Orthogonal Least Squares Algorithm for the Approximation of a Map and its Derivatives with a RBF Network

C. Drioli [*],

Dip.artimento di Elettronica e Informatica, University of Padova, Via Gradenigo 6/a, 35131 Padova, Italy

D. Rocchesso

Dipartimento di Informatica, University of Verona, Ca' Vignal 2, Strada Le Grazie 15, 37134 Verona, Italy

---

Abstract

Radial Basis Function Networks (RBFNs) are used primarily to solve curve-fitting problems and for non-linear system modeling. Several algorithms are known for the approximation of a non-linear curve from a sparse data set by means of RBFNs. Regularization techniques allow to define constraints on the smoothness of the curve by using the gradient of the function in the training. However, procedures that permit to arbitrarily set the value of the derivatives for the data are rarely found in the literature. In this paper, the Orthogonal Least Squares (OLS) algorithm for the identification of RBFNs is modified to provide the approximation of a non-linear single-input single-output map along with its derivatives, given a set of training data. The interest in the derivatives of non-linear functions concerns many identification and control tasks where the study of system stability and robustness is addressed.

The effectiveness of the proposed algorithm is demonstrated with examples in the field of data interpolation and control of non-linear dynamical systems.

Key words: Radial Basis Function Networks, OLS learning, curve fitting, iterated map stability, chaotic oscillators.

1 Introduction

The Orthogonal Least Squares (OLS) algorithm [5] is one of the most popular procedures for the training of Radial Basis Function Networks (RBFNs). An RBFN is a two-layer neural network model especially suited for non-linear function approximation and appreciated in the fields of signal processing [10,11,8], non-linear system modeling, identification and control [1,4,14,13], and time-series prediction [20,3]. With respect to other non-linear optimization algorithms, such as gradient descent or conjugate gradients, the OLS algorithm for RBFNs is characterized by a faster training and improved convergence.

In function approximation and data interpolation by means of such universal approximation schemes, the derivatives of the underlying function can play an important role in the attempt to improve the model performance. Regularization techniques, for example, allow to define constraints on the smoothness of the curve by using the information on the gradient during the training phase. Regularized networks are recognized to perform better than non-regularized

* Corresponding author
Email addresses: `drioli@dei.unipd.it` (C. Drioli), `rocchesso@sci.univr.it` (D. Rocchesso).

networks in function learning and interpolation [16]. Moreover, in many applications the approximation of the mapping alone will not suffice, and learning of the constraints on differential data becomes important as well as learning input-output relations. For example, in the simulation of mechanical systems or plants by physical modeling, the physical knowledge is given in the form of partial differential equations (PDEs), or constraints on differential data [12]. In dynamical system modeling and control tasks the stability of the identified system depends on the gradient of the map [17,6], so that, exploiting the derivatives, the modeling can be improved and the stabilization of desired dynamical behaviors can be achieved.

Despite of the importance of learning differential data, the problem of efficiently approximating a non-linear function along with its derivatives seems to be rarely addressed. Some theoretical results as well as some application examples that apply to generic feedforward neural networks are found in [12,2,9]. More emphasis on the procedural aspects of differential data learning are found in [15], where a back-propagation based algorithm for multilayer neural network is proposed, and [19], where a RBFN with raised-cosine kernels is introduced, that can fit up to first-order differential data.

In this paper, an extended version of the OLS algorithm for the training of single-input single-output RBFNs is presented, which permits to approximate an unknown function by specifying a set of data points along with its desired higher-order derivatives.

The paper is organized as follows: in Section 2, the OLS algorithm is reviewed and modified to add control over the derivative of the function to be approximated. The extension to higher order derivatives is introduced in Section

3. Application examples in the field of function interpolation and non-linear dynamics are given in Section 4. In Section 5, the conclusions are presented.

2 Orthogonal Least Squares Learning Algorithm

The OLS learning algorithm is traditionally tied to the parametric identification of RBF networks, a special two-layer neural network model widely used for the interpolation and modeling of data in multidimensional space. In the following we will restrict the discussion to the single-input single-output RBFN model, which is a mapping $f : \mathbb{R} \to \mathbb{R}$ of the form

$$f(x) = b + \sum_{i=1}^{H} w_i \phi(x, \mathbf{q}_i), \tag{1}$$

where $x \in \mathbb{R}$ is the input variable, $\phi(\cdot)$ is a given non-linear function, b, w_i and $\mathbf{q}_i$, $1 \leq i \leq H$, are the model parameters, and H is the number of radial units. The RBFN can be viewed as a special case of the linear regression model

$$y(k) = b + \sum_{i=1}^{H} w_i p_i(k) + e(k), \tag{2}$$

where $y(k)$ is the desired k-th output sample, $e(k)$ is the approximation error, and $p_i(k)$ are the regressors, i.e. some fixed functions of $x(k)$, where $x(k)$ are the input values corresponding to the desired output values $y(k)$:

$$p_i(k) = \phi(x(k), \mathbf{q}_i). \tag{3}$$

In its original version, the OLS algorithm is a procedure that iteratively selects the best regressors (radial basis units) from a set of available regressors. This set is composed of a number of regressors equal to the number of available data, and each regressor is a radial unit centered on a data point. The selection of radial unit centers is recognized to be the main problem in the parametric identification of these models, while the choice of the non-linear function for the radial units does not seem to be critical. Gaussian-shaped functions, spline, multi-quadratic and cubic functions are some of the commonly preferred choices. Here, we will use the gaussian function $\phi(x, m, \sigma) = exp(\|x - m\|/\sigma)^2$, where $\| \cdot \|$ denotes the euclidean norm, and m and σ denote respectively the center and width of the radial unit. In the following, we assume that the width is unique for all units, and constant during training. For simplicity of notations we thus refer to radial units as $\phi(x, m)$.

2.1 Classic OLS algorithm

Say $\{x(k), y(k)\}$, $k = 1, 2, ..., N$, is the data set given by N input-output data pairs, which can be organized in two column vectors $\mathbf{x} = [x(1) \cdots x(N)]^T$ and $\mathbf{y} = [y(1) \cdots y(N)]^T$. The model parameters are given in vectors $\mathbf{m} = [m_1 \cdots m_H]^T$, $\mathbf{w} = [w_1 \cdots w_H]^T$ and $\mathbf{b} = [b]$, where H is the number of radial units to be used. Arranging the problem in matrix form we have:

$$\mathbf{y} = \begin{bmatrix} \mathbf{P} \ \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \\ b \end{bmatrix} + \mathbf{e} \qquad (4)$$

with

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 \cdots \mathbf{p}_H \end{bmatrix}$$

$$= \begin{bmatrix} \phi(x(1), m_1) & \cdots & \phi(x(1), m_H) \\ \vdots & \ddots & \vdots \\ \phi(x(N), m_1) & \cdots & \phi(x(N), m_H) \end{bmatrix}, \qquad (5)$$

where $\mathbf{p}_i = [\phi(x(1), m_i) \ldots \phi(x(N), m_i)]^T$ are regressor vectors forming a set of basis vectors, $\mathbf{e} = [e(1) \cdots e(N)]^T$ is the identification error, and $\mathbf{1} = [1 \ldots 1]^T$ is a unit column vector of length N. The least squares solution of this problem which satisfies the condition that

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{P} \ \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \qquad (6)$$

is the projection of $\mathbf{y}$ in a vector space spanned by the regressors. If the regressors are not independent, the contribution of each regressor to the total energy of the desired output vector is not clear. The OLS algorithm proceeds iteratively by selecting the next best regressor from a set by applying a Gram-Schmidt orthogonalization, so that the contribution of each vector of this new orthogonal base can be determined individually among the available regressors. This greedy strategy is known to give non-maximally compact representations when non-orthogonal basis, such as the Gaussian, are used [18]. Nevertheless, it proved to be useful and efficient in practice.

2.2 Modified OLS algorithm

The classic algorithm selects a set of regressors from the ones available and determines the output layer weights for the identification of the desired input-output map, but does not explicitly control the derivative of the function. We propose to modify this procedure so as to allow the assignment of a desired value of the function derivative at each data point. The data set will then be organized in three vectors $\mathbf{x} = [x(1) \cdots x(N)]^T$, $\mathbf{y} = [y(1) \cdots y(N)]^T$, and $\mathbf{y}^{(1)} = [y_1(1) \cdots y_1(N)]^T$, $\mathbf{x}$ and $\mathbf{y}$ being the input-output pairs and $\mathbf{y}^{(1)}$ being the respective derivatives. It has to be noted that the original OLS algorithm selects each radial unit from a set of units, each of which is centered on an input data point. The maximum number of units is then limited to the number of data points. When we add requirements on the derivative of the function, a further constraint to the optimization problem is added, and the number of units to be selected in order to reach the desired approximation may be higher than the number of data points. A possible choice is to augment the input vector with points chosen where there is no data available, e.g. a uniform grid with N_e points covering the input interval, and to build a set of N_e regressors centered over these points.

The algorithm can be summarized as follows:

- **First step**, initialization: the set of regressors for selection is obtained by centering the N_e radial units, and the *error reduction ratio* (err) for each regressor vector is computed. Given the regressor vectors

$$\mathbf{p}_i = [\phi(x(1), x(i)), ..., \phi(x(N), x(i))]^T, \quad 1 \le i \le N_e, \tag{7}$$

and defined the first-iteration vectors

$$\mathbf{u}_{1,i} = \mathbf{p}_i, \ \ 1 \leq i \leq N_e. \tag{8}$$

The error reduction ratio associated with the i-th vector is given by

$$\mathrm{err}_{1,i} = (\mathbf{u}_{1,i}^T \mathbf{y})^2 / ((\mathbf{u}_{1,i}^T \mathbf{u}_{1,i})(\mathbf{y}^T \mathbf{y})). \tag{9}$$

In a similar way, the regressor vectors for the derivative of the map are computed:

$$\mathbf{p}_i^{(1)} = [\frac{\partial \phi(x(1), x(i))}{\partial x} \cdots \frac{\partial \phi(x(N), x(i))}{\partial x}]^T, \ \ 1 \leq i \leq N_e, \tag{10}$$

and the first-iteration vectors are defined:

$$\mathbf{l}_{1,i} = \mathbf{p}_i^{(1)}, \ \ 1 \leq i \leq N_e. \tag{11}$$

The error reduction ratio for the derivative is:

$$\mathrm{grad\_err}_{1,i} = (\mathbf{l}_{1,i}^T \mathbf{y}^{(1)})^2 / ((\mathbf{l}_{1,i}^T \mathbf{l}_{1,i})({\mathbf{y}^{(1)}}^T \mathbf{y}^{(1)})), 1 \leq i \leq N_e. \tag{12}$$

The $\mathrm{err}_{1,i}$ and $\mathrm{grad\_err}_{1,i}$ represent the error reduction ratios caused respectively by $\mathbf{u}_{1,i}$ and $\mathbf{l}_{1,i}$, and the total error reduction ratio can be computed by

$$\mathrm{tot\_err}_{1,i} = \lambda \ \mathrm{err}_{1,i} + (1 - \lambda) \ \mathrm{grad\_err}_{1,i}, \tag{13}$$

where λ weights the importance of the map against its derivative. Usually, $\lambda = 0.5$ is the preferred choice when the accuracy requirements for the map and its derivative are the same. The index i_1 is then found, so that:

$$\mathrm{tot\_err}_{1,i_1} = \max_i \{\mathrm{tot\_err}_{1,i}, \ \ 1 \leq i \leq N_e\}. \tag{14}$$

The regressor $\mathbf{p}_{i_1}$ giving the largest error reduction ratio is selected and removed from the set of available regressors. The corresponding center is added to the set of selected centers:

$$\mathbf{u}_1 = \mathbf{u}_{1,i_1} = \mathbf{p}_{i_1}; \tag{15}$$

$$\mathbf{l}_1 = \mathbf{l}_{1,i_1} = \mathbf{p}_{i_1}^{(1)}; \tag{16}$$

$$m_1 = x(i_1). \tag{17}$$

- h**-th iteration**, for $h = 1, ..., H$ and $H \leq N_e$: the regressors selected in the previous steps, having indexes $i_1, ..., i_{h-1}$, have been removed from the set of available regressors. Before computing the error reduction ratio for each regressor still available, the orthogonalization step is performed which makes each regressor orthogonal with respect to those already selected:

$$\mathbf{u}_{h,i} = \mathbf{p}_i - \sum_{j=1}^{h-1} \mathbf{u}_j (\mathbf{p}_i^T \mathbf{u}_j)/(\mathbf{u}_j^T \mathbf{u}_j), \quad i \neq i_1, i_2, ..., i_{h-1}; \tag{18}$$

$$\mathbf{l}_{h,i} = \mathbf{p}_i^{(1)} - \sum_{j=1}^{h-1} \mathbf{l}_j (\mathbf{p}_i^{(1)^T} \mathbf{l}_j)/(\mathbf{l}_j^T \mathbf{l}_j), \quad i \neq i_1, i_2, ..., i_{h-1}; \tag{19}$$

$$\mathrm{err}_{h,i} = (\mathbf{u}_{h,i}^T \mathbf{y})^2/((\mathbf{u}_{h,i}^T \mathbf{u}_{h,i})(\mathbf{y}^T \mathbf{y})), \quad i \neq i_1, i_2, ..., i_{h-1}; \tag{20}$$

$$\mathrm{grad\_err}_{h,i} = (\mathbf{l}_{h,i}^T \mathbf{y}^{(1)})^2/((\mathbf{l}_{h,i}^T \mathbf{l}_{h,i})(\mathbf{y}^{(1)^T} \mathbf{y}^{(1)})), \quad i \neq i_1, i_2, ..., i_{h-1}; \tag{21}$$

$$\mathrm{tot\_err}_{h,i} = \lambda \mathrm{err}_{h,i} + (1-\lambda)\ \mathrm{grad\_err}_{h,i}, \quad i \neq i_1, i_2, ..., i_{h-1}. \tag{22}$$

As before, the regressor with maximum error reduction ratio is selected and removed from the list of availability, and its center is added to the set of selected centers:

$$\mathrm{tot\_err}_{h,i_h} = \max_i \{\ \mathrm{tot\_err}_{h,i}, \quad i \neq i_1, i_2, ..., i_{h-1}\} \tag{23}$$

$$\mathbf{u}_h = \mathbf{u}_{h,i_h}; \tag{24}$$

$$\mathbf{l}_h = \mathbf{l}_{h,i_h}; \tag{25}$$

$$m_h = x(i_h). \tag{26}$$

- **Final step**, computation of output layer weights: once the H radial units have been positioned, the remaining $\mathbf{w}$ and $\mathbf{b}$ parameters can be found with a Moore-Penrose matrix inversion (pseudo-inversion): let us call $\mathbf{P}_H = [\mathbf{p}_{i_1}\mathbf{p}_{i_2}\cdots\mathbf{p}_{i_H}]$ and $\mathbf{P}_H^{(1)} = [\mathbf{p}_{i_1}^{(1)}\mathbf{p}_{i_2}^{(1)}\cdots\mathbf{p}_{i_H}^{(1)}]$ the two sets of selected regressors, and let $\mathbf{1} = [1\dots1]^T$ and $\mathbf{0} = [0\dots0]^T$ be two column vectors of length N. Then we have

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^{(1)} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_H & \mathbf{1} \\ \mathbf{P}_H^{(1)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} + \mathbf{e}_H \tag{27}$$

  whose solution is

$$\begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \left(\begin{bmatrix} \mathbf{P}_H & \mathbf{1} \\ \mathbf{P}_H^{(1)} & \mathbf{0} \end{bmatrix} \right)^+ \begin{bmatrix} \mathbf{y} \\ \mathbf{y}^{(1)} \end{bmatrix}. \tag{28}$$

Usually, it is convenient to stop the procedure before the maximum number of radial units has been reached, as soon as the identification error is considered to be acceptable. To this purpose, one can use equation (28) at iteration h to compute the identification error $\mathbf{e}_h$ in (27) [1].

3 Higher order derivatives

The extension of the algorithm for the identification of a map and its derivatives of order higher than one is straightforward. Given that ϕ is continuous

[1] Note that in this case the length of vector $\mathbf{w}$ and the number of columns of matrices $\mathbf{P}$ in equation (28) is h instead of H

and has continuous derivatives up to order r, the derivatives of order up to r can be identified for the map f. The data set is organized in $r+1$ vectors $\mathbf{x} = [x(1) \cdots x(N)]^T$, $\mathbf{y} = [y(1) \cdots y(N)]^T$, $\mathbf{y}^{(1)} = [y^{(1)}(1) \cdots y^{(1)}(N)]^T$, ..., $\mathbf{y}^{(r)} = [y^{(r)}(1) \cdots y^{(r)}(N)]^T$, where $y^{(d)}(k)$ is the desired d-th derivative for the k-th data point. In the first step, a different set of regressors is computed for each derivative order:

$$\mathbf{p}_i = [\phi(x(1), x(i)), \ldots, \phi(x(N), x(i))]^T, \;\; 1 \le i \le N_e; \tag{29}$$

$$\mathbf{p}_i^{(d)} = [\frac{\partial^d \phi(x(1), x(i))}{\partial x^d}, \cdots, \frac{\partial^d \phi(x(N), x(i))}{\partial x^d}]^T, \\ 1 \le i \le N_e, \;\; 1 \le d \le r. \tag{30}$$

If we now call $\mathbf{u}_{i_{h-1}}$, $\mathbf{l}_{i_{h-1}}^{(1)}, \cdots$, $\mathbf{l}_{i_{h-1}}^{(r)}$ the orthogonalized regressor vectors selected in the $(h-1)$-th iteration, in the h-th iteration the corresponding $r+1$ error reduction ratios can be computed similarly to what was shown in equations (18–21), and the total error reduction ratio can then be computed as the weighted sum of these terms:

$$\mathbf{u}_{h,i} = \mathbf{p}_i - \sum_{j=1}^{h-1} \mathbf{u}_j (\mathbf{p}_i^T \mathbf{u}_j)/(\mathbf{u}_j^T \mathbf{u}_j), \;\; i \ne i_1, i_2, ..., i_{h-1}; \tag{31}$$

$$\mathrm{err}_{h,i} = (\mathbf{u}_{h,i}^T \mathbf{y})^2 / ((\mathbf{u}_{h,i}^T \mathbf{u}_{h,i})(\mathbf{y}^T \mathbf{y})), \;\; i \ne i_1, i_2, ..., i_{h-1}; \tag{32}$$

$$\mathbf{l}_{h,i}^{(d)} = \mathbf{p}_i^{(d)} - \sum_{j=1}^{h-1} \mathbf{l}_j (\mathbf{p}_i^{(d)^T} \mathbf{l}_j)/(\mathbf{l}_j^T \mathbf{l}_j), \;\; i \ne i_1, i_2, ..., i_{h-1}; \tag{33}$$

$$\mathrm{err}_{h,i}^{(d)} = (\mathbf{l}_{h,i}^{(d)^T} \mathbf{y}^{(d)})^2 / ((\mathbf{l}_{h,i}^{(d)^T} \mathbf{l}_{h,i}^{(d)})(\mathbf{y}^{(d)^T} \mathbf{y}^{(d)})), \;\; i \ne i_1, i_2, ..., i_{h-1}; \tag{34}$$

$$\text{tot\_err}_{h,i} = \lambda_0 \text{err}_{h,i} + \sum_{d=1}^{r} \lambda_d \text{err}_{h,i}^{(d)}, \quad i \neq i_1, i_2, ..., i_{h-1}. \tag{35}$$

A common choice is to uniformly weight the importance of the map and its derivatives, i.e. $\lambda_i = 1/(r+1)$, $i = 0, 1, \ldots, r$. However, a non-uniform weighting can be chosen to reflect non-uniform accuracy requirements.

The regressors with maximum error reduction ratio are selected and removed from the list of availability, and the corresponding centers are added to the set of selected centers:

$$\text{tot\_err}_{h,i_h} = \max_i \{ \text{ tot\_err}_{h,i}, \quad i \neq i_1, i_2, ..., i_{h-1} \}; \tag{36}$$

$$\mathbf{u}_h = \mathbf{u}_{h,i_h}; \tag{37}$$

$$\mathbf{l}_h^{(d)} = \mathbf{l}_{h,i_h}^{(d)}, \quad 1 \leqslant d \leqslant r; \tag{38}$$

$$m_h = x(i_h). \tag{39}$$

If we now let

$$\begin{cases} \mathbf{P}_H & = & [\mathbf{p}_{i_1} \cdots \mathbf{p}_{i_H}] \\ \mathbf{P}_H^{(1)} & = & [\mathbf{p}_{i_1}^{(1)} \cdots \mathbf{p}_{i_H}^{(1)}] \\ & \vdots & \\ \mathbf{P}_H^{(r)} & = & [\mathbf{p}_{i_1}^{(r)} \cdots \mathbf{p}_{i_H}^{(r)}] \end{cases} \tag{40}$$

be the final set of orthogonal regressors obtained from the selection procedure, we can compute the output layer parameters by solving the matrix equation

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^{(1)} \\ \vdots \\ \mathbf{y}^{(r)} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_H & \mathbf{1} \\ \mathbf{P}_H^{(1)} & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{P}_H^{(r)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} + \mathbf{e}. \tag{41}$$

A discussion on the invertibility of the concatenated matrices in (27) and in (41) is reported in the Appendix A.

4 Experimental results

In this section, some examples are presented that illustrate the performance of the proposed algorithm. First, an interpolation experiment is proposed that deals with data from a known function. It is shown how the use of differential data can considerably improve the interpolation task. A second example shows how the proposed algorithm can be used to control the dynamics of a simple dynamical system described by a single-input single-output iterated map. To this purpose, the control over fixed points and its derivatives is exploited to drive the system through different dynamical behaviors ranging from stability to chaos. In all the following examples, the grid of points over which the RBFs are centered is not limited to the input data. Instead, a uniform grid that covers the input interval with an opportune number of grid points is used. This choice is adopted for both the proposed and the classic OLS algorithms, when the two are compared.

4.1 Data interpolation

In this experiment, we compare the traditional and the proposed OLS algorithms for the interpolation of data from a known function. To apply the proposed algorithm we make the assumption that information on the derivative is available for each data point. Assume that the function and its derivative are

$$f(x) = \sin(\cos(x)) \tag{42}$$

and

$$\frac{\partial f(x)}{\partial x} = -\cos(\cos(x))\sin(x), \tag{43}$$

with $x \in [-3, 3]$. The input interval $[-3, 3]$ is made discrete using a resolution step $T = 0.03$, that gives a set of 201 points $\{x_i\}_{i=1}^{201}$. Equation (42) is used to compute the corresponding output, and the training set for the traditional OLS algorithm is generated by randomly selecting $N_p = 6$ points from this set of input-output pairs. A uniform grid of 61 points spaced with step 0.1 is used to center the available regressors over the input interval $[-3, 3]$. The width parameter σ of the gaussian radial basis functions was set to one. Figure 1 shows the result of the training procedure by plotting the output of the trained RBFN over the interval $[-3, 3]$. The algorithm reached the required data approximation error, set to $10E-5$, after 4 iterations (i.e., the selected number of RBF units was 4). The distance from the target function and its derivative over the interval $[-3, 3]$ is evaluated through the summed-square error (SSE) function. It can be seen how the trained RBFN correctly fits the input-output data but fails to accurately represent the underlying function in some regions. This is due to the insufficient amount of data. It is common practice to solve this problem by collecting more data and considering a larger training set.

Assume instead that the value of the first derivative of the function is available at the selected N_p data points. The training set for the extended OLS algorithm is generated by adding the value of the derivative to each of the N_p input-output pairs in the original training set (in our example, these values are computed with equation (43)). The grid over which the regressors are centered and the width of the gaussian functions are the same as before. Figure 2 shows the result of the training procedure. The algorithm reached the required data approximation error after 9 iterations, with the parameter λ set to 0.5. It can be seen that the trained RBFN correctly fits the extended data and that the representation of the underlying function along with its first derivative is improved, as confirmed by the reduced values of SSE.

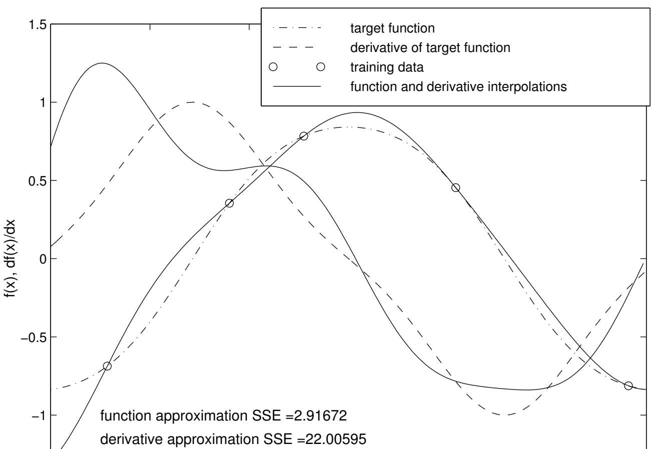

Fig. 1. Interpolation of the function $f(x) = \sin(\cos(x))$ from 5 data points. The RBFN was trained only with the input-output pairs using the standard OLS algorithm, and resulted in a 4-radial basis network.

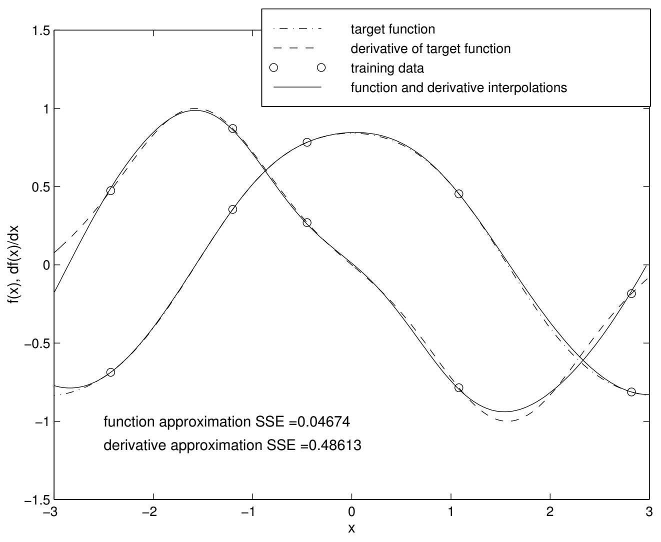

Fig. 2. Interpolation of the function $f(x) = \sin(\cos(x))$ from 5 data points. The RBFN was trained adding the values of the function derivative to the training set and using the proposed algorithm (9 radial basis units where selected).

4.2 Control of fixed point stability in iterated maps

Iterated maps are a common way to describe the discrete-time evolution of dynamical systems [7]. A discrete-time evolution is governed by

$$\mathbf{t}(k+1) = \mathbf{f}(\mathbf{t}(k)) \tag{44}$$

where $\mathbf{t}(k)$ represents the n-dimensional state of the system at discrete-time k and $\mathbf{f}$ is an $\mathbb{R}^n \rightarrow \mathbb{R}^n$ map that transforms the current state in the subsequent state.

In this example we show how to design a single-input single-output map f that can be used to reach different dynamical behaviors, ranging from equilibrium to chaotic motion, by controlling the derivative of the map at a fixed point. We will discuss this example by referring to the properties of the well-known logistic map, defined by the equation $y = f(a, x) = ax(1 - x)$, for all $a \geq 0$

[7]. For $0 \leq a \leq 4$ this equation represents a convex function which intersects the x-axis at 0 and at 1, and maps the interval $[0, 1]$ into itself (refer to figure 3). The intersections of the function with the line $y = x$ are called fixed points. A fixed point (x_f, y_f) is said to be asymptotically stable if $|f'(x_f)| < 1$, which intuitively means that if the initial solution is sufficiently close to the fixed point then the solution remains close to the fixed point thereafter. For $0 \leq a \leq 1$ the fixed point at the origin $x = 0$ is unique and stable. If the value of a is gradually raised over one, the iterated system exhibits an extremely rich spectrum of dynamical behaviours. For $1 < a \leq 3$ the fixed point at the origin becomes unstable, and a second fixed point appears at $x = (a - 1)/a$, which is conversely stable. The iterated system state then diverges from the origin and approaches the stable fixed point as $k \to \infty$.

If $3 < a < 3.449$ the fixed point become unstable and a flip bifurcation with period doubling occurs. The appearance of a periodic motion of period two reflects the fact that the second-generation map of f, defined as $f^2(a, x) = f(a, f(a, x))$, cuts the line $y = x$ at the origin and three other points, the middle of which is a fixed point of f and the rest of which are the two-cycle of f. Let say x_1 and x_2 these two other points. A two-cycle of f is asymptotically stable if the value of $k_2 = f'(x_1)f'(x_2)$ is less then one in magnitude. In general, a period-P orbit $[x_1, x_2, \ldots, x_P]$ is asymptotically stable if $k_P = f'(x_1)f'(x_2)\cdots f'(x_P)$ has magnitude less than one.

As the parameter a is further increased, a variety of stable periodic motions (with period greater than two) and chaotic behaviors succeed, depending on whether the periodic solutions are stable or not.

The qualitative behavior of the logistic map is shared by all smooth and con-

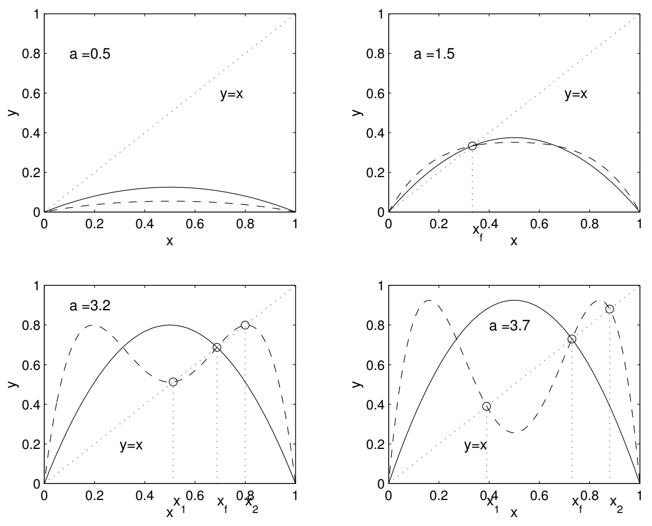

Fig. 3. The logistic map (continuous line) and second-generation map (dashed line) for different values of the parameter.

vex functions $f(x)$ such that $f(0) = 0$, $f(b) = 0$ for some $b > 0$, and $f(x) > 0$ for $0 < x < b$. We take $b = 1$ in our example, and we proceed in the design of a map that exhibits the desired dynamical behavior among stable, periodic and chaotic motion. From the discussion on the logistic map we observe that some of its features, e.g. the derivative at the origin, the fixed point x_f and its derivative, and the fixed points of the second-generation map play a central role in the determination of the dynamic behavior. We focus on these features to design the function, and rely on the relations between derivatives and stability to determine the desired motion.

First, we choose a fixed point within the interval $(0, 1)$, i.e. $y_f = x_f = 0.6$. Next, we can choose the derivative of the function $f(x)$ at $x = 0$, $x = 1$, and $x = x_f$. In order to obtain a convex function, we select a pair of derivatives, e.g. $f'(0) = 1.7$ and $f'(1) = -1.3$, which are compatible with the selected fixed point (x_f, y_f). The derivative at the origin is taken greater than one in magnitude since we want to obtain a motion divergent from the origin and evolving around the value y_f as $k \to \infty$. Finally, we use the derivative at (x_f, y_f) to control the qualitative behavior of the evolution. To this aim, we rely on the fact that, for a derivative less than one in magnitude, an asymptotically stable motion arises. Conversely, for a derivative greater than one in magnitude, either a periodic or a chaotic motion can arise, depending on the stability of the fixed points of the higher-order-generation maps. Unfortunately, there is no straightforward way to control such feature with the proposed method, thus we restrict the control to the first derivatives of the first-generation map f. To show the different qualitative behavior of the iterated system, we choose the following three values for the derivative of the map, $f'(x_f) = \{-.95, -1.2, -2\}$.

To summarize, the training set for the three cases is

$$\begin{array}{lll} x(1) = 0 & y(1) = 0 & y^{(1)}(1) = 1.7 \\ x(2) = 0.6 & y(2) = 0.6 & y^{(1)}(2) = \{-0.95, -1.2, -2.0\} \\ x(3) = 1 & y(3) = 0 & y^{(1)}(3) = -1.3 \end{array}$$

The proposed algorithm is then used to design the map given the selected points and derivatives. A uniform grid of 31 points spaced with step 0.1 is used to center the available regressors over the interval $[-1, 2]$. The width parameter σ of the gaussian radial basis functions was set to 0.5. These parameters are kept the same throughout this example. Figures 4,5, and 6 show the results of the map design and the dynamical behavior of the iterated map in the three cases. The first case corresponds to an asymptotically stable fixed point at $x = 0.6$, with derivative magnitude less than one. The second case reflects the need for a period-two dynamic motion when the fixed point $(0.6, 0.6)$ remains

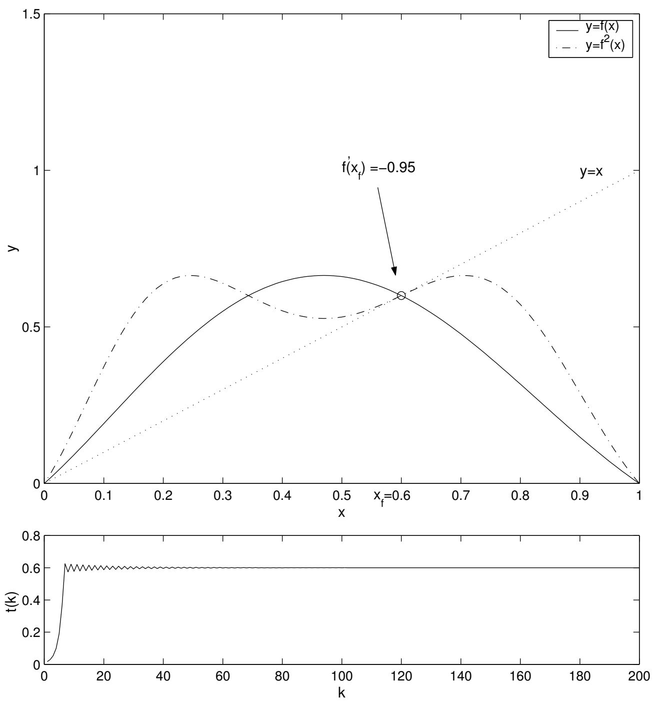

Fig. 4. Upper figure: the map (continuous line) resulting by the training with $f'(0.6) = -0.95$, which gives an asymptotically stable fixed point $(0.6, 0.6)$. The second-generation map (dot-dashed line) is also plotted. Lower figure: autonomous evolution of the dynamical system described by the iterated map (200 iterations). Initial value of the state is $t(0) = 0.01$

the same, although unstable. Note the difference with the simple logistic map case: if we desire a period-two orbit, we can choose $3 < a < 3.449$. However, the fixed point of the map would change depending on a, as it is $y_f = x_f = (a-1)/a$. The periodic motion of the iterated system reflects the fact that the second-generation map of figure 5 now intersects the line $y = x$ in two more points at $x = x_1 = 0.4607$ and $x = x_2 = 0.6976$ other than $x = x_f = 0.6$ and the origin. The third case (figure 6) demonstrate the effect of raising the value of $|f'(0.6)|$ so that $k_2 = f'(x_1)f'(x_2)$ is no longer less than one in magnitude, thus producing an unstable period-two orbit. Since chaotic motion arises in this case, we can deduce that no other P-generation map, with $P > 2$, has any fixed point $[x_1, x_2, \ldots, x_P]$ with $|k_P| = |f'(x_1)f'(x_2)\cdots f'(x_P)| < 1$.

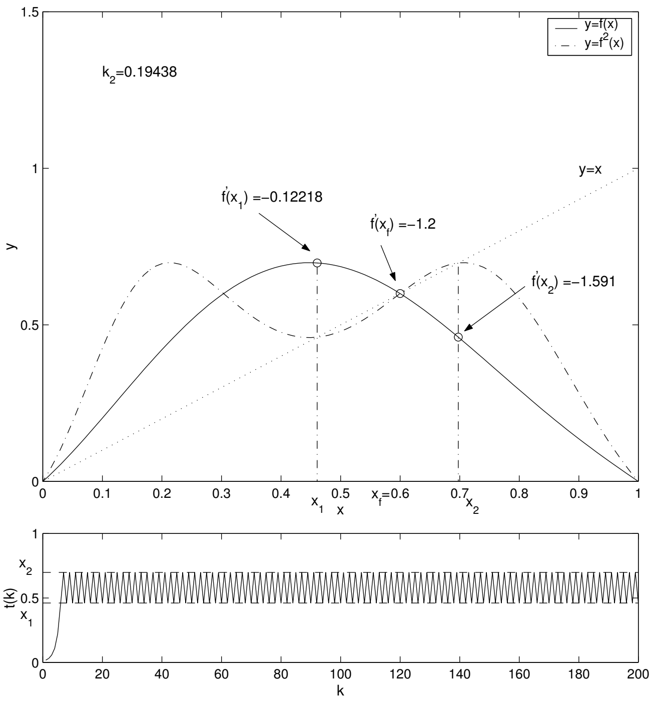

Fig. 5. Upper figure: the map (continuous line) resulting from the training with $f'(x_f) = -1.2$, which makes the fixed point $(0.6, 0.6)$ unstable. From the second-generation map (dot-dashed line) it can be seen how a period-two bifurcation arises, due to the two intersection points of the dot-dashed line with the line $y = x$ (which are stable fixed points of the second-generation map), other than the point $(0.6, 0.6)$ and the origin. The stability of the periodic orbit is due to the stability of the two new second-generation map fixed points. Lower figure: autonomous evolution of the dynamical system described by the iterated map (200 iterations). Initial value of the state is $t(0) = 0.01$

Finally, we may want to stabilize the two fixed points at $x_1 = 0.4607$ and $x_2 = 0.6976$ of the second-generation map in figure 6, with the aim of obtaining a period-two stable orbit with dynamic range $|x_2 - x_1|$. From the previous discussion, we deduce that it is sufficient to put further constraints on the derivatives of the function f so to have $|k_2| = |f'(x_1)f'(x_2)| < 1$. We choose, for

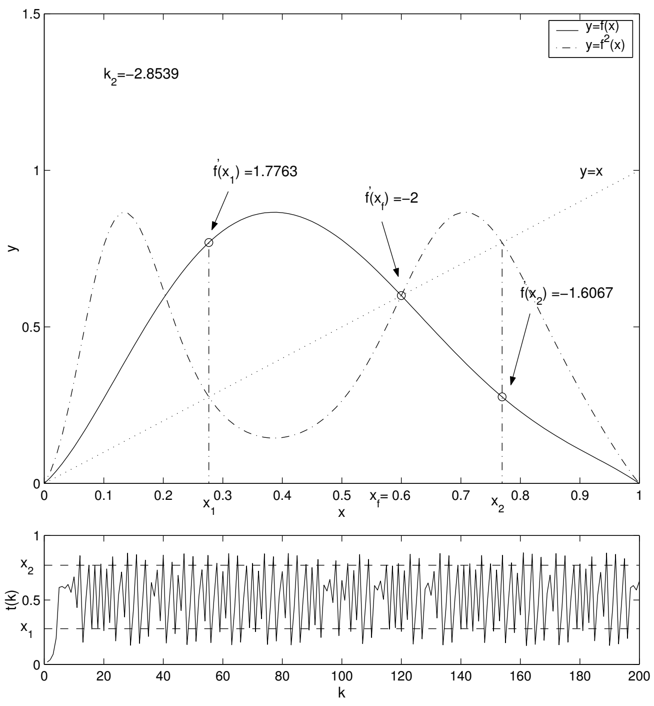

Fig. 6. Upper figure: the map (continuous line) resulting from the training with $f'(x_f) = -2$, which makes the fixed point $(0.6, 0.6)$ unstable for the map. Moreover, this value makes the other two fixed points of the second-generation map unstable as well. No periodic orbit is observable anymore, and a chaotic motion arises. Lower figure: autonomous evolution of the dynamical system described by the iterated map (200 iterations). Initial value of the state is $t(0) = 0.01$

example, $f'(x_1) = 0.9$ and $f'(x_2) = -1.05$, which gives $k = -0.945$. Moreover, the derivatives of the function at $x = 0$ and $x = 1$ were respectively changed to 5 and -4 to keep the function convex. These values were found heuristically. Figure 7 shows the resulting map and the iterated system solution, which asymptotically reaches the desired periodic solution.

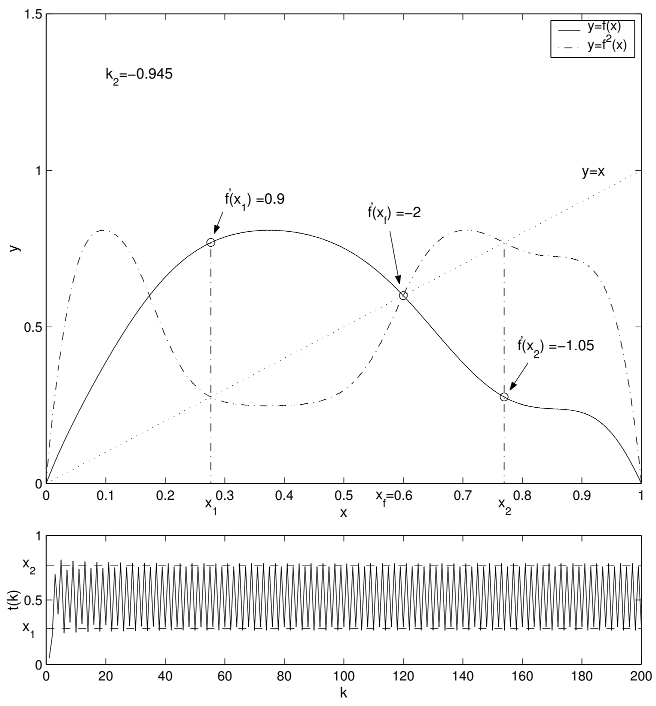

Fig. 7. Upper figure: the map (continuous line) resulting from the training with further constraints $f'(x_1) = 0.9$ and $f'(x_2) = -1.05$. These values make stable the two fixed points of the second-generation map (dashed line) at $x = x_1$ and $x = x_2$, thus allowing a stable period-two orbit. Lower figure: autonomous evolution of the dynamical system described by the iterated map (200 iterations). Initial value of the state is $t(0) = 0.01$

5 Conclusions

The use of the Orthogonal Least Squares algorithm to approximate a non-linear map and its derivatives with radial basis function networks has been investigated. A modified version of the classic OLS algorithm formulation has been proposed, which uses the same orthogonalization approach for both the regressors of the map and the regressors of its derivatives. The usefulness of the method has been illustrated on application examples from the field of function approximation and non-linear system dynamics, and we have stressed the importance of derivatives of the non-linear map to control important features, such as stability and dynamical motion's qualitative behavior.

A Pseudo-invertibility of the regressor matrices

A rectangular matrix $\mathbf{M}$ is pseudo-invertible if the square matrix $\mathbf{M}^T\mathbf{M}$ is invertible, since is $\mathbf{M}^+ = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$. Let us recall now the initial problem in (4) (we have omitted the additive parameter for simplicity, and used $\mathbf{P}_H$ instead of $\mathbf{P}$ to emphasize that the columns in the matrix are regressors selected according to the Gram-Schmidt orthogonalization procedure):

$$\mathbf{y} = \mathbf{P}_H\mathbf{w} + \mathbf{e}. \tag{A.1}$$

The regression matrix $\mathbf{P}_H$ can be decomposed into

$$\mathbf{P}_H = \mathbf{O}_H\mathbf{A}, \tag{A.2}$$

where $\mathbf{O}_H = [\mathbf{u}_1, \ldots, \mathbf{u}_H]$ is a rectangular matrix with orthogonal columns, and $\mathbf{A}$ is a triangular matrix with 1's on the diagonal [5]. The problem thus becomes

$$\mathbf{y} = \mathbf{O}_H\mathbf{v} + \mathbf{e}. \tag{A.3}$$

The triangular matrix $\mathbf{A}$ satisfies the following condition:

$$\mathbf{A}\mathbf{w} = \mathbf{v}. \tag{A.4}$$

We can now compute $\mathbf{w}$ by:

$$\mathbf{w} = \mathbf{P}_H^+\mathbf{y} = \mathbf{A}^{-1}\mathbf{O}_H^+\mathbf{y} = \mathbf{A}^{-1}\mathbf{v}, \tag{A.5}$$

where we used the properties of the pseudo-inverse (see, for example, [21]). In this case, the pseudo-invertibility of $\mathbf{O}_H$ is granted by the fact that $\mathbf{O}_H^T\mathbf{O}_H$

is a diagonal square matrix, and the pseudo-invertibility of $\mathbf{P}_H$ is granted by the pseudo-invertibility of $\mathbf{O}_H$ and by the invertibility of $\mathbf{A}$. When the algorithm is extended to the derivatives of the map, the following matrix has to be pseudo-inverted:

$$\mathbf{O} = \begin{bmatrix} \mathbf{O}_H \\ \mathbf{O}_H^{(1)} \\ \vdots \\ \mathbf{O}_H^{(r)} \end{bmatrix}, \tag{A.6}$$

where $\mathbf{O}_H^{(d)} = [\mathbf{l}_1^{(d)}, \ldots, \mathbf{l}_H^{(d)}]$ is the orthogonal regressor set for the dth derivative, for which is $\mathbf{P}_H^{(d)} = \mathbf{O}_H^{(d)} \mathbf{A}^{(d)}$. The pseudo-invertibility in this case is granted by the fact that

$$\mathbf{O}^T\mathbf{O} = \left[\mathbf{O}_H^T \; \mathbf{O}_H^{(1)^T} \cdots \mathbf{O}_H^{(r)^T}\right] \begin{bmatrix} \mathbf{O}_H \\ \mathbf{O}_H^{(1)} \\ \vdots \\ \mathbf{O}_H^{(r)} \end{bmatrix} = \mathbf{O}_H^T\mathbf{O}_H + \cdots + \mathbf{O}_H^{(r)^T}\mathbf{O}_H^{(r)} \tag{A.7}$$

is a sum of diagonal square matrices. As a result, for the concatenated matrix (A.6) to be pseudo-invertible, it is sufficient that the matrices $\mathbf{O}_H$,$\mathbf{O}_H^{(1)}$,...$\mathbf{O}_H^{(r)}$ be orthogonal, as it happens to be due to the orthogonalization process. Note that mutual orthogonality among sets of bases related to different order of derivatives is not necessary. The last step is to compute the network output weights $\mathbf{w}$:

$$\mathbf{w} = \begin{bmatrix} \mathbf{P}_H \\ \\ \mathbf{P}_H^{(1)} \\ \\ \vdots \\ \\ \mathbf{P}_H^{(r)} \end{bmatrix}^+ \begin{bmatrix} \mathbf{y} \\ \\ \mathbf{y}^{(1)} \\ \\ \vdots \\ \\ \mathbf{y}^{(r)} \end{bmatrix} = \begin{bmatrix} \mathbf{O}_H \mathbf{A} \\ \\ \mathbf{O}_H^{(1)} \mathbf{A}^{(1)} \\ \\ \vdots \\ \\ \mathbf{O}_H^{(r)} \mathbf{A}^{(r)} \end{bmatrix}^+ \begin{bmatrix} \mathbf{y} \\ \\ \mathbf{y}^{(1)} \\ \\ \vdots \\ \\ \mathbf{y}^{(r)} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{A} & \mathbf{0} & \dots & \mathbf{0} \\ \\ \mathbf{0} & \mathbf{A}^{(1)} & & \vdots \\ \\ \vdots & & \ddots & \mathbf{0} \\ \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{A}^{(r)} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{O}_H \\ \\ \mathbf{O}_H^{(1)} \\ \\ \vdots \\ \\ \mathbf{O}_H^{(r)} \end{bmatrix}^+ \begin{bmatrix} \mathbf{y} \\ \\ \mathbf{y}^{(1)} \\ \\ \vdots \\ \\ \mathbf{y}^{(r)} \end{bmatrix} \qquad \text{(A.8)}$$

where the matrix containing the (upper) triangular blocks $\mathbf{A}^{(i)}$ is invertible, being a square matrix with all zeros below the diagonal. We can conclude that the output weights can be computed from the inversion and pseudo-inversion of the two matrices in the last part of (A.8) or, equivalently, from the pseudo-inversion of the matrix containing the blocks $\mathbf{P}_H^{(i)}$.

References

[1] F. M. A. Acosta, Radial basis function and related models: An overview, Signal Processing 45 (1995) 37–58.

[2] P. Cardaliaguet, G. Euvrard, Approximation of a function and its derivatives with a neural network, Neural Networks 5 (1992) 207–220.

[3] M. Casdagli, Nonlinear prediction of chaotic time series, Physica D 35 (1989)

335–356.

[4] S. Chen, S. A. Billings, Neural networks for nonlinear dynamic system modelling and identification, Int. J. of Control 56 (2) (1992) 319–346.

[5] S. Chen, C. F. N. Cowan, P. M. Grant, Orthogonal least squares learning algorithm for radial basis functions networks, IEEE Trans. on Neural Networks 2 (2) (1991) 302–309.

[6] J. T. Connor, R. D. Martin, L. E. Atlas, Recurrent neural networks and robust time series prediction, IEEE Trans. on Neural Networks 5 (2) (1994) 240–254.

[7] P. G. Drazin, Nonlinear Systems, Cambridge Univ. Press, Cambridge, 1992.

[8] C. Drioli, Radial basis function networks for conversion of sound spectra, EURASIP J. on Applied Signal Processing 2001 (1) (2001) 36–44.

[9] A. R. Gallant, H. White, On learning the derivatives on an unknown mapping with multilayer feedforward networks, Neural Networks 5 (1992) 129–138.

[10] S. Haykin, Neural Networks. A Comprehensive Foundation, Macmillan, New York, 1994.

[11] S. Haykin, J. Principe, Making sense of a complex world, IEEE Signal Processing Mag. 15 (3) (1998) 66–81.

[12] K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Networks 3 (1990) 551–560.

[13] R. Langari, L. Wang, J. Yen, Radial basis function networks, regression weights, and the expectation-maximization algorithm, IEEE Trans. on Systems, Man, and Cybernetics-A 27 (5) (1997) 613–623.

[14] G. P. Liu, V. Kadirkamanathan, S. A. Billings, Variable neural networks for adaptive control of nonlinear systems, IEEE Trans. on Systems, Man, and Cybernetics-C 29 (1) (1999) 34–43.

[15] R. Masuoka, Neural networks learning differential data, IEICE Trans. on Information and Systems E83-D (6) (2000) 1291–1300.

[16] T. Poggio, F. Girosi, Networks for approximation and learning, Proc. of the IEEE 78 (9) (1990) 1481–1497.

[17] F. J. Romeiras, C. Grebogy, E. Ott, W. P. Dayawansa, Controlling chaotic dynamical systems, Physica D 58 (1992) 156–192.

[18] A. Sherstinsky, R. W. Picard, On the efficiency of the orthogonal least squares training method for radial basis function networks, IEEE Trans. on Neural Networks 7 (1) (1996) 195–200.

[19] R. Shilling, J. J. Carrol, A. F. Al-Ajlouni, Approximation of nonlinear systems with radial basis function networks, IEEE Trans. on Neural Networks 12 (1) (2001) 1–15.

[20] P. Yee, S. Haykin, A dynamic regularized radial basis function network for nonlinear, nonstationary time series prediction, IEEE Trans. on Signal Processing 47 (9) (1999) 2503–2521.

[21] C. R. Rao, S. K. Mitra, Generalized inverse of matrices and its applications, Wiley, New York, 1971.

List of Figures