ELSEVIER

Neurocomputing 43 (2002) 51–75

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Neural network models in greenhouse air temperature prediction

P.M. Ferreira[a,*], E.A. Faria[b], A.E. Ruano[a,c]

[a] *Faculdade de Ciências e Tecnologia, Universidade do Algarve, Campus de Gambelas, 8000-Faro, Portugal*

[b] *Faculdade de Engenharia de Recursos Naturais, Universidade do Algarve, Campus de Gambelas, 8000-Faro, Portugal*

[c] *Instituto de Sistemas e Robótica, Universidade de Coimbra, 3030 Coimbra, Portugal*

Abstract

The adequacy of radial basis function neural networks to model the inside air temperature of a hydroponic greenhouse as a function of the outside air temperature and solar radiation, and the inside relative humidity, is addressed. As the model is intended to be incorporated in an environmental control strategy both off-line and on-line methods could be of use to accomplish this task. In this paper known hybrid off-line training methods and on-line learning algorithms are analyzed. An off-line method and its application to on-line learning is proposed. It exploits the linear–non-linear structure found in radial basis function neural networks.

Keywords: Radial basis functions; Neural networks; Greenhouse environmental control; Modelling

1. Introduction

Feed-forward layered neural networks (NNs) have extensively been applied in many fields of engineering in order to perform some type of non-linear processing on data generated by a wide variety of systems. In the fields of modelling and identification of non-linear systems, this growing interest is due to several reasons. Some of the most general ones are that no prior knowledge about the structure of the dynamical system is needed, that *multiple-input multiple-output*

* Corresponding author.

E-mail addresses: pfrazao@ualg.pt (P.M. Ferreira), efaria@ualg.pt (E.A. Faria), aruano@ualg.pt (A.E. Ruano).

0925-2312/02/$ - see front matter
PII: S0925-2312(01)00620-8

(MIMO) systems are treated the same way as *single-input single-output* (SISO) systems, and that NNs have a strong ability to perform non-linear information processing. In such contexts NNs act as curve approximators and thus, their design process can be viewed as a curve-fitting problem in a multi-dimensional space. The nature of the mentioned design problem explains the fact that most of the applications consist in approximating unknown systems from an input–output perspective, or in the realization of non-linear decision functions to which classification problems are an example. In situations where the data generating function is a non-linear time-varying function, it is standard practice to train first the networks off-line, and subsequently to adapt the trained neural networks on-line. In this article several off-line training and on-line learning methods are compared for a type of feed-forward layered NN, which in recent years has received growing interest due to its structural simplicity: The *radial basis function* (RBF) NN. The engineering problem for which the networks will be employed is *greenhouse environmental control* (GEC). The networks will be used to model the inside air temperature in a hydroponic vegetable production greenhouse. In the following subsection the general GEC problem and the particular modelling application to which this article is dedicated are briefly introduced. In Section 2, the RBF network is presented and the compared design methods are detailed in Section 3. An explanation of the experiments carried out is done in Section 4, for which results are presented in Section 5. Finally, conclusions are drawn in Section 6.

1.1. Greenhouse environmental control

The main purpose of greenhouses is to improve the environmental conditions in which plants are grown. In greenhouses provided with the appropriate equipment these conditions can be further improved by means of climate control. Environmental factors like the inside air temperature, humidity and CO_2 concentration, to mention a few, are influenced by the heating systems, ventilators and fog systems, among others, which in turn are governed by some type of controller. A good overview of greenhouse climate control can be found in [23]. The greenhouse climate is influenced by many factors, for example the outside weather, the actuators and the crop. Methods aimed at efficiently controlling the greenhouse climate environment must take these influences into account, and that is achieved by the use of models. The design problem being considered is to model the inside air temperature of a hydroponic vegetable production greenhouse as a function of the outside solar radiation and temperature, and the inside relative humidity. This model is intended to be used in a greenhouse adaptive predictive hierarchical (in the sense of [23]) control scheme as shown in Fig. 1. In [5] it has been shown that models of this process are slowly time varying in the parameters. This finding makes both off-line and on-line methodologies important. While the initial design can be done off-line, the network will probably need on-line adaptation when employed in a real-time control system.

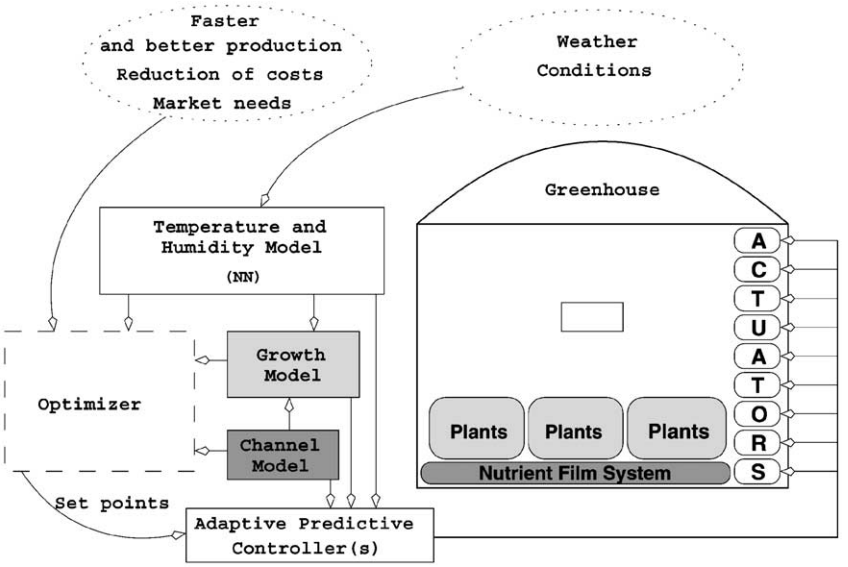

Fig. 1. Hierarchical greenhouse environmental control strategy.

Some examples of previous application of NNs to problems arising from GEC can be found in [20–22]. The *back propagation* (BP) method along with the *multi-layer perceptron* (MLP) is normally employed. RBFNNs are structurally simpler than MLPs, which makes the design and training process an easier task. They are also strongly based on function approximation theory and have a number of proven desirable properties in that line, making them very attractive for the problem in hand. For these reasons they are chosen for this study.

2. RBF neural networks

An RBFN consists of three fully connected layers. The first is the input layer connecting the source nodes to the hidden layer of the network, which is composed of a certain number of units, called neurons. The outputs of the hidden layer are then linearly combined by a set of parameters to produce the overall network response in the output layer. This way the network performs a mapping, f, from an input space, $\mathscr{X}^d$, to an output space, $\mathscr{Y}^m$. The hidden layer applies a non-linear transformation to the inputs generating a hidden space which in general has a higher dimension than $\mathscr{X}$. Given two sets of points, $X=\{x_j\in\mathscr{X}^d\}_{j=1}^N$ and $Y=\{y_j\in\mathscr{Y}^m\}_{j=1}^N$, and a function $F:\mathscr{X}^d\mapsto\mathscr{Y}^m$ such that $Y=F(X)$, the task of the NN is to construct the mapping in such a way that

$$f(x_j)=y_j,\quad j=1,\ldots,N. \tag{1}$$

With this formulation the network is constrained to pass through all the N data points, creating a *strict interpolation* (SI) problem. A known method to solve this multivariate SI problem is the *radial basis function* technique which consists in choosing f of the form

$$f(x_j) = \sum_{i=1}^{N} \alpha_i \varphi(\|x_j - x_i\|), \tag{2}$$

where $\|\cdot\|$ is a norm, usually Euclidean, and $\{\varphi(\|x_j - x_i\|)\}_{i=1}^{N}$ are a set of non-linear functions known as radial basis functions, centered at the data points x_i and weighted by a set of unknown coefficients $\{\alpha_i\}_{i=1}^{N}$. To solve real world problems it might be neither practically feasible nor desirable to have an RBF expansion of dimension N, which could be very large. In [2] an approximation to the RBF SI problem is proposed, using an expansion on the basis of smaller dimensions:

$$f(x_j) = \sum_{i=1}^{n} \alpha_i \varphi(\|x_j - c_i\|), \tag{3}$$

where $n < N$ and the $\{c_i\}_{i=1}^{n}$ are a set of points called centers which, together with the set of weights, $\{\alpha_i\}_{i=1}^{n}$, have to be chosen in order to minimize the distance, from the approximation f to the target F, stated as

$$\mathscr{E}(f) = \sum_{j=1}^{N} (y_j - f(x_j))^2. \tag{4}$$

Defining $\Lambda = [\alpha_i, \ldots, \alpha_n]^{\mathrm{T}}$ as the linear weight vector, Eq. (3) can be written in order to the weights in the following compact form:

$$\Lambda = \Phi^{+}\mathbf{y}, \tag{5}$$

where $\mathbf{y}$ is an N-by-1 vector of the desired target values and Φ is an N-by-n matrix whose elements $\varphi_{j,i}$ are the values of the radial basis functions centered at $\{c_i\}_{i=1}^{n}$ and evaluated at the points $\{x_j\}_{j=1}^{N}$. Φ^{+} denotes the pseudo-inverse of Φ. The most used function in RBFNNs is a Gaussian function of the form:

$$\varphi_i(x_j) = \mathrm{e}^{-\frac{1}{2\sigma_i^2}\|x_j - c_i\|^2}.$$

It is now clear that imposing Eq. (1) to the expansion in Eq. (3) yields an over-constrained system which no longer interpolates F in the SI sense, but instead approximates it. The *regularization network* [17] is another method of approximation also based in an expansion of RBFs, resulting from the application of *regularization theory* to the approximation problem. The quantity to be minimized is now composed of two terms

$$\mathscr{E}_{\lambda}(f) = \sum_{j=1}^{N} (y_j - f(x_j))^2 + \lambda \|Pf\|^2 \tag{6}$$

where P is a differential operator, $\|\cdot\|$ is a norm on the space where f belongs, and λ is a positive real number called the *regularization parameter*. The choice

of the operator P embodies the a priori knowledge about the solution and determines the function φ used in Eq. (3) [11]. The first term on the right-hand side of Eq. (6) is a standard error term measuring the closeness of the approximation to the real data. The second term measures the smoothness of the resulting approximation. The regularization parameter establishes a tradeoff between the two terms. It should be noted that as $\lambda \to 0$, Eq. (4) is recovered. The solution to this regularization problem was then found to be similar to Eq. (2), and again an approximation to this N dimension expansion was pursued by one in less dimensions resulting in a solution similar to Eq. (3), where the weights are now calculated by

$$\Lambda = (\Phi + \lambda I)^{+}\mathbf{y}.$$

This is called a *generalized radial basis function* (GRBF) expansion. Again, if $\lambda \to 0$, the method in Eq. (5) is recovered. From a point of view of function approximation, the regularization network has three desirable properties [17]:

- It is a *universal approximator*. The network can approximate arbitrarily well any multivariate continuous function, given a sufficiently large number of hidden nodes.
- It has the *best approximation property* [12]. Given an unknown non-linear function F, there always exists one set of unknown coefficients that approximates F better than all other possible sets.
- The computed solution is optimal in the sense that it minimizes a functional which measures how much the solution deviates from the true values given in the form of training data.

3. Design methods

3.1. Off-line training

Under the framework of batch processing using epoch learning, three methods will be compared which share some underlying ideas about the training process. The structure of an RBFNN and the nature of the neuron activation functions lead to the idea that training the output linear weights and the neuron-free parameters can be considered as different tasks to which the employment of different optimization techniques makes sense. As opposed to the neuron free parameters the output weights are linear and can be well determined using linear techniques. A multitude of methods may be applied in the determination of the neuron-free parameters but it should be noted that the values determined by a particular method have great influence on the number of hidden units needed to achieve some prescribed error performance and also on the convergence of the linear weights. In what follows two methods will be briefly described, and two variants of a third one are presented.

3.1.1. Off-line method 1

Here, a *hybrid learning* procedure composed of two stages is considered. The selection of the center locations and spreads, $\{c_i, \sigma_i\}_{i=1}^{n}$, is first carried out followed by the determination of the output weights, $\{\alpha_i\}_{i=1}^{n}$, using the linear *least squares* (LS) solution of

$$\mathscr{E} = \frac{1}{2}\sum_{i=1}^{N} e^2(i), \tag{7}$$

where

$$e(i) = t(i) - y(i).$$

Vector $\mathbf{y}$ is defined from Eq. (3) as $\mathbf{y} = [f(x_1), \ldots, f(x_N)]^{\mathrm{T}}$ and $\mathbf{t}$ is the vector of target values. Eq. (7) can also be rewritten as

$$\mathscr{E} = \tfrac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{e} = \tfrac{1}{2}\|\mathbf{t} - \mathbf{y}\|_2^2. \tag{8}$$

The first stage is implemented by a clustering procedure known as the *optimal adaptive k-means algorithm* (OAKM) [4]. The spreads of the neuron activation functions are then determined [13] using

$$\sigma_i = \frac{d_{\max}}{\sqrt{2n}}, \quad i = 1, \ldots, n, \tag{9}$$

where $d_{\max}$ is the maximum distance between the centers determined by the OAKM. This method will be referred to as *off-line method* 1 or OAKM method, throughout the article.

3.1.2. Off-line method 2

Some methods select the center locations from input data. The simplest way of doing this consists in picking randomly a desired number of centers which, if the available number of data points is not large enough and its distribution on input space is not representative of the problem on hand, can result in a larger network for a satisfactory performance and poor behavior on unseen data after the training phase. Methods aimed to build moderately sized networks with a satisfying performance implement the selection of centers based on some criterion. The *orthogonal least squares* (OLS) learning algorithm [3] selects a suitable set of center locations from the set of input data in a constructive fashion. It adds neurons to the network in order to satisfy a prescribed error performance. Again the output linear weights are afterwards computed using LS. An implementation of this learning algorithm is available through the *Neural Network Toolbox* [6] from *The MathWorks, Inc.*, and is used in this study for comparison purposes. It will be referred to as *off-line method* 2 or OLS method.

3.1.3. Off-line method 3

In this approach the center locations, the spreads of centers and the output linear weights are all determined under a *supervised learning procedure* based on unconstrained deterministic optimization. Basically, new parameter values are

calculated in an iterated manner in order to minimize the cost function (7). Let $\mathbf{u}=[\alpha_1,\ldots,\alpha_n]^{\mathrm{T}}$, $\mathbf{v}=[c_1^{\mathrm{T}},\ldots,c_n^{\mathrm{T}},\sigma_1,\ldots,\sigma_n]^{\mathrm{T}}$ and $\mathbf{w}=[\mathbf{v}^{\mathrm{T}},\mathbf{u}^{\mathrm{T}}]^{\mathrm{T}}$. The neurons output N-by-n matrix is $\mathbf{O}=[\varphi_1(x_j),\ldots,\varphi_n(x_j)]_{j=1}^{N}$. The outputs of the network can be described by Eq. (10), where the linear dependence of the network on the output weights and the dependence of $\mathbf{O}$ on $\mathbf{v}$ have been made explicit.

$$\mathbf{y}=\mathbf{O}(\mathbf{v})\mathbf{u}. \tag{10}$$

Eq. (8) now becomes

$$\mathscr{E}(\mathbf{w})=\tfrac{1}{2}\|\mathbf{t}-\mathbf{O}(\mathbf{v})\mathbf{u}\|. \tag{11}$$

The formulation presented so far involves all the network parameters in the optimization procedure. As already mentioned the output weights can be optimally determined by the LS solution. Substituting the target values vector, $\mathbf{t}$, in Eq. (10), denoting matrix $\mathbf{O}(\mathbf{v})$ by $\mathbf{A}$ and solving for $\mathbf{u}$ yields,

$$\hat{\mathbf{u}}=\mathbf{A}^{+}\mathbf{t},$$

where $\mathbf{A}^{+}$ stands for the pseudo-inverse of matrix $\mathbf{A}$. Substituting this result into Eq. (11) gives the new training criterion:

$$\mathscr{E}(\mathbf{v})=\tfrac{1}{2}\|\mathbf{t}-\mathbf{A}\mathbf{A}^{+}\mathbf{t}\|. \tag{12}$$

This new training criterion does not depend on the linear parameters, $\mathbf{u}$, and explicitly incorporates the finding that, whatever values the non-linear parameters $\mathbf{v}$ take, the $\mathbf{u}$ parameters employed are the optimal ones.

Different training algorithms can be employed to minimize (11) and (12). First-order gradient algorithms (known for MLPs as the BP algorithm) or second-order methods, such as quasi-Newton, Gauss–Newton or Levenberg–Marquardt (LM) algorithms can be employed as training algorithm. For non-linear LS problems the LM algorithm is recognized as the best method, as it exploits the sum-of-the squares characteristic of the problem [18]. Therefore, the optimization algorithm used to calculate new parameter values is the LM method [9]. Denoting the training criterion in iteration k by $\Omega(\mathbf{w}_k)$, a search direction, $\mathbf{p}_k$, in parameter space is calculated such that $\Omega(\mathbf{w}_k+\mathbf{p}_k)<\Omega(\mathbf{w}_k)$. This method is said to be of the *restricted step* type because it attempts to define a neighborhood of $\mathbf{w}_k$ in which a quadratic function, $q(\mathbf{p}_k)$, agrees with $\Omega(\mathbf{w}_k+\mathbf{p}_k)$ in some sense. The step, $\mathbf{p}_k$, is restricted by the region of validity of $q(\mathbf{p}_k)$ which is obtained by formulating in terms of $\mathbf{p}_k$ a truncated Taylor series expansion of $\Omega(\mathbf{w}_k+\mathbf{p}_k)$,

$$q(\mathbf{p}_k)\triangleq\tfrac{1}{2}\mathbf{p}_k^{\mathrm{T}}\mathbf{G}_k\mathbf{p}_k+\mathbf{g}_k^{\mathrm{T}}\mathbf{p}_k, \tag{13}$$

where $\mathbf{g}_k$ and $\mathbf{G}_k$ are, respectively, the gradient and Hessian matrix of Ω_k. It is shown in [9] that $\mathbf{p}_k$ can be obtained solving the following system:

$$(\mathbf{G}_k+v_k\mathbf{I})\mathbf{p}_k=-\mathbf{g}_k, \tag{14}$$

where the scalar v_k controls both the magnitude and direction of $\mathbf{p}_k$. The gradient, $\mathbf{g}_k$ must then be obtained as

$$\mathbf{g}_k = \frac{\partial \mathscr{E}}{\partial \mathbf{w}} = \begin{bmatrix} \dfrac{\partial \mathscr{E}}{\partial w_1} \\ \vdots \\ \dfrac{\partial \mathscr{E}}{\partial w_l} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{2}\dfrac{\partial \sum_{i=1}^{N} e_i^2}{\partial w_1} \\ \vdots \\ \dfrac{1}{2}\dfrac{\partial \sum_{i=1}^{N} e_i^2}{\partial w_l} \end{bmatrix}$$

$$= - \begin{bmatrix} \displaystyle\sum_{i=1}^{N} e_i \frac{\partial y_i}{\partial w_1} \\ \vdots \\ \displaystyle\sum_{i=1}^{N} e_i \frac{\partial y_i}{\partial w_l} \end{bmatrix} = - \mathbf{e}^{\mathrm{T}} \mathbf{J},$$

where $\mathbf{J}$ is the Jacobian matrix of the form

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial y_1}{\partial w_1} & \cdots & \dfrac{\partial y_1}{\partial w_l} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial y_N}{\partial w_1} & \cdots & \dfrac{\partial y_N}{\partial w_l} \end{bmatrix},$$

which, looking at the definition of $\mathbf{w}$, can be further decomposed as $\mathbf{J} = [\mathbf{J}_c \ \ \mathbf{J}_\sigma \ \ \mathbf{J}_\alpha]$, where

$$\mathbf{J}_\xi = \begin{bmatrix} \dfrac{\partial y_1}{\partial \xi_1} & \cdots & \dfrac{\partial y_1}{\partial \xi_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial y_N}{\partial \xi_1} & \cdots & \dfrac{\partial y_N}{\partial \xi_n} \end{bmatrix}, \quad \xi \in \{c, \sigma, \alpha\}.$$

The three derivatives needed to compute $\mathbf{J}$, with the standard formulation (7), are

$$\frac{\partial \mathbf{y}}{\partial c_i} = \varphi_i(x) \frac{\alpha_i}{\sigma_i^2} (x - c_i)^{\mathrm{T}},$$

$$\frac{\partial \mathbf{y}}{\partial \sigma_i} = \varphi_i(x) \frac{\alpha_i}{\sigma_i^3} \|x - c_i\|_2^2,$$

$$\frac{\partial \mathbf{y}}{\partial \alpha_i} = \varphi_i(x). \tag{15}$$

Using the new formulation (12), alternative Jacobian matrices are available for this problem. We use the one introduced in [19], which is

$$\mathbf{J} = (\mathbf{A})_v \mathbf{A}^{+} \mathbf{t}, \tag{16}$$

where $(\mathbf{A})_v$ is a three-dimensional quantity denoting the derivatives of the output neuron matrices with respect to the centers and spreads. Note that this Jacobian is very easy to compute, as it is given by the two first lines of (15), where α_i are replaced by their optimal values, in the least-squares sense. Actually, the use of this Jacobian matrix implies that each iteration of the LM method minimizing (12) is cheaper than minimizing (7).

In the context of non-linear LS, the second derivative matrix, $\mathbf{G}_k$, is approximated [8] using information already available in the determination of $\mathbf{g}_k$:

$$\mathbf{G}_k \approx \mathbf{J}^{\mathrm{T}}\mathbf{J}.$$

In order to measure the agreement of $q(\mathbf{p}_k)$ with $\Omega(\mathbf{w}_k + \mathbf{p}_k)$, the ratio between the actual reduction, $\Delta\Omega_k$, and the predicted reduction, $\Delta\hat{\Omega}_k$, in Ω_k is used:

$$r_k = \frac{\Delta\Omega_k}{\Delta\hat{\Omega}_k}.$$

The actual and predicted reduction values are obtained as follows. The predicted error value after taking step $\mathbf{p}_k$ is

$$\hat{\mathbf{e}}_k = \mathbf{e}_k - \mathbf{J}_k\mathbf{p}_k.$$

This way the prediction reduction in Ω_k becomes

$$\Delta\hat{\Omega}_k = \Omega(\mathbf{w}_k) - \frac{\hat{\mathbf{e}}_k^{\mathrm{T}}\hat{\mathbf{e}}_k}{2}.$$

The actual reduction on Ω_k is simply

$$\Delta\Omega_k = \Omega(\mathbf{w}_k) - \Omega(\mathbf{w}_k + \mathbf{p}_k).$$

The LM method finds a value $v_k \geqslant 0$ which ensures positive definiteness for $(\mathbf{G}_k + v_k\mathbf{I})$ and then solves Eq. (14) for $\mathbf{p}_k$. On these conditions the solution $\mathbf{p}_k$ is a unique global solution on the minimization of Eq. (13). One iteration of the LM method can be stated as

(i) Given $\mathbf{w}_k$ and v_k, calculate $\mathbf{e}_k$, $\mathbf{g}_k$ and $\mathbf{J}_k$.
(ii) If $(\mathbf{G}_k + v_k\mathbf{I})$ is not positive definite, then $v_k = 4v_k$ and repeat the test.
(iii) Obtain $\mathbf{p}_k$ from Eq. (14).
(iv) Calculate $\Omega(\mathbf{w}_k + \mathbf{p}_k)$ and r_k.
(v) If $r_k < 0.25$ then $v_{k+1} = 4v_k$,
else if $r_k > 0.75$ then $v_{k+1} = v_k/2$,
else $v_{k+1} = v_k$.
(vi) If $r_k \leqslant 0$ then $\mathbf{w}_{k+1} = \mathbf{w}_k$,
else $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{p}_k$.

The optimization procedure is iterated until a set of termination criteria is met [10]. Assume θ_k is a measure of absolute accuracy, where τ_f is a measure of the desired number of correct figures in the objective function:

$$\theta_k = \tau_f(1 + \Omega_k).$$

The optimization stops when all the following conditions are met:

$$\begin{aligned} &\Omega_{k-1} - \Omega_k < \theta_k, \\ &\|\mathbf{v}_{k-1} - \mathbf{v}_k\| < \sqrt{\tau_f}(1 + \|\mathbf{v}_k\|), \\ &\|\mathbf{g}_k\| \leqslant \sqrt[3]{\tau_f}(1 + |\Omega_k|). \end{aligned} \tag{17}$$

In the following this method will be referred to as *off-line method* 3 or LM method.

3.2. On-line learning

In order to integrate a RBFNN temperature model in a real-time GEC methodology like the one depicted in Fig. 1, two possibilities can be considered. The use of a network trained off-line by a method such as those presented in the last section or the employment of an on-line learning algorithm capable of adjusting the network-free parameters as data arrives at every sample time. For the latter, depending on each particular learning strategy, some initial condition for the network parameters must be chosen. Generally, this is accomplished in one of two broad ways: initial values are chosen by means of heuristics, randomly or minimizing some criterion, or from an off-line trained network. Previous experimental studies in the context of real-time dynamic temperature models identification, have shown that such models present time variance in the parameters [7]. In the context of NN modelling, on-line learning algorithms can be used to circumvent this possibility. Three of such methods will be compared in this work.

3.2.1. On-line method 1

The first learning method considered is a constructive one. Constructive methods usually start with an empty hidden layer and then, based on the novelty of the input training samples, start adding units to the network. This means that no initial start point must be provided. A major contribution to this kind of technique was made by Platt [16] with its *resource allocating network* (RAN), followed by Kadirkamanathan and Niranjan [15], who improved RAN using an *extended Kalman filter* (EKF) instead of the *least mean-squares* (LMS) estimate for the network parameters, creating the RANEKF network, which is more compact and has a better accuracy. In order to overcome a common drawback of both RAN and RANEKF, in the sense that they could add new units to the network but never remove them, Yingwei et al. [25] proposed a sequential learning algorithm adopting the basic ideas in RANEKF but improving it with a pruning strategy in order to obtain a *minimal-RAN* (M-RAN) network. The resulting approach revealed itself to be better than RAN or RANEKF. For the reasons pointed out already and those presented in a performance study of M-RAN [26], this method will be considered here. The basic idea of M-RAN is to add or subtract hidden units from the network, based on its performance on the current input sample and also in a sliding window of a certain size over the training samples. Then, if no units are

added or subtracted, all the free parameters of the network are adjusted by the EKF algorithm.

3.2.2. On-line method 2

The second method studied is based on the SI problem with GRBFs with regularization [24]. Regularization networks depend on a regularization parameter which establishes a trade-off between a standard error measure and the smoothness of the mapping performed by the network. In this approach the SI equation is solved recursively, at each time step k, over a subset $\mathscr{Z}_M(k)=\{\mathscr{X}_j,\mathscr{Y}_j\}_{j=k-M+1}^{k}$ of the training data, formed from the last M input–output pairs. The size of the network, in terms of number of hidden units, equals M and the centers are located at the input points in $\mathscr{Z}_M(k)$. The basis functions used are of the form

$$\varphi_i(x_j)=\mathrm{e}^{-\frac{1}{2}\|x_j-x_i\|_U^2},$$

where U is a d-by-d diagonal norm weighting matrix, d being the dimension of $\mathscr{X}$. The diagonal elements of this matrix are the variances of each element in x_j over $\mathscr{Z}$ [14]. Since the center locations are known, the only parameters which have to be learned are the output weights. This is done by solving the GRBF SI equation recursively. Basically, all data in this equation pertaining to time-step $k-M+1$ are discarded, the weights are updated and finally, a new center and weight relative to time-step $k+1$ are added. Due to its own nature this method does not require any initial condition for the parameters.

3.2.3. On-line method 3

The last learning algorithm considered comes from the LM method presented in Section 3.1.3 and the reasoning behind its on-line implementation follows. The LM optimization method is iterated a certain number of times, at each time-step k, over a subset $\mathscr{Z}_M(k)=\{\mathscr{X}_j,\mathscr{Y}_j\}_{j=k-M+1}^{k}$ of the training data, until the termination criteria (17) is met. At $k+1$ the first input–output pair in $\mathscr{Z}$ is discarded and the one pertaining to time-step $k+1$ is added. Assuming that the dimension of $\mathscr{Z}$ is large enough two conclusions can be drawn: its statistical properties at $k+1$ are essentially the same as in k, and its distribution on input space is representative of the process data to some extent. As a consequence, point $\mathbf{w}$ in parameter space that minimizes Ω at time $k+1$ will be the same as in k with a slight correction. If at the initial time step, with $v=1$ in the first iteration (please recall the role of v), the LM method succeeds to converge, v will tend to a small value in the last iteration. This means that the correction made to $\mathbf{w}$ is small and that we are in the vicinity of the optimum, $\mathbf{w}^*$, which agrees with the consequence of the assumptions taken. In the subsequent time steps the value of v in the first iteration is the same as in the last iteration of the previous instant and it can be expected that the number of iterations spent will be near 1. The choice of M is application and problem dependent and care should be taken with its choice in order to satisfy the assumptions made. For the minimization of criterion (12) initial values must be provided for the center locations and for the spreads of the neuron activation

functions. This is achieved by means of one iteration of the OAKM clustering procedure and by the use of Eq. (9).

4. Experiments

The networks will try to model the inside air temperature in an hydroponic greenhouse as a function of the outside solar radiation, air temperature and the inside relative humidity. The input–output model structure was selected from a previous work [5], where several hypotheses were tested and the best one chosen by means of the Akaike information criterion [1]. It is a second-order model with one delay from the outside solar radiation to the inside air temperature. The data set are composed of 4257 points acquired with a sample rate of 5 min. It is graphically shown in Fig. 2. The first 1000 points are used, for the off-line methods, as training set and the remaining for testing the networks. All DC terms were subtracted from the signals, which were then scaled to an amplitude one, $[-0.5, 0.5]$, interval. Table 1 shows the value of the DC terms subtracted to the signals and the amplitude interval from which they were scaled.

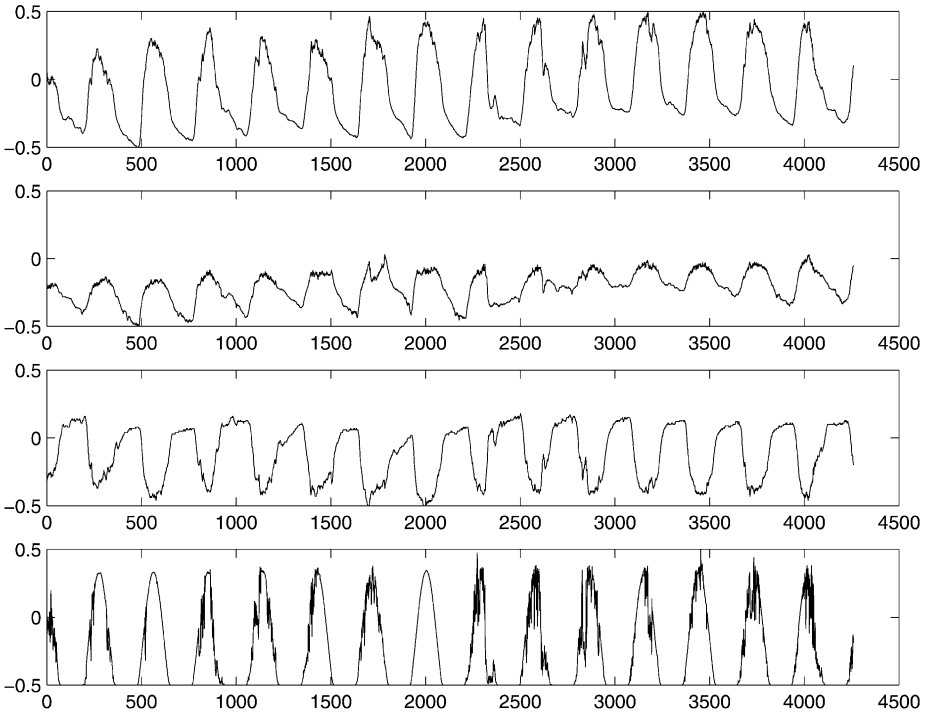

Fig. 2. Input–output data. From top: inside air temperature, outside air temperature, inside relative humidity and outside solar radiation.

Table 1
Values involved in signal pre-processing

Signal	DC term	Amplitude interval
Inside air temperature ($^{\circ}$C)	13.1	[0...24]
Outside air temperature ($^{\circ}$C)	10.8	[0...24]
Outside solar radiation (W/m^2)	0	[0...1070]
Inside relative humidity (%)	19	[0...100]

5. Results and discussion

5.1. Off-line methods

Regarding the methods presented in Section 3.1, Table 2 shows the results. This table is divided into four groups off cells, each presenting results for one method in particular. From top to bottom and left to right these are the LM method minimizing (7) and (12), off-line method 1 and the OLS method, respectively. For each group of cells, the first column, *Nc*, denotes the number of centers. The second and third are the *sum-of-square of the errors* (*sse*) for the training and test sets, respectively. Finally, the fourth column is the norm of the linear output weights. The OLS method is of the constructive type, which means that the number of centers was chosen by the algorithm.

Fig. 3 presents the curve approximation, output error and error distribution for the LM methods minimizing (12) and (7), the OAKM method and OLS method, respectively. The number of centers is 16 for all methods except for the OLS, where 13 centers were employed. The leftmost plots correspond to a detail from the test data set from $k = [2000 \ldots 2600]$. The solid line represents measured values and the dotted line the NN approximation.

5.1.1. OAKM and OLS methods

From the data presented in Table 2 we can conclude that the OLS method did not achieve good linear parameter conditioning. Its error performance was good with the exception of the value obtained for the test set using the largest network ($Nc = 256$). The OAKM method presents the worst error performance among all. This can be confirmed looking at Fig. 3 where clearly this method achieves the poorer curve fitting.

5.1.2. LM based methods

It can be observed that, in terms of *sse* for both training and test sets, the LM methods outperformed the other hybrid methods, usually by a factor of one order of magnitude in the training set. Comparing the two LM methods, the use of the new criterion (12) outperformed the standard criterion (7). The error values obtained with the test set put in evidence the improvement in terms of generalization that the LM method minimizing (12) achieves. The termination criterion (a value of

Table 2
Results obtained with the off-line training methods

Nc	Training *sse*	Test *sse*	$\|\mathbf{u}\|$	*Nc*	Training *sse*	Test *sse*	$\|\mathbf{u}\|$
8	0.029	0.902	1.65	8	0.028	0.379	1.92
16	0.029	1.033	1.71	16	0.029	0.849	2.33
24	0.024	1.333	2.58	24	0.027	0.819	4.33
32	0.023	1.833	2.47	32	0.021	1.319	6.05
8	1.172	28.191	1.35	7	0.093	1.700	1.84E+02
16	0.329	21.676	3.80	13	0.046	0.889	2.04E+02
24	0.399	24.161	2.61	58	0.025	0.769	1.09E+04
32	0.238	21.162	3.08	256	0.013	131.633	3.99E+07

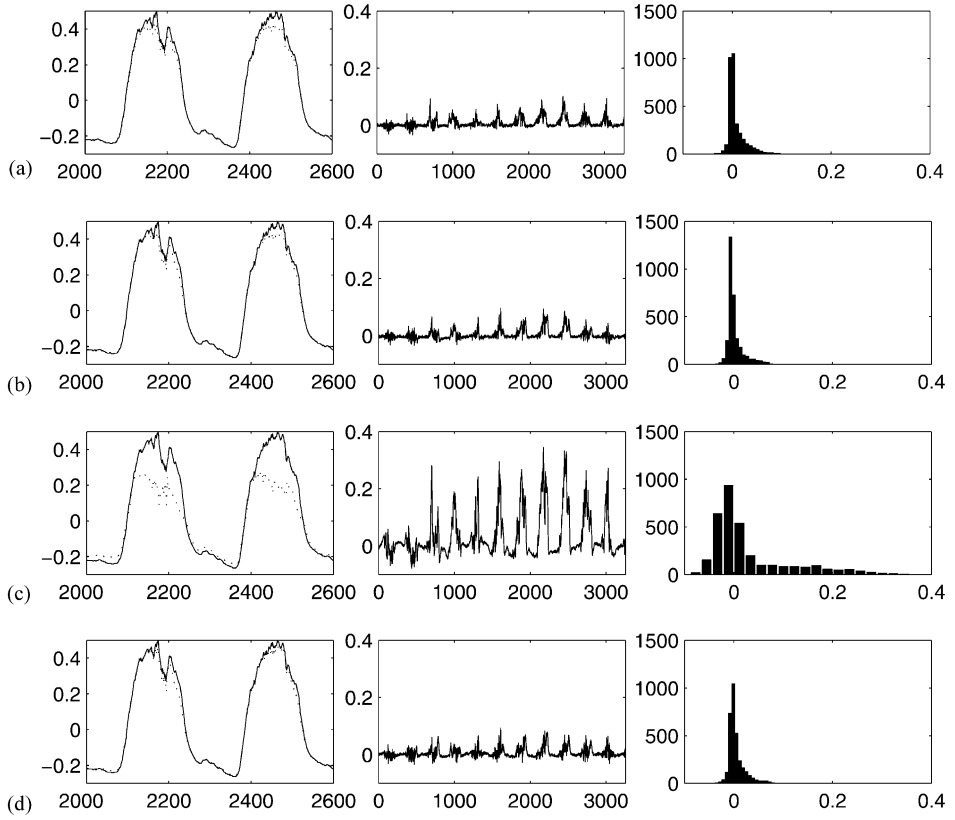

Fig. 3. Curve fittings and error for the off-line methods ($k = [2000 \ldots 2600]$).

$\tau_f = 0.001$ was employed) not only ensures good generalization capability, but also good conditioning in the linear parameters. Also, faster convergence rates were

Table 3
Results obtained with the M-RAN method

Network	$e_{\min}$	$e'_{\min}$	Nc	$rmse$
N_1	0.1	0.01	14	0.0390
N_2	0.1	0.001	14	0.0390
N_3	0.05	0.01	24	0.0445
N_4	0.05	0.001	24	0.0445
N_5	0.01	0.01	42	0.0450
N_6	0.01	0.001	42	0.0450

observed for the LM method minimizing (12). Taking into account, on the one hand, the bad linear parameter conditioning of the OLS method and the network sizes it obtains and, on the other hand, the fittings, error distributions and error values obtained by the LM methods, we conclude that the latter achieves a better performance.

5.2. On-line methods

5.2.1. M-RAN method

The M-RAN method depends on several parameters. A detailed description of their role is beyond the scope, but the same notation from [26] will be used. A common choice to all networks is done for some of the parameters: $\varepsilon_{\max} = 1.0$, $\varepsilon_{\min} = 0.2$, $\gamma = 0.4$, $\delta = 0.01$, $\kappa = 0.3$, $P_0 = 1$, $R_n = 1$, $Q_0 = 0.00001$ and $M = 288$. The value of M determines the size of a sliding window over the output error values and the outputs of all neurons, which are the decision basis for the pruning and addition operations. The chosen value corresponds to 1 day of data. Table 3 shows a summary of the values chosen for the two remaining parameters and the corresponding results obtained. These parameters are $e_{\min}$, a threshold for the absolute value of the output error, and $e'_{\min}$, a threshold for the values of RMSE over a window of size M. Nc is the number of centers obtained and the last column presents the *root mean square error* (*rmse*) of the predicted output for all simulation points.

Clearly, in this problem $e_{\min}$ dominates the design of the network. The RMSE values are greater for larger networks when the opposite is expected. Tighter error restrictions provoke a greater number of pruning and addition operations, which lead the EKF to converge to a new point in parameter space. Probably, this results in a worst RMSE behavior. Fig. 4 shows results obtained with the M-RAN method for networks $N_{\{1,3,5\}}$.

For each row, the top left plot presents the evolution of the network size. The plot below represents the output error obtained. On the right-hand side, the curve fittings for the three networks can be observed.

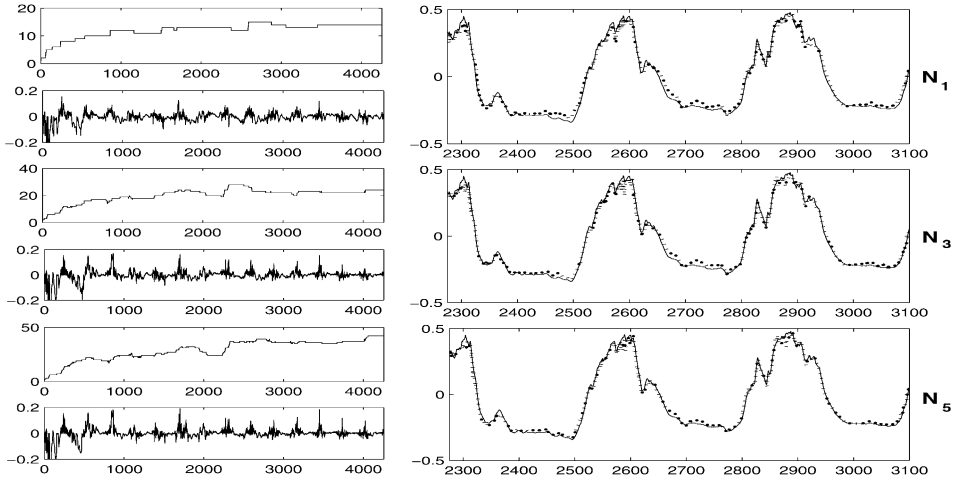

Fig. 4. Results for the M-RAN method.

Table 4
Results obtained with the regularized network for $\kappa = 4.0$

Nc	λ					
	1	0.1	0.01	0.001	0.0001	0.00001
5	0.0986	0.0208	0.0255	0.0528	0.0106	0.0106
6	0.0905	0.0431	0.0192	0.0111	0.0106	0.0106
7	0.0838	0.0232	0.0365	0.0107	0.0106	0.0106
8	0.0781	0.0435	0.1867	0.0110	0.0106	0.0106
9	0.0733	0.0680	0.0468	0.0124	0.0107	0.0106
10	0.0692	0.0283	0.0426	0.0149	0.0107	0.0106
11	0.0658	0.2710	0.0973	0.0186	0.0107	0.0106
12	0.0629	0.5028	0.0515	0.0142	0.0108	0.0106
13	0.0604	0.7114	0.2378	0.0246	0.0107	0.0107
14	0.0582	0.1022	0.6323	0.0339	0.0110	0.0107
15	0.0563	0.1527	0.1359	0.1952	0.0116	0.0107
16	0.0547	0.9885	0.1945	0.0436	0.0112	0.0107
17	0.0534	0.1293	0.1393	0.0214	0.0142	0.0108
18	0.0522	0.1311	0.1787	0.1345	0.0208	0.0107

5.2.2. *On-line method 2*

For the on-line method 2, the network size is varied from 5 to 15 neurons and for each network different values of the regularization parameter were employed: $\lambda = \{1, 0.1, 0.01, 0.001, 0.0001, 0.00001\}$. Table 4 presents the RMSE values obtained for these combinations with $\kappa = 4.0$ (read below for the role of κ).

Table 5
Results obtained with the regularized network for $\kappa = 1.0$

Nc	λ					
	1	0.1	0.01	0.001	0.0001	0.00001
5	0.0987	0.0205	0.0258	0.0528	0.0106	0.0106
6	0.0908	0.0223	0.0186	0.0110	0.0106	0.0106
7	0.0842	0.0210	0.0283	0.0107	0.0106	0.0106
8	0.0786	0.0215	0.0251	0.0111	0.0110	0.0106
9	0.0739	0.0212	0.0188	0.0145	0.0106	0.0106
10	0.0699	0.0211	0.0516	0.0145	0.0107	0.0106
11	0.0666	0.0362	0.4321	0.0142	0.0109	0.0106
12	0.0637	0.4923	0.0722	0.0162	0.0139	0.0107
13	0.0611	0.0395	0.2760	0.0364	0.0108	0.0147
14	0.0589	0.2029	0.6275	0.0195	0.0126	0.0107
15	0.0569	0.0291	0.1482	0.0477	0.0172	0.0108
16	0.0551	0.0400	0.2694	0.0286	0.0119	0.0107
17	0.0536	0.0394	0.1154	0.0187	0.0109	0.0108
18	0.0524	0.0285	0.4286	0.5326	0.0109	0.0108

Table 5 presents the same results for $\kappa = 1.0$. Some numerical instability was verified in the experiments and as suggested in [24], algorithm resetting was used to counteract this problem. The error is monitored at each iteration, k, and the algorithm is restarted when the following condition is met:

$$|e[k] - \mu([e(i)]_{i=k-n}^{k-1})| > \kappa\sigma([e(i)]_{i=k-n}^{k-1}),$$

where μ and σ are the sample mean and standard deviation of their argument values.

κ is a threshold parameter which directly affects the number of reset operations. Fig. 5 shows plots pertaining to the values in Tables 4 and 5. The values from 1 to 6 in the λ-axis correspond to $\{1, 0.1, 0.01, 0.001, 0.0001, 0.00001\}$, respectively. From the RMSE values in Fig. 5(a) and (c), it can be seen that larger networks and bigger regularization parameters do not produce good results, and that the method is sensitive to those n, λ combinations. Small networks for all tested λ values and small λ for all network sizes produced acceptable RMSE values. By looking at Fig. 5(b) and (d) it can be observed that $\kappa = 1.0$ forces a large number of reset operations. For $\kappa = 4.0$ this number is acceptable, varying from 23 to 199. From $n = 7$ to 18 and small λ values this number lies within $[31 \ldots 95]$. Fig. 5(e) presents the difference in RMSE resulting from the two values of κ and reinforces the idea that the method is sensitive to the combination of larger networks with bigger λ values. The other combinations of n, λ are not affected by the resetting procedure in terms of RMSE. The reset rate is clearly varying with the n, λ combination. Better results were achieved by the use of smaller values for the regularization parameter. Considering its role, this fact can indicate that the data set is a good representation of the underlying system.

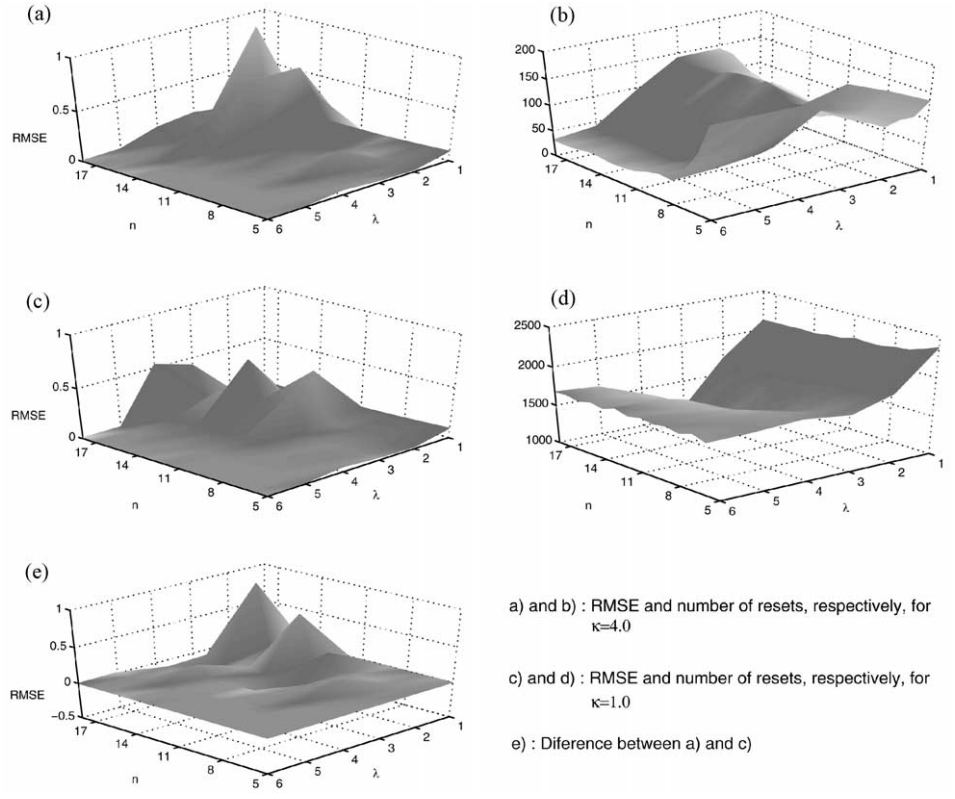

Fig. 5. RMSE values for on-line method 2.

5.2.3. LM based methods

Regarding the on-line LM methods, the network size, Nc, was varied from 3 to 15 neurons and two values of τ_f, $\{0.1, 0.01\}$, were used for each network. The size of the training set, M, in each time step k was 288 corresponding to one day of data. This results in a total of 3969 input–output pairs available for simulation. Tables 6 and 7 present the results obtained for the LM algorithm minimizing the new criterion (12) and the standard criterion (7), respectively. Each table presents the results for the two values of τ_f in separate groups of cells. In each group of cells the third and fourth columns denote the mean of the values obtained for the norm of the linear weights and for the error criterion. NI stands for the total number of iterations spent and the last column is the mean value of v. Table 6 presents the results only for networks up to 10 neurons. This is due to the bad linear weight conditioning verified, starting with $Nc = 7$. This problem should be solved employing a regularized implementation of the LS solution of (12). It can be observed that in general better error performance is obtained with a tighter error restriction with almost no cost in terms of number of iterations spent. With

Table 6
Results for the on-line LM method minimizing (12)

Nc	*rmse*	$\|\bar{\mathbf{u}}\|$	$\bar{\Omega}$	*NI*	$\bar{v}$
$\tau_f = 0.1$					
3	0.0089	1.60	0.110	3972	0.125
4	0.0065	3.54	0.088	3970	0.5
5	0.0067	3.14	0.086	3970	0.5
6	0.0068	1.97	0.087	3970	0.5
7	0.0073	2.02	0.080	3970	0.5
8	0.0072	10.68	0.077	3970	0.5
9	0.0072	17.06	0.079	3970	0.5
10	0.0076	16.43	0.076	3970	0.5
$\tau_f = 0.01$					
3	0.0078	2.26	0.098	3976	0.0156
4	0.0066	14.99	0.083	3974	0.0313
5	0.0066	3.87	0.085	3972	0.1250
6	0.0067	2.34	0.083	3972	0.1250
7	0.0080	28.7E04	0.080	3972	0.1250
8	0.0072	28.7E03	0.077	3972	0.1250
9	0.0078	1.9E03	0.076	3971	0.2501
10	0.0078	18E03	0.072	4031	0.1795

$\tau_f = 0.1$ the termination criteria is easily met hence the relatively large values of v obtained.

For the other value of τ_f the termination conditions are not met so easily and in the initial time step a few more iterations are needed to bring v to smaller values. Table 7 puts in evidence the slower convergence rates obtained with criterion (12), as it can be seen from the values of *NI*. In terms of the error performance although the mean values of Ω are smaller for $\tau_f = 0.01$ as expected, this fact shows no correspondence on the values of the RMSE. With the tighter error restriction much more iterations were needed and some over-fitting may have occurred. Also the **u** parameters obtained in the last time step are not the optimal ones for the current time step, making convergence more difficult. Comparing the two LM methods for error performance, it can be seen that the new error criterion leads, in general, to better values. In terms of convergence rate and computational cost the improvement is obvious. In all experiments good linear parameter convergence could be observed with some smooth time variability. Figs. 6 and 7 present results regarding the LM methods minimizing (12) and (7), respectively, for $Nc = 6$ and $\tau_f = 0.1$.

The fitting, error sequence and distribution, linear parameter evolution and learning criterion evolution can be observed. In terms of real values of the modeled quantity, the error is bounded within the range $[-1^{\circ}\mathrm{C}, \ldots, 1^{\circ}\mathrm{C}]$ and concentrated on the interval $[-0.2^{\circ}\mathrm{C}, \ldots, 0.2^{\circ}\mathrm{C}]$. For both methods good linear parameter convergence can be observed.

Table 7
Results for the on-line LM method minimizing (7)

Nc	*rmse*	$\|\bar{\mathbf{u}}\|$	$\bar{\Omega}$	*NI*	$\bar{v}$
$\tau_f = 0.1$					
3	0.0063	3.52	0.088	3978	0.0625
4	0.0063	3.21	0.085	3974	0.0313
5	0.0073	4.65	0.087	4014	3.5527
6	0.0068	2.96	0.078	3978	0.0400
7	0.0074	7.50	0.073	4696	0.0377
8	0.0074	4.81	0.080	4148	0.7191
9	0.0075	4.07	0.075	4122	0.1763
10	0.0101	8.00	0.061	5018	0.0340
11	0.0102	7.19	0.058	5338	0.4211
12	0.0111	5.38	0.083	4209	7.4484
13	0.0102	7.61	0.062	4950	0.0409
14	0.0130	6.42	0.063	4634	1.1504
15	0.0120	7.36	0.052	5569	0.0263
$\tau_f = 0.01$					
3	0.0066	4.18	0.091	3977	0.0625
4	0.0069	4.90	0.090	4160	4.0802
5	0.0076	11.25	0.075	6607	0.0078
6	0.0087	7.00	0.075	4838	0.0328
7	0.0280	9.39	0.074	5901	0.0083
8	0.0086	6.78	0.073	4825	0.0804
9	0.0136	6.80	0.063	6055	0.1236
10	0.0114	6.25	0.067	5087	0.1870
11	0.0147	8.60	0.059	7680	0.0405
12	0.0130	8.53	0.054	8079	0.0196
13	0.0141	6.46	0.070	5201	0.9633
14	0.0105	8.29	0.063	6181	0.0368
15	0.0117	6.85	0.056	6576	0.0376

5.2.4. On-line methods comparison

In order to compare all the on-line methods, Table 8 shows two sets of results organized by network size, presenting the RMSE values obtained. The first set concerns all methods except method 3 minimizing (12). The regularization method and the LM method minimizing (7) have a clear advantage over the M-RAN method. The second set deals with all methods except M-RAN. Again the LM method minimizing (12) achieves the best result.

5.3. Off-line or on-line?

From the comparisons and reasonings made so far, it can be said that, whether off-line or on-line, the LM method minimizing the new criterion achieves the best results. It has the best error performance at smaller network size and better general-

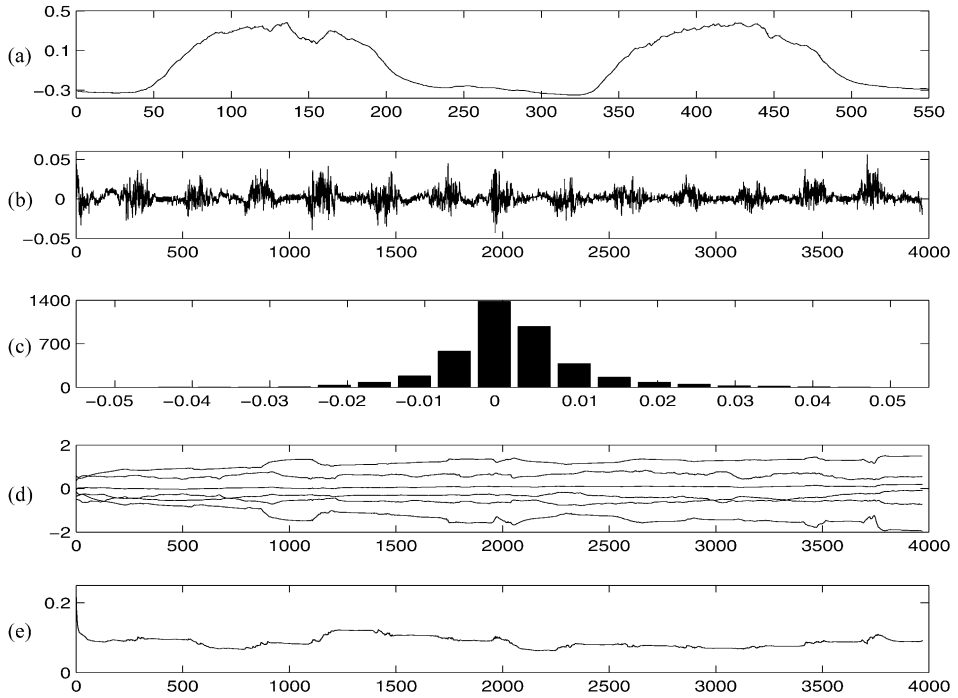

Fig. 6. LM minimizing (12). From top: curve fitting, error sequence, error distribution, linear parameter evolution and training criterion evolution.

Table 8
Comparison of all on-line methods

Method	Observations	*Nc*	*rmse*
Method 1	$e_{\min} = 0.1$ $e'_{\min} = 0.01$	14	0.0390
Method 2	$\lambda = 0.00001$ $\kappa = 4.0$	14	0.0107
Method 3 minimizing (7)	$\tau_f = 0.01$	14	0.0105
Method 2	$\lambda = 0.00001$ $\kappa = 4.0$	6	0.0106
Method 3 minimizing (7)	$\tau_f = 0.01$	6	0.0087
Method 3 minimizing (12)	$\tau_f = 0.01$	6	0.0067

ization capability (in the off-line case). The smaller RMSE value obtained off-line (for the test set) was 0.0108 with a eight neurons network. For a similar sized network adapted on-line, a value of 0.0072 was achieved. Comparing these values we can conclude that the on-line version of the LM method minimizing (12) is the best method for the GEC problem. This conclusion is also supported by the

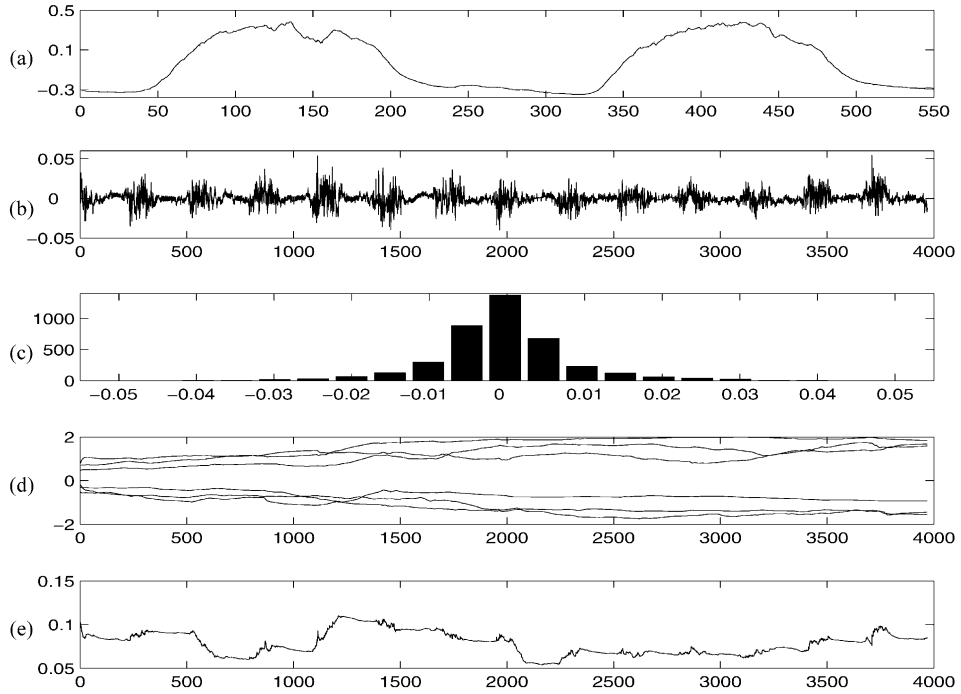

Fig. 7. LM minimizing (7). From top: curve fitting, error sequence, error distribution, linear parameter evolution and training criterion evolution.

best curve tracking achieved by the on-line version and its linear parameter slow time variance.

6. Conclusions

The results obtained lead to the conclusion that RBFNNs can model the greenhouse internal temperature. All methods achieved good fittings and acceptable one step ahead prediction errors. Results for a new algorithm, based on a Levenberg–Marquardt method, that explicitly exploits the non-linear–linear structure of RBFNNs have been presented. A strategy for its on-line application was also discussed. It has been shown that, whether on-line or off-line, better results were obtained by the two LM methods, compared with other hybrid batch and adaptive methods. The LM method, exploiting the separability of parameters, achieves the best results in terms of error performance and parameter convergence, and with smaller computational costs.

With a view to greenhouse environmental control, more work has to be done in terms of the model input selection. The performance of these methods have to be evaluated over greater prediction horizons, where we think that the benefits of the

on-line version of the LM method will be more evident. Finally, neural network models have to be compared also with conventional models.

Acknowledgements

The authors would like to acknowledge the FCT (project MGS/33906/2000 and grant SFRH/BD/1236/2000) for supporting this work.

References

[1] H. Akaike, A new look at the statistical model identification, IEEE Trans. Automat. Control AC-19 (1974) 716–723.
[2] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Systems 2 (1988) 321–355.
[3] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Trans. Neural Networks 2 (2) (1991) 302–309.
[4] C. Chinrungrueng, C.H. Squin, Optimal adaptive K-means algorithm with dynamic adjustment of learning rate, IEEE Trans. Neural Networks 6 (1) (1995) 157–169.
[5] J.B. Cunha, A.E. Ruano, E.A. Faria, Dynamic temperature models of a soilless greenhouse, in: Proceedings of the Second Portuguese Conference on Automatic Control, Vol. 1, Porto, Portugal, 1996, Portuguese Association of Automatic Control, pp. 77–81.
[6] H. Demuth, M. Beale, Neural Network Toolbox Users Guide, The MathWorks, Inc., Natick, MA, 1998.
[7] P.M. Ferreira, E.A. Faria, A.E. Ruano, Real-time data acquisition system for the identification of dynamic temperature models in a hydroponic greenhouse, Acta Horticulturae 519: Computers and Automation, Electronic Information in Horticulture, Brussels, Belgium, January 2000, XXV International Horticultural Congress, pp. 191–196.
[8] R. Fletcher, Practical Methods of Optimization, 2nd Edition, Wiley, New York, 1996, p. 100.
[9] R. Fletcher, Practical Methods of Optimization, 2nd Edition, Wiley, New York, 1996, p. 95.
[10] P.E. Gill, W. Murray, M.H. Wright, Practical Optimization, Academic Press, Inc., New York, 1981, p. 306.
[11] F. Girosi, M. Jones, T. Poggio, Priors, stabilizers and basis functions: from regularization to radial, tensor and additive splines, A.I. Memo N. 1430, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, June 1993.
[12] F. Girosi, T. Poggio, Networks and the best approximation property, Biol. Cybernet. 63 (1990) 169–176.
[13] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1998, p. 299.
[14] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ, 1998.
[15] V. Kadirkamanathan, M. Niranjan, A function estimation approach to sequential learning with neural networks, Neural Comput. 5 (1993) 954–975.
[16] J. Platt, A resource allocating network for function interpolation, Neural Comput. 3 (1991) 213–225.
[17] T. Poggio, F. Girosi, Networks for approximation and learning, Proceedings of the IEEE, Vol. 78, September 1990, pp. 1481–1497.
[18] A.E. Ruano, P.J. Fleming, D.I. Jones, A connectionist approach to pid autotuning, IEE Proceedings, Part D, Vol. 139, Brighton, UK, 1992, pp. 279–285.
[19] A.E. Ruano, D.I. Jones, P.J. Fleming, A new formulation of the learning problem for a neural network controller, IEEE Conference on Decision and Control, Brighton, UK, 1991, pp. 865–866.

[20] I. Seginer, Some artificial neural network applications to greenhouse environmental control, Comput. Electron. Agric. 18 (23) (1997) 167–186.
[21] I. Seginer, T. Boulard, B.J. Bailey, Neural network models of the greenhouse climate, J. Agric. Eng. Res. 59 (1994) 203–216.
[22] I. Seginer, Y. Hwang, T. Boulard, J.W. Jones, Mimickink an expert greenhouse grower with a neural-net policy, Trans. ASAE 39 (1996) 299–306.
[23] G. van Straten, J.W. Bentum, R.F. Tap, Paradigms in greenhouse climate control: on hierarchy and energy savings, in: A. Munack and H.J. Tantau (Eds.), Mathematical and Control Applications in Agriculture and Horticulture, Hannover, Germany, September 1997, Pergamon, Oxford, pp. 307–312.
[24] P. Yee, S. Haykin, A dynamic regularized radial basis function network for nonlinear, nonstationary time series prediction, IEEE Trans. Signal Process. 47 (9) (1999) 2503–2521.
[25] Lu Yingwei, N. Sundararajan, P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function neural networks, Neural Comput. 9 (1997) 461–478.
[26] Lu Yingwei, N. Sundararajan, P. Saratchandran, Performance evaluation of a sequential minimal radial basis function (RBF) meural network learning algorithm, IEEE Trans. Neural Networks 9 (2) (1998) 308–318.

Eugénio de Araújo Faria was born in 1940. He has a degree in Agronomic Engineering by the Instituto Superior de Agronomia of the Universidade Técnica de Lisboa (1976). He achieved his doctorate in 1984. He is a Full Professor at the Universidade do Algarve (UAlg), since 1995, and he is President of the Scientific Council, since 1999. He was the national Co-ordinator of the European Master (Management de la Filière Fruits et Légumes, Groupement Européen d'Intérêt Economique-FORmations Universitaires Méditerranéennes) between 1992 and 1996. He was the Scientific Co-ordinator of the Center for Development of Sciences & Techniques of Vegetable Production (Unit I&D n°52/94, FCT), (1995–2000). He was Co-ordinator of the National Program (PRAXIS XXI) in the Area of Horticulture and member of the Assessment Subcommittees (evaluation of projects proposals) on Agricultural Sciences of the Fundação para a Ciência e a Tecnologia (1994–1995). He was the President of Organizing Commission (1995–1997) of the II Congresso Iberoamericano—III Congresso Ibérico de Ciências Hortícolas (11–15, March 1997). Concerning scientific research, he has worked mainly in the area of Plant Nutrition especially in Food and Crop Quality. He has several papers published in national and international scientific journals and he supervised Master and Doctorate thesis in these areas.

Pedro Frazão Ferreira was born in 1970 in Santarém, Portugal. He received a five-year first degree in Systems Engineering and Computing from the University of Algarve, Portugal, in 1996. In 1995 and 1996 he worked as a Student Assistant at the Department of Electronic Engineering and Computing of the Faculty of Sciences and Technology of the University of Algarve, where he is a Ph.D. student in Electronic and Computing Engineering since 1998. From 1997 to 2000 he joined the School of Technology of the University of Algarve as a Teaching Assistant. His main research interests are neural modelling, soft computing methods and environmental control.

Antonio Ruano was born in 1959 in Espinho, Portugal. He received the First Degree in Electronic and Telecommunications Engineering from the University of Aveiro, Portugal, in 1982, the M.Sc. in Electrothecnic Engineering from the University of Coimbra, Portugal, in 1989, and the Ph.D. degree in Electronic Engineering from the University of Wales in 1992. In 1992 he joined the Department of Systems Engineering and Computing of the Faculty of Sciences & Technology of the Universidade do Algarve, where in 1996 he became an Assistant Professor of Automatic Control. His main research interests are neural control (both theoretical issues and applications), environmental control and parallel processing techniques applied to real-time control. He has over 80 research publications, he is an Associate Editor for Automatica, he is a member of the Editorial Board of International Journal of Systems Science, and serves as reviewer for other journals and international Conferences.