One Dimensional IIR Digital Filter Modeling Based on Recurrent Neural Network

Stefanova S.A.

Department of Electronic Engineering and Technologies, Technical University of Sofia,
8 Kliment Ohridski Str.,
Sofia, 1797, Bulgaria

***Abstract* - One approach for modeling of 1-D IIR digital filter based on two layer recurrent neural network is proposed. The Lagrange multipliers method has been applied to the training process of the neural network. The set of time domain data is generated and used as a target function in the training procedure. To demonstrate the effectiveness of the proposed neural network model some simulations have been done using input harmonic signals with different frequencies. The analysis of the behaviour of neural network model and target filter frequency responses shows good approximation results.**

I. INTRODUCTION

Neural networks have increasingly been used in many areas of signal processing. In last year various approaches for modeling and simulation of the analog and discrete dynamic systems based on different type of neural networks have been developed [1-9]. Basic results related to the discrete dynamical systems approximation using neural networks are discussed in [1]. Some of these methods are successfully applied to the design problem in time domain for analog [2] the non-recursive, recursive and adaptive digital filters [3-5]. One approach for the 1-D FIR digital filter design based on the weighted mean square method and neural network to state the approximation problem is proposed in [3]. Some methods for the non-linear digital filters design using neural networks are considered in [4]. One approach for time domain design of 1-D and 2-D recursive digital filter, based on recurrent neural network is proposed in [5].

A various methods have been developed for the purposes of the training process of the recurrent neural networks [6-9]. The backpropagation through time method is the basic approach for recurrent neural network training, discussed in [6]. Some training algorithms based on the sensitivity theory are considered in [7, 8]. Reference [5] describes the learning algorithm for the neural network structure based on the state space representation of 1-D and 2-D IIR digital filters.

In this paper an IIR or recursive digital filter model based on two layer recurrent neural network is proposed. The structure and mathematical description of the recurrent neural network is given in section II. The training process of the neural network using Lagrange multipliers method is discussed in Section III. A set of time domain data has been generated and used as a target function in the training procedure. The neural network model has been trained in such a way that with given predetermined input signal, the output variable approximates the target function in mean square sense. The neural network model of the recursive digital filter is considered in section IV. The effectiveness of the proposed neural network model of the recursive digital filter is demonstrated in Section V. The modeling results have been obtained for two different cases – a Nyquist recursive digital filter witch has a special type of the impulse response and a 4-th order bandpass recursive digital filter. Some simulations are realized using harmonic signals with different frequencies in the filter's passband and stopband. The analysis of the behavior of the neural network model and the target filter frequency responses shows good approximation results. The proposed approach can be extended and successfully applied to the case of modeling of nonlinear IIR digital filters.

II. STRUCTURE AND DESCRIPTION OF THE NEURAL NETWORK

The two layer recurrent neural network is considered. The structure of neural network is shown in Fig. 1.

The recurrent neural network is described with the recurrent system of equations as follows:

$$\begin{bmatrix} s_1(k) \\ s_2(k) \\ \cdots\cdots \\ s_n(k) \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \ldots & w_{1n} & \vdots \ldots w_{1n_z} \\ w_{21} & w_{22} \ldots & w_{2n} & \vdots \ldots w_{2n_z} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \vdots\cdots\cdots \\ w_{n1} & w_{n2} \ldots & w_{nn} & \vdots \ldots w_{nn_z} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \cdots\cdots \\ x_n(k) \\ u_1(k) \\ \cdots\cdots \\ u_{n_u}(k) \end{bmatrix},$$

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \cdots\cdots \\ x_n(k+1) \end{bmatrix} = \begin{bmatrix} f(s_1(k)) \\ f(s_2(k)) \\ \cdots\cdots \\ f(s_n(k)) \end{bmatrix} \tag{1}$$

where f(.) is the activation function for the first layer of the neural network

Let be introduced the vector function $\mathbf{f}(\mathbf{s})$ as follows:

$\mathbf{f}(\mathbf{s}) = [f(s_1), f(s_2), \ldots, f(s_n)]^T$,

then the neural network description in the matrix form is:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{W}\mathbf{z}(k)), \quad \mathbf{x}(k_0) = \mathbf{x}_0 \quad , \tag{2}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) \tag{3}$$

M. Iskander et al. (eds.), *Technological Developments in Education and Automation*,
DOI 10.1007/978-90-481-3656-8_52, © Springer Science+Business Media B.V. 2010

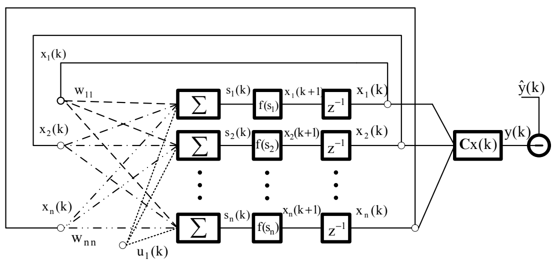

Fig. 1. Recurrent neural network structure for digital filter modeling

where

$\mathbf{x} \in \mathbf{R}^n$ is the state space vector,

$\mathbf{u} \in \mathbf{R}^{n_u}$ is the vector of neural network input excitations

$\mathbf{y} \in \mathbf{R}^m$ is the vector of neural network output,

$\mathbf{W} \in \mathbf{R}^{n \times n_z}$ and $\mathbf{C} \in \mathbf{R}^{m \times n}$, $m \le n$, $\mathbf{C} = \lfloor c_{ij} \rfloor$,

$c_{ij} = \begin{cases} c_i & i = j \\ 0 & i \neq j \end{cases}$ are the matrixes of system coefficients,

n - is the number of neurons at the first layer;

n_u - is the number of input excitations;

$n_z = n + n_u$ - is the number of neural network inputs.

The equations (2), (3) can be used to describe the recurrent neural network shown in Fig. 1.

The following additional vectors have been defined:

- a vector of the first layer inputs of neural network

$$\mathbf{z} = [\, x_1, x_2, \ldots x_n, u_1, u_2, \ldots, u_{n_u} \,]^T, \ \mathbf{z} \in \mathbf{R}^{n_z} \tag{4}$$

- a vector of the neural network outputs

$$\mathbf{y}(k) = [\, y_1(k), y_2(k) \ldots y_m(k) \,]^T \tag{5}$$

- a vector of the neural network weighting coefficients

$$\mathbf{p} = \lfloor w_{11}\ w_{12} \ldots w_{1n_z}, w_{21}\ w_{22} \ldots w_{2n_z}, \ldots w_{nn_z}, c_1, c_2 \ldots c_m \rfloor,$$

$$\mathbf{p} \in \mathbf{R}^{n_p} \tag{6}$$

where $n_p = n.n_z + m$ is the number of neural network coefficients and the elements of matrix $\mathbf{W} \in \mathbf{R}^{n \times n_Z}$ are introduced row by row.

Mean square error objective function is defined in following form:

$$\mathrm{J}(\mathbf{p}) = \frac{1}{2} \sum_{k=k_0}^{k_f - 1} \sum_{i=1}^{m} [\mathrm{y_i(k)} - \hat{\mathrm{y}}_\mathrm{i}(\mathrm{k})]^2 \tag{7}$$

where $\{\, \hat{y}_i(k) \,\}$ is a set of samples of the target function.

III. Algorithm of Lagrange Multipliers

The main problem in the neural network training process is the gradient calculation of the mean square objective function (7) with respect to weights. The algorithm of Lagrange multipliers [9] is used as a training procedure of the neural network model. The Lagrange multipliers algorithm has been written in matrix form.

The vectors of Lagrange multipliers are defined as:

$\boldsymbol{\lambda}(k) = [\, \lambda_1(k), \lambda_2(k), \ldots, \lambda_n(k) \,]^T$;

$\boldsymbol{\Gamma}(k) = [\, \Gamma_1(k), \Gamma_2(k), \ldots, \Gamma_{n_p}(k) \,]^T$;

and the Hamiltonian of the optimization problem (2),(3), (7) is stated in the form:

$$H = \frac{1}{2} \sum_{i=1}^{m} (\, y_i(k) - \hat{y}_i(k))^2 + \boldsymbol{\lambda}^T (k+1) [\, f(\sum_{j=1}^{n_z} w_{1j} z_j \,), \\ f(\sum_{j=1}^{n_z} w_{2j} z_j \,), \ldots, f(\sum_{j=1}^{n_z} w_{nj} z_j \,) \,]^T + \boldsymbol{\Gamma}^T (k+1) \mathbf{p} \tag{8}$$

Using (8) the conjugated system is composed as follows:

$$\boldsymbol{\lambda}(k) = \frac{\partial H}{\partial x(k)} \qquad \boldsymbol{\lambda}(\kappa_f) = 0\,; \tag{9}$$

$$\boldsymbol{\Gamma}(k) = \frac{\partial H}{\partial \mathbf{p}(k)} \qquad \boldsymbol{\Gamma}(\kappa_f) = 0\,. \tag{10}$$

The gradient of the objective function (7) can be calculated using the following algorithm:

Step 1. Calculate and store the set of values $\{\, \mathbf{x}(k) \,\}$ from (2) for $\mathbf{x}(k_0) = \mathbf{x}_0;\ k = k_0, k_0 + 1, \ldots, k_f - 1$.

Step 2. Solve the conjugate system (9) и (10) for $k = k_f - 1, \ldots, k_0$ (backwards in time).

Step 3. Obtain the objective function gradient from the solution of (10) for $k = k_0$

$$\nabla J(\mathbf{p}) = \boldsymbol{\Gamma}(k_0)\,.$$

The conjugated system (9), (10) can be written in matrix form. Then it is necessary to define the following matrix and vectors:

- a sub-matrix $\mathbf{W}_x$ of the weighting coefficients matrix $\mathbf{W}$:

$$\mathbf{W}_x = \begin{bmatrix} w_{11} & w_{12} \cdots & w_{1n} \\ w_{21} & w_{22} \cdots & w_{2n} \\ \cdots & \cdots & \cdots \\ w_{n,1} & w_{n,2} \cdots & w_{n,n} \end{bmatrix}; \tag{11}$$

- a matrix of Lagrange multipliers for the first layer of the neural network:

$$\boldsymbol{\Gamma}^w(k) = \begin{bmatrix} \Gamma_{11}(k)\, \Gamma_{12}(k) \ldots \Gamma_{1,n_z}(k) \\ \Gamma_{21}(k)\, \Gamma_{22}(k) \ldots \Gamma_{2,n_z}(k) \\ \cdots \\ \Gamma_{n1}(k)\, \Gamma_{n2}(k) \ldots \Gamma_{n,n_z}(k) \end{bmatrix}; \tag{12}$$

- a vector of Lagrange multipliers for the second layer of the neural network:

$$\boldsymbol{\Gamma}_c(k) = \left[\Gamma_{c1}(k)\, \Gamma_{c2}, \ldots, \Gamma_{cm} \right]^T; \tag{13}$$

- a diagonal matrixes

$$\mathbf{D}_1 = diag(\, f'(s_1(k))\, f'(s_2(k)), \ldots, f'(s_n(k)) \,)$$

$$\mathbf{D}_2 = diag(\, x_1(k)\, x_2(k), \ldots, x_m(k) \,)\,. \tag{14}$$

The error vector from the objective function (7)