

Important Copyright Notice:

The provision of this paper in an electronic form in this site is only for scholarly study purposes and any other use of this material is prohibited. What appears here is a near-publication draft of the final paper as appeared in the journal or conference proceedings. This is subject to the copyrights of the publishers. Please observe their copyrights.

Missing Data Handling by A Multi-Step Ahead Predictive Filter

R. HoseinNezhad¹, A. Bab-Hadiashar¹, P. Harding²

¹School of Engineering and Science, Swinburne University of Technology, John St., Hawthorn, Victoria 3122, Australia

E-mail: {rhoseinnezhad, abab-hadiashar}@swin.edu.au

²Pacifica Group Technologies, 264 East Boundary Rd., East Bentleigh, Victoria 3165, Australia
E-mail: peter_harding@pgt.com.au

Abstract

A multi-step ahead predictive filter for missing data handling is presented in this paper. This is a simple FIR filter, useful for time and memory critical applications. Furthermore, our proposed algorithm is formulated in such a way that only one set of FIR weights are tuned (by steepest gradient descent method), memorised and used by the system for different numbers of steps ahead in prediction. Hence, this filter is also suitable for memory critical applications. Our proposed filter, the first order hold method and a non-linear neural predictive filter were applied to missing data handling by prediction of up to four missing samples of sensor data in the HMI section of a real time and memory critical application. Experimental results show that prediction error of our proposed filter is decreased by 80% with respect to the first order hold method and it is comparable to the neural filter. Moreover, execution time of data missing handling by our proposed method is 30 times shorter than execution time of the neural filter. Similarly, the number of weights required to be memorised in our method, is 30 times lower than the number of weights in the neural filter. Hence, a high prediction performance is achieved with low computational overhead, which makes our method suitable for real time predictive filtering in time and memory critical applications.

1 Introduction

In time and memory critical applications where prediction of missing data is required, predictive filtering is a useful tool. FIR (Finite Impulse Response) predictive filters are usually used to predict future values of a signal and simultaneously perform low-pass filtering. Heinonen et.al. [1] applied FIR predictive filters in combination with median filters to measure systolic blood pressure. Harju et.al. [2] introduced a method of designing predictive IIR (Infinite Impulse Response) filters via feedback extension of FIR predictive filters. They showed that the advantage of the IIR structures over their FIR counterparts is a more desirable frequency response obtained with considerably smaller memory and computational overhead. Händel [3] proposed a multi-step ahead predictive filter for narrow band waveforms with m distinct spectral peaks. He showed that the noise gain of the filter can be minimised and the coefficients of an optimal L -th order FIR predictor are obtainable, if $L \geq 2m-1$. He also studied the feedback extension of this predictor. Koppinen et.al. [4,5] proposed a filter which is able to generate any window of a linearly predictable signal, including the polynomial, weighted exponential and Hamming-window responses. Tanskanen et.al. [6] presented a General Parameter (GP) extension to polynomial and sinusoidal FIR predictors for extended sinusoidal and Rayleigh fading signal prediction. With a single adaptive general parameter, they extended prediction capabilities of polynomial and sinusoidal FIR predictors beyond polynomial input signals or beyond the nominal design frequencies. This allows for more accurate prediction of input signals with unknown time varying statistics.

In this paper, we introduce a new filter to predict missing samples of sensor data in time and memory critical applications, where we are limited to choose simple and stable algorithms for signal processing. Furthermore, in such applications, data prediction may be required after different numbers of missing samples, i.e. we may need to perform missing data

handling by one-step, two-steps or three-step ahead prediction and so on. If any of the FIR structures available in current literature are applied, then different predictive filters should be designed for different numbers of missing data samples (i.e. for different numbers of steps ahead in prediction). Thus, different sets of filter weights should be saved in the system memory and this is undesirable due to memory critical issues. Moreover, some of the system processing time will be spent for selection of a filter with required number of steps ahead in prediction. This is also undesirable due to time critical issues.

We have attempted to solve the above problem by using only one set of FIR weights for different numbers of steps ahead in prediction. A new weight tuning algorithm has been formulated based on the steepest gradient method. In the case of one-step ahead prediction (one missing sample), the output of the filter, i.e. predicted current sample, is a linear combination of previous samples and the filter is simply a FIR filter. In the case of having more consecutively missing samples, the filter outputs same linear combination of previous data samples and estimated samples. Hence, less memory is required for saving and real-time adaptation of filter coefficients.

In section 2, we present the problem statement, then our new algorithm for tuning of filter weights is introduced in section 3. Experimental results are explained in section 4 and concluded in section 5.

2 Problem Statement

In an application, where a processing unit is receiving data samples from a sensor, via a network, some of the data samples are missed because of different types of faults such as instantaneous short circuits, network faults, and so on. The processing unit should replace the missing samples with some predicted values. It is assumed that maximally L consecutive missing data samples are handled. In the case of more than L consecutive missing samples, processing unit will switch to another policy in which the control command is determined based on the other available sensory data. We have designed and formulated our missing data handling scheme for $L=4$ which is enough for many real time applications, where a large number of missing samples is intolerable.

Based on the number of recently missed samples, a missing sample is handled by one-step ahead, two-step ahead, ... or L -step ahead prediction. The predicted value of each missing sample is a linear combination of the recent P values, as follows:

$$\hat{y}(k) = \sum_{i=1}^P a_i y'(k-i) \quad (1)$$

where $y'(k-i)$ is either equal to the previous data sample $y(k-i)$ or the previously predicted sample $\hat{y}(k-i)$. Because of the available limitations in memory, complexity and code execution delay, it is not desired to search for different a_i coefficients for one-step ahead, two-step ahead, ... and L -step ahead prediction. Hence, we intend to tune the same a_i values for prediction in all cases. In the case of one-step ahead prediction, the first missing sample is given by:

$$y_1(k) = \sum_{i=1}^P a_i y(k-i) \quad (2)$$

In the case of two-step ahead prediction, a second missing sample is estimated by the same linear combination of the recently predicted sample and $P-1$ previous data samples, as below:

$$y_2(k) = a_1 y_1(k-1) + \sum_{i=2}^P a_i y(k-i) \quad (3)$$

Replacing $y_1(k-1)$ from (2) in (3) results the following equation:

$$y_2(k) = a_1 a_P y(k-P-1) + \sum_{i=2}^P (a_1 a_{i-1} + a_i) y(k-i) \quad (4)$$

In the case of three-step ahead prediction, similarly the predicted sample is formulated as below:

$$y_3(k) = a_1 y_2(k-1) + a_2 y_1(k-2) + \sum_{i=3}^P a_i y(k-i) \quad (5)$$

and by replacing $y_1(k-2)$ from (2) and $y_2(k-1)$ from (4), the following formula is obtained:

$$\begin{aligned} y_3(k) = & (a_1^2 a_{P-1} + a_1 a_P + a_2 a_{P-1}) y(k-P-1) + (a_1^2 a_P + a_2 a_P) y(k-P-2) \\ & + \sum_{i=3}^P (a_1^2 a_{i-2} + a_1 a_{i-1} + a_2 a_{i-2} + a_i) y(k-i) \end{aligned} \quad (6)$$

Finally four-step ahead prediction, is formulated as follows:

$$\begin{aligned} y_4(k) = & a_1 y_3(k-1) + a_2 y_2(k-2) + a_3 y_1(k-3) + \sum_{i=4}^P a_i y(k-i) \\ = & \sum_{i=4}^{P+3} A_i y(k-i) \end{aligned} \quad (7)$$

where:

$$\begin{aligned} A_i = & a_i + a_1 a_{i-1} + a_2 a_{i-2} + a_3 a_{i-3} + a_1^2 a_{i-2} + a_1^3 a_{i-3} + 2a_1 a_2 a_{i-3} \quad (4 \leq i \leq P) \\ A_{P+1} = & a_3 a_{P-2} + a_2 a_{P-1} + a_1 a_P + a_1^2 a_{P-1} + a_1^3 a_{P-2} + 2a_1 a_2 a_{P-2} \\ A_{P+2} = & a_3 a_{P-1} + a_2 a_P + a_1^2 a_P + a_1^3 a_{P-1} + 2a_1 a_2 a_{P-1} \\ A_{P+3} = & a_3 a_P + a_1^3 a_P + 2a_1 a_2 a_{P-1} \end{aligned} \quad (8)$$

Equations (2), (4), (6), (7) and (8) show that our proposed predictor is a FIR filter in all cases. The problem addressed in this paper is to tune a_i weights in such a way that the resulting filter can optimally predict missing values of a signal by (2), (3), (5) or (7).

3 Filter Weights Tuning Algorithm

In our proposed tuning algorithm, we try to search for a_i values that minimise the following error function:

$$\begin{aligned} E = & C_1 \sum_{k=1}^N (y(k) - y_1(k))^2 + C_2 \sum_{k=1}^N (y(k) - y_2(k))^2 + C_3 \sum_{k=1}^N (y(k) - y_3(k))^2 \\ & + C_4 \sum_{k=1}^N (y(k) - y_4(k))^2 \end{aligned} \quad (9)$$

where N is the total number of samples used for off-line tuning and C_i constants are selected based on the priority of the performance of one to four-step ahead predictions. This

prioritisation depends on the application. In most real time applications, a lower number of steps ahead in prediction has a higher priority, i.e. $C_1 > C_2 > C_3 > C_4$.

In order to optimise this error function, steepest gradient method is utilised. In this method, the weight vector $\Theta = [a_1 \ a_2 \ \dots \ a_p]^T$ is iteratively moved in the reverse direction of gradient vector of the error function with respect to the weight vector. Hence, calculation of this gradient is required. It is expressed as below:

$$\nabla_{\Theta} E = \left[\frac{\partial E}{\partial a_1} \quad \frac{\partial E}{\partial a_2} \quad \dots \quad \frac{\partial E}{\partial a_p} \right]^T \quad (10)$$

$$\frac{\partial E}{\partial a_j} = -2 \sum_{i=1}^4 \left\{ C_i \sum_{k=1}^N \frac{\partial y_i(k)}{\partial a_j} (y(k) - y_i(k)) \right\} \quad (11)$$

Equations (4), (6), (7) and (8) show that direct calculation of $\partial y_i(k)/\partial a_j$ and its formulation are complicated for $i > 1$. Instead, an indirect iterative calculation based on (2), (3), (5) and (7), is easier to formulate and compute. Partial differentiation of (2) with respect to a_j simply results:

$$\frac{\partial y_1(k)}{\partial a_j} = y(k-j) \quad (12)$$

Partial differentiation of (3) with respect to a_j and replacing $\partial y_1/\partial a_j$ from (12) results:

$$\begin{aligned} \frac{\partial y_2(k)}{\partial a_1} &= y_1(k-1) + a_1 y(k-2) \\ \frac{\partial y_2(k)}{\partial a_{j>1}} &= y_1(k-j) + a_1 y(k-j-1) \end{aligned} \quad (13)$$

Partial differentiation of (5) with respect to a_j and replacing $\partial y_1/\partial a_j$ and $\partial y_2/\partial a_j$ from (12) and (13) results:

$$\begin{aligned} \frac{\partial y_3(k)}{\partial a_1} &= y_2(k-1) + a_1 y_1(k-2) + (a_1^2 + a_2) y(k-3) \\ \frac{\partial y_3(k)}{\partial a_2} &= y_1(k-2) + a_1 y(k-3) + (a_1^2 + a_2) y(k-4) \\ \frac{\partial y_3(k)}{\partial a_{j>2}} &= y(k-j) + a_1 y(k-j-1) \end{aligned} \quad (14)$$

Finally, partial differentiation of (7) with respect to a_j and replacing $\partial y_1/\partial a_j$, $\partial y_2/\partial a_j$ and $\partial y_3/\partial a_j$ from (12) to (14) results:

$$\begin{aligned} \frac{\partial y_4(k)}{\partial a_1} &= y_3(k-1) + a_1 y_2(k-2) + (a_1^2 + a_2) y_1(k-3) + (a_1^3 + 2a_1 a_2 + a_3) y(k-4) \\ \frac{\partial y_4(k)}{\partial a_2} &= y_2(k-2) + a_1 y_1(k-3) + (a_1^2 + a_2) y(k-4) + (a_1^3 + 2a_1 a_2 + a_3) y(k-5) \\ \frac{\partial y_4(k)}{\partial a_3} &= y_1(k-3) + a_1 y(k-4) + (a_1^2 + a_2) y(k-5) + (a_1^3 + 2a_1 a_2 + a_3) y(k-6) \\ \frac{\partial y_4(k)}{\partial a_{j>3}} &= y(k-j) + a_1 y(k-j-1) + (a_1^2 + a_2) y(k-j-2) + (a_1^3 + 2a_1 a_2 + a_3) y(k-j-3) \end{aligned} \quad (15)$$

The gradient vector in (10) and (11) can be iteratively calculated by (12) to (15) and then applied to weight adaptation by the following rule:

$$\Delta\Theta(l) = -\eta \nabla_{\Theta} E(l) + \alpha \Delta\Theta(l-1) \quad (16)$$

where l is the iteration number of the search process over the whole ensemble of N samples, $-\eta \nabla_{\Theta} E(l)$ is the steepest gradient term and $\alpha \Delta\Theta(l-1)$ is the momentum term, added to avoid local minima.

In our search algorithm, filter weights vector is initialized to $[1 \ 0 \ 0 \ \dots \ 0]^T$ which is the weights vector of a zero order hold filter as a simple and general predictor. Then the weights are tuned off-line by (10) to (16) and then used for missing data handling in our real-time application. If for any reason, dynamic characteristics of the sensor signal change, filter weights can be adaptively tuned by using the equations (10) to (16) in real-time i.e. the summations in (11) are ignored and l in (16) is replaced with the signal sampling time k .

4 Experimental Results

We tuned the weights of a FIR filter with our proposed method and used this filter for missing data handling by prediction of up to four missing samples of sensor data in the HMI (Human-Machine Interface) section of a real time and memory critical application. Sensory data were measured and recorded in two different ensembles, one for off-line weight tuning or training and the other for validation purposes. For comparison purposes, two other methods were also applied: first order hold filtering method as a simpler approach and a neural predictor as a more sophisticated approach.

A first order hold filter assumes that the missing sample follows the same slope of the signal as in the two recent data samples. More precisely, a first order hold filter as L -step ahead predictor, estimates the missing sample by the following equation:

$$y_L(k) = (L+1)y(k-L) - L y(k-L-1) \quad (17)$$

A TDNN (Time Delay Neural Network) was applied for prediction as a neural predictor. Because of their high computational overhead, neural networks are rarely used as nonlinear predictive filters in time critical applications. Instead, they are extensively applied to off-line prediction applications such as time series analysis and prediction [7,8] where computational overhead is not an important issue. In spite of complexity, neural networks have shown to be powerful in learning and realisation of nonlinear models which are difficult to be modeled by other techniques. We used a neural network to learn the nonlinear dynamics of the sensor signal and applied it for prediction and compared the performance of this neural predictor with our proposed filter.

Architecture of the TDNN is shown in Figure 1. As a p -th order nonlinear filter, the neural network block is a multi-layered perceptron with an input layer of p neurons, one hidden layer of $n_h=20$ neurons and a single output neuron which gives the estimated value for missing sensory data sample. The number of delays in each of the delay boxes in this architecture depends on the number of steps ahead in prediction. For instance, if the signal itself and estimated signal are shown by y and \hat{y} , respectively, then in one-step ahead prediction, the inputs of the neural network will consist of $\{y(k-1), \dots, y(k-p)\}$, while in four-step ahead prediction the inputs will include $\{\hat{y}(k-1), \hat{y}(k-2), \hat{y}(k-3), y(k-4), \dots, y(k-p)\}$.

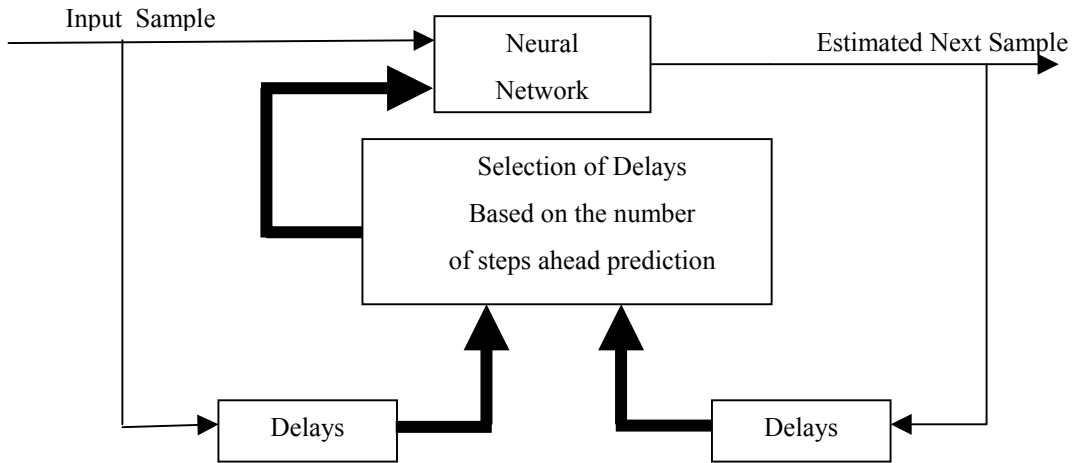


Figure 1: Architecture of the TDNN, used for missing data handling

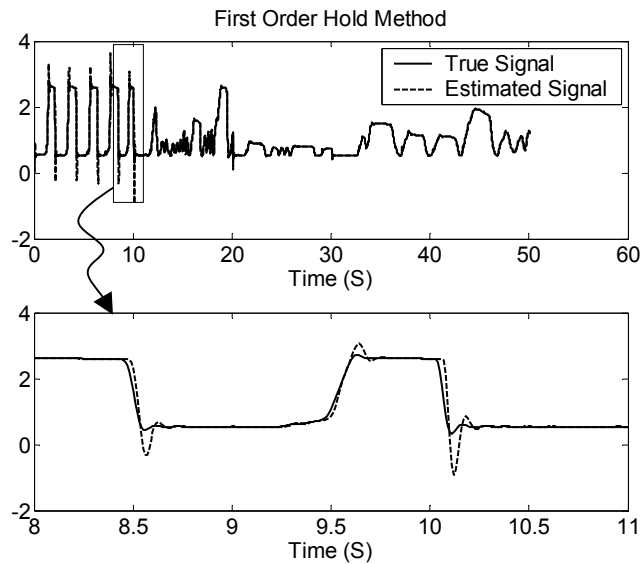


Figure 2: (Upper) Validation data samples and their estimated values by first order hold method (Lower) The two signals zoomed in the time interval of 8 to 11 seconds

Figure 2 shows validation data samples and their four-step ahead predicted values by first order hold method. In order to reveal the prediction error as the difference between real and predicted data, the two signals in the time interval of 8-11 seconds has also been zoomed and re-plotted in Figure 2. Same plots for the case of using a neural predictor are also depicted in Figure 3. In this case, a $p=5^{\text{th}}$ order nonlinear filter, realised by a TDNN (as shown in Figure 1), was first trained by using the ensemble of training samples and then applied to prediction of validation data samples. We predicted the validation data samples by $p=5^{\text{th}}$ order FIR filter whose weights were tuned by our proposed method, using the training data samples. Figure 4 shows the results of four-step ahead prediction by our method.

In terms of the distance between true and estimated signals, Figure 2, 3 and Figure 4 show that prediction error of our proposed filter is lower than the first order hold method and comparable to the neural predictor. In order to achieve quantitative comparison results, the following prediction MSE (Mean Square Error) is defined:

$$E = \sqrt{\frac{1}{4} \sum_{i=1}^4 \left\{ \frac{\sum_{k=1}^N (y(k) - y_i(k))^2}{N} \right\}} \quad (18)$$

where $y(k)$ is the k -th data sample and $y_i(k)$ is its i -step ahead predicted value. This MSE was calculated for each of the three prediction methods. Furthermore, execution times were also recorded for the three methods, implemented by MATLAB software on a Pentium IV 1.6 GHz platform.

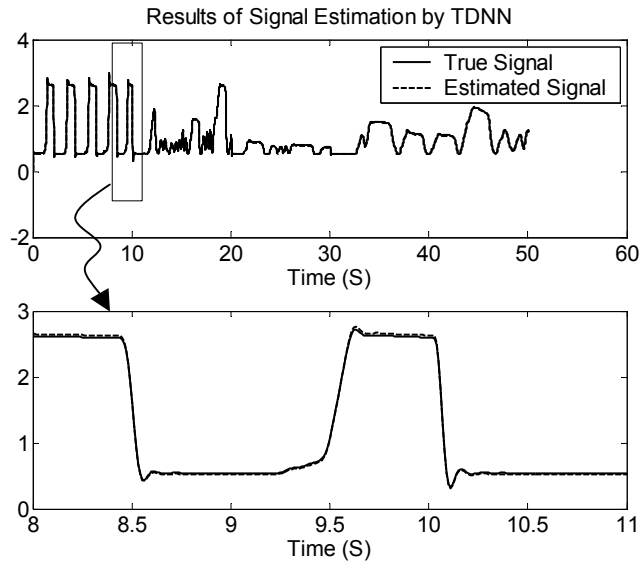


Figure 3: (Upper) Validation data samples and their estimated values by TDNN (Lower) The two signals zoomed in the time interval of 8 to 11 seconds

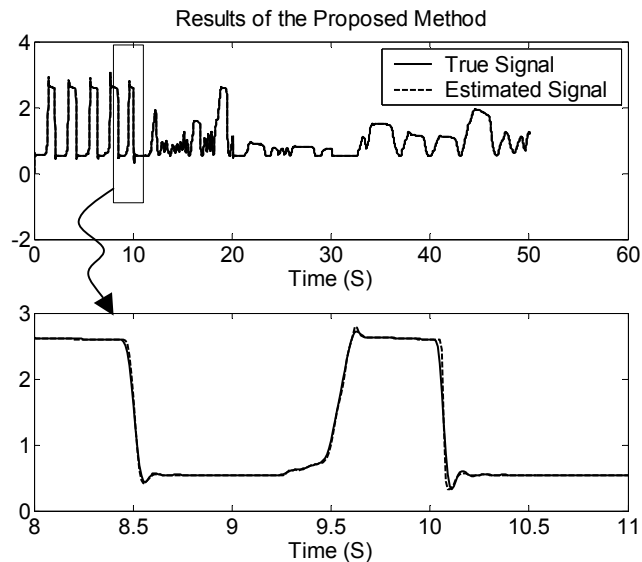


Figure 4: (Upper) Validation data samples and their estimated values by a FIR filter whose weights were tuned by our new method, based on steepest gradient method (Lower) The two signals zoomed in the time interval of 8 to 11 seconds

Table 1 summarises the results of MSE, execution times and the number of weights required to be memorised in each method. These results show that while prediction error of our proposed FIR filter is five times lower than the first order hold filter, it is almost equal to

MSE in the case of prediction by the neural filter. The number of weights to be memorised in FIR filter and the execution time, however, are almost 30 times lower than the number of weights of neural synapses in the neural filter and its execution time.

Table 1: Summary of missing data handling results by three different prediction methods: first order hold, TDNN and our proposed adaptive method

Prediction Method	Prediction MSE	Execution Time (mS)	Number of Weights
1 st Order Hold	0.1845	1.1	2
TDNN	0.0340	98.9	147
Proposed Adaptive Method	0.0369	3.7	5

5 Conclusion

We introduced a new multi-step ahead predictive filter for missing data handling. This filter is a simple FIR filter. In contrast to well-known FIR filters, in our proposed filter, only one set of weights are tuned (by steepest gradient descent method), memorised and applied for different numbers of steps ahead prediction. Hence, this filter is appropriate for time and memory critical applications.

In our experiments, we performed missing data handling by prediction of up to four missing samples of sensor data in HMI section of a real time and memory critical application. Two ensembles of sensory data were measured and recorded: one for off-line weight tuning or training and the other for validation purposes. In addition to our proposed predictive filter, we applied both first order hold method and a nonlinear predictive filter realised by a time delay neural network (TDNN). The results show that prediction error of the FIR filter is five times better than a first order hold filter. Also, it is almost as accurate as the neural predictive filter, but both the number of weights to be memorised for our FIR filter and its execution time are almost 30 times lower compared to the neural filter. Hence our proposed predictive filtering and its weight tuning algorithm are appropriate for time and memory critical applications.

References

- [1] P. Heinonen & Y. Neuvo, FIR-median hybrid filters with predictive FIR structures, *IEEE Trans. Acoust. Speech Signal Process.*, Vol. 36, Issue 6, 892-899, 1988.
- [2] P.T. Harju, T.I. Laakso & S.J. Ovaska, Applying IIR predictors on Rayleigh fading signals, *Signal Processing*, Vol. 48, No. 1, 91-96, 1996.
- [3] P. Händel, Predictive digital filtering of sinusoidal signals, *IEEE Trans. Signal Proc.*, Vol. 46, No. 2, 364-374, 1998.
- [4] K. Koppinen, Efficient filter structures for linearly predictable responses, *Proc. Third IEEE Nordic Signal Process. Symp. (NORSIG2000)*, Kolmarden, Sweden, 13-15 June, 2000.

- [5] K. Koppinen & J. Astola, Generalized IIR polynomial predictive filters, *Proc. Eighth European Signal Process. Conf. (EUSIPCO-2000)*, Tampere, Finland, 5-8 September, 2000.
- [6] J.M.A. Tanskanen, S.J. Ovaska & O. Vainio, Adaptive general parameter extension to FIR predictors, *Proc. 8th European Signal Process. Conf. (EUSIPCO-2000)*, Tampere, Finland, 1005-1008, 5-8 Sept., 2000.
- [7] G. Dorfner, Neural networks for time series processing, *Neural Network World*, Vol. 6, No. 4, 447-468, 1996.
- [8] R.J. Frank, N. Davey & S.P. Hunt, Time series prediction and neural networks, *Journal of Intelligent and Robotic Systems*, Vol. 31, Issues 1-3, 91-103, 2001.