On realizability of neural networks-based input-output models

Ü.Kotta, F.N. Chowdhury, S. Nõmm

Abstract

In this paper, we present the Additive Nonlinear Auto Regressive Moving Average (ANARMA) structure as an excellent choice for neural-networks-based input-output models. The advantage of the ANARMA model is that the time-steps in the argument are pair-wise decomposed, which allows the ANARMA model to be realized in state-space, and to linearize the model via dynamic output feedback. Results of recursive training of such NN-ANARMA models are presented; this training can be carried out on-line.

Keywords

Nonlinear discrete-time systems, Input-output model, Neural networks, Realizability.

Ü. Kotta is with the Institute of Cybernetics, Tallinn Technical University, Akadeemia tee 21, 12618, Estonia. Fax: +372 620 41 51, E-mail: kotta@cc.ioc.ee

F.N. Chowdhury is with the University of Louisiana at Lafayette, P.O. Box No.43890, Lafayette, LA 70504-3890, USA E-mail: fnchowdh@louisiana.edu

S. Nõmm is with the Institute of Cybernetics, Tallinn Technical University, Akadeemia tee 21, 12618, Estonia. Fax: +372 620 41 51, E-mail: sven@cc.ioc.ee

This work was partially supported by the Estonian Science Foundation grant N 3738 and by the US National Science Foundation grant No. ECS-9753084.

For correspondence address to Ü. Kotta Institute of Cybernetics, Tallinn Technical University, Akadeemia tee 21, 12618, Estonia. Fax: +372 620 41 51, E-mail: kotta@cc.ioc.ee

I. Introduction

Designing controllers for unknown (possibly nonlinear) engineering plants is a challenging task. Often we start with experimentally obtained input-output data, and little else; our task is to first obtain a mathematical model of the system, and then design the controller. Nonlinear system identification is an active research field, and the most successful identification techniques are currently based on recursive input-output (i/o) models. These are usually NARMA-type models

$$y_t = \Psi(y_{t-1}, \ldots, y_{t-n}, u_{t-1}, \ldots, u_{t-n}) \tag{1}$$

that can be estimated by various techniques. Due to their good function approximation ability function Ψ is often chosen to be a feedforward neural network with $2n$ inputs and one output.

The field of nonlinear control system synthesis has seen a lot of progress during the last two decades. However, practically all the previous work in this field is concentrated on systems having a state space representation [1], [2]. It is interesting to note that despite the large volume of work going on in both nonlinear system identification and (state-space based) controller design fields, there is serious gap between the two. Most system identification is input-output based, but converting these i/o models to state-space has not received the attention it deserves. Typically, for applying control techniques, the NARMA-type model (1) is transformed into an *extended* state-space realization. Specifically, this representation is obtained from equation (1) by taking x_t as the following state vector, involving both past outputs and past inputs: $x_t = (y_{t-1}, \ldots, y_{t-n}, u_{t-1}, \ldots, u_{t-n})$. The disadvantage of the extended state-space realization is that it is nonminimal, and therefore either non-controllable or non-observable or both. Hereinafter, if we speak about realizability, we mean finding the accessible and observable realization of (1).

We show that the typical NN-based NARMA-type model *does not* have a state space

realization, and suggest a new subclass of NN-based models, NN-ANARMA, that can be easily realized in the classical state-space form. Moreover, we extend the existing methods of NN training to estimate the parameters of our model. The approximation ability of the special function corresponding to this subclass of NN-based models has not yet been established theoretically and remains the topic of the future research, but as the simulation in Section 5 shows, this subclass of models can successfully mimic structures that do not actually belong to this class.

Besides the realizability property, the NN-ANARMA model allows to simplify the controller design task also in the input-output domain since this model can be linearized via dynamic output feedback [13].

II. Nonrealizability of typical NN-based NARMA model

Consider a nonlinear system Σ described by input output (i/o) equation (1) where $u \in \mathcal{U} \subset I\!R$ is the scalar input variable, $y \in \mathcal{Y} \subset I\!R$ is the scalar output variable and Ψ is a real analytic function defined on $\mathcal{Y}^n \times \mathcal{U}^n$. Assume that either $\partial\Psi(\cdot)/\partial y_{t-n}$ or $\partial\Psi(\cdot)/\partial u_{t-n}$ is different from zero (Assumption 1). In the realization problem we are looking for transforming the i/o equation (1) into the classical state-space form

$$\begin{array}{rcl} x_{t+1} & = & f(x_t, u_t) \\ y_t & = & h(x_t). \end{array} \tag{2}$$

In [3] the intrinsic and coordinate-free generic necessary and sufficient conditions have been given for the existence of a minimal, i.e. an observable and an accessible state space realization (2) of a nonlinear i/o equation (1). These conditions were formulated in terms of integrability of certain subspaces of one-forms, classified according to their relative degree and associated with the i/o model, via the so-called extended state-space system Σ_e with input u_t and state $z_t = [y_{t-1}, \ldots, y_{t-n}, u_{t-1}, \ldots, u_{t-n}]^T$ defined as

$$z_{t+1} = f_e(z_t, u_t) \tag{3}$$

where $f_e(\cdot) = [\Psi(z), z_1, \ldots, z_{n-1}, u, z_{n+1}, \ldots, z_{2n-1}]^T$. Note that (3) is not an accessible realization.

Let $\mathcal{K}$ denote the field of meromorphic functions in a finite number of variables $\{z_0, u_t, t \geq 0\}$. The forward-shift operator $\delta : \mathcal{K} \to \mathcal{K}$ is defined by $\delta\zeta(z_t, u_t) = \zeta(f_e(z_t, u_t), u_{t+1})$. The pair $(\mathcal{K}, \delta)$ is a difference field. Over the field $\mathcal{K}$ one can define a difference vector space $\mathcal{E} := \mathrm{span}_{\mathcal{K}}\{\mathrm{d}\varphi \mid \varphi \in \mathcal{K}\}$. The operator δ induce the operator $\Delta : \mathcal{E} \to \mathcal{E}$ by $\sum_i a_i \mathrm{d}\varphi_i \mapsto \sum_i (\delta a_i)\mathrm{d}(\delta\varphi_i)$, $a_i, \varphi_i \in \mathcal{K}$. $(\mathcal{K}, \delta)$ is not inversive in general. Nevertheless, it is always possible to imbed $\mathcal{K}$ into an inversive difference overfield $\mathcal{K}^*$, called the inversive closure of $\mathcal{K}$. In [4] an explicit construction of $(\mathcal{K}^*, \delta^*)$ is given. Hereinafter, we use the same symbol to denote the difference field $(\mathcal{K}, \delta)$ and its inversive closure.

The relative degree r of a one-form $\omega \in \mathcal{E}$ is defined to be the least integer such that $\Delta^r\omega \notin \mathrm{span}_{\mathcal{K}}\{\mathrm{d}z\}$. If such an integer does not exist, we set $r = \infty$. A sequence of subspaces $\{\mathcal{H}_k\}$ of $\mathcal{E}$ is defined by

$$\begin{array}{rl} \mathcal{H}_1 & = \mathrm{span}_{\mathcal{K}}\{\mathrm{d}z_0\} = \mathrm{span}_{\mathcal{K}}\{\mathrm{d}y_{-1}, \ldots, \mathrm{d}y_{-n}, \mathrm{d}u_{-1}, \ldots, \mathrm{d}u_{-n}\} \\ \mathcal{H}_{k+1} & = \{\omega \in \mathcal{H}_k \mid \Delta\omega \in \mathcal{H}_k\},\ k \geq 1. \end{array} \tag{4}$$

There exists an integer $k^* \leq 2n$ such that for $0 \leq k \leq k^*$, $\mathcal{H}_{k+1} \subset \mathcal{H}_k$ but $\mathcal{H}_{k+1} \neq \mathcal{H}_k$ and $\mathcal{H}_{k^*+1} = \mathcal{H}_{k^*+2} = \ldots = \mathcal{H}_\infty$. The subspaces $\mathcal{H}_k$ contain the one-forms whose relative degree is equal to k or greater than k, and are invariant under the (extended) state space diffeomorphism.

Recall that the system (2) is said to be locally generically observable if $\mathrm{rank}_{\mathcal{K}}(\partial(y_t, y_{t+1}, \ldots, y_{t+n-1}))/(\partial x_t) = n$ and generically (forward) accessible [4] if $\mathcal{H}_\infty = \{0\}$. The i/o model is said to be irreducible [3] if the associated extended system $\sum_e$ satisfies the accessibility condition $\mathcal{H}_\infty = \{0\}$.

Theorem 1. [3] *The nonlinear system described by the irreducible (minimal) input-output difference equation (1) has a generically observable and accessible state space realization iff for* $1 \leq k \leq n+1$ *the subspaces* $\mathcal{H}_k$ *defined by (4) are completely integrable.*

Moreover, the state coordinates can be obtained by integrating the basis of $\mathcal{H}_{n+1}$.

The $\mathcal{H}_k$ subspaces are sometimes very difficult to find due to the fact their computation requires to find the backward shift operator δ^{-1} for control system which in turn requires the solution of a system of nonlinear algebraic equations. Instead, a related sequence of decreasing subspaces $\{\mathcal{I}_k\}$ defined by $\mathcal{I}_1 = \mathcal{H}_1, \mathcal{I}_{k+1} = \mathcal{I}_k \cap \Delta\mathcal{I}_k, k \geq 1$ can be used to check the realizability of the NARMA model (1). The $\mathcal{I}_k$ subspaces are much more straightforward to find but they cannot be used to construct the coordinates of the state vector.

Lemma 2. [5] $\mathcal{I}_k$ *is completely integrable iff* $\mathcal{H}_k$ *is completely integrable.*

The main purpose of this section is to demonstrate that the majority of existing neural networks (NNs) type models given by

$$y_t = \sum_{i=1}^{l} c_i \phi(w_{i1} y_{t-1} + \ldots + w_{in} y_{t-n} + w_{i,n+1} u_{t-1} + \ldots + w_{i,2n} u_{t-n}) = C\phi[W X_{t-1}] \tag{5}$$

where X_{t-1} is the input vector composed of past outputs $y_{t-1}, \ldots, y_{t-n}$ and past inputs $u_{t-1}, \ldots, u_{t-n}$, that is, $X_{t-1} = [y_{t-1}, \ldots, y_{t-n}, u_{t-1}, \ldots, u_{t-n}]^T$, $\phi(\cdot)$ is a saturation-type smooth nonlinear function, l is the number of hidden neurons, determined by trial and error, and C and W are synaptic weight matrices of appropriate dimensions, do not satisfy those conditions, i. e. are not realizable in the classical state-space form. The reason is that for the typical NN-type model the $\mathcal{H}_k$ subspaces are not integrable.

Note that we assume system (5) to be shift-invariant which means that system (5) is equivalent to

$$y_{t+n} = C\phi[W X_{t+n-1}]. \tag{6}$$

The latter form is better suited for proofs and feedback design.

Theorem 3. *The NN-type NARMA model (5) is, in general, not realizable in the*

classical state space form.

Proof: The proof is a direct consequence of Theorem 1 and Lemma 2, and that of Frobenius Theorem. Note that $\mathcal{I}_1$ and $\mathcal{I}_2$ are always integrable by definition. Compute $\mathcal{I}_3$ for (5)

$$\begin{aligned}\mathcal{I}_3 \;=\; & \operatorname{span}_{\mathcal{K}}\{\mathrm{d}y_{t+2},\ldots,\mathrm{d}y_{t+n},\sum_{i=1}^{l} c_i\delta\phi'(w_{i1}y_t+\ldots+w_{i,2n}u_{t+n-1})\times \\ & \times[w_{i1}\mathrm{d}y_{t+1}+w_{i,n+1}\mathrm{d}u_{t+1}],\mathrm{d}u_{t+2},\ldots,\mathrm{d}u_{t+n-1}\} := \operatorname{span}_{\mathcal{K}}\{v_1,\ldots,v_{2n-2}\}.\end{aligned}$$

By the Frobenius Theorem, a subspace of $\mathcal{E}$, $V=\operatorname{span}_{\mathcal{K}}\{v_1,\ldots,v_r\}$ is closed if and only if $\mathrm{d}v_k\wedge v_1\wedge\ldots\wedge v_r=0$, for any $k=1,\ldots,r$. If V is closed, then there exists locally a system of coordinates $\{\zeta_1,\ldots,\zeta_r\}$ such that $V=\operatorname{span}_{\mathcal{K}}\{\mathrm{d}\zeta_1,\ldots,\mathrm{d}\zeta_r\}$. In this case V is said to be completely integrable. Compute $\mathrm{d}v_n=\mathrm{d}\sum_{i=1}^{l} c_i\delta\phi'(w_{i1}y_t+\ldots+w_{i,2n}u_{t+n-1})[w_{i1}\mathrm{d}y_{t+1}+w_{i,n+1}\mathrm{d}u_{t+1}]=\sum_{i=1}^{l} c_i\mathrm{d}\delta\phi'(w_{i1}y_t+\ldots+w_{i,2n}u_{t+n-1})\wedge[w_{i1}\mathrm{d}y_{t+1}+w_{i,n+1}\mathrm{d}u_{t+1}]$. Applying the Frobenius Theorem to $\mathcal{I}_3$ we get that

$$\begin{aligned}&\mathrm{d}v_n\wedge v_1\wedge\ldots\wedge v_{2n-2}= \\ &\qquad\Big[\sum_{i=1}^{l} c_i w_{i1}w_{i,2n}\delta\phi''(w_{i1}y_t+\ldots+w_{i,2n}u_{t+n-1})\Big] \quad\times \\ &\times\quad\Big[\sum_{j=1}^{l} c_j w_{j,n+1}\delta\phi'(w_{j1}y_t+\ldots+w_{j,2n}u_{t+n-1})\Big] \\ &-\quad\Big[\sum_{k=1}^{l} c_k w_{k,n+1}w_{k,2n}\delta\phi''(w_{k1}y_t+\ldots+w_{k,2n}u_{t+n-1})\Big] \quad\times \\ &\times\quad\Big[\sum_{r=1}^{l} c_r w_{r1}\delta\phi'(w_{r1}y_t+\ldots+w_{r,2n}u_{t+n-1})\Big]\end{aligned} \tag{7}$$

So, the Frobenius condition is not satisfied already for $\mathcal{I}_3$ and of course there is no need to check integrability of $\mathcal{I}_4,\ldots,\mathcal{I}_{n+1}$. ■

III. A new subclass of NN-based models: the ANARMA structure

It is the structure of the typical NN-type model that is creating the problem. In (5), the nonlinear function has in its argument all the time steps together. Namely, the realizability problems are caused by the fact that there is no way to separate (decompose) the different time instances. In this section, we present the Additive Nonlinear AutoRegressive Moving

Average (ANARMA) structure as an excellent choice for neural-networks-based input-output models. The advantage of the ANARMA model is that the time steps in the argument are pair-wise decomposed, which allows the ANARMA model to be realized in classical state-space form. Therefore for such class of NN-type models it is not necessary to develop the control algorithms on the basis of the i/o models: we may apply a wide variety of existing state space techniques. Moreover, this NN-ANARMA model, unlike the general NNs-based model, can be linearized via dynamic output feedback. The NN-ANARMA model has the following structure

$$y_t = \sum_{i=1}^{n} C_i \phi(W_i Z_{t-i}), \tag{8}$$

where $Z_{t-i} = [y_{t-i}, u_{t-i}]^T$, C_i and W_i are $1 \times l$ and $l \times 2$ dimensional matrices, and applying the result of [6] it would be easy to realize this in the classical state space form:

$$\begin{array}{rcl} x_{t+1}^1 & = & x_t^2 + C_1 \phi(W_1 (x_t^1, u_t)^T) \\ x_{t+1}^2 & = & x_t^3 + C_2 \phi(W_2 (x_t^1, u_t)^T) \\ & \vdots & \\ x_{t+1}^{n-1} & = & x_t^n + C_{n-1} \phi(W_{n-1} (x_t^1, u_t)^T) \\ x_{t+1}^n & = & C_n \phi(W_n (x_t^1, u_t)^T) \\ y_t & = & x_t^1 \end{array} \tag{9}$$

A schematic diagram of this model with n sub-NNs is shown in Figure 1. Note that the ith sub-NN has hidden weights W_i.

In the next sections, we describe the structure and implementation of the ANARMA model, and provide the examples of recursive training it with input-output data from the nonlinear plants. If we start with i/o data and have little information about the underlying process that generated the data, it is up to us to choose the model structure before we do system identification. The goodness of fit between the experimental data and the chosen model can be judged after the system identification is complete.

We would like to point out that the nonclassical NN structure (8) is not completely new: in several papers (see for example [7]) the so-called affine NN is used and claimed to retain

the superior properties of NN models. The affine NN is basically a linear combination of several feedforward NNs; in a way, our model is somewhat similar. It should be noted that we do not claim this new structrure to be superior to the typical time-delay-neural-network (TDNN) structure in any way: we only claim that this structure is more convenient because it allows a state-space realization, and is linearizable via dynamic output feedback.

Finally, let us mention that y_t in (8) can be a sum of less that n functions, each being a function of more than two arguments

$$y_t = \sum_{i=1}^{n-k} \tilde{C}_i \phi(\tilde{W}_i \tilde{Z}_{t-i}) \tag{10}$$

for any $k = 0, 1, \ldots, n-1$ where $\tilde{Z}_{t-1} = [y_{t-i}, \ldots, y_{t-i+k}, u_{t-i}]^T$, $\tilde{C}_i$ and $\tilde{W}_i$ are $1 \times l$ and $l \times (2+k)$ dimensional vectors. The model (10) is still realizable in the classical state space form [6].

IV. Structure and training of the additive NARMA model

The additive NARMA model (ANARMA) is defined by equation

$$y_t = \sum_{i=1}^{n} \Psi_i(y_{t-i}, u_{t-i}) \tag{11}$$

and represents a structural restriction of the general model (1), obtained by constraining the form of the function Ψ. Although the model can be justified simply on the basis that in the absence of any knowledge of the underlying dynamics of the process we are free to assume any suitable structure, we present another justification here. Suppose it is known that the original NARMA model (1) is a close representation of the dynamic system under investigation. Then, it can be shown that this system can be approximated by a family of linear time-varying systems [8]. At each discrete time step t, a linear system can match the input-output data of the time-invariant nonlinear plant. This produces a time-varying

family of ARMA (TVARMA) models:

$$y_t = \sum_{i=1}^{n} a_t(i) y_{t-i} + \sum_{i=1}^{n} b_t(i) u_{t-i}. \tag{12}$$

Here, $a_t(i)$ and $b_t(i)$ are the coefficients of the model at time-step t. It is simple to re-arrange (12) pair-wise and obtain: $y_t = [a_t(1)y_{t-1} + b_t(1)u_{t-1}] + [a_t(2)y_{t-2} + b_t(2)u_{t-2}] + ...[a_t(n)y_{t-n} + b_t(n)u_{t-n}]$. Now, replace each pair-wise term by a nonlinear function $\Psi_i(y_{t-i}, u_{t-i})$ and obtain (11).

Although equation (11) does not require any more modification, we propose to replace it by a more convenient form for identification by neural network training:

$$y_t = \sum_{i=1}^{n} C_i q_i(y_{t-i}, u_{t-i}). \tag{13}$$

Each of the functions q_i can be thought of as the output of a sub-neural-network and the terms C_i as the weights with which these sub-NNs are combined to get the output y_t.It is clear that the above equation (13) is equivalent to (11).

This structure, called Additive NARMA or ANARMA, is particularly convenient for certain control design purposes. It also opens the possibility of experimentally determining the model order of the unknown system, since the addition of each sub-NN represents exactly one-order increase.

V. Implementation: Training the ANARMA model with neural networks in real-time

In this paper, we address the issue of on-line training only, where the input-output data are used in a step-by-step fashion. We propose a combination of Kalman-filter/backpropagation for on-line training, but claim that in many practical situations, only the Kalman filter will suffice. Here, we assume that each sub-NN has a hidden layer weight matrix

$W_i(l,2)$, where l is the number of hidden neurons in the (i-th) sub-NN; clearly, the input r_i to each sub-NN has dimension $(2,1)$. The output neuron of each sub-NN is linear, therefore all of them can be combined into one output neuron with a weight vector $C(n,l) = [c_{11} \dots c_{1l}; \dots c_{n1} \dots c_{nl}]^T$. This is the vector that is optimized by casting the problem into the format of a Kalman filter. In order to create a Kalman-filter like format, we compute the vectors $q_{i,t} = \phi(W_i r_{i,t})$ for each sub-NN, and put all $q_{i,t}$ vectors into one vector of length n: $H_t = [q_{1,t}; \dots; q_{n,t}]^T$. Note that $q_{i,t}$ implies "output vector of the i-th sub-NN at time step t", and $r_{i,t} = Z_{t-i} = [y_{t-i}, u_{t-i}]^T$. Using the equation $y_t = H_t^T C_t$ as the observation equation of a *fictitious* dynamic system, and assuming as a *fictitious* state equation: $C_{t+1} = C_t$, we design a discrete-time Kalman filter to estimate C_t. See [8] for the details of a slightly different version of the same estimation technique, with $C_{t+1} = C_t + w_t$,[1] where w_t is an "artificial" process noise, and without the nonlinearities $\phi(.)$. The Kalman estimates are guaranteed to converge provided the data come from a stable plant. The error between the estimated output and the actual output is minimized in a stochastic recursive-least-squares sense, conditioned on the choice of the hidden-layer weight matrices W_i. If this error is higher than the acceptable threshold, we update the hidden-layer weights using ordinary backpropagation, and repeat the Kalman optimization of the output-layer weights. The following is a step-by-step description of the training algorithm.

- Generate the weight matrices W_i randomly, using a Gaussian distribution.
- Compute the vectors $q_{i,t} = \phi(W_i r_{i,t})$ for each sub-NN. Note that $q_{i,t}$ implies "output vector of the i-th sub-NN at time step t", and $r_{i,t} = [y_{t-i}, u_{t-i}]^T$.
- Put all $q_{i,t}$ vectors in one vector of length n: $H_t = [q_{1,t}; \dots; q_{n,t}]^T$
- Generate the vector C_0 randomly as an initial guess.
- Using the equation $y_t = H_t^T C_t$ as the observation equation of a *fictitious* dynamic system,

[1] The random walk model

and assuming as a *fictitious* state equation: $C_{t+1} = C_t$, estimate C_t recursively. Details of the Kalman-filter based training technique can be found in [12].

A. An example of implementaion of the training procedure

We use a plant described by the following input-output equation:

$$\begin{aligned} y_{t+3} &= 0.43y_{t+2} + 0.681y_{t+1} - 0.149y_t + 0.396u_{t+2} + 0.014u_{t+1} - 0.071u_t \\ &- 0.351y_{t+2}u_{t+2} - 0.03y_{t+1}^2 - 0.135y_{t+1}u_{t+1} - 0.027y_{t+1}^3 - 0.108y_{t+1}^2u_{t+1} \\ &- 0.099u_{t+1}^3; \end{aligned} \tag{14}$$

and representing a model of a liquid level system of interconnected tanks [11]. We treat equation (14) as our unknown dynamic system, which is approximated by the ANARMA model, estimated by a neural network. The NN model thus obtained can be shown to admit a classical state space realization, according to the results of [10], [6]. We simulated this plant with a sinusoidal input, and trained a neural network with one hidden layer. We chose a 3rd order model of type (8) with the MATLAB logsig (i.e. $\phi(x) = 1/(1+\exp(-x))$) as the nonlinearity. The training took place for 800 time-steps, after which all weights of the NN were kept constant.

In Figure 2, the solid line indicates the plant output and dashed line indicates NN output. Figure 3 shows the residuals of the process, and Figure 4 shows how the outer-layer weights evolve with time. Note that after the training is stopped, modelling error remains within the same range. The resulting NN-ANARMA model of the type (8) and the matrices C_i and W_i are:

$$\begin{aligned} C_1 &= [-1.0944, 2.1594, -10.6249] \\ C_1 &= [3.7680, -0.6351, 6.4588] \\ C_1 &= [70.6338, -13.8388, 33.0411] \end{aligned} \qquad \begin{aligned} W_1 &= \begin{bmatrix} -0.9304 & -1.5717 & -1.4179 \\ -0.4243 & -0.4608 & -0.4195 \end{bmatrix} \\ W_2 &= \begin{bmatrix} 1.3708 & 0.2031 & -0.2420 \\ -0.8852 & 0.3737 & 0.3589 \end{bmatrix} \\ W_3 &= \begin{bmatrix} -0.0222 & 0.1182 & 0.3005 \\ 0.3058 & 0.0702 & -0.5000 \end{bmatrix} \end{aligned} \tag{15}$$

If we increase the number of hidden neurons from three to seven for each sub-NN, the

residuals of the process are smaller, see Figure 5.

As the second example, we mention some results obtained for a plant taken from [9]:

$$\begin{aligned} y_{t+4} \;=\; & -0.00113 - 0.0628u_{t+2} - 0.0675u_{t+1} - 0.0215u_t + 0.84y_{t+3} \\ & -0.0526u_{t+1}y_{t+2} - 0.053u_{t+2}y_{t+3} + 0.0613y_{t+3}^2 - 0.0071u_{t+2}u_{t+1} \\ & -0.0234u_{t+2}^2y_{t+3} - 0.044u_{t+1}^2y_{t+3} + 0.0573u_{t+2}y_{t+3}^2 - 0.02y_{t+1}^2, \end{aligned} \tag{16}$$

which does not have the structure (11). The model describes the relationship between the varying part of current u and the frequency y of the generated voltage in an electrical generator. We simulated this model with a random input, and trained a neural network with the ANARMA model. We chose a 3rd order model with three hidden neurons for each sub-NN ($l = 3$, so the total number of hidden neurons was 9) and with the MATLAB logsig (i.e. $\phi(x) = 1/(1 + e^{-x})$) as the nonlinearity. The training took place for 250 time-steps, after which all weights on the NN were kept constant. In Figure 6, the solid line indicates the plant output and dashed line, NN output. Figure 7 shows the residuals of the modelling process. Figure 8 shows evolution of outer-layer weights with time. The resulting NN-ANARMA model of the type (8) and the matrices C_i and W_i are:

$$\begin{aligned} C_1 &= [0.5240, 0.5088, 0.3432] & W_1 &= \begin{bmatrix} 0.2492 & 0.0960 & -0.0383 \\ -0.1877 & -0.0691 & 1.2199 \end{bmatrix} \\ C_1 &= [0.9061, 0.5464, 0.7219] & W_2 &= \begin{bmatrix} 0.3078 & 1.4120 & 1.3940 \\ -1.6976 & -0.0626 & -0.6442 \end{bmatrix} \\ C_1 &= [0.52190.3655, 0.3152] & W_3 &= \begin{bmatrix} -2.4781 & -0.4490 & -0.7373 \\ 0.4005 & -0.7535 & -1.0483 \end{bmatrix} \end{aligned} \tag{17}$$

The training technique was verified in many other examples, and details would be available from the authors upon request.

VI. Conclusions

We have demonstrated that the majority of existing neural networks (NNs) type models are not realizable in the classical state-space form. The focus of this paper is on a new subclass of input-output models: the Additive NARMA model. This model has the

advantage that it allows a classical state space realization, which helps in the task of controller design. Moreover, this subclass of models is linearizable via simple dynamic output feedback [13]. In this paper, we have presented the detailed structure of this model and shown a method of on-line recursive training. This training process, based on a novel application of the Kalman filter, is very fast and efficient. The goal of our future work is to formalize the controller design methodology on the bases of the state-space realization for this subclass of NN-based input-output models as well on the bases of linearization of a NN-based ANARMA model via dynamic output feedback. The NN-based state-space model (9) can be alternatively used in combination with numerous existing in the literature NN controllers for state-space models. For example, in [14] a NN controller is proposed to perform feedback linearization with rigorous stability analysis to be used ultimately for output tracking with an acceptable bounded error. The controller suggested in [14] makes the closed-loop system passive so that the additional unknown bounded disturbance do not destroy the stability and tracking of the system.

References

[1] Isidori, A. "Nonlinear Control Systems", 1989, Berlin, 2nd Ed.

[2] Nijmeijer, H. and van der Schaft, A.J. "Nonlinear Dynamical Control Systems", 1990, Springer Verlag, New York.

[3] Kotta, Ü., Zinober, A. and Liu, P. "Transfer equivalence and realization of nonlinear higher order input-output difference equations.", *Automatica*, 2001, **37**, pp. 1771-1778.

[4] Aranda-Bricaire, E., Kotta, Ü. and Moog, C., "Linearization of discrete-time systems", *SIAM J. Control and Optimization*, 1996, **34**, No 6, pp. 1999–2023.

[5] Kotta, Ü., Liu, P. and Zinober, A. S. I., "Transfer equivalence and realization of nonlinear higher order i/o difference equations using MAPLE", *Proc. of the 14th IFAC World Congress*, Beijing, 1999, vol. E, pp. 249–254.

[6] Kotta, Ü. and Sadegh, N. "Two approaches for state space realization of NARMA models: bridging the gap", *Proc of the 3rd IMACS Conf. "Mathmod"*, I. Troch, F. Breitenecker (Eds.), Vienna, Vienna Univ. of Technology, 2000, **1**, pp. 415–419.

[7] M. A. Botto, T. J. J. van der Boom, A. Krijgsman and J. Sa Da Costa. "Predictive control based on neural

network models with I/O feedback linearization", *Int. J. Control*, 1999, **17**, 1538–1559.

[8] Chowdhury, F. N., "Input-output modeling of nonlinar systems with time-varying linear models", *IEEE Transactions on Automatic Control*, 2000, **7**, pp. 1355-1358

[9] Haber, R. and Unbehauer, H., "Structure identification of nonlinear dynamic systems – a survey on input/output approaches", *Automatica*, 1990, **26**, pp. 651–677.

[10] Chowdhury, F. N., Kotta Ü. and Nõmm, S., "On realizability of neural-networks-based input-output models", *Proc. of the 3rd Int. Conf. on Differential Equations and Applications*, St. Petersburg, Russia, 2000, **6**, pp. 47-51.

[11] Billings, S. A. and Fadzil, M. B., "The practical identification of systems with nonlinearities", *Proc. of 7th IFAC/IFORS Symp. Identification Syst. Parameter Estimation*, York, UK, 1985, pp. 155-160.

[12] Chowdhury, F. N., "A Novel Method for On-line Training of Dynamic Neural Networks", *Proc. IEEE Conf. Control Applications*, Mexico City, Mexico, Sept. 2001

[13] Pothin, R., Kotta, Ü. and Moog, C. H., "Output feedback linearization of nonlinear discrete-time systems", *Proc. of the IFAC Conf. on Control System Design*, Bratislava, 2000, pp. 174-179.

[14] Lewis, P. L. Jagannathan, S. Yesildirek, A., "Neural Network Control of Robot Manipulators and Nonlinear Systems", 1999, Taylor & Francis.

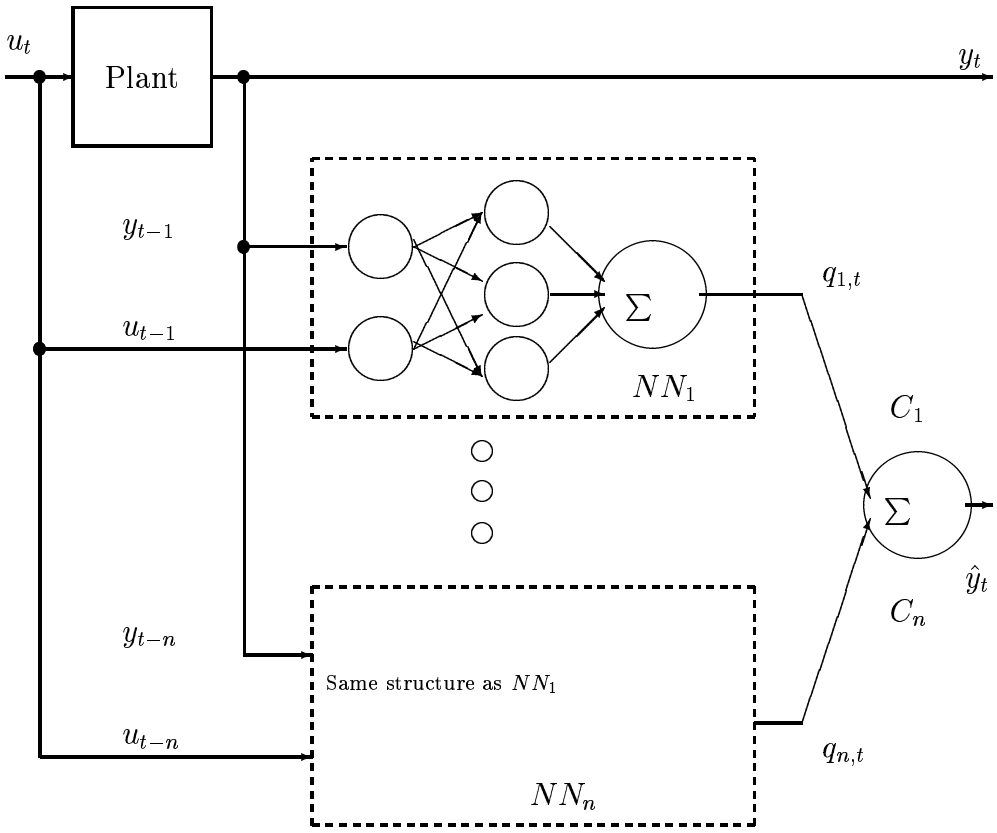

Fig. 1. Additive NARMA model represented by a neural network

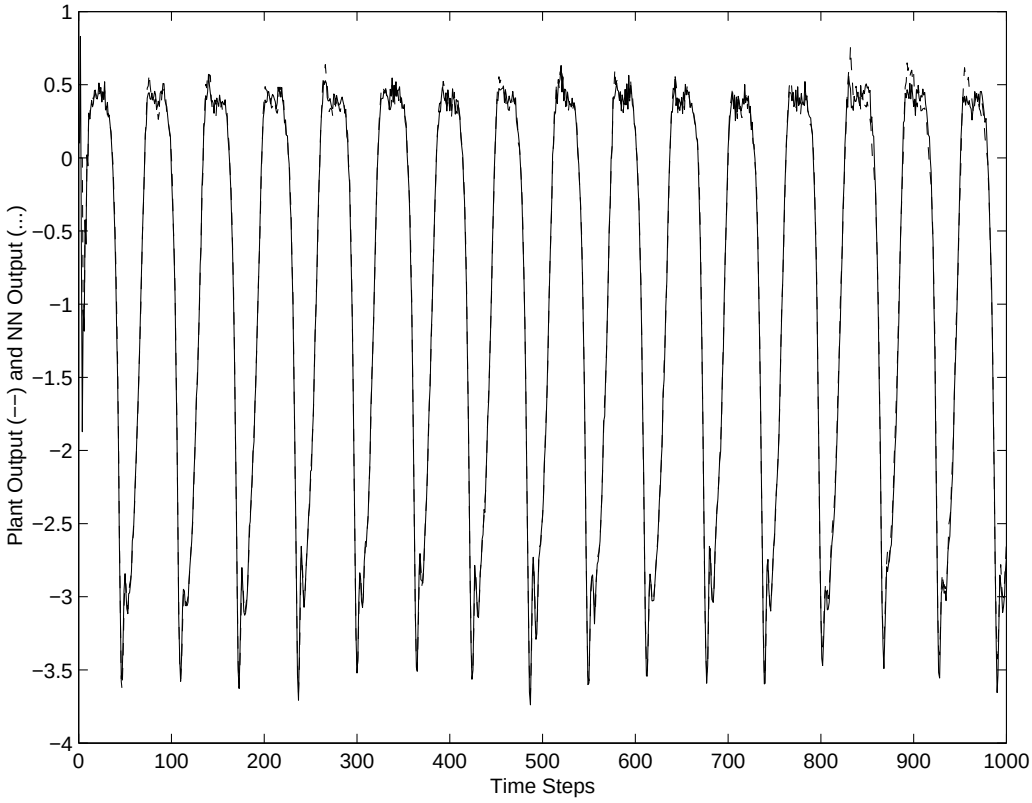

Fig. 2. Plant and NN outputs: ANARMA model

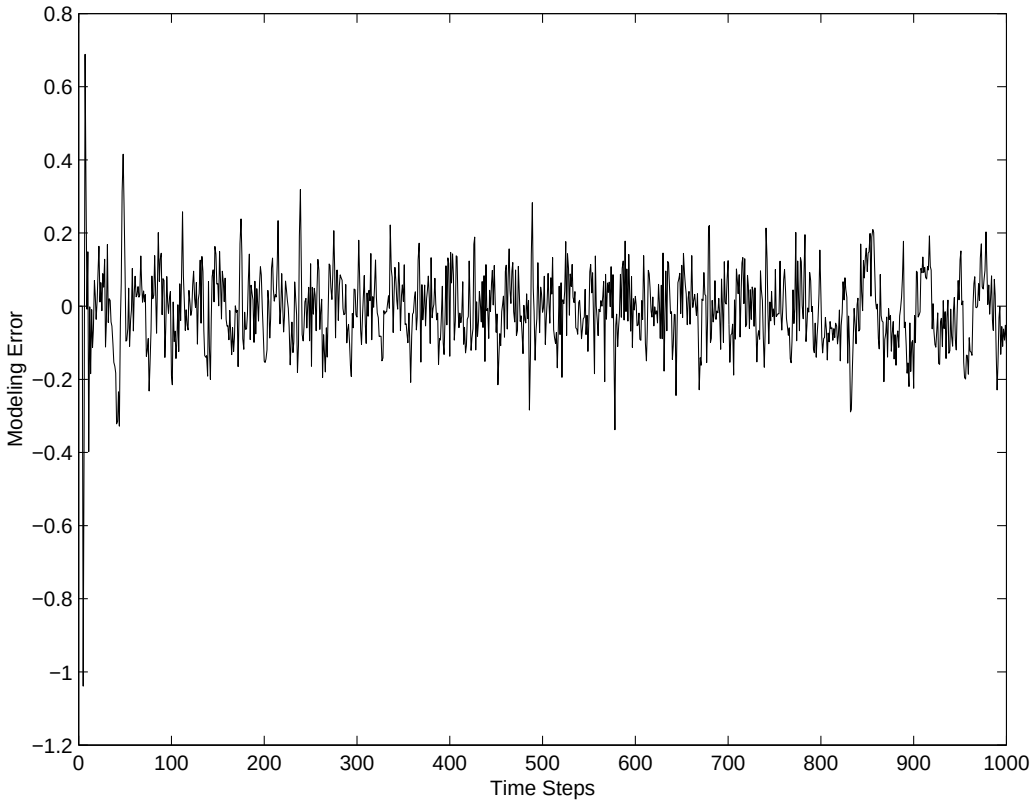

Fig. 3. Residuals of the modeling process

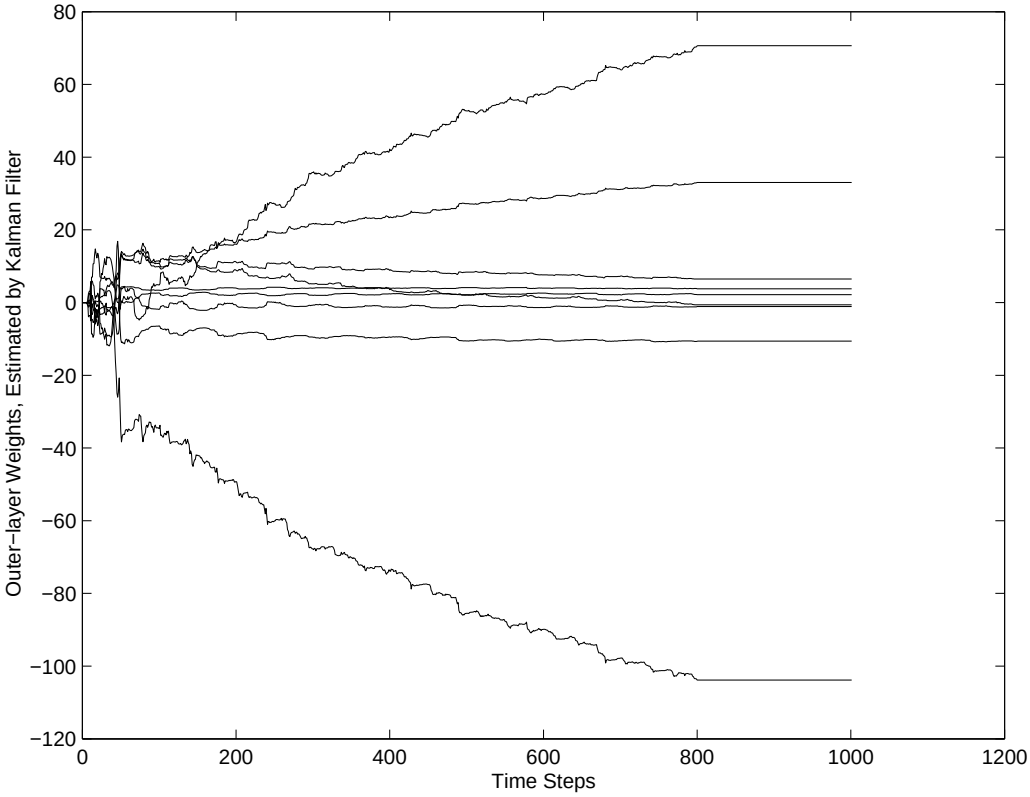

Fig. 4. Evolution of the weights of the outer layer of the ANARMA NN

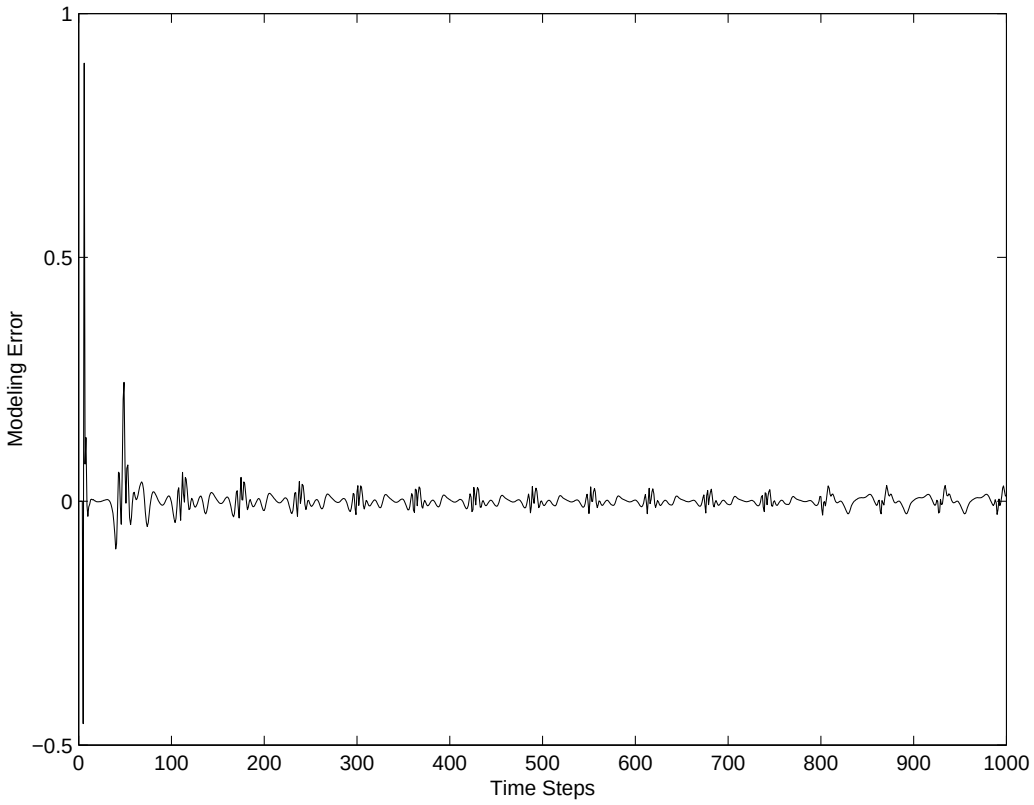

Fig. 5. Residuals of the modeling process

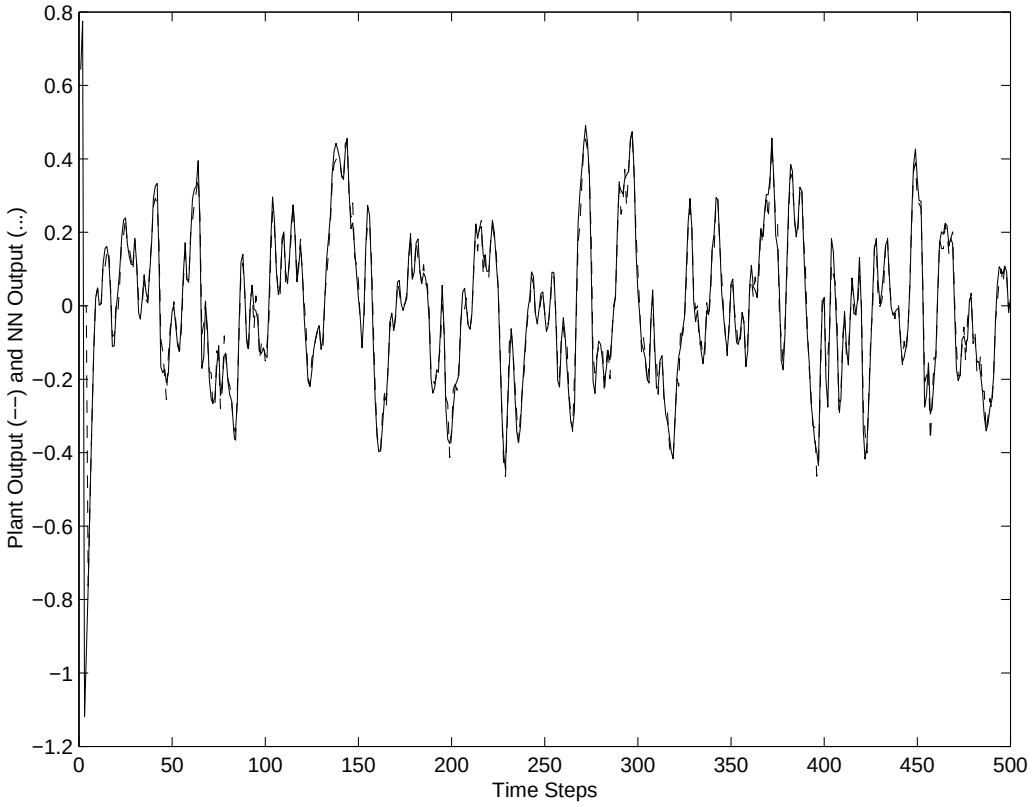

Fig. 6. Plant and NN outputs: ANARMA model

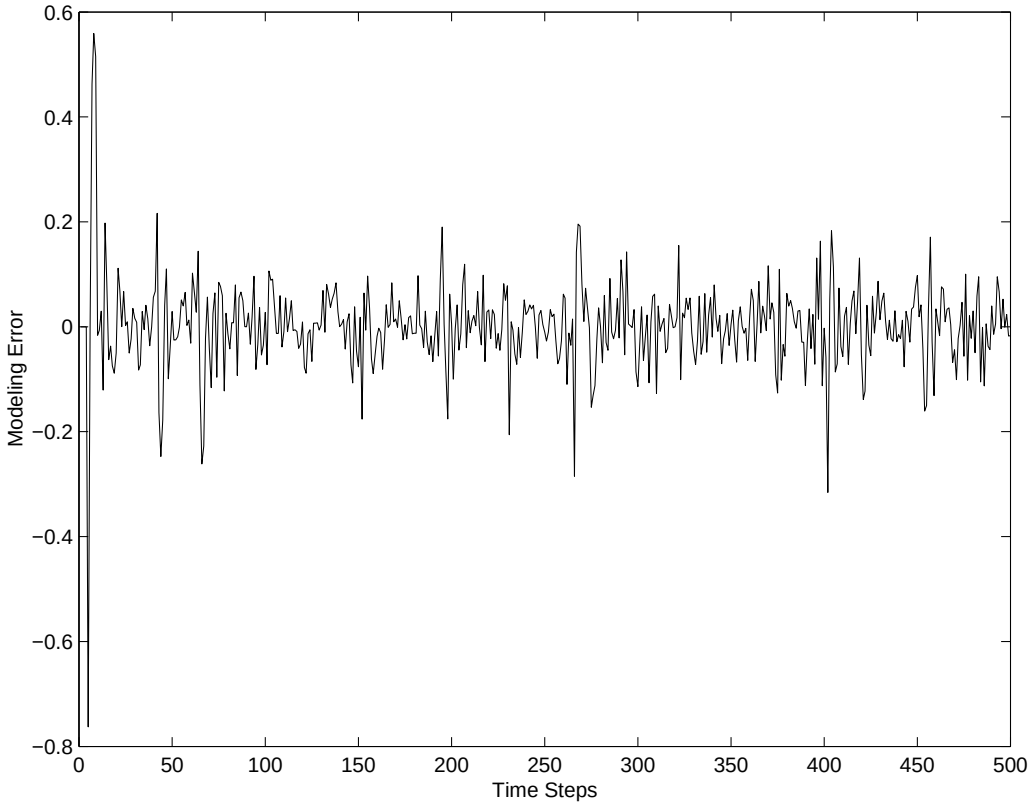

Fig. 7. Residuals of the modeling process

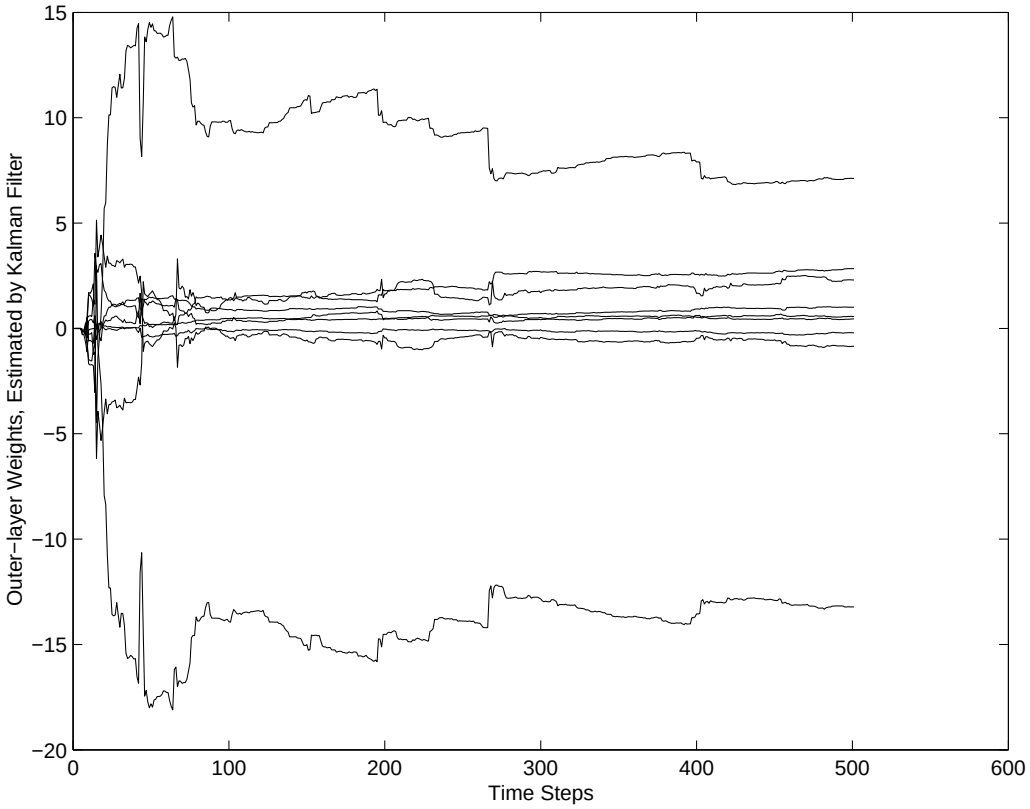

Fig. 8. Evolution of the weights of the outer layer of the ANARMA NN