

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 Compensation for Safety-Critical Components in a Drive-by-Wire System

Reza Hoseinnezhad and Alireza Bab-Hadiashar, *Senior Member, IEEE*

Abstract—In this article, a new multistep ahead predictive filtering scheme is introduced. The proposed technique is essential for proper operation of safety-critical components in a drive-by-wire car. Because limited computational time and memory are available in drive-by-wire systems, the main advantage of our scheme is that it only requires one set of finite impulse response (FIR) filter weights to be tuned while it can be used for different numbers of steps ahead predictions. To verify and compare the proposed filter, our model and four competing methods were applied to predict up to four missing samples of displacement sensor data from a brake pedal in a brake-by-wire system. Experimental results show that prediction performance of our proposed FIR filter is higher than, or at least comparable to, other filters with the same memory requirements and the computational overhead of data missing handling by our proposed method is considerably lower than other comparable methods. Hence, the proposed filter has a superior performance in missing data compensation for drive-by-wire systems.

Index Terms—Braking, brake-by-wire, data handling, drive-by-wire, prediction methods.

I. INTRODUCTION

IN the automotive industry, by-wire technology replaces the traditional mechanical and hydraulic systems with electronic control systems, electromechanical actuators, and human-machine interfaces, such as pedal and steering feel emulators. Hence, the traditional components, such as the steering column, intermediate shafts, pumps, hoses, fluids, belts, brake boosters, and master cylinders, are eliminated from the vehicle. The management system of a by-wire car, in schematic form, is shown in Fig. 1. The system is mainly comprised of five types of elements:

- 1) Processors, including an electronic control unit (ECU) and other local processors;
- 2) Memory (mainly integrated into the ECU);
- 3) Sensors;
- 4) Actuators;
- 5) Communication network.

This article focuses on the system sensors, the significance of their data, and the issue of missing sensory data samples in the system. In addition to discussing different methods to resolve this issue, this article also considers computational complexity and memory usage of each method.

Manuscript received November 10, 2004; revised January 11, 2004. This work was supported in part by Research Centre for Advanced By-Wire Technology (RABiT) and Pacifica Group Technologies Pty Ltd. The review of this paper was coordinated by Dr. M. A. Masrur.

R. Hoseinnezhad and A. Bab-Hadiashar are with the Faculty of Engineering and Industrial Sciences, Swinburne University of Technology, Hawthorn, Victoria 3122, Australia (e-mail: rhoseinnezhad@swin.edu.au; abab-hadiashar@swin.edu.au).

A. Sensors

In a by-wire car, some sensors are safety-critical components, and their failure will disrupt the vehicle function and endanger human lives. Two examples are the brake pedal sensors and the wheel speed sensors. The ECU must always be informed of the driver's intentions to brake or to stop the vehicle. Therefore, missing the pedal sensor data is a serious problem for functionality of the vehicle control system. Wheel speed data are also vital in a brake-by-wire system to avoid skidding.

The design of a by-wire car should provide safeguards against missing some of the data samples provided by the safety-critical sensors. Popular solutions are to provide redundant sensors [1] and to apply a fail-safe mechanism [2]. In addition to a complete sensor loss, the ECU may also suffer an intermittent (temporary) data loss. For example, sensor data can sometimes fail to reach the ECU. This may happen due to a temporary problem with the sensor itself or with the data transmission path. It may also result from an instantaneous short circuit or disconnection, a communication network fault, or a sudden increase in noise. In such cases, for a safe operation, the system has to be compensated for missing data samples.

Our approach here is to continuously predict the data samples and replace the missing ones with their previously predicted values. In most cases, where multiple consecutive samples are missing, *multistep ahead predictive filters* are needed. The associated predictive filtering computation is one of the many tasks that should be performed by the ECU in real time.

B. Memory Usage and Computational Complexity

The system memory is often divided into two main parts. One part is to store the program code and the constant parameters of the vehicle and its control system (symbolically shown as a_i in Fig. 1). Another part keeps track of the current and past values of the system states and control variables (shown as y_i in Fig. 1). To achieve a competitive cost, the design should not require a lot of memory. Particularly, there are limitations in using the dynamic part of memory (usually implemented using high-speed RAM) with fast read-write capability. In a brake-by-wire system, real-time signal processing algorithms need dynamic memory with fast read-write response, which is costly. Memory-efficient algorithms will help producers achieve a competitive cost in mass production and are highly preferred in industrial applications. The design of a by-wire car is a time- and memory-critical application, where the amount of memory required by different algorithms and their computational complexity are important matters, and simple and stable algorithms are preferred because they remove the need for fast and expensive hardware and memory.

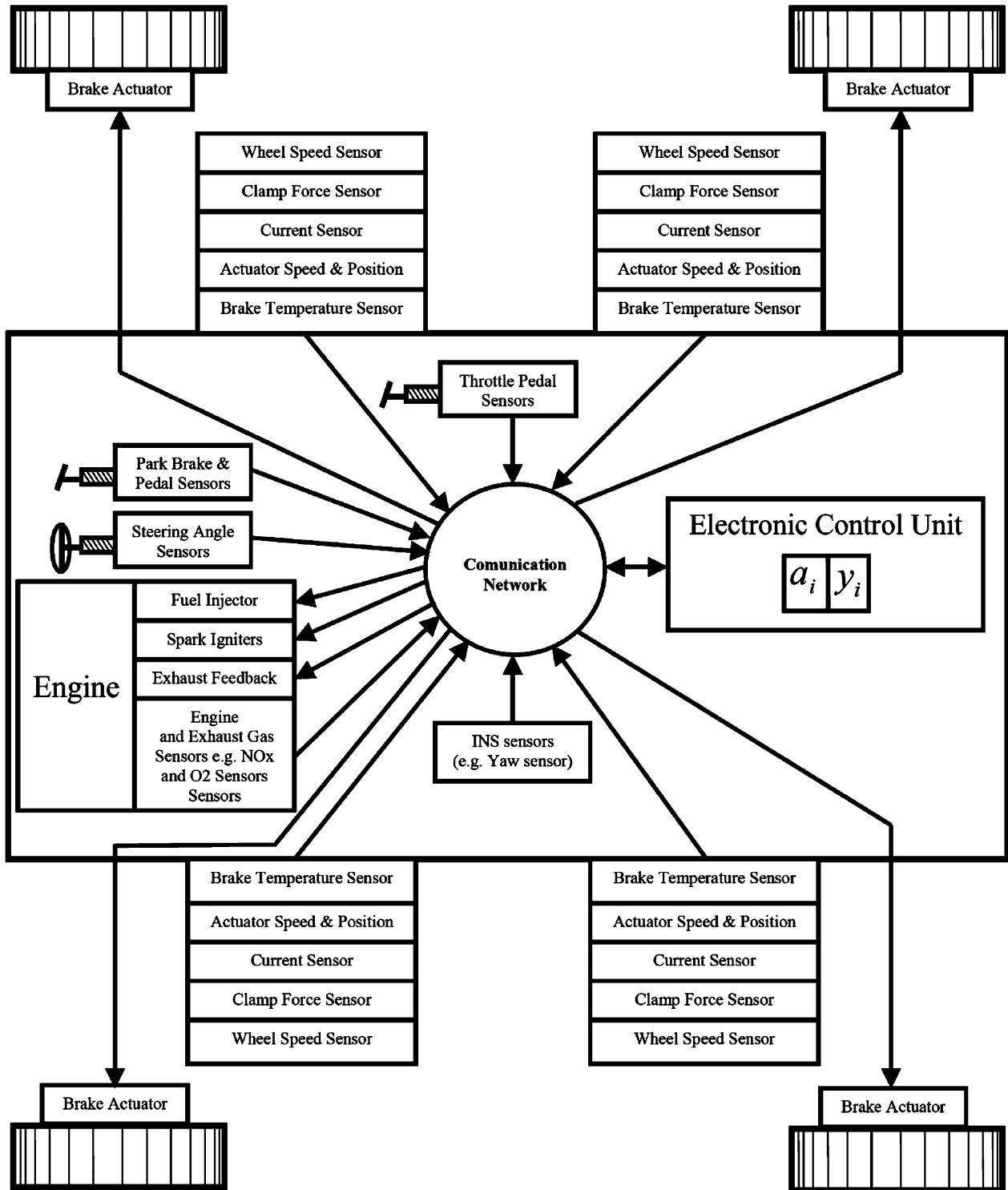


Fig. 1. The management system of a drive-by-wire system in schematic form.

In this article, we introduce a new multistep ahead predictive filter as a compensation scheme for missing data samples of a safety-critical component in the time- and memory-critical system of a by-wire car. An important feature of the proposed method is that it uses only one set of weights for different steps ahead in prediction. Hence, less memory is required to save filter coefficients and data samples. The weight-tuning al-

gorithm has been formulated based on the steepest gradient method.

In Section II, a brief literature review for predictive filtering methods in control and instrumentation is provided. In Section III, we present our proposed method of missing data compensation for drive-by-wire systems. The new multistep ahead predictive filtering scheme is also explained in this section. Tuning

of the filter weights is discussed in Section IV, and experimental results are presented in Section V. Section VI concludes the discussion.

II. PREDICTIVE FILTERING METHODS: A LITERATURE REVIEW

A variety of approaches have been used to design predictive filters in different applications. In practice, zero- and first-order hold, finite impulse response (FIR), and infinite impulse response (IIR) filters are the simplest choices for predictive filtering in control instrumentation. Some authors have modified these traditional methods and improved their performance for specific applications. Heinonen *et al.* applied FIR predictive filters combined with median filters to measure systolic blood pressure [3]. Harju *et al.* introduced a method to design predictive IIR filters via feedback extension of FIR predictive filters [4]–[6]. 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 a lower computational overhead. Handel proposed a multistep ahead predictive filter for narrow-band waveforms with multiple distinct spectral peaks [7]. He obtained the coefficients of an optimal FIR predictor by minimization of the noise gain of the filter and studied the feedback extension of this predictor. Koppinen *et al.* proposed a filter that can generate any window of a linearly predictable signal, including the polynomial, weighted exponential, and Hamming-window responses [8], [9].

Polynomial predictive filters have been widely used in control instrumentation [10]. They are designed for prediction of *polynomial-like* signals that can be accurately approximated as a piecewise polynomial function of sampling time. Smooth signals can be modeled as piecewise low-degree polynomials with sufficient accuracy within narrow time windows. Polynomial filters are also designed to predict narrow-band signals with variable sampling rate and to remove the undesirable effects of varying sampling rate (e.g., aliasing or imaging frequency components) [10], [11].

Nonlinear predictive filters have also been applied to systems with nonlinear dynamics. Crassidis *et al.* used the least mean square method to tune the parameters of a nonlinear predictive filter for attitude estimation [13]. Ferrah *et al.* applied an adaptive nonlinear predictive filter as a dynamic speed estimator for induction motor drives [14]. The parameters of this filter are adaptively tuned by recursive least squares technique.

Intelligent and soft computation approaches have been applied to design and implement predictive filters for applications with high levels of complexity and uncertainty. Fuzzy predictive filters are the filters whose linear weights or nonlinear parameters are tuned by fuzzy IF-THEN rules. They have been used in different model predictive control applications [15], [17], [18]. Ovaska *et al.* [18] and Schoen [19] used genetic algorithms to aid the design of predictive filters.

Due to their computational load and complexity, nonlinear and soft computation approaches are not appropriate choices for a drive-by-wire system. Similarly, the polynomial predictive filters are also not suitable for missing data compensation scheme mainly because the signals from safety critical com-

ponents in a drive-by-wire system comprise occasional sharp jumps and fast variations. In fact, from safety point of view such sharp changes are important as they carry critical information. Thus, those portions of the signal must be compensated for when they are missing.

By the previous discussion, we would argue that FIR filters are the most suitable method for missing data handling in a drive-by-wire application. Because there are strict limitations on the requirement of memory and computation time, regular adaptation of filter weights would be infeasible in practice, and consequently, using a filter with fixed but optimally tuned weights is desired.

III. PROPOSED MULTISTEP AHEAD PREDICTIVE FILTERING SCHEME

The main motivation behind the proposed approach is that if the traditional predictive filters 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). In that case, different sets of filter weights have to be saved in the system memory, which in turn increases the memory usage. Moreover, in that case, some of the system processing time has to be spent on selection of the appropriate filter with required number of steps ahead in prediction. This is also undesirable due to time limitations.

Assume in a by-wire system a particular safety critical component or sensor is compensated for a maximum number of L consecutive missing data samples. If more than L consecutive data samples are missing, then the by-wire system will switch to another policy in which the relevant control commands are determined based on other available sensory data. We assume the sensory signals do not follow sharp jumps, thus this switching policy will not affect the stability.

Depending on the number of recently missed samples, compensation of the sensor for its new missing sample may require a one-step, two-step, \dots or L -step ahead prediction. The predicted value of each missing sample is formulated as a linear combination of its 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 memory and computational time constraints, it is desired to tune the same a_i values for prediction in all cases.

In the case of one-step ahead prediction, the first missing sample at time k is replaced with the value $y_1(k)$ given by (1), in which y' is replaced with y as shown:

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

where P is the order of the predictive filter. The transfer function of this filter is given by

$$H_{\text{FIR}}^1(z^{-1}) = Y_1(z^{-1})/Y(z^{-1}) = \sum_{i=1}^P a_i z^{-i}. \quad (3)$$

For two-step ahead prediction, a second missing sample at time k is replaced with $y_2(k)$. It is the same linear combination of the recently predicted data sample and the recent $P - 1$ available data samples, as shown:

$$y_2(k) = a_1 y_1(k-1) + \sum_{i=2}^P a_i y(k-i). \quad (4)$$

Replacing $y_1(k-1)$ from (2) into (4) results in the following equation:

$$y_2(k) = a_1 a_P y(k-P-1) + \sum_{i=2}^P (a_1 a_{i-1} + a_i) y(k-i). \quad (5)$$

The transfer function of this two-step ahead predictive filter is

$$H_{\text{FIR}}^2(z^{-1}) = Y_2(z^{-1})/Y(z^{-1}) = \sum_{i=2}^P (a_1 a_{i-1} + a_i) z^{-i} + a_1 a_P z^{-P-1}. \quad (6)$$

Similarly, the j th missing data sample is replaced with $y_j(k)$ given by the following equation:

$$y_j(k) = \sum_{i=0}^{j-1} a_i y_{j-i}(k-i) + \sum_{i=j}^P a_i y(k-i) \quad (7)$$

where $2 \leq j \leq L$ and $a_0 = 0$. We assume $P \geq L$, which is reasonable because the maximum tolerable number of missing data samples, L , is usually small in real-time control systems, particularly in the drive-by-wire system. By iterative substitution of previously predicted y_{j-i} values in (7), the following general formula is derived:

$$y_j(k) = \sum_{i=j}^{P+j-1} A_i^j y(k-i); \quad 1 \leq j \leq L \quad (8)$$

where the A_i^j multipliers can be recursively computed by the following equations:

$$A_i^1 = a_i; \quad 1 \leq i \leq P \quad (9)$$

$$A_i^j = \begin{cases} 0; & i < j \\ a_i + \sum_{l=1}^{j-1} a_l A_{i-l}^{j-l}; & j \leq i \leq P \\ \sum_{l=1}^{j-1} a_l A_{i-l}^{j-l}; & P < i < j + P \\ 0; & i \geq j + P \end{cases} \quad (10)$$

where $2 \leq j \leq L \leq P$. Equation (8) shows that the proposed predictive filter is an FIR filter with the following transfer function:

$$H_{\text{FIR}}^j(z^{-1}) = Y_j(z^{-1})/Y(z^{-1}) = \sum_{i=j}^{P+j-1} A_i^j z^{-i} \quad (11)$$

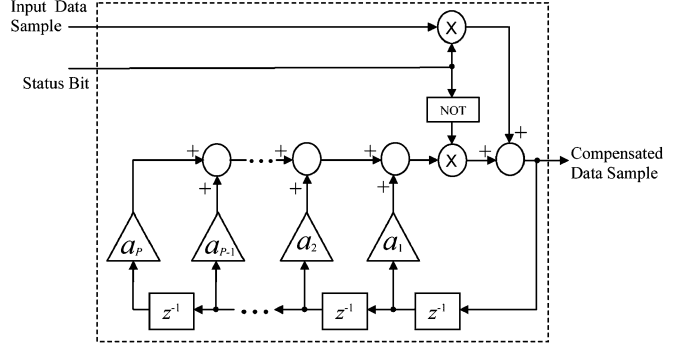


Fig. 2. Block diagram of our proposed multistep ahead predictive filtering as a compensation scheme for missing data samples.

where $1 \leq j \leq L$.

Fig. 2 shows the block diagram of the multistep ahead predictive filtering scheme. For each safety critical component in this system, a status bit accompanies the data stream. The status bit is reset if, for any reason, the data sample is invalid and considered missing. The missing sample is then replaced by a linear combination of previous valid or missing (but predicted and replaced) samples. Indeed, (7) is implemented in this diagram. It is important to note that, as shown in Fig. 2, our proposed scheme only requires to store the P constant weights of the filter and the recent P values of the compensated module's output signal, regardless of how many consecutive samples are missing. Also, in terms of computation, it only comprises two multiplications and $P - 1$ summations.

IV. FILTER WEIGHT-TUNING ALGORITHM

The problem addressed in this section is how to tune the a_i weights in such a way that the resulting filter can optimally predict missing data samples of a signal by (7). Our weight-tuning algorithm is a data-driven technique. We try to search for optimal a_i weights that minimise the following error function:

$$E = \frac{1}{2} \sum_{j=1}^L \left\{ C_j \sum_{k=1}^N [y(k) - y_j(k)]^2 \right\} \quad (12)$$

where N is the total number of samples used for offline tuning and the nonnegative C_j constants are selected based on the given priority of the performance of one- to L -step ahead predictions. This prioritization depends on the application.

To solve this nonlinear least square problem, different data-driven algorithms can be used (e.g., steepest gradient search method, simulated annealing, evolutionary and genetic algorithms). Our search algorithm for weight tuning is formulated based on the steepest gradient search. In this method, the weight vector $\Theta = [a_1 \ a_2 \ \dots \ a_p]^T$ is iteratively moved toward the reverse direction of the gradient vector of the error function (with respect to the weight vector), which is given by the following equation:

$$\nabla_{\Theta} E = \left[\frac{\partial E}{\partial a_1} \quad \frac{\partial E}{\partial a_2} \quad \dots \quad \frac{\partial E}{\partial a_p} \right]^T. \quad (13)$$

The partial derivatives of the error functions can be calculated from (12) as given:

$$\frac{\partial E}{\partial a_j} = \sum_{i=1}^L \left\{ C_i \sum_{k=1}^N \frac{\partial y_i(k)}{\partial a_j} (y_i(k) - y(k)) \right\}. \quad (14)$$

Equations (5) and (8)–(10) show that direct calculation of $\partial y_i(k)/\partial a_j$ is complicated specially for $i > 1$. However, an indirect iterative calculation based on (2), (4), and (7) is easier to formulate. Partial differentiation of (2) with respect to a_j simply results in the following equation:

$$\frac{\partial y_1(k)}{\partial a_j} = y(k - j). \quad (15)$$

Partial differentiation of (4) with respect to a_j and replacing $\partial y_1/\partial a_j$ from (15) results in

$$\frac{\partial y_2(k)}{\partial a_j} = \begin{cases} y_1(k - 1) + a_1 y(k - 2); & j = 1 \\ y(k - j) + a_1 y(k - j - 1); & j > 1 \end{cases} \quad (16)$$

By a similar approach, the following general formula for $\partial y_i(k)/\partial a_j$ can be achieved: (Please see the equation at the bottom of the page.) where $1 \leq j \leq P$, $2 \leq i \leq L$ and the multipliers B_l can be recursively calculated by the following equations:

$$B_l = \begin{cases} 1; & l = 0 \\ \sum_{i=1}^{\min(l,P)} a_i B_{l-i}; & l > 0 \end{cases} \quad (18)$$

The gradient vector in (13) can be calculated by (14), (17), and (18) and applied to weight adaptation based on the rule of steepest gradient descent as follows:

$$\Delta\Theta(l) = -\eta \nabla_{\Theta} E(l) + \alpha \Delta\Theta(l - 1) \quad (19)$$

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

The weight vector is first initialized to a zero-order hold filter $[1 \ 0 \ 0 \ \dots \ 0]^T$. This initialization provides a suitable starting point for the search operation in the P -dimensional space. More precisely, it is practically assumed for most of the times during vehicle motion that the sensory signal does not change rapidly and that sharp peaks and jumps do not happen frequently. This is a valid assumption for all safety-critical sensors such as the brake pedal and wheel speed sensors. With this assumption, the total error calculated by (12) will be small if $y_j(k)$ is the output of a zero hold filter. Hence, the zero-order hold initialization provides a starting point in the search space that is close to the global minimum point. Besides, addition of the momentum term helps the search process avoid being trapped by local minima. Appropriate choices for the two constants α and η by trial and error will also decrease the likelihood of the process to be trapped in local minima.

The weight vector is tuned offline by (12)–(19) and then used for missing data compensation in the real-time application. If for

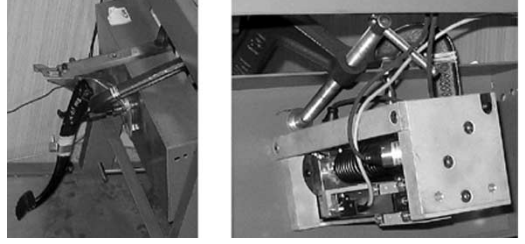


Fig. 3. Brake pedal and its sensors in our test rig.

any reason dynamic characteristics of the sensor signal change, then filter weights can be adaptively tuned by (12)–(19) in real time. In the adaptive scheme, the summations in (12) and (14) are ignored and l in (19) is replaced with the signal sampling time k .

V. EXPERIMENTAL RESULTS

A. Test Rig Setup

As stated, the brake pedal sensors in a brake-by-wire system are safety-critical components. The driver's brake demand must always be available to the by-wire system to generate appropriate brake commands that are sent to brake actuators. If the brake demand is not received, then the by-wire management will be faced with an extremely dangerous situation.

To test the previous scenario, we have prepared a test rig comprising a brake pedal equipped with one displacement and two force sensors. Fig. 3 shows photographs of the pedal and its sensors.

During our experiments, we have measured and collected data samples from the displacement sensor of the pedal. The reliability of prediction and missing data handling by our proposed method heavily depends on proper tuning of the filter weights. If the filter weights are tuned based on a particular set of test data, then the reliability of the predictor for different circumstances will be questionable. Thus, we have repeated the experiment several times and captured almost all possible pedal movements in our data collection, including standstill pedal, frequent soft pushes, frequent hard pushes, and sudden continuous hard brake.

B. Other Compared Methods

To evaluate our proposed technique, we have implemented and compared four other methods, including a first-order hold filtering method as a simpler approach, the classic and linear smoothed Newton filters that are polynomial predictive filters, and a neural predictor as a highly sophisticated approach. The required amount of memory for each method is evaluated in terms of *memory units*. Each memory unit means a number of bytes or words in the system memory, required to save signals, parameters, or weights in fixed-point, floating point, or double

$$\frac{\partial y_i(k)}{\partial a_j} = \begin{cases} \sum_{l=0}^{i-j-1} B_l y_{i-j-l}(k - l - j) + \sum_{l=i-j}^{i-1} B_l y(k - l - j); & j < i \\ \sum_{l=0}^{i-1} B_l y(k - l - j); & j \geq i \end{cases} \quad (17)$$

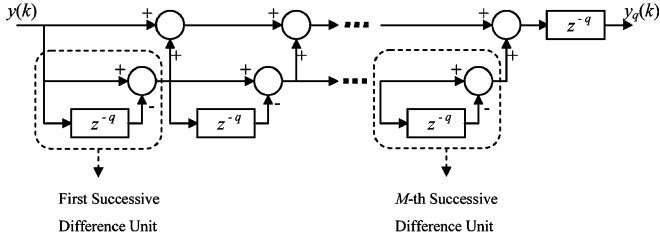


Fig. 4. Block diagram of Newton polynomial filter for q -step ahead prediction of M -th-order polynomial-like signals.

precision format, depending on the realization of the technique in the system. Before presenting the comparative performance results, we briefly explain these filters.

1) *First-Order Hold*: A first-order hold filter assumes the missing sample follows the same slope of the signal as in the two most recent available data samples. As a j -step ahead predictor, assuming j consecutive samples $y(k), y(k-1), \dots, y(k-j+1)$ are missing, a first-order hold filter estimates $y(k)$, in terms of the recent available samples by the following equation:

$$y_j(k) = (j+1)y(k-j) - jy(k-j-1). \quad (20)$$

The transfer function of this filter is given by

$$H_{\text{FOH}}^j(z) = Y_j(z^{-1})/Y(z^{-1}) = z^{-j}(1 + j(1 - z^{-1})). \quad (21)$$

This filter needs two memory units to save the two constant multipliers and two memory units to save the data samples $y(k-j)$ and $y(k-j-1)$. For $L=4$, eight memory units for the multipliers and two memory units for saving data samples are required by this filter.

2) *Newton Polynomial Filters*: A j -step ahead (classic) Newton predictor is defined by the following transfer function [15]:

$$\begin{aligned} H_{M,\text{Newton}}^j(z^{-1}) &= Y_j(z^{-1})/Y(z^{-1}) \\ &= z^{-j} \sum_{i=0}^M (1 - z^{-j})^i \end{aligned} \quad (22)$$

where M is the polynomial degree of the polynomial-like signals under prediction. Fig. 4 shows the block diagram of a Newton predictive filter. This filter needs M units of memory to implement the successive difference units $(1 - z^{-j})^i$ and one memory unit to implement the z^{-j} delay. For $L=4$, the total number of required memory units is $4(1 + M)$.

A linear smoothed newton (LSN) predictive filter has the following transfer function [13]:

$$\begin{aligned} H_{M',\text{LSN}}^j(z^{-1}) &= Y_j(z^{-1})/Y(z^{-1}) \\ &= z^{-j} \sum_{i=0}^{M'-1} (1 - z^{-j})^i + S(z^{-1})(1 - z^{-j})^{M'} \end{aligned} \quad (23)$$

where M' plays the same role as M for Newton predictive filter and $S(z^{-1})$ is the transfer function of a low-pass filter (e.g., a moving averaging filter of length $N=8$). The block diagram of

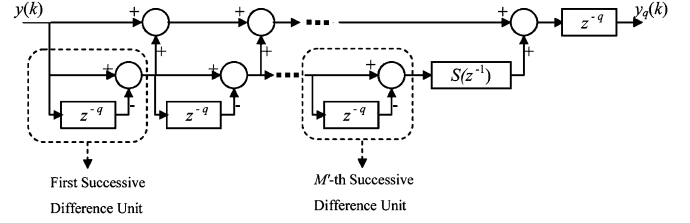


Fig. 5. Block diagram of the LSN polynomial filter for q -step ahead prediction of M' -th-order polynomial-like signals.

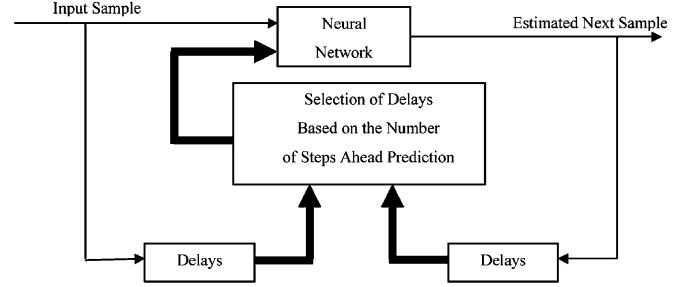


Fig. 6. Structure of the TDNN, used for missing data compensation.

this filter is shown in Fig. 5. The total number of memory units required by this filter is $4(1 + M' + N) = 4(9 + M')$.

3) *Neural Predictive Filter*: A time delay neural network (TDNN) is a neural network in which some of its inputs are the delayed version of its outputs or other inputs. TDNNs have been widely used for identification and recognition purposes in different applications [20], [22], [23]. 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 offline prediction applications such as analysis and prediction of time series [24], [25], where computational overhead is not an important issue. We used a neural network to learn the nonlinear dynamics of the sensor signal and applied it for prediction. Architecture of the TDNN is shown in Fig. 6.

As a m -th-order nonlinear filter, the neural network block is a multilayered perceptron with an input layer of m neurons and a bias neuron, one hidden layer of n_h neurons and a bias neuron, and a single output neuron that gives the estimated value for the missing sensory data sample. The number of delays in each delay box in this architecture depends on the number of steps ahead in prediction. For instance, if the signal itself and the estimated signal are shown by y and \hat{y} , respectively, then in one-step ahead prediction the inputs of the neural network are $\{y(k-1), \dots, y(k-m)\}$, and in four-step ahead prediction they are $\{\hat{y}(k-1), \hat{y}(k-2), y(k-3), y(k-4), \dots, y(k-m)\}$. The total number of memory units required to memorise the trained weights of this neural network is $(m+2)n_h + 1$. It also needs m units of memory to save the recent m samples of data $\{\hat{y}(k-i)\}$ or previously estimated data $\{\hat{y}(k-i)\}$. Thus, the total number of required memory units is $(m+2)n_h + m + 1$.

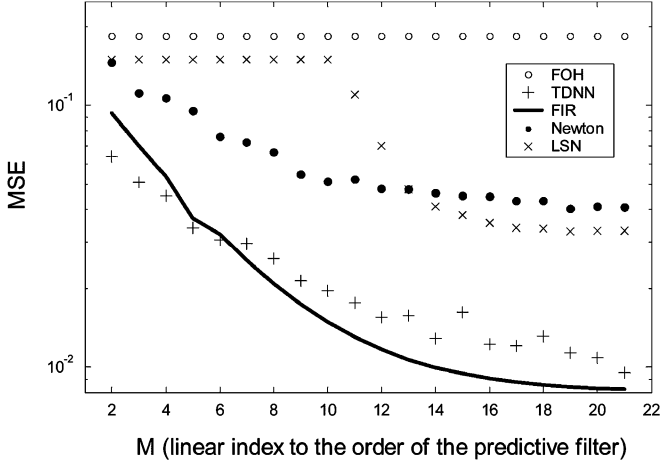


Fig. 7. Prediction MSE of the five methods in one-step to four-step ahead prediction of the ensemble of validation data samples: Corresponding to each M value, the orders of the filters are chosen in such a way that they required the same number of memory units.

C. Comparative Results

We have computed the weights of our proposed filter and used this filter for missing data compensation by prediction of up to four missing samples in our experiments (i.e., $L = 4$). Sensory data were measured and recorded in two different ensembles, one for offline weight tuning (or training) and the other for validation purposes. The following prediction mean square error (MSE) is defined:

$$E = \sqrt{\frac{1}{LN} \sum_{i=1}^L \sum_{k=1}^N (y(k) - y_i(k))^2} \quad (24)$$

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 five predictive filtering methods. The results are shown in Fig. 7. As stated previously, our proposed multistep ahead predictor only requires $2P$ memory units.

To achieve comparable results, we used filters with the same number of required memory units, except the first-order hold filter, which only needs a fixed number of memory units. For our proposed FIR filter and the neural filter, we used the training data to tune the weights offline, prior to applying the filters to the validation data. Each M value in Fig. 7 corresponds to a Newton polynomial predictor with $4(1 + M)$ memory units. The corresponding LSN filter has the same number of memory units (i.e., M' in (23) is replaced with $M - 8$, except for $1 < M < 10$, where it is replaced with 2 to be meaningful). Besides, corresponding to each value of M , a TDNN and a P th-order multistep predictive FIR filter with the same number of memory units were trained and then applied to predict validation data (i.e., $P = 2(1 + M)$ and $(m + 2)n_h + m = 4M + 3$).

The prediction MSEs of various filters with different orders, depicted in Fig. 7, show that the prediction performance of the proposed FIR filter is comparable to the neural predictor and is higher than other filters. Even for higher orders, it predicts more accurately than the TDNN. That is because the TDNN overfits

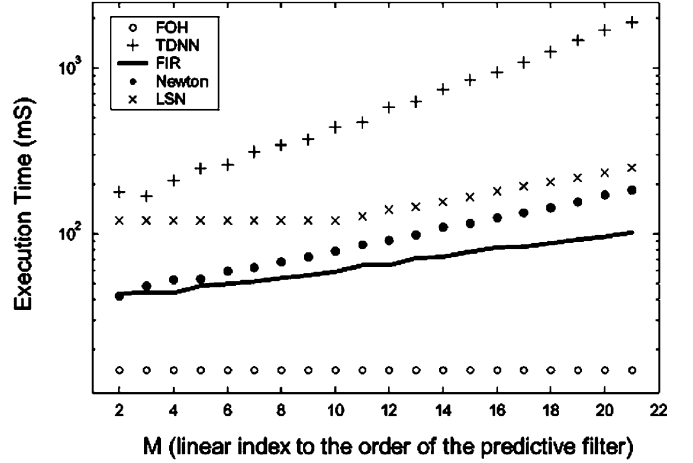


Fig. 8. Execution times of the five methods for one-step to four-step ahead prediction of the ensemble of validation data samples: Corresponding to each M value, the orders of the filters are chosen in such a way that they require the same number of memory units.

TABLE I
AVERAGED PREDICTION MSE VALUES AND EXECUTION TIMES

Prediction Method	Average Prediction MSE	Average Execution Time (mS)
1 st Order Hold Method for Prediction	0.1845	15.016
TDNN – The Neural Predictive Filter	0.0234	706.338
Newton Polynomial Predictive Filter	0.0637	97.669
Linear Smoothed Newton (LSN) Predictive Filter	0.0930	154.566
Proposed FIR Multi-Step Ahead Predictive Filter	0.0235	67.202

to the training data, and its generalization power to validation data does not increase when the order of the filter increases.

Furthermore, execution times were also measured for the five methods, implemented using MATLAB software on a Pentium IV 1.6 GHz processor. Fig. 8 shows the execution times for prediction of all validation data samples. Similar to Fig. 7, the methods with the same number of required memory units are compared with each other. The results show that the computational overhead of our proposed FIR filtering method is less than other methods, except for the simple first-order hold whose prediction error is very high.

For the sake of quantitative comparison, we calculated the average of MSE values and execution times over different orders. The results are summarized in Table I. The results confirm that although our proposed FIR multistep ahead predictive filter estimates missing data samples as accurately as a sophisticated neural predictive filter, and more accurately than other methods, its computational overhead is lower than other filters.

VI. CONCLUSION

We have introduced a new multistep ahead predictive filtering technique as a compensation scheme for missing data samples in a drive-by-wire system. In contrast to commonly used FIR filters, our proposed filter only requires one set of weights to

be tuned, memorized, and applied to different numbers of steps ahead prediction.

In our experiments, a displacement sensor in a by-wire brake pedal was compensated for its up to four consecutive missing data samples. In addition to our proposed predictive filter, we have also applied a first-order hold method, a Newton polynomial predictive filter, a linear smoothed Newton filter, and a non-linear predictive filter realized by a time-delay neural network. The results show that prediction MSE of the proposed filter is comparable to the neural filter and better than other filters. Its computational overhead is lower than other filters, except for the first-order hold whose prediction error is undesirably high. Hence, our proposed predictive filtering technique is an ideal tool for missing data compensation in drive by-wire systems as well as any other time- and memory-critical application.

REFERENCES

- [1] O. Rooks, M. Armbruster, S. Buchli, A. Sulzmann, G. Spiegelberg, and U. Kiencke, "Redundancy management for drive-by-wire computer systems," in *Lecture Notes in Computer Sciences 2788, Taken at 22nd International Conference on Computer Safety, Reliability and Security*, Edinburgh, Scotland, Sep. 23–26, 2003, pp. 249–262.
- [2] M. Lubaszewski and B. Courtois, "A reliable fail-safe system," *IEEE Trans. Comput.*, vol. 47, no. 2, pp. 236–241, 1998.
- [3] P. Heinonen and Y. Neuvo, "FIR-median hybrid filters with predictive FIR structures," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 36, no. 6, pp. 892–899, 1988.
- [4] P. T. Harju, "Finite word length implementation of IIR polynomial predictive filters," in *Proc. 1997 IEEE Instrum. Meas. Technol. Conf. (IMTC)*, vol. 1, Ottawa, Canada, May 19–21, pp. 60–65.
- [5] P. T. Harju, "Round off noise properties of IIR polynomial predictive filters," in *Proc. 1997 IEEE Instrum. Meas. Technol. Conf. (IMTC)*, vol. 1, Ottawa, Canada, May 19–21, pp. 66–70.
- [6] P. T. Harju, T. I. Laakso, and S. J. Ovaska, "Applying IIR predictors on Rayleigh fading signals," *Signal Process.*, vol. 48, no. 1, pp. 91–96, 1996.
- [7] P. Handel, "Predictive digital filtering of sinusoidal signals," *IEEE Trans. Signal Process.*, vol. 46, no. 2, pp. 364–374, 1998.
- [8] K. Koppinen, "Efficient filter structures for linearly predictable responses," in *Proc. Third IEEE Nordic Signal Process. Symp. (NORSIG2000)*, Kolmarden, Sweden, Jun. 13–15, 2000.
- [9] K. Koppinen and J. Astola, "Generalized IIR polynomial predictive filters," in *Proc. 8th European Signal Process. Conf. (EUSIPCO-2000)*, Tampere, Finland, Sep. 5–8, 2000.
- [10] S. Valiviita, S. J. Ovaska, and O. Vainio, "Polynomial predictive filtering in control instrumentation: A review," *IEEE Trans. Ind. Electron.*, vol. 46, no. 5, pp. 876–888, 1999.
- [11] F. P. Dawson and L. Klaffke, "Variable-sample-rate delayless frequency-adaptive digital filter for synchronized signal acquisition and sampling," *IEEE Trans. Ind. Electron.*, vol. 46, no. 5, pp. 889–896, 1999.
- [12] T. L. Leung, S. Valiviita, and S. J. Ovaska, "Adaptive and delayless filtering system for sinusoids with varying frequency," in *Proc. IEEE SOUTHEASTCON*, Lexington, KY, Mar. 25–28, 1999, pp. 149–153.
- [13] O. Vainio and S. Valiviita, "Predictive interpolation and decimation of narrow-band signals," *IEEE Trans. Ind. Electron.*, vol. 46, no. 5, pp. 897–903, 1999.
- [14] J. L. Crassidis and F. L. Markley, "Predictive filtering for attitude estimation without rate sensors," *J. Guid. Control Dyn.*, vol. 20, no. 3, pp. 522–527, 1997.
- [15] A. Ferrah, K. J. Bradley, M. S. Woolfson, and G. M. Asher, "New sensorless dynamic speed-estimator for induction motor drives using predictive adaptive filtering," in *Proc. Joint 1996 IEEE Instrum. Meas. Technol. Conf. & IMEKO Technical Committee*, vol. 1, Brussels, Belgium, Jun. 4–6, pp. 458–463.
- [16] S. Da Costa Sousa, M. Joao, and M. Setnes, "Fuzzy predictive filters in model predictive control," *IEEE Trans. Ind. Electron.*, vol. 46, no. 6, pp. 1225–1232, 1999.
- [17] J. M. Sousa and M. Setnes, "Model predictive control: A data-driven approach using simple fuzzy tools," in *Proc. 9th IEEE Int. Conf. Fuzzy Syst.*, vol. 2, San Antonio, TX, May 7–10, 2000, pp. 1017–1020.
- [18] S. Valiviita, X. Z. Gao, and S. J. Ovaska, "Polynomial predictive filters: Complementing technique to fuzzy filtering," in *Proc. IEEE Int. Conf. on Syst. Man Cybern.*, vol. 5, San Diego, CA, Oct. 11–14, 1998, pp. 4648–4652.
- [19] S. J. Ovaska, T. Bose, and O. Vainio, "Genetic algorithm-aided design of predictive filters for electric power applications," in *Proc. IEEE Int. Conf. on Syst. Man Cybern.*, vol. 2, Washington, DC, Oct. 5–8, 2003, pp. 1463–1468.
- [20] M. P. Schoen, "Dynamic compensation of intelligent sensors," in *Proc. 2003 ASME Int. Mech. Eng. Congress*, Washington, DC, Nov. 15–21, pp. 1249–1256.
- [21] K. Lang, A. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Netw.*, vol. 3, pp. 23–43, 1990.
- [22] C. Lin, H. Nein, and W. Lin, "A space-time delay neural network for motion recognition and its application to lipreading," *Int. J. Neural Syst.*, vol. 9, no. 4, pp. 311–334, 1999.
- [23] P. Lingras, S. Sharma, and M. Zhong, "Prediction of recreational travel using genetically designed regression and time-delay neural network models," *Trans. Res. Record*, vol. 1805, pp. 16–24, 2002.
- [24] H. Zhang, M. O. Balaban, and J. C. Principe, "Improving pattern recognition of electronic nose data with time-delay neural networks," *Sens. Actuators B, Chem.*, vol. 96, no. 1–2, pp. 385–389, 2003.
- [25] G. Dorfner, "Neural networks for time series processing," *Neural Netw. World*, vol. 6, no. 4, pp. 447–468, 1996.
- [26] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *J. Intell. Robot. Syst.*, vol. 31, no. 1–3, pp. 91–103, 2001.



Reza Hoseinnezhad was born in Tehran, Iran, in 1973. He received the B.E., M.E., and Ph.D. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 1994, 1996, and 2002, respectively.

From 2002 to 2003, he was an Assistant Professor at the University of Tehran. Since July 2003, he has been a Postdoctoral Research Fellow at the Swinburne University of Technology, Hawthorn, Victoria, Australia, where he has engaged in the design of drive-by-wire systems. His research is focused on

new methodologies for signal processing and sensor data fusion techniques applied in drive-by-wire systems. He is the co-author of 7 journal papers and 14 conference papers, and holds two Australian patents. His research interests are signal processing, control system design, and autonomous mobile robotics, in general, and sensor data fusion, in particular.

Dr. Hoseinnezhad's doctoral research was recognized as the distinguished Ph.D. thesis by the University of Tehran in 2003.



Alireza Bab-Hadiashar (SM'04) was born in Iran in 1964. He received the B.E. degree in 1988 from the University of Tehran, Tehran, Iran, the M.E. degree in 1994 from the University of Sydney, Sydney, Australia, and the Ph.D. degree in 1997 from the Monash University, Clayton, Victoria, Australia.

Since 1997, he has held various academic positions at both Monash University and the Swinburne University of Technology, Hawthorn, Victoria, Australia, where he is currently an Associate Professor and Program Manager of Robotics and Mechatronics.

His research interest is the development of robust data analysis techniques for engineering applications, in general, and computer vision, in particular.