memory

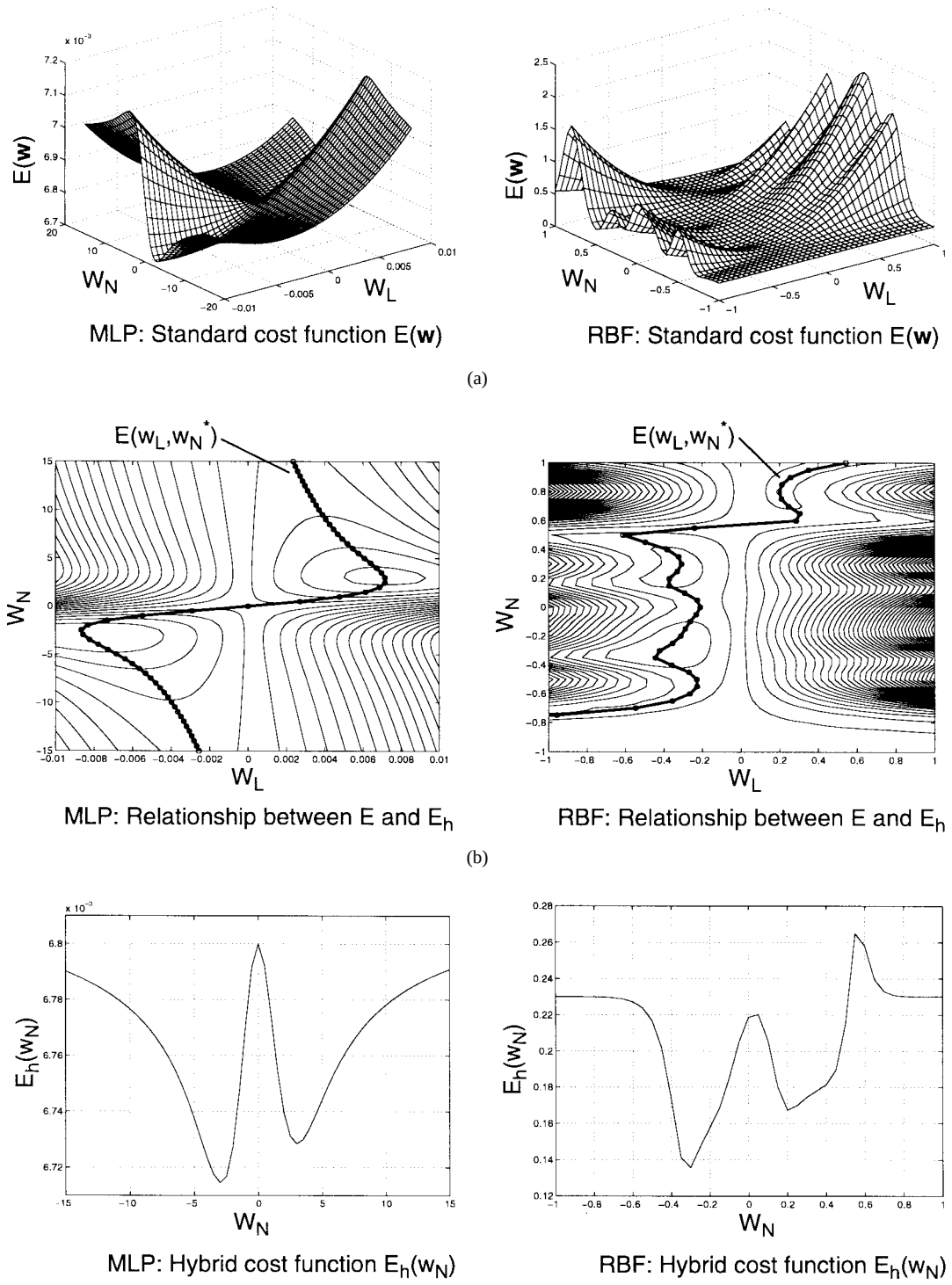


Fig. 1. Visualization of hybrid cost functions for two-parameter neural networks.

(memory-less) Broyden–Fletcher–Golfarb–Shanno (BFGS) algorithm (LM) by Battiti *et al.* [1], the scaled conjugate gradient (SCG) algorithm by Møller [20], and the conjugate gradient (CG) algorithm with Powell restarts by Van Der Smagt [28]. However recent developments in computer technology have meant that the use of the more powerful, memory intensive methods such as full memory BFGS (FM) and Levenberg–Marquardt [17] are now feasible for much larger problems. In this paper the FM technique has been employed. For further details of this and other second-order training methods see McLoone *et al.* [18].

B. Linear Training

When only the linear output layer weights $(\boldsymbol{w}_L)$ of a feedforward neural network are considered for optimization, training can be posed as a linear least-squares problem with

TABLE I
COMPUTATIONAL AND MEMORY REQUIREMENTS OF HYBRID FM AND FM

	Full Memory BFGS		Hybrid Full Memory BFGS	
	Computation	Memory	Computation	Memory
$E(\mathbf{w})$, $E_H(\mathbf{w})$	$O(N_v N_w)$	$O(N_w)$	$O(N_v N_w + N_v N_L^2 + N_L^3)$	$O(N_w + N_v N_L + N_L^2)$
$\mathbf{g}$, $\mathbf{g}_H$	$O(2N_v N_w)$	$O(2N_w)$	$O(2N_v N_N)$	$O(2N_N)$
other	$O(3N_w^2)$	$O(1.5N_w^2)$	$O(3N_N^2)$	$O(1.5N_N^2)$

a cost function

$$E(\boldsymbol{w}_L) = \sum_{i=1}^{N_V} [\boldsymbol{r}_i^T \boldsymbol{w}_L - d_i]^2 \tag{4}$$

where $\boldsymbol{r}_i$ is the vector of hidden layer outputs for the ith training vector

$$\boldsymbol{r}_i = [r_{i1} \quad r_{i2} \quad \cdots \quad r_{iN_h}]^T \tag{5}$$

and element r_{ij} is defined as the jth hidden neuron output for the ith training input. The network output y_i is therefore given by $\boldsymbol{r}_i^T \boldsymbol{w}_L$ and the corresponding desired output is denoted by d_i. The number of linear weights N_L is generally the same as the number of hidden layer outputs (N_h). However, if a bias is included on the linear output layer neuron, as is typically the case with MLP networks, then $N_L = N_h + 1$ and $\boldsymbol{r}_i$ is augmented by an additional element $r_{i(N_h+1)} = 1$. Equation (4) can also be expressed in matrix form as

$$E(\boldsymbol{w}_L) = [R\boldsymbol{w}_L - \boldsymbol{d}]^T [R\boldsymbol{w}_L - \boldsymbol{d}] \tag{6}$$

where

$$\boldsymbol{d} = [d_1 \quad d_2 \quad \cdots \quad d_{N_v}]^T \tag{7}$$

and R is the regressor matrix defined as

$$R = [r_{ij}] = [\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \cdots \quad \boldsymbol{r}_{N_v}]^T. \tag{8}$$

The main advantages of adapting only the linear weights are as follows.

- It leads to a significant reduction in problem dimension.
- The resulting cost function has a single global minimum.
- The minimum can be determined in one step by solving the linear system of equations

$$R\boldsymbol{w}_L = \boldsymbol{d}. \tag{9}$$

In general neural-network problems are over-determined, that is, there are many more training vectors than weights. Consequently R is not square and (9) does not have a unique solution. In these circumstances the minimum error solution

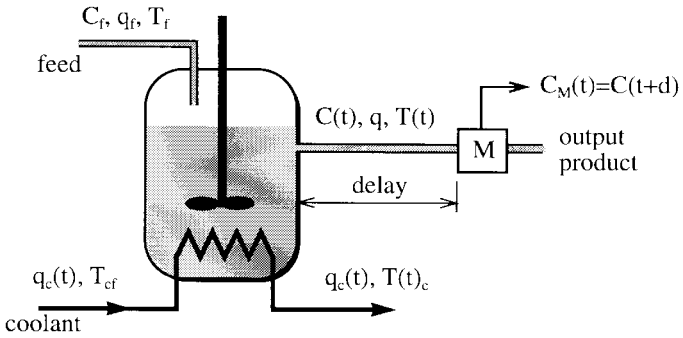


Fig. 2. Continuous stirred tank reactor.

for $\boldsymbol{w}_L$ is given by

$$\boldsymbol{w}_L^* = [RR^T]^{-1} R\boldsymbol{d} \tag{10}$$

where $[RR^T]^{-1}R$ is the pseudoinverse of R. This is the linear least-squares solution and is generally evaluated using singular value decomposition (SVD). SVD is a powerful algorithm which can deal with both over- and underdetermined systems and is recognized as the most numerically robust technique for dealing with ill-conditioned problems.

In practice the critical aspect of linear training is not the optimization itself, but rather the network initialization to decide on the number of neurons and nonlinear parameter values. Since this determines the performance achievable with a given network it is included for completeness.

Initialization of RBF Weights: The intuitive nature of the RBF network structure, in which the nonlinear weights can be interpreted as "centers" and "widths," makes informed selection of these parameters possible. The centers are usually placed on a uniform grid and the widths chosen as a function of the inter neuron spacing. While this gives good results it is not feasible for large input dimensions and is also very inefficient in problems where only a small portion of the input space is active. Consequently practical center allocation algorithms involve the placement of a small number of neurons in a manner which reflects the distribution of the training data. The simplest approach is to employ a randomly selected subset of the training data as centers. Alternatively an optimum subset can be determined systematically using the orthogonal least-squares procedure proposed by Chen *et al.* [6]. However, both these strategies restrict centers to the position of data points and in general lead to suboptimal placements. An optimal

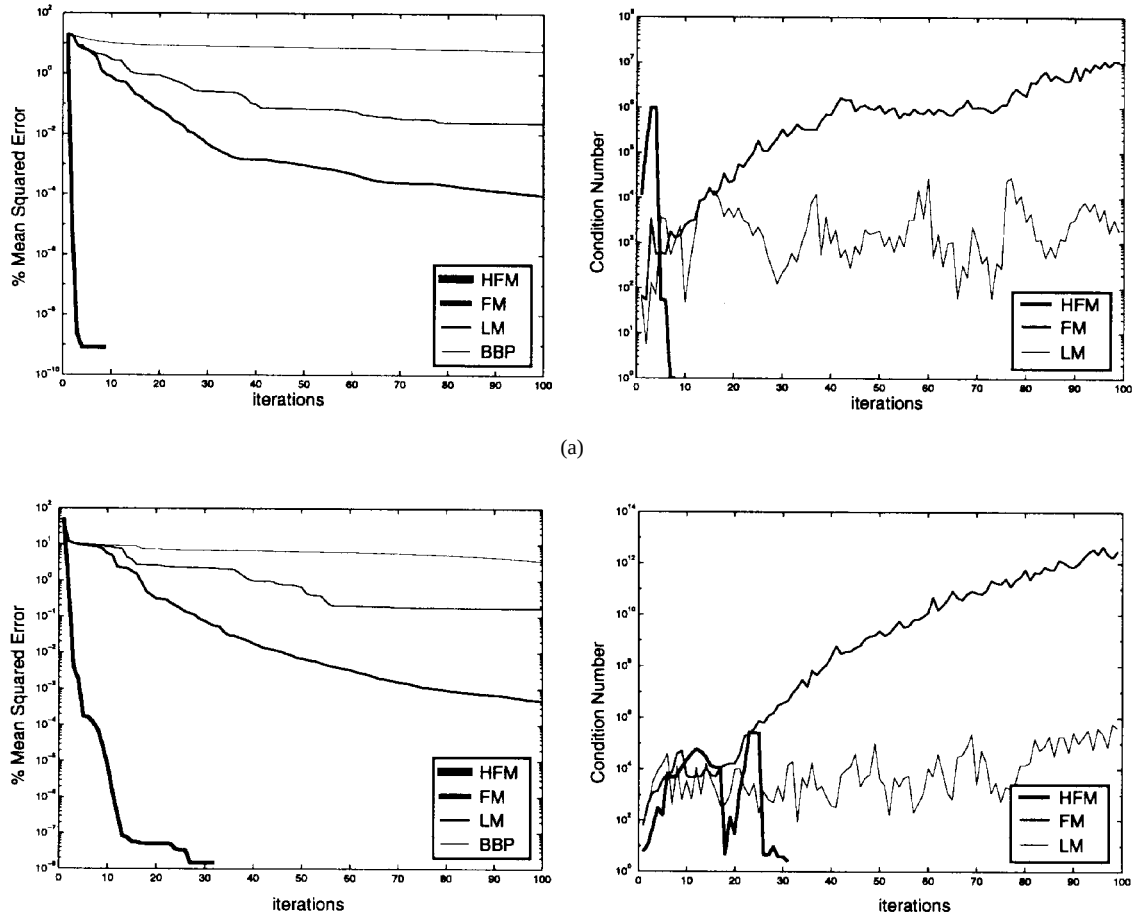


Fig. 3. Training algorithm performance for MLP's on Problem 1. (a) MLP (1,5,1). (b) MLP (1,2,1).

placement with respect to the distribution of the data is one which minimizes the total Euclidean distance (E_k) between the training patterns and the closest centers. This can be determined using an unsupervised training procedure such as k-means clustering [5].

A major drawback of these heuristics is their inability to take into account the dynamics of the output. Clearly it would be desirable to have more neurons in regions where the output is changing significantly and fewer neurons in flat regions. Chen *et al.* [4] propose a modification of the k-means clustering algorithm which has the effect of skewing the center placements toward areas of the input space which exhibit large output variations.

Once the centers have been located the widths are usually determined using a nearest-neighbor criteria, such that, the width of a given activation function is set equal to the mean distance to its p nearest neighbors. Again Chen *et al.* [4] suggest varying the number of nearest neighbors used in the width computation according to the smoothness of the output. The resulting hybrid nearest-neighbor algorithm generates large widths in flat regions of the input space and small widths in areas with significant output variation.

Initialization of MLP Weights: The development of useful heuristics for initial weight assignment in MLP's has not been possible due to its black box characteristic. Consequently, most strategies are based on random initialization of the weights and are thus very much trial-and-error processes. Random initialization is important to ensure the creation of a general structure. Symmetric weight distributions, a major source of redundancy within an MLP, are therefore unlikely to occur. Such distributions severely restrict the performance of gradient-based training algorithms which are unable to break weight symmetry. The random weights method [18] is the simplest and most widely used strategy for MLP weight initialization. Here, the weights are chosen from a random distribution in the range $[-\alpha, \alpha]$, where α is commonly chosen as one. Wessels and Barnard [32] use a statistical and empirical analysis of the problem, to show that $3/\sqrt{N_I}$ is the optimal choice for α. In a variation, called the scaled random weights method [18], the weights are first initialized using the random weights method and then individually scaled to ensure that each neuron is active over its full dynamic range. The scaling factor for the weights connected to the ith neuron in the jth hidden layer (ρ_{ij}) is given by

$$\rho_{ij} = \frac{D_{ij}}{\alpha \cdot n_{ij}} \tag{11}$$

where D_{ij} is the dynamic range of the activation function, α is the range of the random distribution and n_{ij} is the number of weights connected to that neuron.

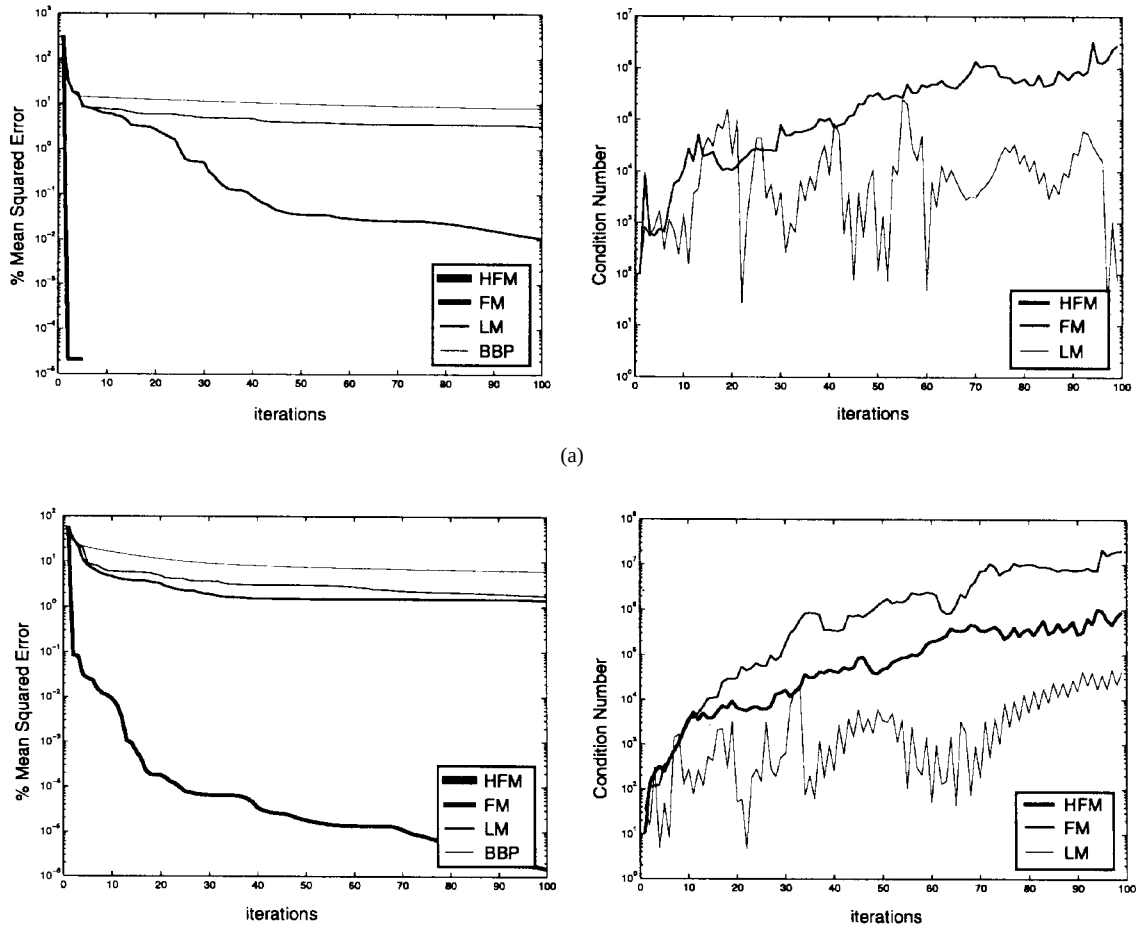


Fig. 4. Training algorithm performance for MLP's on Problem 2. (a) MLP (1,10,1). (b) MLP (1,5,1).

While the foregoing review reflects the fact that nonlinear optimization is normally associated with MLP's and linear optimization is the usual method for training RBF networks, it does not preclude the possibility of training each network by either method. This contribution describes a new hybrid training algorithm that integrates the best features of each of these methods into one routine which can be used to train MLP's, RBF's, or any other feedforward network with linear output layer weights. The new algorithm is presented in Section II. Its extension to local model networks (LMN's), a developing and important research theme in the nonlinear modeling and control literature is dealt with in Section III. Section IV contains a comprehensive set of simulation results which confirm the benefit of the new training approach compared with existing techniques. The final conclusions are given in Section V.

II. New Hybrid Optimization Approach

When a neural network is linear in its adjustable parameters they can be determined using SVD as discussed in the previous section. However, when both linear and nonlinear weights have to be adapted nonlinear training procedures such as FM and CG have to be employed. These techniques do not differentiate between the linear and nonlinear weights and therefore cannot exploit SVD or its properties. Here a hybrid training algorithm is proposed which achieves this objective by adapting the nonlinear weights using the FM BFGS algorithm while employing SVD to compute the optimum linear weights at each iteration.

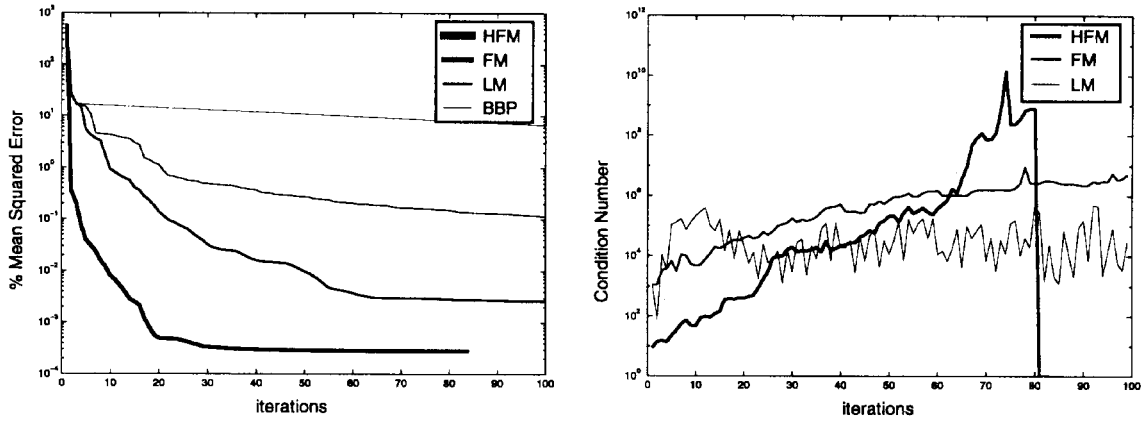
Structuring the overall optimization in this way can lead to a number of advantages in training the network. Since the number of independent parameters in the network is reduced, it should be expected that the time taken to reach a minimum of the nonlinear cost function will be less. Secondly, the network is always in a state where the sum squared error is at a global minimum with respect to the output layer weights, since these are obtained by linear optimization.

A mathematical development of the algorithm is now obtained by viewing training as a standard nonlinear optimization problem applied to a hybrid cost function.

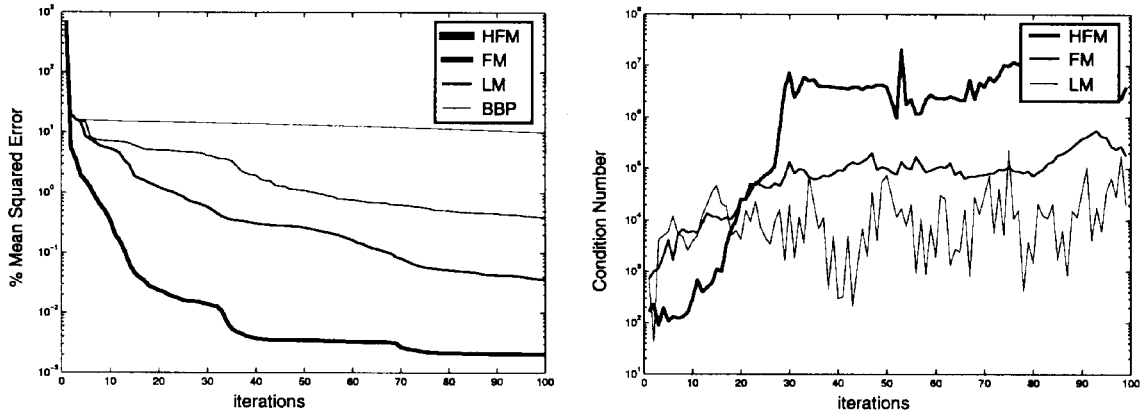
A. Hybrid Cost Function

Suppose the overall weights vector, $\boldsymbol{w}$, for a feedforward neural network with linear output layer weights is partitioned as

$$\boldsymbol{w} = [\boldsymbol{w}_N^T; \boldsymbol{w}_L^T]^T \tag{12}$$



(a)



(b)

Fig. 5. Training algorithm performance for MLP's on Problem 3. (a) MLP (2,10,1). (b) MLP (2,5,1).

where $\boldsymbol{w}_L$ and $\boldsymbol{w}_N$ are column vectors containing the N_L linear parameters and N_N nonlinear parameters. $E(\boldsymbol{w})$ in (1) can then be expressed as

$$E(\boldsymbol{w}_N, \boldsymbol{w}_L) = \sum_{i=1}^{N_v} [\boldsymbol{r}_i(\boldsymbol{w}_N)^T \boldsymbol{w}_L - d_i]^2. \tag{13}$$

This can also be expressed in matrix form as

$$E(\boldsymbol{w}_N, \boldsymbol{w}_L) = [R(\boldsymbol{w}_N)\boldsymbol{w}_L - \boldsymbol{d}]^T [R(\boldsymbol{w}_N)\boldsymbol{w}_L - \boldsymbol{d}] \tag{14}$$

where $\boldsymbol{d}$ and R are as previously defined in (7) and (8).

A hybrid cost function, in which the optimum linear weights are chosen for each combination of nonlinear weights, can now be expressed as

$$E_h(\boldsymbol{w}_N) = \min_{\boldsymbol{w}_L} \{E(\boldsymbol{w}_N, \boldsymbol{w}_L)\} = E(\boldsymbol{w}_N, \boldsymbol{w}_L^*(\boldsymbol{w}_N)). \tag{15}$$

Here $\boldsymbol{w}_L^*(\boldsymbol{w}_N)$ are the optimum linear weights given by

$$\boldsymbol{w}_L^*(\boldsymbol{w}_N) = [R(\boldsymbol{w}_N)R(\boldsymbol{w}_N)^T]^{-1} R(\boldsymbol{w}_N)\boldsymbol{d}. \tag{16}$$

This is the linear least-squares solution and is evaluated using SVD.

Visualization of the hybrid cost function is possible for simple, two-parameter MLP and RBF networks as defined in (17) and (18), respectively,

$$y = w_L \cdot \mathrm{sig}(w_N \cdot u) \tag{17}$$

$$y = w_L \cdot \exp[-(u - w_N)^2]. \tag{18}$$

Here, u is the input, y is the output and w_L and w_N are the linear and nonlinear parameters, respectively, of each network. The cost functions obtained with these networks for the training set $u\varepsilon[-0.5\ -0.2\ 0.0\ 0.3\ 0.8]^T, d\varepsilon[-0.2\ -0.3\ -0.1\ -0.3\ 0.2]^T$ are shown in Fig. 1. In each case the standard cost function $E(w)$ and the hybrid cost function $E_h(w_N)$ are presented along with a contour plot of $E(w)$. The locus of $E_h(w_N)$ is superimposed on the contour plots to highlight the relationship between the two cost functions. These examples clearly show how E_h is contained within E and can be interpreted as E subject to the constraint specified in (16). In this two-dimensional case the hybrid error can be plotted as a one-dimensional function, while generally E_h is constrained to a subspace of E, resulting in a subproblem of dimension N_N.

B. Hybrid Gradient

The hybrid cost function gradient $\boldsymbol{g}_H$, defined as $\partial E_h / \partial \boldsymbol{w}_N$, is needed by gradient-based optimization routines such as the FM and CG algorithms. Using the chain rule to differentiate

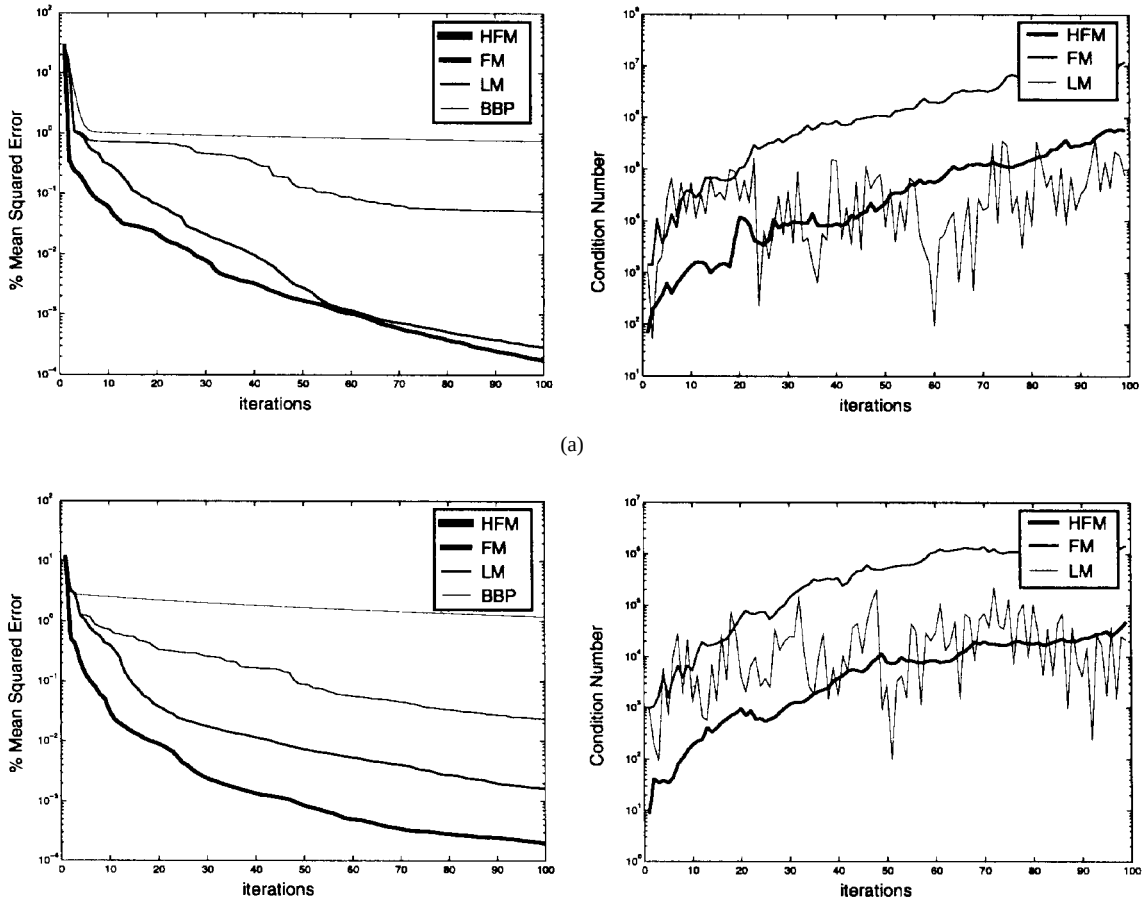


Fig. 6. Training algorithm performance for MLP's on Problem 4. (a) MLP (20,10,1). (b) MLP (20,5,1).

(15), $\boldsymbol{g}_H$ can be expressed as

$$\boldsymbol{g}_H = \boldsymbol{g}_N + J_{LN}^T \boldsymbol{g}_L. \tag{19}$$

Here $\boldsymbol{g}_N$ and $\boldsymbol{g}_L$ are the nonlinear and linear portions of the standard batch gradient $(\boldsymbol{g})$, that is,

$$\boldsymbol{g} = [\boldsymbol{g}_N^T; \boldsymbol{g}_L^T] = \left[\frac{\partial E^T}{\partial \boldsymbol{w}_N}; \frac{\partial E^T}{\partial \boldsymbol{w}_L}\right]^T = \frac{\partial E}{\partial \boldsymbol{w}} \tag{20}$$

and as such are straightforward to evaluate. The Jacobian matrix J_{LN} in (19), defined as

$$J_{LN} = \frac{\partial \boldsymbol{w}_L^*}{\partial \boldsymbol{w}_N} \tag{21}$$

requires the differentiation of (16) for each of the nonlinear weights and can be determined on a column-by-column basis. Thus

$$J_{LN}^k = \frac{\partial \boldsymbol{w}_L^*}{\partial w_N^k} = PQ_k^T \boldsymbol{d} - P[R^T Q_k + Q_k^T R]PR^T \boldsymbol{d} \tag{22}$$

where J_{LN}^k is the kth column of J_{LN} and w_N^k is the kth nonlinear weight. The matrices P and Q_k are defined as

$$P = [R^T R]^{-1}, \quad Q_k = \frac{\partial R}{\partial w_N^k}. \tag{23}$$

While P need only be computed once per iteration, and Q_k will have only one nonzero column for a given weight, this computation is extremely complex $(O(N_N N_L^5))$ and is susceptible to ill-conditioning problems. Fortunately, in the course of normal hybrid gradient computation, J_{LN} need not be calculated because $\boldsymbol{g}_L$ is always zero. Thus $\boldsymbol{g}_H$ is given by

$$\boldsymbol{g}_H = \boldsymbol{g}_N \tag{24}$$

which has a computational complexity of $O(N_v N_H)$. This simplification occurs because the optimum linear weights $\boldsymbol{w}_L^*$ are those which minimize the error in the subspace defined by the linear weights. Hence, by definition, the components of the gradient in this subspace must be zero. In terms of the standard cost function $E(\boldsymbol{w}_N, \boldsymbol{w}_L)$, this means that the gradient at any point satisfying (16) has the form

$$\boldsymbol{g}(\boldsymbol{w}_N, \boldsymbol{w}_L^*) = [\boldsymbol{g}_N; \boldsymbol{0}]. \tag{25}$$

The hybrid gradient is therefore simply obtained by computing the nonlinear component of the standard batch gradient for the optimum set of linear weights, that is,

$$\boldsymbol{g}_H = \left.\frac{\partial E}{\partial \boldsymbol{w}_N}\right|_{\boldsymbol{w}_L = \boldsymbol{w}_L^*}. \tag{26}$$

The computational complexity and memory requirements of the main components of the hybrid FM and standard FM algorithms are summarized in Table I.

TABLE II
COMPARISON OF ITERATION TIMES AND WEIGHT MAGNITUDES FOR HFM AND FM TRAINING

Problem No.	1a	1b	2a	2b	3a	3b	4a	4b
Linear/nonlinear weights ratio	6/10 0.6	3/4 0.75	11/20 0.55	6/10 0.6	11/30 0.37	6/15 0.4	11/210 0.052	6/105 0.057
Iteration time ratio (HFM/FM)	1.413	1.572	1.45	1.71	2.42	3.09	1.14	3.19
Weight magnitude ratio (HFM/FM)	19300	2153	48744	5252	737.2	44.9	28.24	4.93

Here N_v, N_w, N_L, and N_N are the number of training vectors, total number of weights, number of linear weights and number of nonlinear weights, respectively. The entry "other" is primarily the matrix computations in the computation of the BFGS search direction. From the table it can be seen that there is a significant reduction in the gradient and "other" computational and memory requirements if $N_N < N_w$. However, there is a huge increase in these quantities for the cost function evaluation in the hybrid algorithm because of the SVD computation.

III. EXTENSION TO LOCAL MODEL NETWORKS

The representational ability of RBF neural networks can be generalized to form the LMN structure. The basis functions in this instance are used to weight general functions of the inputs as opposed to simply the weights [11]. The network output can be expressed as

$$y = F(\boldsymbol{\psi}, \boldsymbol{\phi}) = \sum_{i=1}^{N_M} f_i(\boldsymbol{\psi}) \rho_i(\boldsymbol{\phi}) \tag{27}$$

where the N_M local models $f_i(\boldsymbol{\psi})$ are linear or nonlinear functions of the measurement vector, $\boldsymbol{\psi}$ (dimension N_d), and are multiplied by a basis function $\rho_i(\boldsymbol{\phi})$ that is a function of the current operating region vector, $\boldsymbol{\phi}$ (dimension N_k). This operating region does not necessarily need to be the full model input vector $\boldsymbol{\psi}$, but can be a subset of the measurement data available. In the Gaussian RBF neural network, the functions $f_i(\boldsymbol{\psi})$ are constants and the basis functions $\rho_i(\boldsymbol{\phi})$ are radial. In the LMN the basis functions $\rho_i(\boldsymbol{\phi})$ are commonly chosen to be normalized Gaussian functions of the form

$$\rho_i(\boldsymbol{\phi}) = \left[\exp\left(\frac{\|\boldsymbol{\phi} - \boldsymbol{c}_i\|^2}{2\delta_i^2}\right)\right] \Big/ \left[\sum_{i=1}^{N_M} \exp\left(-\frac{\|\boldsymbol{\phi} - \boldsymbol{c}_j\|^2}{2\delta_j^2}\right)\right]. \tag{28}$$

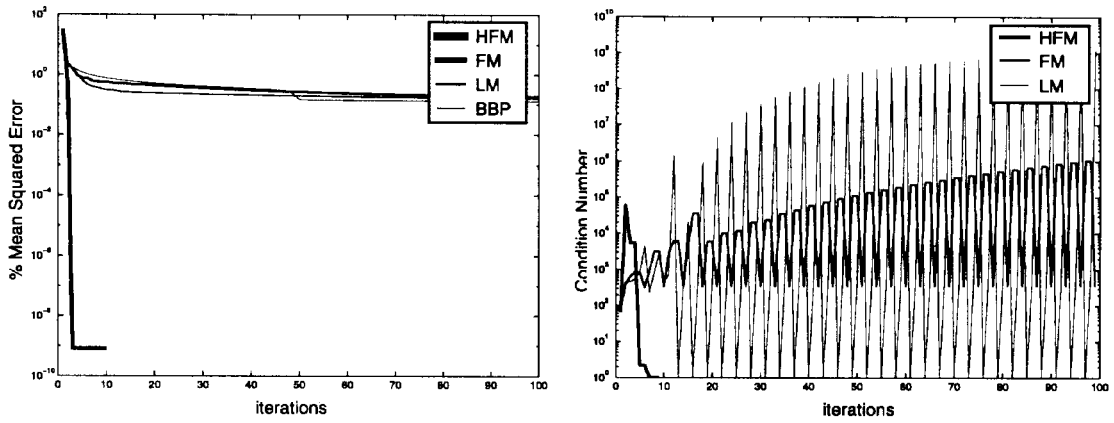
Thus the $\rho_i(\boldsymbol{\phi})$ can be interpreted as functions that give a value close to one in regions of $\boldsymbol{\phi}$ where the local f_i is a good approximation to F, and a value of zero elsewhere. They sum to one at every point in $\boldsymbol{\phi}$, thus ensuring that at least one model remains valid outside the normal operating range. The underlying assumption for the LMN strategy is that the systems to be modeled undergo significant changes in operating conditions. For most batch and continuous processes in the chemical, biotechnological, and power industries, definite regimes can be identified during procedures such as start-up, shut-down, and product shifts. The transparency of the overall model can be improved by employing simpler models in each operating region. For example, local state-space, and ARMAX models can be formed and then interpolated to give global nonlinear state-space and NARMAX models [12].

TABLE III
COMPARISON OF ITERATION TIMES AND WEIGHT MAGNITUDES FOR HFM AND FM TRAINING

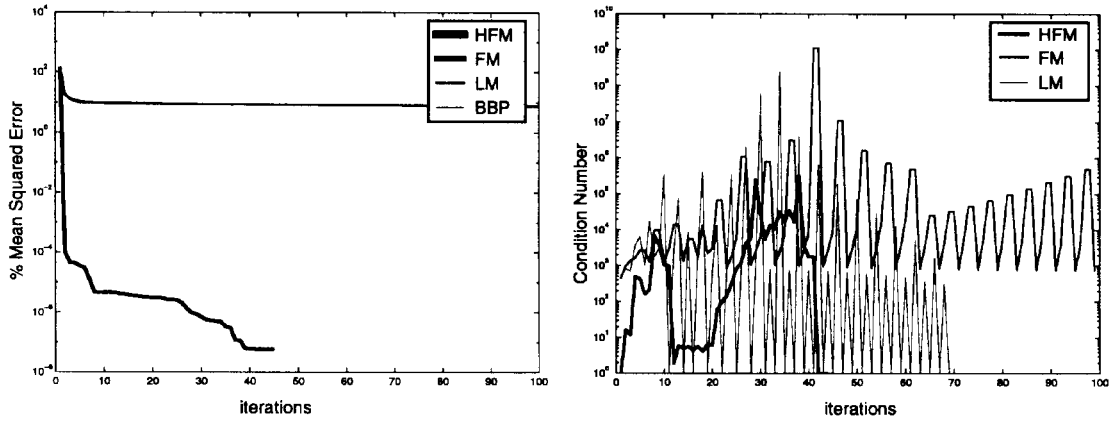
Comparison between FM and HFM solutions and iteration times.				
Problem No.	1	2	3	4
Linear/nonlinear weights ratio	5/10 0.5	10/20 0.5	10/40 0.25	10/400 0.025
Iteration time ratio (HFM/FM)	1.517	1.402	1.302	0.949
Weight magnitude ratio (HFM/FM)	61.32	40.77	46.8	10^5

The identification of local operating regimes for an unknown plant is difficult. Any such identification strategy has to take into account the complexity of the target mapping, the representational ability of the local models associated with the basis functions and the availability of the data. The problem is therefore to identify those variables which describe the system operating behavior. A *priori* knowledge of the plant can be used at this stage. When little knowledge of the actual regimes exists, however, it may be beneficial to use unsupervised learning methods, such as k-means clustering and nearest neighbors, to give an initial estimate of the normalized Gaussian activation regions. These clustering methods are valid in this case since many plants tend to operate in distinct regions. Despite these difficulties, successful applications of LMN's have been reported in the biotechnology, chemical engineering and automotive industries [3], [13], [14], [21].

The hybrid optimization strategy described in Section II can be extended to the LMN by considering the centers and widths of the normalized Gaussian interpolation functions as the nonlinear weights in the network, and the parameters of the local linear models as the linear weights. Thus the hybrid



(a)



(b)

Fig. 7. Training algorithm performance for RBF networks. (a) Problem 1, RBF (1,5,1). (b) Problem 2, RBF (1,10,1).

cost function in (15) for the LMN is given by

$$E(\boldsymbol{w}_N, \boldsymbol{w}_L^*) = \sum_{j=1}^{N_v} \left[\sum_{i=1}^{N_M} f_i(\boldsymbol{w}_L^*, \boldsymbol{\psi}_j) \rho_i(\boldsymbol{w}_N, \boldsymbol{\phi}_j) - d_j \right]^2 = \sum_{j=1}^{N_v} [F_j - d_j]^2 \quad (29)$$

The gradient with respect to the cost function is given as

$$\frac{\partial E}{\partial \boldsymbol{w}_N} = 2 \sum_{j=1}^{N_v} [F_j - d_j] \frac{\partial F_j}{\partial \boldsymbol{w}_N}. \quad (30)$$

Partial differentiation with respect to $\boldsymbol{w}_N$ of the LMN output for the jth training vector (assuming normalized Gaussian activation functions and local linear models) produces the expressions

$$\frac{\partial F_j}{\partial c_{mk}} = \left[\frac{\phi_{jk} - c_{mk}}{\delta_m^2} \right] \rho_m(\boldsymbol{\phi}_j)[f_m(\boldsymbol{\psi}_j) - F_j] \quad (31)$$

$$\frac{\partial F_j}{\partial \delta_{mk}} = \left[\frac{||\phi_{jk} - c_{mk}||^2}{\delta_m^3} \right] \rho_m(\boldsymbol{\phi}_j)[f_m(\boldsymbol{\psi}_j) - F_j] \quad (32)$$

where $j = \{1, \cdots, N_v\}, k = \{1, \cdots, N_k\}$ and $m = \{1, \cdots, N_M\}$.

IV. Performance Evaluation

The following test problems will be used to illustrate the performance of the new hybrid training approach.

Problem 1: A single-input/single-output (SISO) feedforward network trained to represent the quadratic function $y = (x - 2)(x + 1)$ in the range $[-3, +3]$. The training set consists of 25 uniformly distributed vectors (i.e., $[x, y]$ points).

Problem 2: A SISO feedforward network trained to represent the one-dimensional mapping $y = x \cdot \sin(x) + 1.5$. The training set consists of 25 points drawn uniformly from x and y in the range $[-5, +5]$).

Problem 3: A two-input/one-output feedforward network trained to represent the three-dimensional mapping $z = x \cdot \sin(x) + y \cdot \cos(y) - 0.75$. The training set consists of 225 points drawn uniformly from x and y in the range $[-2, +2]$).

Problem 4: A 20-input/one-output feedforward network trained as a five-step ahead neural predictive model of a continuous stirred tank reactor (CSTR) [15]. The CSTR is a highly nonlinear SISO plant and is a useful example for testing neural modeling techniques. The output is the product concentration with the input being the flow rate of a coolant. The chemical reaction is exothermic, which raises the temperature and hence reduces the reaction rate.

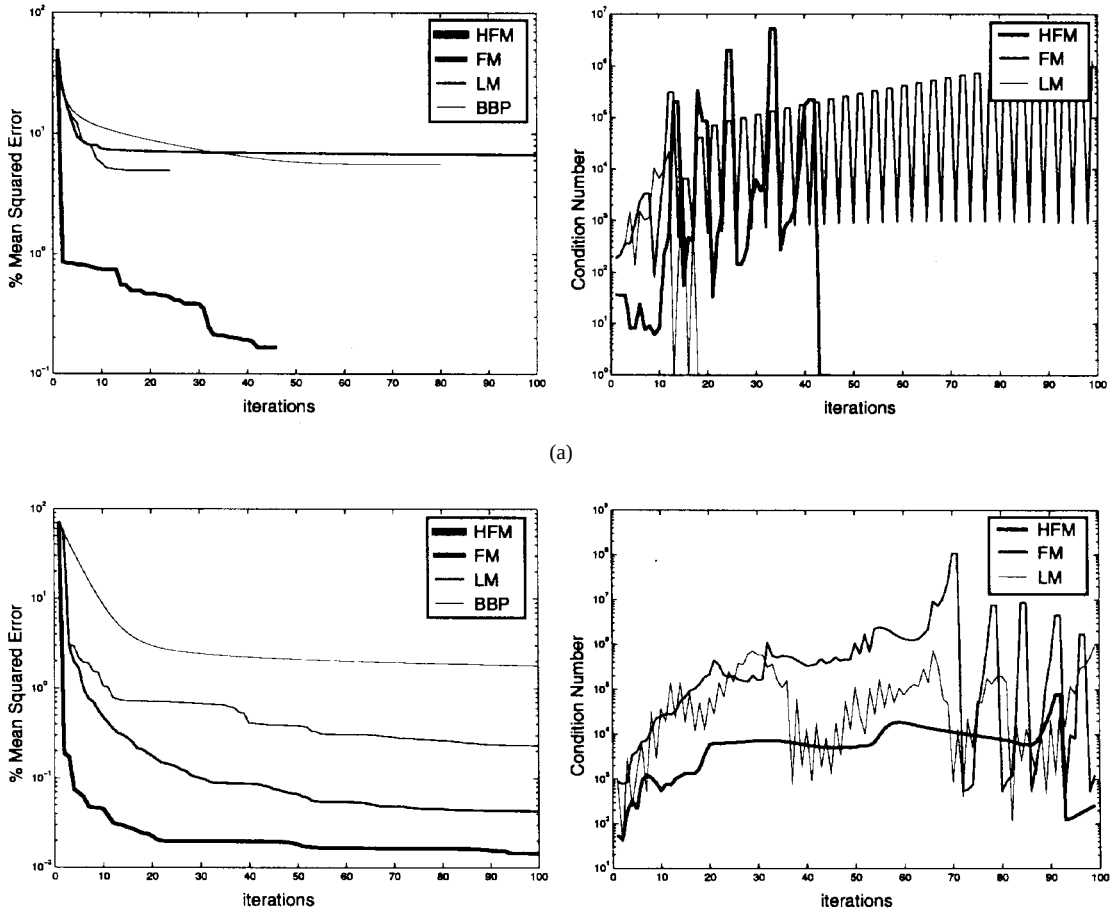


Fig. 8. Training algorithm performance for RBF networks. (a) Problem 3, RBF (2,10,1). (b) Problem 4, RBF (20, 10,1).

The introduction of a coolant allows the manipulation of the reaction temperature and hence the product concentration can be controlled. The reaction takes place in a container of fixed volume and the product flow rate, input concentration, temperature, and output flow rate are assumed constant at their nominal values (Fig. 2). The plant has a time delay (d) of 30 s, consisting of a transport delay at the output and a measurement delay when determining the product concentration. A simulation of this plant was used to generate training data of 400 vectors for the purpose of producing neural predictive plant models.

A. MLP Networks

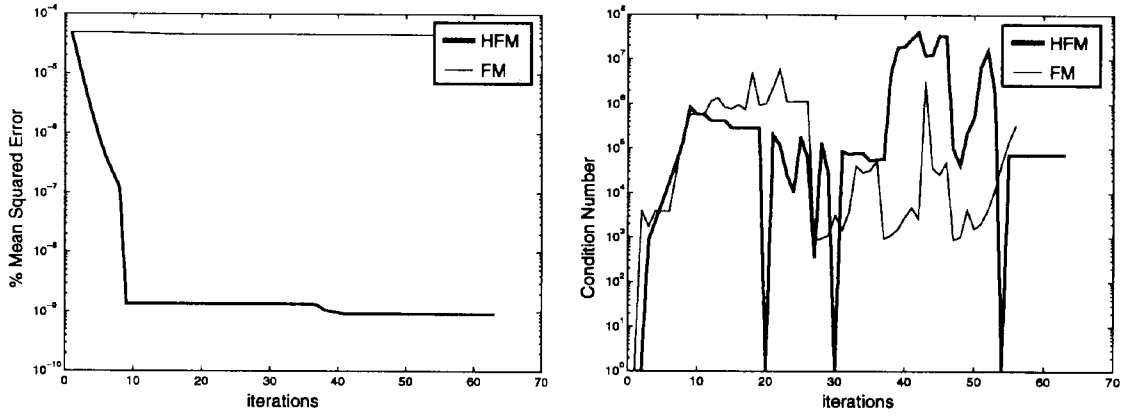
Figs. 3–6 compare the performance of the hybrid FM (HFM), FM, LM, and BBP algorithms for the four test problems outlined above when using MLP networks. In each case plots of the learning curves and condition number are provided for two different sizes of network. The condition number in question is the condition number of the inverse Hessian approximation used by the BFGS based methods. While this is only relevant for the FM and HFM techniques, it has also been included for the one-step Hessian approximation used by the LM algorithm for completeness. The percentage mean squared error (%MSE) used in the learning curve plots is a normalized form computed as

$$\%\text{MSE} = 100 \cdot \frac{d_{\max} - d_{\min}}{N_V} \cdot \sum_{i=1}^{N_v} (y_i - d_i). \qquad (33)$$

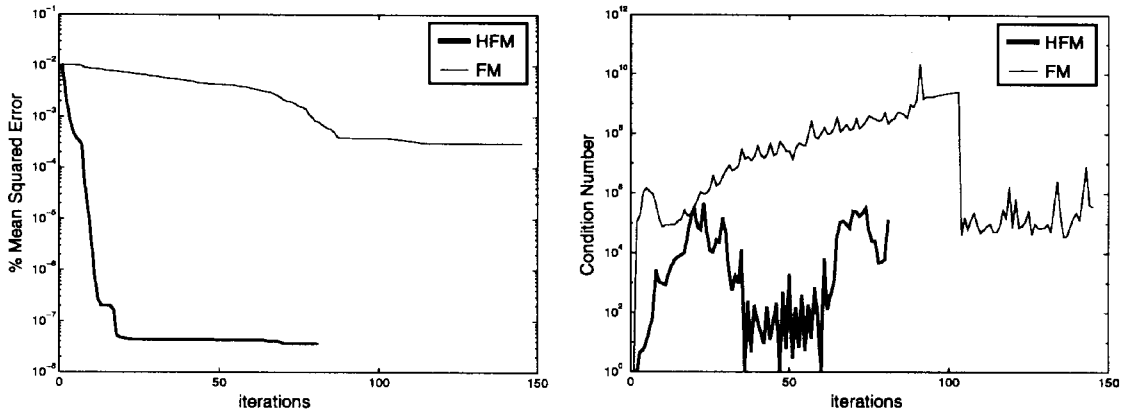
Here $d_{\max}$ and $d_{\min}$ are the maximum and minimum values of the training data and y_i, d_i, and N_v are as previously defined (1). The learning curves highlight the rate of convergence of each algorithm, while the condition number plots illustrate the link between conditioning and algorithm performance.

These results clearly demonstrate that HFM training is vastly superior to both FM and LM, in terms of rate of convergence and the final error level achieved, whenever the number of linear parameters is comparable to the number of nonlinear parameters (see Table I). When this is not the case, as in Problem 4, performance is similar to that of FM. In Problems 1 and 2 HFM achieves a huge error reduction in the first few iterations which reflects the excess dynamic capability within the networks in these examples. In these situations it is possible to train MLP's using only linear optimization of the output layer weights. With smaller networks a more gradual error reduction is achieved, indicating that both the linear and nonlinear weights are being adapted.

The condition number plots confirm that, in general, MLP training problems are extremely ill-conditioned. For the FM



(a)



(b)

Fig. 9. Training algorithm performance for LMN's. (a) LMN (1–1,3,1). (b) LMN (1–1, 6,1).

and HFM algorithms, the condition number for the full weight space (FM) is significantly greater than that for the nonlinear weights subspace (HFM), in many problems, and this is reflected in the rates of convergence of the two algorithms.

The ratios of the average iteration times for the HFM and FM algorithms are presented in Table II. As expected, HFM iterations take significantly longer than FM iterations due to the SVD computation, but the vastly superior rate of convergence more than compensates for this.

The third row of this table shows the ratio between the average magnitude of the linear weight solutions generated by HFM and FM for each problem. In general the linear weights generated by HFM are many orders of magnitude greater than those of the FM algorithm. This characteristic arises because SVD is able to exploit the dynamic flexibility achievable from the addition of opposite sign, large magnitude basis functions to give accurate fits to the underlying function. A theoretical explanation and an illustrative example of this phenomenon can be found in [19]. A similar effect is not seen with the other algorithms because the linear weights change much more slowly, giving the algorithms time to adapt the nonlinear parameters of the basis functions to compensate. Large magnitude solutions may be unacceptable because they are subject to numerical instability and are highly sensitive to parameter variations. They can be prevented in many problems by limiting the number of neurons so that linear optimization alone does not give good results. Alternatively, FM can be used to train the network for an initial period so that the nonlinear weights have reasonable values before HFM is applied. The most effective solution however is to employ regularization to penalize large magnitude weights. This strategy, which is discussed in detail in [19] has the added advantage of generating solutions with good generalization properties.

B. RBF Networks

Figs. 7 and 8 compare the training performances of the BBP, LM, FM, and HFM algorithms for Problems 1–4, when RBF's, are employed to form the mappings. These results are typical of those obtained when attempting to train RBF's networks using nonlinear optimization and clearly show that even the highly efficient FM algorithm has great difficulty in training RBF networks, performing only marginally better than BBP in most problems. The HFM algorithm on the other hand works extremely well with RBF networks and is thus the preferred method for this network paradigm.

Table III shows the linear/nonlinear weight ratios, iteration time ratios, and average linear weight magnitude ratios for the RBF networks trained using the HFM and FM algorithms. With the exception of Problem 4, the magnitudes of the

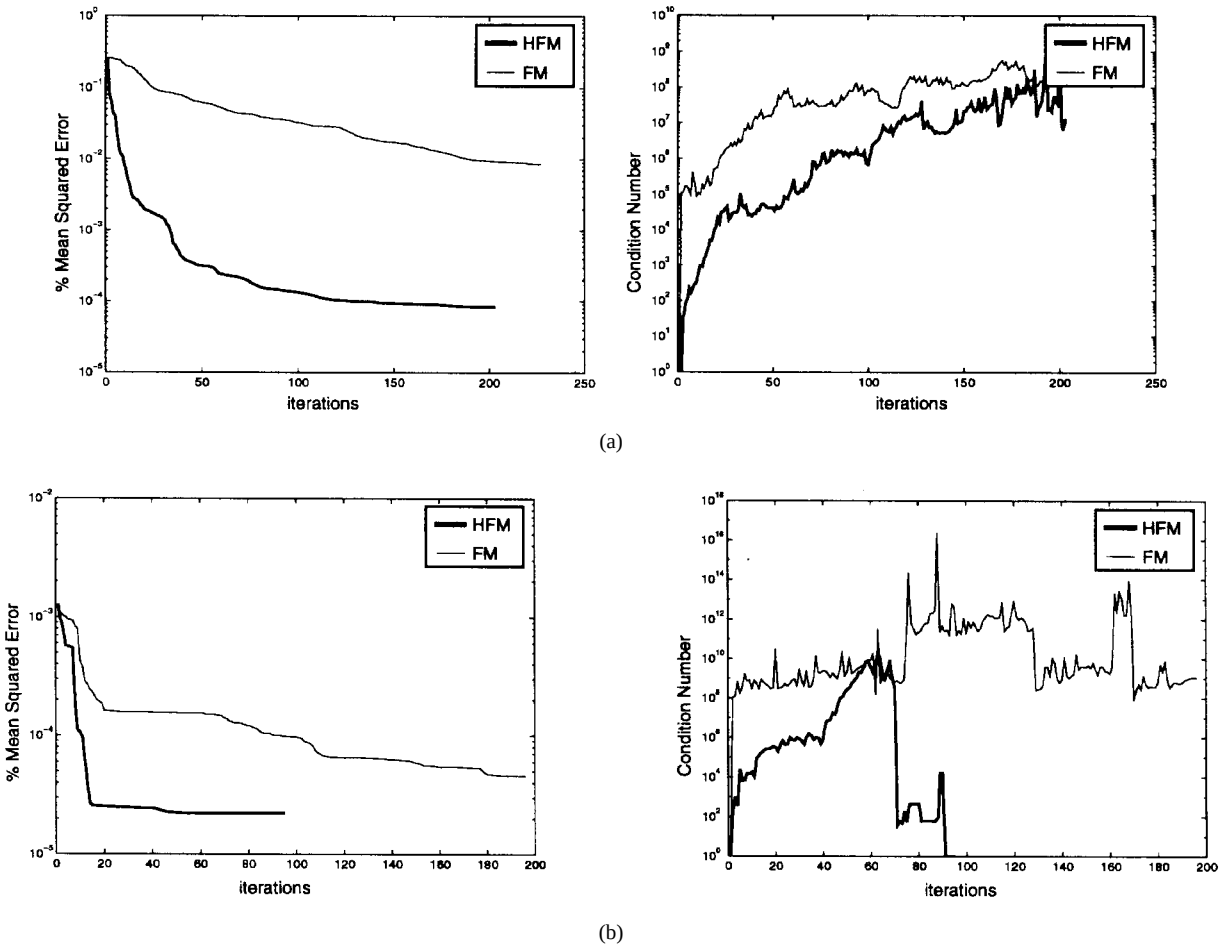


Fig. 10. Training algorithm performance for LMN's. (a) LMN (2.2, 6,1). (b) LMN (6–1, 3,1).

TABLE IV
TRAINING RESULTS FOR THE PROBEN BENCHMARK PROBLEMS

Problem Details	Alg.	Training Set		Validation Set		Test Set	
		mean	stddev	mean	stddev	mean	stddev
MLP(14,10,3) on *build1a*	BBP	0.6090	0.8640	2.9263	4.0248	1.7360	2.5239
	LM	0.2496	0.4169	1.4469	1.9564	1.0000	1.0512
	FM	0.1449	0.3186	0.8567	1.4768	0.7190	1.0512
	HFM	0.0872	0.2660	0.7447	1.3746	0.6340	1.0635
MLP(24,10,3) on *flare1a*	BBP	0.5255	2.9944	0.4926	2.0164	0.7292	4.0734
	LM	0.4040	2.9854	0.3396	2.3588	0.5773	3.8132
	FM	0.3376	2.0341	0.3924	2.1730	0.5733	3.8132
	HFM	0.3619	2.4753	0.3446	2.2107	0.5215	3.3629
RBF(14,10,3) on *build1a*	BBP	0.5302	0.8367	3.1576	4.2830	1.8291	2.7111
	LM	0.3281	0.6030	1.9222	3.0842	1.2734	2.1583
	FM	0.2167	0.3789	1.2180	1.7426	0.8801	1.3745
	HFM	0.2012	0.3728	0.9847	1.5059	0.8322	1.2745
RBF(24,10,3) on *flare1a*	BBP	0.4439	3.4128	0.4073	2.6539	0.6234	4.1582
	LM	0.4446	3.5868	0.3832	2.9019	0.6344	4.5187
	FM	0.4079	3.2012	0.3301	2.3135	0.5717	4.0450
	HFM	0.3758	2.9052	0.3140	1.9480	0.5420	3.6621

weights generated by HFM are much less than those obtained for MLP's. The opposite is true for Problem 4 because, in this example, the underlying mapping is a smooth, slowly varying function. In general MLP's are more suited to this type of approximation problem, while RBF networks are more suited to the approximation of functions with large variations.

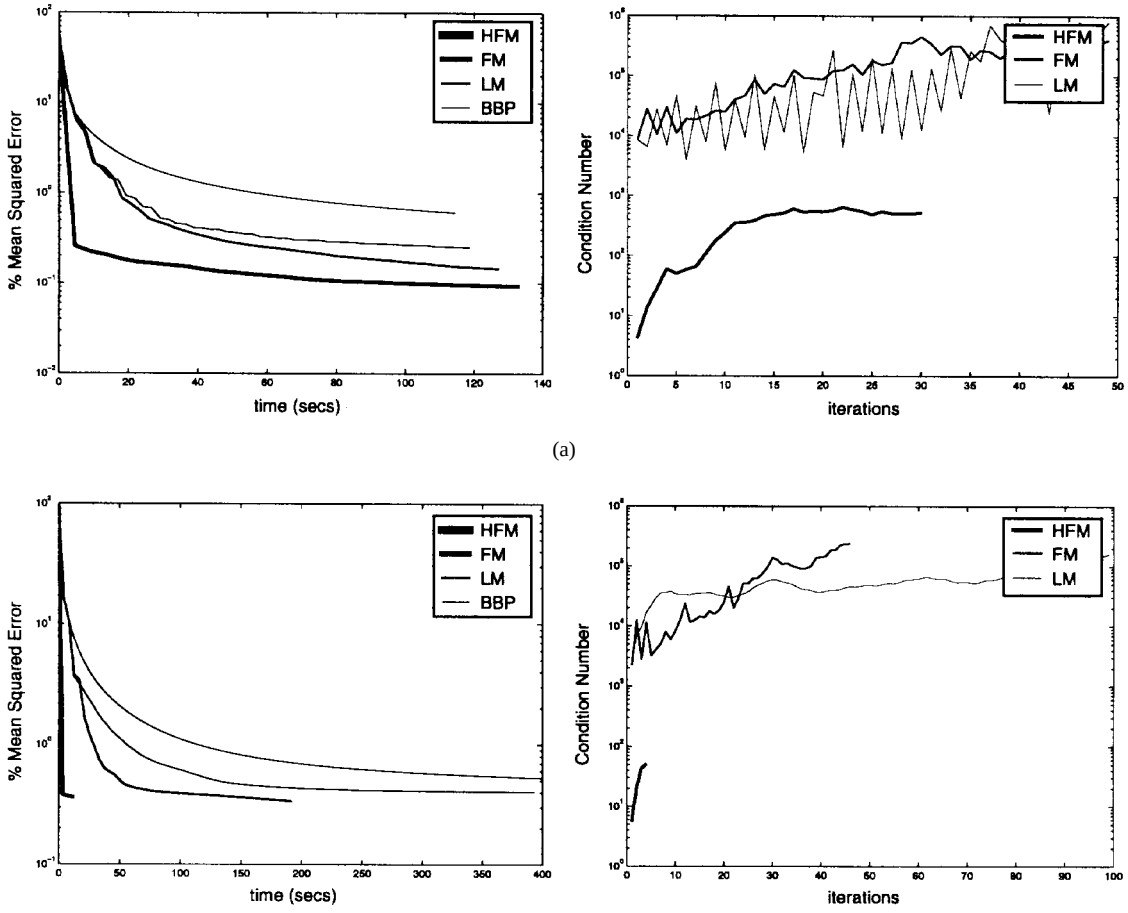


Fig. 11. Training algorithm performance on Proben benchmark problems. (a) MLP (14, 10, 3) on *building1a*. (b) MLP (24, 10, 1) on *flare1a*.

Note also that in Problem 4 the average iteration time for HFM is less than that for FM. This occurred because the repeated failures of the BFGS update in this problem result in a significant increase in the average number of function evaluations needed by the line search algorithm. Failure is indicated by spikes in the condition number plot which occur every time the M matrix is reset.

The poor performance obtained with most of the training algorithms is related to the fact that the RBF network structure is inherently extremely ill-conditioned [18]. These difficulties are overcome to some extent by the hybrid optimization scheme because it effectively separates the variables into groups which form subproblems that have better conditioning.

C. Local Model Networks

Figs. 9 and 10 show some typical results when using the HFM and FM algorithms to train LMN's. Again, the hybrid method converges much more rapidly than FM and achieves a much lower error level for most problems. The variation in condition number of the respective optimization problems during training is also shown. As expected there is a strong correlation between the condition number and each algorithm.

Here the modeling process is more complex since a measurement vector $\boldsymbol{\psi}$ and an operating region vector $\boldsymbol{\phi}$ have to be defined in each case. For the static mapping examples (Problems 1–3) these are simply the input variable(s) of the functions. In the modeling example (Problem 4) three third-order models are scheduled on the current concentration. The structural notation $(n_1 - n_2, n_3, n_4)$ used with LMNS thus indicates the dimension of the measurement vector, the dimension of the operating region vector, the number of local models and the number of outputs, respectively. Further details on the design process involved in modeling with LMN's can be found in [3].

D. Benchmark Evaluation

Results are presented here for *building1a* and *flare1a*, two benchmark neural-network training problems taken from the Proben benchmark suite [23]. The former is an energy consumption prediction task characterized by a 14 input, three output, 4208 training vector dataset, while the latter is a 24 input, three output, 1066 training vector dataset describing a solar flare prediction problem. Details of each benchmark, along with guidelines for training and presentation of results, can be found in [23].

Following these guidelines, MLP and RBF networks were trained on each data set using the BBP, LM, FM, and HFM algorithms. In each case the minimum mean and standard deviation of the percentage squared error on the training,

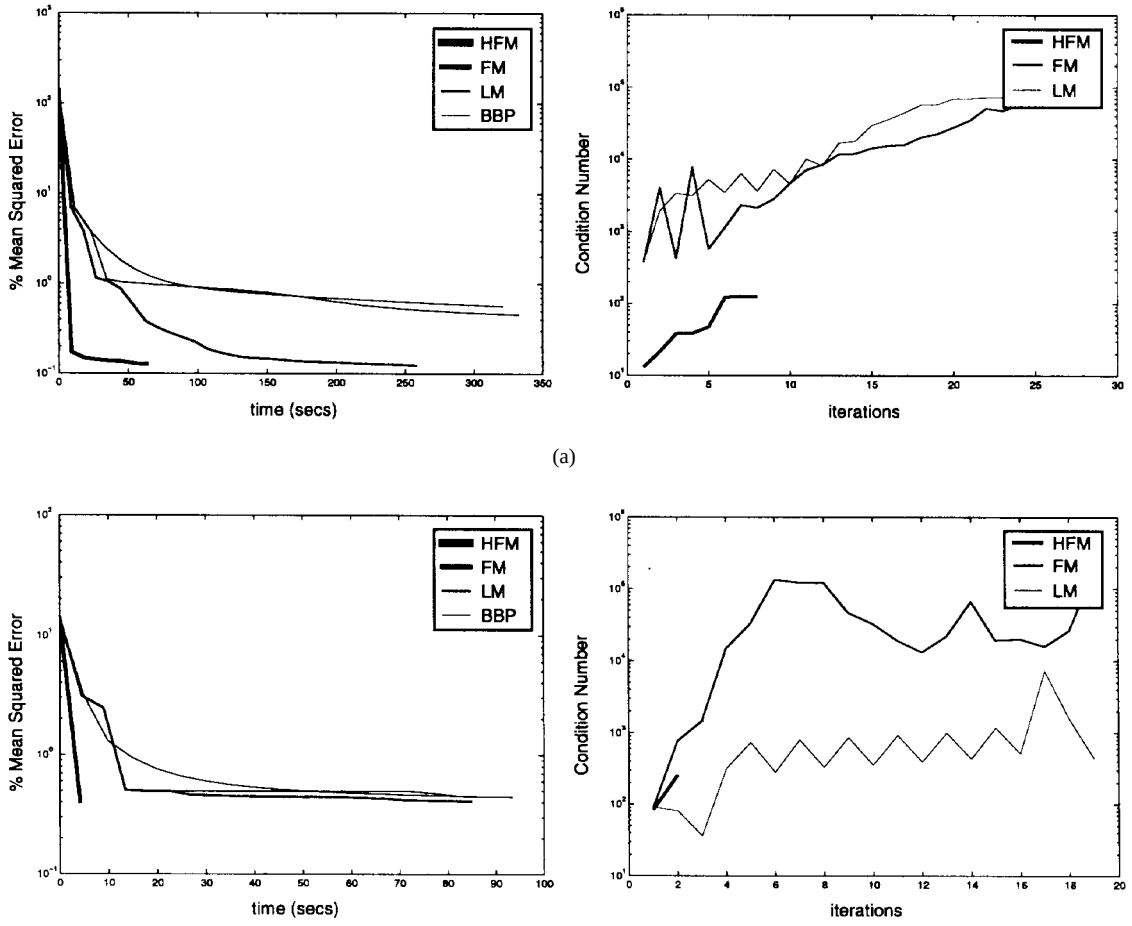


Fig. 12. Training algorithm performance on Proben benchmark problems. (a) RBF (14, 10, 1) on *building1a*. (b) RBF (24, 10,1) on *flare1a*.

validation, and test sets were recorded and are included in Table IV below. The corresponding learning curves and associated condition number plots are given in Fig. 11 for the MLP's and Fig. 12 for the RBF's. Note that in these examples the learning curves are plotted against time to show explicitly the speed-ups achievable with the new HFM approach.

As with the previous examples, HFM convergences much more quickly than the other techniques for both benchmark problems and for both networks. Furthermore, as expected, the condition number of the inverse Hessian approximation is consistently less in HFM than in FM. Table IV shows that generalization performance is also better when using HFM. However, note that these results may not be optimal since the stopped minimization approach, normally used to determine optimum generalization, is not particularly effective with HFM because of its speed of convergence. In these circumstances regularization, a much more elegant method for controlling generalization, can be applied [26]. The extension of regularization techniques to the HFM algorithm is described in [19].

V. Conclusions

A new general hybrid training algorithm has been proposed for feedforward neural networks in which nonlinear optimization of the hidden layer weights and linear optimization of the output layer weights are combined in one algorithm. While the complexity of HFM is much greater than FM, because of the need for SVD computations every iteration, the benefit is vastly improved convergence properties. This has been demonstrated for MLP, RBF, and local model networks. Indeed, HFM was the only algorithm capable of giving good results with RBF networks which are inherently extremely ill-conditioned. The structure of HFM in which nonlinear and linear are combined in one algorithm is ideally suited to the training of LMN's where linear models are used as local models.

References

[1] R. Battiti and F. Masulli, "BFGS optimization for faster automated supervised learning," in *Int. Neural-Network Conf.*, vol. 2, 1990, pp. 757–760.

[2] R. Battiti, "First- and second-order methods for learning: Between steepest descent and newton's method," *Neural Comput.*, vol. 4, pp. 141–166, 1992.

[3] M. D. Brown, G. W. Irwin, and G. Lightbody, "Local model networks for nonlinear system identification," in *Proc. 11th IFAC Syst. Identification Conf., SYSID '97*, Fukuoka, Japan, vol. 2, July 1997, pp. 709–714.

[4] C. L. Chen, W. C. Chen, and F. Y. Chang, "Hybrid learning algorithm for Gaussian potential function networks," *Proc. Inst. Elect. Eng.*, vol. 140, pt. D, no. 6, pp. 442–448, 1993.

[5] S. Chen, S. A. Billings, and P. M. Grant, "Recursive hybrid algorithm for nonlinear system identification using radial basis function networks," *Int. J. Contr.*, vol. 55, no. 5, pp. 1051–1070, 1992.

[6] S. Chen, C. F. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.
[7] S. E. Fahlman, "Fast learning variations on backpropagation: An empirical study," in *Proc. 1988 Connectionist Models Summer School*, D. S. Touretzky, G. Hinton, and T. Sejnowski, 1988, pp. 38–51.
[8] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[9] S. J. Huang and S. L. Huang, "Static security assessment of a large-scale power system using genetic-enhanced neural-network approaches," in *Proc. Nat. Sci. Council*, China, Part A: Physical Science and Engineering, vol. 20, no. 2, Mar. 1996, pp. 228–235.
[10] A. Jacobs, "Increased rates of convergence through learning adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.
[11] T. A. Johansen and B. A. Foss, "Constructing NARMAX models using ARMAX models," *Int. J. Contr.*, vol. 58, no. 5, pp. 1125–1153, 1993.
[12] ——, "State-space modeling using operating regime decomposition and local models," in *Proc. IFAC World Congr.*, vol. 4, no. 251, pp. 39–42, 1993.
[13] ——, "Identification of nonlinear system structure and parameters using regime decomposition," *Automatica*, vol. 31, no. 2, pp. 321–326, 1995.
[14] T. A. Johansen, B. A. Foss, and A. V. Sorensen, "Nonlinear predictive control using local models—applied to a batch fermentation process," *Contr. Eng. Practice*, vol. 3, no. 3, pp. 389–396, 1995.
[15] G. Lightbody, "Identification and control using neural networks," Ph.D. dissertation, Advanced Control Engineering Research Group, Queen's Univ. Belfast, U.K., May 1993.
[16] S. McLoone and G. W. Irwin, "Fast gradient based off-line training of multilayer Perceptrons," in *Neural-Network Engineering in Dynamic Control Systems*, K. J. Hunt, G. W. Irwin, and K. Warwick, Eds. New York: Springer-Verlag, 1995.
[17] S. F. McLoone and G. W. Irwin, "Fast parallel off-line training of multilayer perceptrons," *IEEE Trans. Neural Networks*, vol. 8, pp. 646–653, May 1997.
[18] S. F. McLoone, "Neural-Network Training for Modeling and Control," Ph.D. dissertation, Advanced Control Engineering Research Group, Queen's University of Belfast, U.K., Sept. 1996.
[19] S. F. McLoone and G. W. Irwin, "Improving neural-network training solutions using regularization," submitted to *Neurocomputing*, 1997.
[20] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525–533, 1993.
[21] O. Nelles, S. Sinsel, and R. Isermann, "Local basis function networks for identification of a turbocharger," in *Proc. UKACC Int. Conf. Contr. '96*, Exeter, U.K., July 1996, pp. 7–12.
[22] A. G. Parlos, B. Fernandez, F. Atiya, J. Muthusami, and W. K. Tsai, "An accelerated learning algorithm for multilayer perceptron networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 493–397, 1994.
[23] L. Prechelt, "Proben 1—A set of neural network benchmark problems and benchmarking rules," Tech. Rep. 21/94, Fakultät für Informatik, Univ. Karlsruhe, Germany, Sept 1994. Available FTP: ftp.ira.uka.de in /pub/neuron/proben1.tar.gz
[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., vol. 1. Cambridge, MA: MIT Press, 1986, ch. 8.
[25] W. Schiffmann, M. Joost, and R. Werner, "Optimization of the backpropagation algorithm for training multilayer perceptrons," Univ. Koblenz, Inst. Physics, Rheinau 3-4, W-5400 Koblenz, Germany.
[26] J. Sjöberg and L. Ljung, "Overtraining, regularization, and searching for minimum with application to neural networks," *Int. J. Contr.*, vol. 62, no. 6, pp. 1391–1407.
[27] T. Ichimura, T. Takano, and E. Tazaki, "Learning of neural networks using hybrid genetic algorithm," in *4th European Congr. Intell. Techniques and Soft Computing (EUFIT'96)*, 1996.
[28] P. Van Der Smagt, "Minimization methods for training feedforward neural networks," *Neural Networks*, vol. 7, no. 1, pp. 1–11, 1994.
[29] P. D. Wasserman, *Neural Computing, Theory and Practice*. New York: Van Nostrand Reinhold, 1989.
[30] P. B. Watta, M. H. Hassoun, and J. Meisel, "Design of optimal neurocontrollers for the separately excited dc motor using a hybrid genetic algorithm-neural network approach," in *Proc. SPIE—Int. Soc. Opt. Eng.*, vol. 2760, pp. 230–241, 1996.
[31] A. R. Webb, D. Lowe, and M. D. Bedworth, "A comparison of nonlinear optimization strategies for feedforward adaptive layered networks," *Royal Signals and Radar Establishment*, Memorandum 4157, Controller HMSO, London, U.K., 1988.
[32] L. F. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections," *IEEE Trans. Neural Networks*, vol. 3, pp. 899–905, 1992.
[33] M. J. Willis, C. Di Massimo, G. A. Montague, and G. A. Tham, "Artificial neural networks in process engineering," *Proc. Inst. Elect. Eng., Control Theory and Applicat.*, vol. 128, no. 3, pp. 256–204, 1991.

Seán McLoone received the M.Eng. degree with distinction in electrical and electronic engineering from the Queen's University of Belfast, U.K., in 1992, and the Ph.D. degree in control engineering from the same university in 1996.

From March 1995 to April 1996 he worked as a Research Fellow in the School of Electrical Engineering and Computer Science, Queen's University Belfast conducting research on modeling and control of power station plant using neural-network techniques. In December 1996 he joined the Advanced Control Engineering Research Centre at Queen's as a Postdoctoral Research Fellow, investigating the application of advanced modeling and multivariable statistical process control techniques to quality control and predictive maintenance of chemical processes, before taking up his current appointment as Lecturer in Control Engineering at the university in January 1998. His current research interests include nonlinear system identification, neural-network training, parallel computing, optimization, nonlinear adaptive control, statistical process control, and fault diagnosis.

Michael D. Brown received the B.Eng. and Ph.D. degrees in electrical engineering from the Queen's University of Belfast, U.K., in 1988 and 1991, respectively.

From 1991 to 1994 he was a SERC Postdoctoral Research Fellow at Queen's University, studying parallel adaptive control of generator systems and neural-network modeling of power station boilers. From July 1994 to July 1995 he was a Senior Applications Engineer with GE power systems in New York, involved in various industrial projects relating to torsional effects on turbine-generator sets. Subsequently, he spent two years with Queen's University, studying local model neural networks applied to industrial processes. He is currently a Lecturer in Control Systems in the School of Mechanical Engineering of the University of Leeds, U.K. His research interests include neural networks, optimization, modeling and control of power plant, automotive systems, and industrial systems.

George Irwin (M'83–SM'89) received the bachelor's degree with 1st class honors in electrical and electronic engineering in 1972 and the Ph.D. degree in control theory in 1976 from The Queen's University of Belfast, U.K.

Returning to Queen's in 1980, he was awarded a personal chair in Control Engineering in 1989. His current research, much of which is in collaboration with industry, includes identification and control using neural networks, fuzzy neural systems and local model networks, with applications in the aerospace, chemical, and electric power fields. He has over 200 publications, including six edited research books.

Dr. Irwin was awarded prizes for journal papers including four IEE Premiums (1985, 1987, 1991, 1996) and the 1995 Honeywell Prize from the U.K. Institute of Measurement and Control (1995). He is a member of the Control and Instrumentation College of the U.K. Engineering and Physical Sciences Research Council. He is the current chair of the Institute of Electrical Engineers Control division and chairs the Northern Ireland Centre of the IEE. He is a Chartered Engineer, a Fellow of the IEE, and a Fellow of the Institute of Measurement and Control. In addition to being Editor-in-Chief of *Control Engineering Practice*, he serves on the International Editorial Board of the *IEE Proceedings on Control Theory and Applications* and has wide experience on the program and organizing committees of international conferences. He is active in two IFAC Technical Committees, "Algorithms and Architectures for Real-time Control" and "Fuzzy and Neural Systems," and is an executive committee member of the U.K. Automatic Control Council.

Gordon Lightbody received the M.Eng. degree with distinction in electrical and electronic engineering from Queen's University, Belfast, U.K., in 1989. He received the Ph.D. degree in 1993 for research which investigated the application of neural networks for nonlinear modeling and control.

After completing a one-year postdoctoral position funded by DuPont (U.K.), in which neural modeling techniques were successfully used for polymer viscosity prediction, he was recruited by Queen's University as a Lecturer in Modern Control Systems. At Queens, his research focused primarily on the application of intelligent control to the chemical process industry. In 1997 he joined University College Cork, Ireland, as a College Lecturer in control engineering. His current research interests include local model networks for process modeling and control, nonlinear model based predictive control, fuzzy/neural systems, and nonlinear control. He has published more than 30 papers in these areas.

Dr. Lightbody has received prizes at ACC 1995, UKACC Control 1994, and an honorable mention at the IFAC World Congress, 1996. He was a member of the Applied Control Techniques committee (C9) of the Institute of Electrical Engineers from 1996 to 1997, and has recently joined the new Intelligent Control Committee (B4) of the IEE.