International Journal of Control, 2014
Vol. 87, No. 3, 622–632, http://dx.doi.org/10.1080/00207179.2013.852250

Taylor & Francis
Taylor & Francis Group

Low-complexity piecewise-affine virtual sensors: theory and design

Matteo Rubagotti[a,*], Tomaso Poggi[b], Alberto Oliveri[c], Carlo Alberto Pascucci[d], Alberto Bemporad[d] and Marco Storace[c]

[a]*Department of Robotics, Nazarbayev University, Kabanbay Batyr Avenue 56, 010000 Astana, Kazakhstan;* [b]*RF group, ESS-Bilbao, Paseo Landabarri, 2, 48940 Leioa, Spain;* [c]*Department of Electrical, Electronic, Telecommunications Engineering, and Naval Architecture, University of Genoa, Via Opera Pia 11a, 16145 Genova, Italy;* [d]*IMT Institute for Advanced Studies, Piazza S. Ponziano 6, 55100 Lucca, Italy*

(*Received 31 March 2013; accepted 2 October 2013*)

This paper is focused on the theoretical development and the hardware implementation of low-complexity piecewise-affine direct virtual sensors for the estimation of unmeasured variables of interest of nonlinear systems. The direct virtual sensor is designed directly from measured inputs and outputs of the system and does not require a dynamical model. The proposed approach allows one to design estimators which mitigate the effect of the so-called 'curse of dimensionality' of simplicial piecewise-affine functions, and can be therefore applied to relatively high-order systems, enjoying convergence and optimality properties. An automatic toolchain is also presented to generate the VHDL code describing the digital circuit implementing the virtual sensor, starting from the set of measured input and output data. The proposed methodology is applied to generate an FPGA implementation of the virtual sensor for the estimation of vehicle lateral velocity, using a hardware-in-the-loop setting.

Keywords: piecewise-affine functions; nonlinear observers; digital circuits

1. Introduction

The estimation of unmeasurable variables of a nonlinear dynamical system is a widely studied problem in control theory. Whenever possible, a model of the system is employed to obtain a model-based observer, e.g. extended Kalman filters, unscented Kalman filters, ensemble Kalman filters, particle filters, sliding-mode observers and moving horizon estimators. In many practical applications, however, a reliable model of the system is not available, and one usually follows a *two-step procedure*, obtaining first a black-box model by system identification techniques, and then designing a model-based observer. In this way, the overall performance is usually far from optimal, which suggests the adoption of the so-called *filter design from data*, where the observer is directly designed as a function of past inputs and outputs of the system from a training set of measured data. In this case, we call the observer a direct virtual sensor (DVS). This approach has been proposed in Milanese, Novara, Hsu, and Poolla (2006), and then further developed and applied in Milanese, Novara, Hsu, and Poolla (2009), Ruiz, Taragna, and Milanese (2009), Novara, Fagiano, and Milanese (2013), Novara, Ruiz, and Milanese (2011), Canale, Fagiano, Ruiz, and Signorile (2008), Poggi, Rubagotti, Bemporad, and Storace (2012) and Novara, Ruiz, and Milanese (2013).

This paper proposes a new DVS setting in which the estimate of the current unmeasured variable of interest is obtained as the sum of piecewise-affine simplicial (PWAS) functions of past inputs, measurable outputs and past estimates in a given time window. The main reason for using PWAS functions is that they can be implemented very efficiently in digital circuits (e.g. field-programmable gate arrays, FPGAs (Storace & Poggi, 2011) or application specific integrated circuits (ASICs; Di Federico, Poggi, Julián, & Storace, 2010)). Then, whenever size, power consumption and speed are concerns, this kind of function becomes more competitive with respect to other classes of functions, which are more suitable for modelling purposes, but require richer hardware resources (PCs, DSP boards, etc.) to be implemented. Among the others (see e.g. Castro, Agamennoni, & Álvarez, 2010), we refer to the approach of Poggi et al. (2012), in which a single PWAS function is used for DVS design, as Standard DVS (S-DVS), whose main drawback is that the complexity of the filter increases exponentially with the number of inputs, measurable outputs, and past data used. This effect, known as 'curse of dimensionality' (Bellman, 1957), prevents the use of the S-DVS for medium-size or large-size problems.

* Corresponding author. Email: matteo.rubagotti@nu.edu.kz

In this paper, we present a new formulation, according to which the estimate is obtained as the sum of lower-dimensional PWAS functions, which allows the circuit designer to strongly reduce the effect of the curse of dimensionality with respect to the standard approach. Therefore, we refer to the new approach as reduced-complexity DVS (RC-DVS). Also, we allow the use of past values of the estimated output as further inputs for the DVS, which was not considered in Poggi et al. (2012). We will show that the proposed approach leads to theoretical results on minimum variance and will outperform an estimator based on the two-step procedure. The actual digital implementation of the PWAS DVS is presented and discussed, together with an automated toolchain – included in the MOBY-DIC Toolbox for MATLAB[1] – implementing the described method, which allows one to create the VHDL[2] description of the digital circuit directly from the data sets, after tuning the required parameters. Note that the same toolbox also provides a toolchain for design, simulation and circuit implementation of embedded control systems, as shown in Oliveri et al. (2012). Finally, the proposed methodology is applied to a simulation example (Lorenz system) and to a case study concerned with the estimation of lateral velocity in road vehicles. In the latter case, a circuit architecture implementing the DVS is generated with the toolbox and programmed on an FPGA. Hardware-in-the-loop (HIL) simulations are performed in Simulink by connecting the FPGA to a personal computer in order to validate the performance of the RC-DVS under fixed-point representation of data. A preliminary description of the theoretical development proposed in this paper, focusing on the convergence properties of the RC-DVS, is presented in Rubagotti, Poggi, Bemporad, and Storace (2012).

The paper is organised as follows. Section 2 describes the design and circuit implementation of the proposed DVS. Section 3 deals with convergence properties, whereas Section 4 introduces the main functionalities of MOBY-DIC toolbox, related to the virtual sensor design. The simulation example is presented in Section 5, in order to compare the proposed method with previous approaches, and Section 6 describes the automotive case study. Conclusions are drawn in Section 7.

2. Description of the RC-DVS

2.1 *System description*

In the following, we provide a concise description of the main system characteristics, as stated by Milanese et al. (2009). Consider the nonlinear discrete-time dynamical system $\mathcal{S}$

$$\mathcal{S}: \begin{cases} x(t+1) = g(x(t), u(t)) \\ y(t) = h_y(x(t)) \\ z(t) = h_z(x(t)) \end{cases}, \tag{1}$$

where the state vector is $x \in \mathbb{R}^{n_x}$, the input vector is $u \in \mathbb{R}^{n_u}$, the vector of measurable outputs is $y \in \mathbb{R}^{n_y}$, and t represents the discrete-time instant. Vector $z \in \mathbb{R}^{n_z}$ collects a set of variables to be estimated. We assume that only during preliminary experiments $z(t)$ can be measured by a *real sensor* at time instants $t = 0, \ldots, T-1$. A portion of these measurements, from $t = 0$ to $t = T_t - 1$, is referred to as *training set* and is used to design the virtual sensor, which will operate without measuring $z(t)$; the remaining data, from $t = T_t$ to $t = T - 1$, constitute the *validation set* and are used to validate the estimation capabilities of the DVS. The functions $g(\cdot,\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, $h_y(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ and $h_z(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$ are not known. Without loss of generality we set $n_z = 1$ (the case $n_z > 1$ can be managed component-wise by solving n_z scalar problems).

The possibility of estimating $z(t)$ is related to the concept of observability. Indeed, as stated in Milanese et al. (2009), the observability of the system implies that z can be uniquely determined using a finite number $0 \le M_u \le n_x$ of samples of u, a finite number $1 \le M_y \le n_x$ of samples of y, and a finite number $0 \le M_z \le n_x$ of samples of z. In particular, if $\mathcal{S}$ is *observable*, there exists a function f_z such that $z(t) = f_z(U(t), Y(t), Z(t))$, with

$$\begin{aligned} U(t) &\triangleq [\ u(t-M_u+1)'\ \ u(t-M_u+2)'\ \cdots\ u(t)'\ ]' \\ Y(t) &\triangleq [\ y(t-M_y+1)'\ \ y(t-M_y+2)'\ \cdots\ y(t)'\ ]' \\ Z(t) &\triangleq [\ z(t-M_z+1)'\ \ z(t-M_z+2)'\ \cdots\ z(t-1)'\ ]', \end{aligned}$$

where $'$ denotes transposition. Two cases can be distinguished:

- if $\mathcal{S}$ is *fully observable*, the function f_z can be defined such that $z(t) = f_z(U(t), Y(t))$, i.e. the past values of z are not needed, since the whole state x can be reconstructed from $Y(t)$ and $U(t)$, and z is a static function of x; this was the case considered in Poggi et al. (2012);
- if $\mathcal{S}$ is *partially observable*, it is not possible to reconstruct the whole state vector x, and then past values of z are needed to reconstruct $z(t)$.

Vector $U(t)$ is empty when the system is autonomous, while $Z(t)$ is empty when the system is fully observable. In this paper, it is assumed that (1) is *partially observable*, i.e. it is possible to reconstruct the value of $z(t)$ as a function of a finite number of past values of y, u and z (of course, all the results of this paper will also hold for fully observable systems as a particular case).

2.2 *General formulation of the RC-DVS*

Assuming that system $\mathcal{S}$ is partially observable, the DVS is a function providing the estimate $\hat{z}(t)$ of z at time t. We assume that noisy measurements of y, u and z at past time instants

are available, $\tilde{u}(t) = u(t) + \eta_u(t)$, $\tilde{y}(t) = y(t) + \eta_y(t)$ and $\tilde{z}(t) = z(t) + \eta_z(t)$, where η_u, η_y and η_z are unknown stochastic variables. For given values $M_u, M_y, M_z \geq 0$, the inputs of the DVS will be noisy sequences of measurements of y and u, and a vector of past values of $\hat{z}$, namely

$$\begin{aligned}
\tilde{U}(t) &\triangleq \left[\, \tilde{u}(t - M_u + 1)' \;\; \tilde{u}(t - M_u + 2)' \;\; \cdots \tilde{u}(t)' \,\right]' \\
\tilde{Y}(t) &\triangleq \left[\, \tilde{y}(t - M_y + 1)' \;\; \tilde{y}(t - M_y + 2)' \;\; \cdots \tilde{y}(t)' \,\right]' \\
\hat{Z}(t) &\triangleq \left[\, \hat{z}(t - M_z)' \;\; \hat{z}(t - M_z + 1)' \;\; \cdots \hat{z}(t - 1)' \,\right]'
\end{aligned}$$

Remark 1: If a model of the system were available, it would be possible to check the observability, and to analytically determine suitable values for M_u, M_y and M_z. However, in practice, a model is not available, and then observability is simply assumed a priori, and the values of M_u, M_y and M_z become tuning parameters.

For the sake of compactness, henceforth the input of the DVS is referred to as

$$\Xi(t) \triangleq [\, \tilde{U}'(t) \; \tilde{Y}'(t) \; \hat{Z}'(t) \,]' \in \mathbb{R}^{n_\xi},$$

where $n_\xi \triangleq M_u n_u + M_y n_y + M_z$, and $'$ denotes the transposition operator.

Assume to split vector Ξ into $\nu \in \mathbb{N}$ subsets $\Xi^1, \Xi^2, \ldots, \Xi^\nu$, such that all elements of Ξ are included in one and only one of these subsets. Each subset Ξ^j ($j = 1, \cdots, \nu$) has dimension equal to n_j, such that $1 \leq n_j \leq n_\xi$ and $n_1 + n_2 + \cdots + n_\nu = n_\xi$. The n_j elements of each Ξ^j are denoted as $\xi_{j,1}, \xi_{j,2}, \ldots, \xi_{j,n_j}$. The proposed DVS is referred to as $\mathcal{V}_\alpha(w)$ and is defined through a sum of PWAS functions f_α^j, $j = 1, \ldots, \nu$, each being the weighted sum of N_j PWAS basis functions (the choice of the basis functions will be discussed in Section 2.3):

$$\begin{aligned}
\hat{z}(t) = f_\alpha(\Xi(t); w) &= \sum_{j=1}^{\nu} f_\alpha^j(\Xi^j(t); w) \\
&= \sum_{j=1}^{\nu} \sum_{k=1}^{N_j} w_{j,k} \alpha_{j,k}(\Xi^j(t)),
\end{aligned} \tag{2}$$

where $f_\alpha : \mathbb{R}^{n_\xi} \to \mathbb{R}$ (for fixed w), $\{\alpha_{j,k}\}$ is a basis of PWAS functions and N_j is the number of basis functions in each domain Ξ^j (see Section 2.3). Also,

$$w \triangleq [\, w_{1,1} \cdots w_{1,N_1} \; w_{2,1} \cdots w_{2,N_2} \cdots w_{\nu,1} \cdots w_{\nu,N_\nu} \,]'$$

with $w \in D_w \subset \mathbb{R}^{N_\xi}$, D_w being a compact set and $N_\xi = \sum_{j=1}^{\nu} N_j$. The vector of parameters w (which determines the shape of f_α) is obtained by solving the least-squares problem

$$w^* = \arg\min_w \left\{ \sum_{t=M}^{T_t - 1} [\tilde{z}(t) - f_\alpha(\Xi(t); w)]^2 \right\}, \tag{3}$$

where T_t is the number of measurements in the training set and $M = \max(M_u, M_y, M_z)$, $M \ll T_t$.

Remark 2: As observed above, the past values of $\hat{z}$ were not considered in Poggi et al. (2012), where $\Xi(t) \triangleq [\, \tilde{U}'(t) \; \tilde{Y}'(t) \,]'$, so the approach was only applicable to fully observable systems. Also, in Poggi et al. (2012) there was no partitioning of Ξ, and then the PWAS S-DVS was directly defined over a single domain of dimension n_ξ. This might cause serious implementation problems, since the number of coefficients in (2) increases exponentially with n_ξ. Splitting the domain into subspaces can lead to huge practical advantages, as shown in Section 5. At the moment, there exists no systematic way to determine how to split the vector Ξ into its different subsets, in order to maximise the performance. A possibility that has shown good practical results has consisted in merging in a single subset Ξ^j all the variables that refer to a specific time instant. However, the results can be strongly system-dependent, and the search for a systematic procedure to determine the subsets will be object of future research.

Remark 3: If the DVS starts receiving measurements at time $t = 0$, the needed past values of $\tilde{u}$ and $\tilde{y}$ will be available at time $t_{uy} \triangleq \max(M_u, M_y)$. Therefore, if $M_z = 0$, the DVS will start providing its output at time t_{uy}. In case $M_z > 0$, past values of $\hat{z}$ would be needed, and then the DVS will be able to generate its output at time $t = M - 1$, using an initial guess for the past values of $\hat{z}$. In practical applications, the initial guess can be related to an a priori knowledge of the initial condition. For example, when estimating the lateral velocity of a vehicle (see Section 6), since the DVS starts working when the engine is turned on, it is reasonable to assume that the vehicle is not moving, which implies that the lateral velocity is equal to zero.

2.3 *Digital implementation of the RC-DVS*

To efficiently implement the RC-DVS (2) on a digital circuit, we consider a class of continuous and regular PWAS functions, defined over hyper-rectangular domains

$$S_j = \left\{ \Xi^j \in \mathbb{R}^{n_j} : \underline{\xi}_{j,i} \leq \xi_{j,i} \leq \bar{\xi}_{j,i},\; j = 1, \ldots, \nu,\; i = 1, \ldots, n_j \right\}. \tag{4}$$

Each domain is partitioned into a set of regular *simplices* by taking $p_{j,i}$ ($i = 1, \ldots, n_j$) uniformly distributed points along each $\xi_{j,i}$ axis; therefore $N_j = \prod_{i=1}^{n_j} p_{j,i}$ vertices $v_{j,k}$, $k = 1, \ldots, N_j$ are obtained. Any PWAS function f_α^j of this kind can be expressed as a weighted sum of N_j PWAS basis functions as in (2); different types of continuous basis

functions can be defined; we use here the so-called α-basis (Julián, Desages, & D'Amico, 2000). For the use of other bases to represent PWAS functions the reader is referred to Storace, Repetto, and Parodi (2003) and Repetto, Storace, and Parodi (2003). Each function $\alpha_{j,k}(\Xi^j)$ in (2) is a PWAS hyper-pyramid, which takes the value 1 at the vertex $v_{j,k}$ (i.e. the kth vertex of the jth domain) and 0 at all the other vertices $v_{j,q}, q \neq k$. For details on simplices and PWAS functions in general, the reader is referred to Poggi et al. (2012) and the references therein.

Digital architectures implementing PWAS functions were proposed in Storace and Poggi (2011) and are used here for the implementation of the PWAS virtual sensor (2). Essentially, they calculate the value of a PWAS function at a given point ξ_j by interpolating the values of the function itself at the vertices of the simplex containing ξ_j. Other architectures were proposed for the implementation of PWAS functions in Echevarria, Martínez, Echanobe, del Campo, and Tarela (2007) and Rovatti, Fantuzzi, and Simani (2000).

A high-level block scheme of the circuit implementing the RC-DVS is shown in Figure 1.

At each sampling time t, the values of the system inputs $u(t)$ and measurable outputs $y(t)$ are loaded into First-In-First-Out (FIFO) blocks which behave as buffers, storing the data at current and past time instants (i.e. $\tilde{U}(t)$ and $\tilde{Y}(t)$). Once the FIFO blocks are full, the computation of the estimated output is performed by the *PWAS*_i ($i = 1, \ldots, \nu$) blocks, which are responsible for the evaluation of the ν PWAS functions. A detailed description of these blocks is available in Storace and Poggi (2011). Each *PWAS_i* block communicates the end of its computation with a *ready_i* signal; as soon as all *ready_i* signals are active, the results of each function evaluation (*fpwas_i*) are added up by an adder block, which provides the estimation of the unmeasurable output $z(t)$, and a global *ready* signal is set to logic value '1' indicating that the result is available. If $M_z > 0$, the estimation is brought back to the *FIFO_z* block and is used as input for next estimations. An affine scaling of $z(t)$ (performed by block *SCALE_z*) is necessary to bring the value to the correct range needed by circuit inputs.

Two kinds of architectures (serial and parallel) are available for blocks *PWAS_i*. If the serial architectures are used, the overall latency of the virtual sensor circuit is equal to $\max\{n_i\} + 7$ ($i = 1, \ldots, \nu$) clock cycles; otherwise, if the parallel architectures are chosen, the latency is reduced to six clock cycles, at the cost of employing $\sum_{i=1}^{\nu} n_i + \nu + 1$ multipliers instead of only $\nu + 1$ for the serial circuit.[3]

3. Convergence properties of the estimation error

This section presents the main theoretical properties of the proposed RC-DVS. Consider a standard two-step procedure to obtain a minimum-variance filter $\mathcal{K}(\theta)$, estimating a state-space model of (1) from a set of available measurements, and then designing $\mathcal{K}(\theta)$ based on this model. The filter $\mathcal{K}(\theta)$ will be based on the set of parameters $\theta \in D_\theta$ (D_θ being a compact set), and designed relying on a class of models $\mathcal{M}(\theta)$ of (1). In particular, we consider the model

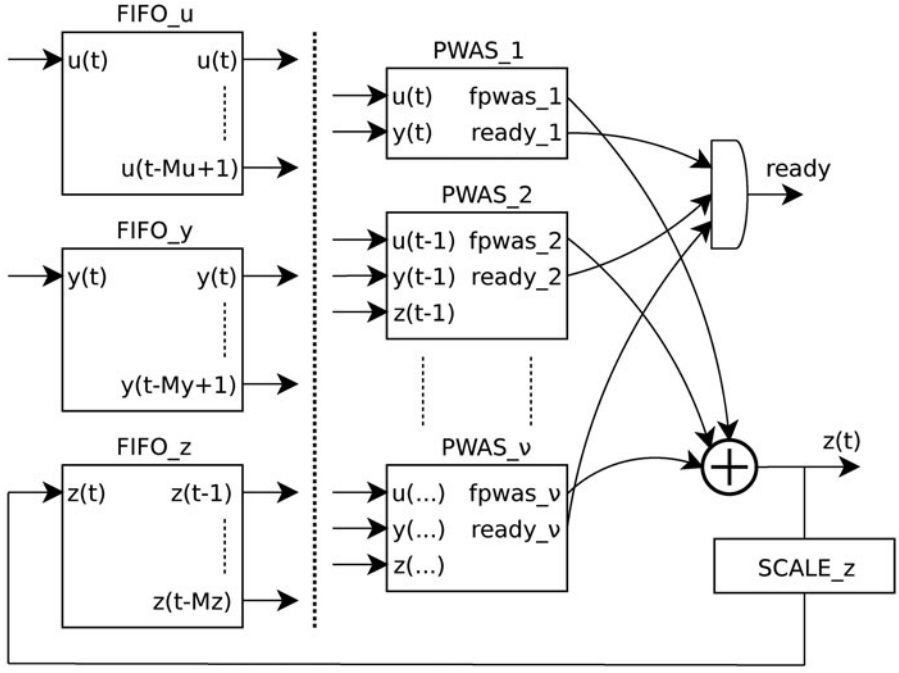

Figure 1. Block scheme of the circuit implementing the RC-DVS. The inputs of block $PWAS_\nu$ are generic since they depend on the values of M_u, M_y and M_z. The vertical dotted line masks all connections, which are easily understandable looking at the signal names.

$\mathcal{M}(\theta^*)$, obtained using a prediction error method from a set of measured data, and the corresponding filter realisation $\mathcal{K}(\theta^*)$, which is assumed to have fading memory. It is possible to represent the estimate given by $\mathcal{K}(\theta^*)$ in regression form as

$$\hat{z}_{\mathcal{K}}(t+1) = f_{\mathcal{K}}(\Xi(t); \theta^*). \tag{5}$$

The following theorem describes the properties of the proposed RC-DVS (2) in comparison with (5).

Theorem 3.1: *Let system* (1) *be partially observable. Consider a minimum-variance filter* $\mathcal{K}(\theta^*)$ *in* (5)*, and the virtual sensor* $\mathcal{V}_\alpha(w^*)$ *in* (2)*, whose parameter vector* w^* *is obtained from* (3)*. Let* $\hat{z}_{\mathcal{V}}$ *be the value of the estimate obtained with a RC-DVS* $\mathcal{V}_\alpha(w^*)$ *in* (2)*. Then, denoting expected values by* $E[\cdot]$*, the following results hold with probability* 1 *as* $T_t \to \infty$*:*

(i) *The vector of parameters defined in* (3) *guarantees the minimisation of the variance of the estimation error among all the virtual sensors with the same structure, i.e.* $\mathcal{V}_\alpha(w^*) = \arg\ \min_{\mathcal{V}_\alpha(w)} E[(z(t) - \hat{z}_{\mathcal{V}})^2]$.

(ii) *If there exists* w *such that* $\mathcal{K}(\theta^*) = \mathcal{V}_\alpha(w)$ *(i.e. it is possible to express the two-step observer in regression form as a particular realisation of the virtual sensor), one obtains that* $E[(z(t) - \hat{z}_{\mathcal{K}}(t))^2] \geq E[(z(t) - \hat{z}_{\mathcal{V}}(t))^2]$*, i.e. the performance of the RC-DVS is better than or equal to that of* (5).

(iii) *If there exists* $\theta^o \in D_\theta$ *such that* $\mathcal{S} = \mathcal{M}(\theta^o)$ *(i.e. there exists a set of parameters of the two-step observer that describes exactly the system), and there exists a vector* w *such that* $\mathcal{K}(\theta^o) = \mathcal{V}_\alpha(w)$*, then* $\mathcal{V}_\alpha(w^*)$ *is a minimum-variance filter.*

Proof: Analogously to Milanese et al. (2006) and Poggi et al. (2012), one needs to show that three specific conditions listed in Ljung (1978) hold, which leads to the fulfilment of (*i*), (*ii*) and (*iii*). The first condition (referred to as 'S3' in Ljung, 1978) refers to the data set and is satisfied if we assume that system (1) is partially observable. The second condition ('C1') refers to the choice of vector w and is fulfilled if the quadratic criterion in (3) is adopted. The third condition ('M1') requires to check if the proposed DVS retains the following property: there exist two scalars $C > 0$ and λ, $0 < \lambda < 1$, such that

(1) the estimate is bounded at the origin, namely

$$|f_\alpha(\Xi_0(t); w)| \leq C \tag{6}$$

for $\Xi_0(t) \triangleq [\,\tilde{U}_0'(t)\ \tilde{Y}_0'(t)\ \hat{Z}_0'(t)\,] = 0 \in \mathbb{R}^{n_\xi}$;

(2) the virtual sensor (2) has exponential fading memory

$$\begin{aligned} &|f_\alpha(\Xi_1(t); w) - f_\alpha(\Xi_2(t); w)| \\ &\quad \leq C \sum_{s=0}^{t} \lambda^{t-s} \big[\|\tilde{u}_1(s) - \tilde{u}_2(s)\|_1 \\ &\qquad + \|\tilde{y}_1(s) - \tilde{y}_2(s)\|_1 + \|\hat{z}_1(s) - \hat{z}_2(s)\|_1 \big] \end{aligned} \tag{7}$$

for any $\Xi_1(t)$, $\Xi_2(t)$;

(3) function f_α is differentiable with respect to w for all $w \in D_w$ and the following exponential fading property is satisfied:

$$\begin{aligned} &\|\nabla_w f_\alpha(\Xi_1(t); w) - \nabla_w f_\alpha(\Xi_2(t); w)\|_1 \\ &\quad \leq C \sum_{s=0}^{t} \lambda^{t-s} \big[\|\tilde{u}_1(s) - \tilde{u}_2(s)\|_1 \\ &\qquad + \|\tilde{y}_1(s) - \tilde{y}_2(s)\|_1 + \|\hat{z}_1(s) - \hat{z}_2(s)\|_1 \big] \end{aligned} \tag{8}$$

for any $\Xi_1(t)$, $\Xi_2(t)$.

In the remainder of the proof we will prove that all three properties hold. Recalling that $0 \leq \alpha_k(\cdot) \leq 1$ holds for all k, it yields

$$\begin{aligned} |f_\alpha(\Xi_0(t); w)| &= \left| \sum_{j=1}^{\nu} \sum_{k=1}^{N_j} w_{j,k} \alpha_{j,k}(\Xi_0^j(t)) \right| \\ &\leq \sum_{j=1}^{\nu} \sum_{k=1}^{N_j} |w_{j,k}| \triangleq C_1 > 0 \end{aligned}$$

which implies the fulfilment of (6).

Consider the left-hand side of (7):

$$\begin{aligned} &|f_\alpha(\Xi_1(t); w) - f_\alpha(\Xi_2(t); w)| \\ &\quad \leq \left| \sum_{j=1}^{\nu} \sum_{k=1}^{N_j} w_{j,k} \big(\alpha_{j,k}(\Xi_1^j(t)) - \alpha_{j,k}(\Xi_2^j(t)) \big) \right| \\ &\quad \leq \sum_{j=1}^{\nu} \sum_{k=1}^{N_j} \left| w_{j,k} \left[\alpha_{j,k}(\Xi_1^j(t)) - \alpha_{j,k}(\Xi_2^j(t)) \right] \right| \\ &\quad = \sum_{j=1}^{\nu} \sum_{k=1}^{N_j} |w_{j,k}| \big| \alpha_{j,k}(\Xi_1^j(t)) - \alpha_{j,k}(\Xi_2^j(t)) \big|. \end{aligned}$$

The structure of the α-basis chosen for the design of the virtual sensor implies that each basis function $\alpha_{j,k}$ is Lipschitz continuous with respect to the input $\Xi(t)$. More precisely, there exists $\beta > 0$ such that

$$|\alpha_{j,k}(\Xi_1^j(t)) - \alpha_{j,k}(\Xi_2^j(t))| \leq \beta \big\| \Xi_1^j(t) - \Xi_2^j(t) \big\|_1 \tag{9}$$

for all $(j, k) \in (1, \ldots, v \times 1, \ldots, N_j)$. Then,

$$\begin{aligned}
&\sum_{j=1}^{v}\sum_{k=1}^{N_j} |w_{j,k}| \left|\alpha_{j,k}\left(\Xi_1^j(t)\right) - \alpha_{j,k}\left(\Xi_2^j(t)\right)\right| \\
&\quad \le \beta \sum_{j=1}^{v}\sum_{k=1}^{N_j} |w_{j,k}| \left\|\Xi_1^j(t) - \Xi_2^j(t)\right\|_1 \\
&\quad \le \beta C_1 \sum_{j=1}^{v}\sum_{k=1}^{N_j} \left\|\Xi_1^j(t) - \Xi_2^j(t)\right\|_1 \\
&\quad = \beta C_1 \sum_{j=1}^{v} N_j \left\|\Xi_1^j(t) - \Xi_2^j(t)\right\|_1 \\
&\quad \le \beta C_1 n_\xi \sum_{j=1}^{v} \left\|\Xi_1^j(t) - \Xi_2^j(t)\right\|_1 \\
&\quad = \beta C_1 n_\xi \left\|\Xi_1(t) - \Xi_2(t)\right\|_1. \qquad (10)
\end{aligned}$$

Consider the right-hand side of (7), and take any λ, $0 < \lambda < 1$. Moreover, let $C_2 > 0$ be a constant to be determined. Recalling that $M = \max(M_u, M_y, M_z)$, it yields

$$\begin{aligned}
&C_2 \sum_{s=0}^{t} \lambda^{t-s} [\|\tilde{u}_1(s) - \tilde{u}_2(s)\|_1 \\
&\qquad + \|\tilde{y}_1(s) - \tilde{y}_2(s)\|_1 + \|\hat{z}_1(s) - \hat{z}_2(s)\|_1] \\
&\quad \ge C_2 \left(\sum_{s=t-M_u+1}^{t} \lambda^{t-s} \|\tilde{u}_1(s) - \tilde{u}_2(s)\|_1 \right. \\
&\qquad + \sum_{s=t-M_y+1}^{t} \lambda^{t-s} \|\tilde{y}_1(s) - \tilde{y}_2(s)\|_1 \\
&\qquad \left. + \sum_{s=t-M_z+1}^{t} \lambda^{t-s} \|\hat{z}_1(s) - \hat{z}_2(s)\|_1 \right) \\
&\quad \ge C_2 \lambda^{M-1} \left(\sum_{s=t-M_u+1}^{t} \|\tilde{u}_1(s) - \tilde{u}_2(s)\|_1 \right. \\
&\qquad \left. + \sum_{s=t-M_y+1}^{t} \|\tilde{y}_1(s) - \tilde{y}_2(s)\|_1 + \sum_{s=t-M_z+1}^{t} \|\hat{z}_1(s) - \hat{z}_2(s)\|_1 \right) \\
&\quad = C_2 \lambda^{M-1} \left\|\Xi_1(t) - \Xi_2(t)\right\|_1 .
\end{aligned}$$

If we define $C_2 \triangleq \lambda^{1-M} C_1 \beta n_\xi$, we obtain $C_1 \beta n_\xi \|\Xi_1(t) - \Xi_2(t)\|_1 = C_2 \lambda^{M-1} \|\Xi_1(t) - \Xi_2(t)\|_1$, which implies the fulfilment of (7). Function f_α is differentiable with respect to w, and its gradient is

$$\begin{aligned}
\nabla_w f_\alpha(\Xi(t); w) &= \nabla_w \left(\sum_{k=1}^{N_1} w_{1,k} \alpha_{1,k}(\Xi^1(t)) \right. \\
&\quad \left. + \cdots + \sum_{k=1}^{N_v} w_{1,k} \alpha_{v,k}(\Xi^v(t)) \right) \\
&= [\, V'_{\alpha_1}(\Xi^1(t))\ V'_{\alpha_2}(\Xi^2(t)) \cdots\ V'_{\alpha_v}(\Xi^v(t))\,],
\end{aligned}$$

where $V_{\alpha_j}(\Xi^j(t)) = \left[\alpha_{j,1}(\Xi^j(t)) \alpha_{j,2}(\Xi^j(t)) \cdots \alpha_{j,N_j}(\Xi^j(t))\right]'$, with $j = 1, \ldots, v$. Considering the left-hand side of (8), from (9) we obtain

$$\begin{aligned}
&\|\nabla_w f_\alpha(\Xi_1(t); w) - \nabla_w f_\alpha(\Xi_2(t); w)\|_1 \\
&\quad = \sum_{j=1}^{v}\sum_{k=1}^{N_j} \left|\alpha_{j,k}\left(\Xi_1^j(t)\right) - \alpha_{j,k}\left(\Xi_2^j(t)\right)\right| \\
&\quad \le \beta n_\xi \|\Xi_1(t) - \Xi_2(t)\|_1 = \beta n_\xi \|\Xi_1(t) - \Xi_2(t)\|_1 .
\end{aligned}$$

Noting that the right-hand side of (8) coincides with that of (7), by setting $C_3 = \lambda^{1-M} \beta n_\xi$, we obtain $\beta n_\xi \|\Xi_1(t) - \Xi_2(t)\|_1 \le C_3 \lambda^{M-1} \|\Xi_1(t) - \Xi_2(t)\|_1$, which leads to the fulfilment of (8). The existence of C_1, C_2 and C_3 implies that, for any choice of λ, $0 < \lambda < 1$, by choosing $C = \max(C_1, C_2, C_3)$, conditions (6)–(8) are satisfied, which completes the proof.

Remark 4: The proposed RC-DVS retains all the positive features of the general DVS framework of Milanese et al. (2006, 2009). Even assuming of being able to compute $\mathcal{K}(\theta^*)$ based on a perfect model, the two-step procedure would perform no better than the direct approach. In the presence of modelling errors, the virtual sensor $\mathcal{V}(w^*)$ would be the minimum-variance estimator among the selected approximating class of filters (although it may be suboptimal). The listed properties are analogous to those proven in Poggi et al. (2012), but referred to a wider range of implementations, since the S-DVS can be seen as a particular case of the RC-DVS.

4. A MATLAB toolbox for DVS design

We developed a software toolchain in MATLAB that is able to automatically generate VHDL files for the implementation of S-DVS or RC-DVS, starting from training and validation data sets. This tool has been included in the MOBY-DIC Toolbox.[4] The project flow of the toolbox is shown in Figure 2. First of all, a *virtualsensor* object must be created by specifying the number of measurable inputs (n_u) and outputs (n_y) of the system:

```
vs = virtualsensor(nu,ny,vsOpts).
```

`vsOpts` is a structure in which sensor specifications can be set, such as the use of standard or reduced-complexity approach. Second, the time windows M_u, M_y

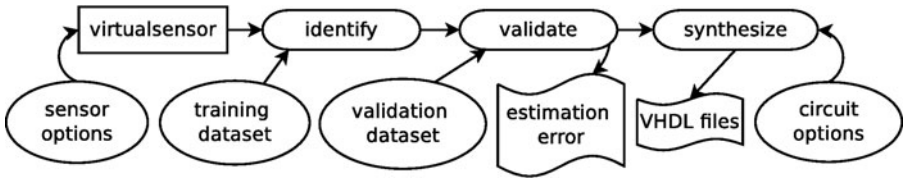

Figure 2. Project flow of MOBY-DIC toolbox for the implementation of PWAS virtual sensors.

and M_z must be specified as well as the number of subdivisions per dimensions of the simplicial partitions (i.e. the vector p containing the values $p_{j,i}$ defined in section 2.3):

```
vs = vs.setInputTimeWindow(Mu);
vs = vs.setOutputTimeWindow(My);
vs = vs.setAutoregressiveTimeWindow
     (Mz);
vs = vs.setNumberOfPartitions(p).
```

The identification (training) of the sensor starting from a training data set composed by T_t samples of noisy measurements $\tilde{u}$, $\tilde{y}$ and $\tilde{z}$ (u, y and z in the code) is performed with method *identify*:

```
vs = vs.identify(u,y,z,p).
```

The validation of the sensor on a validation data set composed by $T_v = T - T_t$ samples of $\tilde{u}$, $\tilde{y}$ and $\tilde{z}$ (uv, yv and zv in the code) is carried out by method *validate*:

```
[zh err] = vs.validate(uv,yv,zv).
```

The output variable zh contains the estimated values ($\hat{z}$) and err contains the value of the root mean square estimation error (RMSEE), defined as

$$\text{RMSEE}(z,\hat{z}) = \sqrt{\frac{\sum_{t=T_t}^{T-1}[z(t)-\hat{z}(t)]^2}{T_v}} \tag{11}$$

and the maximum error between real and estimated data. A plot of these data is also automatically provided.

Once the validation has been performed, method *synthesize* allows one to generate the VHDL files for the circuit implementation of the virtual sensor:

```
vs = vs.synthesize(cirOpts).
```

cirOpts contains the circuit parameters such as the chosen kind of architecture (serial or parallel), the number of bits to code inputs (*N_BIT*) and coefficients (*N_BIT_COEFF*), and the system sampling time. A log file containing information about circuit latency, number of multipliers and memory occupation is also generated.

5. Simulation results

Before introducing the case study, we test the performance of the RC-DVS on a simulation example that permits a direct comparison with the S-DVS proposed in Poggi et al. (2012) and the approach of Milanese et al. (2006).

Consider the discrete-time Lorenz system

$$\begin{cases} x_1(t+1) = (1-\tau s)x_1(t) + \tau s x_2(t) \\ x_2(t+1) = (1-\tau)x_2(t) - \tau x_1(t)x_3(t) + \tau\rho x_1(t) \\ x_3(t+1) = (1-\tau\beta)x_3(t) + \tau x_1(t)x_2(t) \\ \tilde{y}_1(t) = x_1(t)x_2(t) + \eta_{y_1}(t) \\ \tilde{y}_2(t) = x_2^2(t) + \eta_{y_2}(t) \\ \tilde{z}(t) = \sin(0.1x_3(t)) + \eta_z(t), \end{cases} \tag{12}$$

where $\tau = 0.01$ is the sampling time, $s = 10$, $\beta = 8/3$ and $\rho = 28$ are fixed parameters, and $\eta_{y_1}(t)$, $\eta_{y_2}(t)$ and $\eta_z(t)$ are Gaussian processes with zero mean and standard deviations equal to 0.02, 0.02 and 0.01, respectively. With this set of parameters, system (12) exhibits a chaotic behaviour. The S-DVS has already been tested on the same system in Poggi et al. (2012), and compared with the approach of Milanese et al. (2006). Simulations were carried out using the RMSEE calculated over a test set. As a result, the values of the RMSEE for the two approaches were very close to each other.

In the following, a RC-DVS and a S-DVS are derived from a set of $T_t = 60{,}000$ samples of $\tilde{z}(t)$ and $\tilde{y}(t)$. The parameters (M_y, M_z) of the two DVS have been varied in order to show the differences between the two methods. Note that the Lorenz system is autonomous ($n_u = 0$), so that M_u can be ignored.

In order to derive the RC-DVS, ν has been set to $\nu = \max\{M_y, M_z\}$, i.e. $\Xi(t)$ is divided into a number of subsets equal to the number of past samples used by the RC-DVS itself. As a consequence, the estimate $\hat{z}(t)$ is given by the sum of ν PWAS functions. We select a uniform partition with three subdivisions along each dimension. The remaining parameters used for the virtual sensors are reported in Table 1. Table 1 also shows the RMSEE, calculated over $T_v = 3000$ samples, for RC-DVS and S-DVS.

Table 1. Parameters and simulation results for RC-DVS and S-DVS applied to Lorenz's system.

Simulation	Method	N_ξ	M_z	M_y	RMSEE
A	S-DVS	256	0	2	0.201
	RC-DVS	32	0	2	0.225
B	S-DVS	4096	2	2	0.042
	RC-DVS	64	2	2	0.060
C	RC-DVS	4096	64	64	0.002

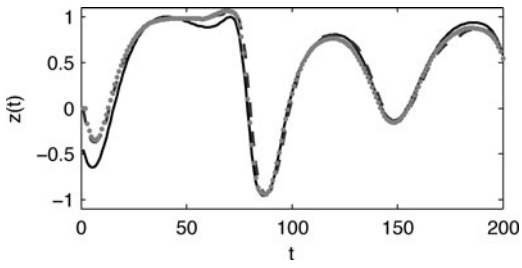

Figure 3. Time evolution of true data (solid black), S-DVS estimate (dashed grey) and RC-DVS estimate (dotted grey).

Simulation A shows that the RMSEE of S-DVS is lower than for RC-DVS if the same value of M_y (i.e. past samples of the measurable output) is used. Nevertheless, the complexity in terms of coefficients is higher in the case of S-DVS (256 instead of 32).

In Simulation B we allowed the RC-DVS and the S-DVS to use past estimates $\hat{z}(t)$, i.e. we set $M_z = 2$. Figure 3 shows the evolution of the estimation error before converging to a neighbourhood of the origin, for both the S-DVS and the RC-DVS virtual sensors. From Table 1, it is apparent that the performance in case B is better than in case A, at the cost of a higher complexity for both virtual sensors.

Finally, Simulation C shows the results obtained by increasing the value of M_y and M_z for the RC-DVS until the same number of coefficients of the S-DVS of Simulation B is reached. In this case, the estimation error of the RC-DVS is much lower than the error obtained with the S-DVS of Simulation B. Note that in this case it is not possible to derive a practical realisation of the S-DVS with $M_y = M_z = 64$, since the resulting PWAS function would be defined by more than 10^{115} coefficients.

This example shows that, in case a high number of past values is required, the S-DVS becomes impossible to implement, due to the curse of dimensionality (which is not the case for the RC-DVS).

6. Case study: estimation of lateral velocity in road vehicles

6.1 *Problem formulation*

In order to test the developed algorithm on a significant case study, we consider the problem of estimating the lateral velocity in road vehicles, a classical problem in the automotive industry (see e.g. Farrelly & Wellstead, 1996). Our objective is to estimate the lateral velocity $z = v_y$ with input vector $u = [a_y\, r]'$ (i.e. lateral acceleration and yaw rate), and output $y = v_x$ (i.e. longitudinal velocity). In Farrelly and Wellstead (1996), it is shown that, using a simplified kinematic model of the car, the system is fully observable when $r(t) \neq 0$. Without making use of simplified models, we use the above-mentioned input and output variables of the system to design a RC-DVS from data, which will provide an estimate of the lateral velocity v_y as a sum of PWAS functions of a_y, r and v_x in a given window of past values. We use a sampling frequency of 10 Hz for data acquisition, which is fast enough to describe all the relevant dynamic behaviours of the vehicle in the given maneuvers (the actual signals are provided at a frequency of 100 Hz, and subsampled with a factor equal to 10). As shown in Section 6.3, the latency of the circuit implementing the virtual sensor is much lower than the used sampling time. Nevertheless, note that the DVS will be just a component of a more complex system. Thus, such a low estimation latency allows for an easier design of other components (e.g. controllers or other signal processing algorithms) whose computation time would stack together.

In the considered case study, the data set describes different kinds of maneuvers, provided by Ford Forschungszentrum Aachen and generated by using an internally developed high-precision simulator, whose parameters are tuned upon a real car. The data set is mainly composed of three types of maneuvers: step steer, sine steer and lane change, for a total of more than 200 maneuvers. Each maneuver is characterised by different values of the lateral acceleration a_y, the longitudinal velocity v_x, and the yaw rate r. Over the whole data set, the lateral acceleration a_y ranges from -10 to 10 m/s^2, the longitudinal velocity v_x spans the interval from 3 to 41 m/s, and finally the yaw rate r is in the range from -1.54 to 1.25 rad/s. The lateral velocity v_y is ranges from -20 to 20 m/s, while the so-called sideslip angle $\alpha_s \triangleq \tan^{-1}(v_x/v_y)$ ranges from $-\pi/4$ to $\pi/4$ rad. Note that, if the sideslip angle were always too close to zero, the nonlinearity in the vehicle dynamics would be negligible. 143 maneuvers (72%) are used for training the RC-DVS, and the remaining 57 (28%) are used for validation purposes.

While designing the RC-DVS, we have $n_u = 2$ (a_y and r) and $n_y = 1$ (v_x). After trying different values for M_u, M_y and M_z, we decided to set $M_u = M_y = 5$, $M_z = 1$, which led to a good trade off between performance and memory occupation. The vectors $\Xi^j, j = 1, \ldots, \nu$ are

$$\Xi^j = \begin{bmatrix} \tilde{u}(t-j+1) \\ \tilde{y}(t-j+1) \end{bmatrix},\ j = 1,3,4,5;\ \ \Xi^2 = \begin{bmatrix} \tilde{u}(t-1) \\ \tilde{y}(t-1) \\ \hat{z}(t-1) \end{bmatrix}.$$

Note that the number of elements of each vector Ξ^j is $n_j = 3$ for $j = 1, 3, 4, 5$, and $n_2 = 4$. Finally, we subdivide each component of each Ξ^j into eight segments of equal length ($p_{j,i} = 9, j = 1, \ldots, \nu, i = 1, \ldots, N_j$).

Using the described tuning parameters, we obtain a total of $N_v = 8748$ vertices. Note that using the S-DVS (i.e. setting $\nu = 1$ and keeping all the other parameters as they are) would lead to a number of vertices equal to $N_v = 2 \times 10^{15}$, which would be clearly impossible to implement.

The value of the RMSEE between the estimation $\hat{z}$ performed with MATLAB in floating point precision and the

data z in the validation set is equal to 0.0646, which is less than 0.2% of the interval where the values of v_y are defined.

6.2 *Circuit implementation*

The VHDL files defining the circuit architecture described in Section 2.3 for the FPGA implementation of the lateral velocity estimator were generated in an automated way by using the previously described toolchain. A serial architecture was chosen, since the sampling time of the system (100 ms) is quite high if compared to the circuit latency (see Section 6.3) and therefore the circuit with lowest complexity was preferred. The selection of the best values for the number of bits to code inputs (*N_BIT*) and coefficients (*N_BIT_COEFF*) was performed by analysing the results of several simulations, carried out as described below.

A Simulink model was created in which a Xilinx System Generator *Black Box* block simulates the VHDL files describing the virtual sensor architecture. This block allows taking into account the effects of the fixed point representation and of the circuit delays.

The samples of the measurable system inputs and outputs related to all maneuvers in the validation data set are loaded from the MATLAB workspace, scaled (with an affine transformation) and provided to the *Black Box* block, together with a *reset* signal, which is activated at the beginning of each maneuver, in order to avoid that different maneuvers influence each other. The *Black Box* block estimates variable z by simulating the digital circuit, whose output is scaled back with coefficients α and β and saved to MATLAB workspace as variable z_{cir}. The estimates provided by the circuit are compared to the exact measurements of the validation set (containing T_v elements) and two kinds of global errors are evaluated, the RMSEE(z, z_{cir}) (see (11)) and the relative error RE(z, z_{cir}), defined as follows:

$$\text{RE}(z, z_{\text{cir}}) = \frac{\sum_{t=T_i}^{T-1} |z(t) - z_{\text{cir}}(t)|}{\sum_{t=1}^{T_v} |z(t)|}$$

The target FPGA for the implementation of the DVS is a Virtex5 XC5VLX50T, which is equipped with 18 $\times$ 25 bit multipliers. Therefore, the pairs *N_BIT*, *N_BIT_COEFF* are chosen in order not to exceed the multipliers' range. We performed several simulations by varying the number of bits for representing inputs and coefficients and for each of them we measured the two aforementioned errors, together with the resource occupation (number of registers and look-up tables) of the selected FPGA. A summary of the main results is shown in Table 2.

It can be noticed that the percentage of used FPGA registers exhibits negligible variations, while the percentage of used FPGA LUTs mainly depends on *N_BIT_COEFF*. Moreover, the estimation errors are most of all influenced by *N_BIT*. Since the errors have significant drops until *N_BIT* = 18 we decided to fix *N_BIT* = 18 and *N_BIT_COEFF* = 12 and to use the obtained circuit architecture to perform the HIL simulation of the virtual sensor described next.

Table 2. Estimation errors and FPGA resources exploitation by varying the number of bits for inputs and coefficients.

N_BIT	N_BIT_COEFF	RMSEE	RE	Regs	LUTs
12	25	3.1372	0.9681	3%	76%
14	12	0.9469	0.2679	3%	37%
14	18	0.9353	0.2646	3%	57%
14	25	0.9345	0.2645	3%	78%
16	12	0.2366	0.0674	3%	39%
16	18	0.2364	0.0654	3%	59%
16	25	0.2360	0.0653	4%	80%
18	12	0.1086	0.0250	3%	40%
18	18	0.1118	0.0237	4%	60%
18	25	0.1114	0.0237	4%	82%
20	12	0.1015	0.0183	4%	41%
20	18	0.1049	0.0160	4%	62%
22	12	0.1015	0.0174	4%	44%
22	18	0.1063	0.0150	5%	64%
25	12	0.1054	0.0159	5%	51%
25	18	0.1068	0.0149	5%	66%

So far, to the best of the authors' knowledge, no automatic procedure for tuning the circuit parameters (N_BIT, N_BIT_COEFF) or the RC-DVS structure (M_u, M_y, M_z, etc.) has been proposed.

6.3 *Hardware-in-the-loop simulation*

A Digilent Genesys Development Board equipped with a Xilinx Virtex5 XC5VLX50T FPGA was programmed with the virtual sensor architecture and connected to a PC via USB cable. The board is equipped with a 50 MHz clock, thus allowing to obtain a circuit latency of 220 ns, which is much lower than the circuit sampling time (100 ms). The estimated power consumption of the circuit resulted being of 600 mW. A Simulink model for the HIL simulation of the virtual sensor was designed. The model is analogous to the one described in Section 6.2, only the *Black Box* has been replaced by a block (named *Virtex5*), which allows sending inputs to and retrieving outputs from the FPGA physically connected to the PC. This allows to validate in a very easy way the functioning of the designed circuit.

In order to test the effect of noise-corrupted data on our circuit implementation of the RC-DVS, a Gaussian noise with zero mean and different standard deviations was summed to the validation data coming from the vehicle simulator. We call $\tilde{z}_{\text{cir}}$ the estimation performed by the FPGA considering this noise and z_{cir} the estimation obtained with nominal inputs. Figure 4 shows the mean value $\bar{e}$ and standard deviation σ_e of the error $z_{\text{cir}} - \tilde{z}_{\text{cir}}$, with respect to the standard deviation σ_n of the Gaussian noise. It can be

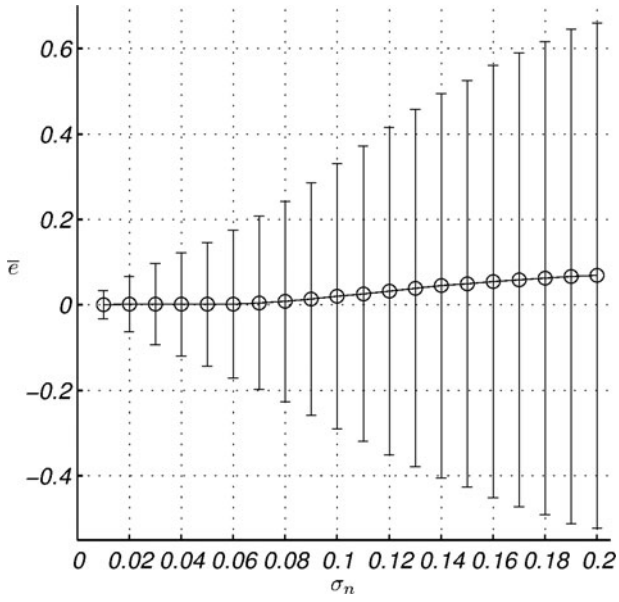

Figure 4. Mean value and standard deviation of the error $z_{\text{cir}} - \tilde{z}_{\text{cir}}$ by varying the standard deviation of the Gaussian noise.

noticed that the mean value remains almost zero till $\sigma_n = 0.06$; for higher values of the standard deviations an offset appears in the estimate.

Figure 5 shows z_{cir} compared to the real data (z) for some maneuvers taken from the three categories (step steer, sine steer, lane change). The plots are related to a Gaussian noise with standard deviation $\sigma_n = 0.05$ and show that the virtual sensor has an acceptable degree of robustness with respect to the noise introduced in the simulation. The local errors $e = z - z_{\text{cir}}$ and $\tilde{e} = z - \tilde{z}_{\text{cir}}$ on the 57 maneuvers in the validation set are shown in Figure 6.

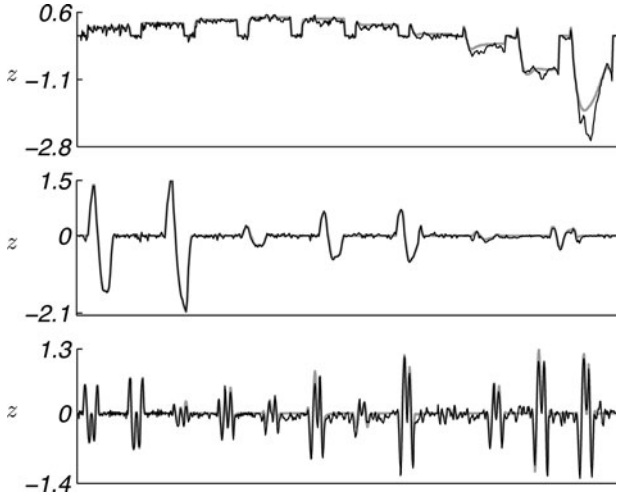

Figure 5. Real data values (grey line) and estimation performed by the FPGA (black thin line) related to step steer (top panel), sine steer (middle panel) and lane change (bottom panel) maneuvers.

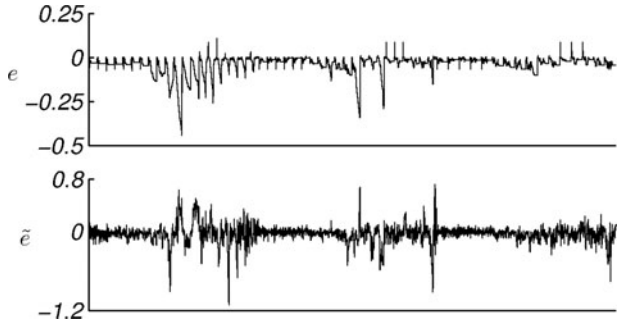

Figure 6. Estimation error related to all maneuvers between the true data and the values computed by the circuit with (bottom panel) and without (top panel) added Gaussian noise (with null mean and standard deviation $\sigma_n = 0.05$).

7. Conclusions

This paper has described the design, the theoretical properties and the FPGA implementation of RC-DVSs in PWAS form. The contribution of this work relies on the following aspects. First, the proposed approach overcomes the curse of dimensionality observed in Poggi et al. (2012), and is proved to be a minimum-variance estimator among all the virtual sensors with the same structure. Second, the proposed DVS is proved to be easily implementable on digital devices, also due to the use of an automated toolchain. Its practical implementation at very fast rates makes the approach appealing for industrial applications, mainly when unmeasurable variables of relatively low-order systems must be estimated with high sampling frequencies.

Acknowledgements

The authors would like to thank Dr Mohsen Lakehal-Ayat and Dr Urs Christen from Ford Forschungszentrum Aachen, for providing guidance, support, and data for the case study.

Funding

This work was partially supported by the European Commission under project FP7-CNECT-ICT-248858 'MOBY-DIC - Model-based synthesis of digital electronic circuits for embedded control' (http://www.mobydic-project.eu/) and by the University of Genoa.

Notes

1. The toolbox can be downloaded at http://ncas.dibe.unige.it/software/MOBY-DIC_Toolbox/.
2. VHSIC Hardware Description Language, where VHSIC stands for Very High Speed Integrated Circuit.
3. One of these multipliers is used by block *SCALE_z*.
4. http://ncas.dibe.unige.it/software/MOBY-DIC_Toolbox/

References

Bellman, R.E. (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Canale, M., Fagiano, L., Ruiz, F., & Signorile, M. (2008, December). *A study on the use of virtual sensors in vehicle control*. Paper presented at the 47th IEEE Conference on Decision and Control, Cancun, Mexico.

Castro, L.R., Agamennoni, O.E., & Álvarez, M.P. (2010). From linear to nonlinear identification: One step at a time. *Mathematical and Computer Modelling, 51*, 473–486.

Di Federico, M., Poggi, T., Julián, P., & Storace, M. (2010). Integrated circuit implementation of multi-dimensional piecewise-linear functions. *Digital Signal Processing, 20*, 1723–1732.

Echevarria, P., Martínez, M., Echanobe, J., del Campo, I., & Tarela, J.M. (2007). Digital hardware implementation of high dimensional fuzzy systems. In F. Masulli, S. Mitra, & G. Pasi (Eds.), *Applications of fuzzy sets theory*, Lecture notes in computer science (pp. 245–252). Berlin: Springer.

Farrelly, J., & Wellstead, P. (1996). Estimation of vehicle lateral velocity. In *IEEE International Conference on Control Applications* (pp. 552–557). Dearborn, MI.

Julián, P., Desages, A., & D'Amico, B. (2000). Orthonormal high level canonical PWL functions with applications to model reduction. *IEEE Transactions on Circuits and Systems I, 47*, 702–712.

Ljung, L. (1978). Convergence analysis of parametric identification methods. *IEEE Transactions on Automatic Control, 23*, 770–783.

Milanese, M., Novara, C., Hsu, K., & Poolla, K. (2006, June). *Filter design from data: Direct vs. two-step approach*. Paper presented at the Proceedings of the American Control Conference, Minneapolis, MN.

Milanese, M., Novara, C., Hsu, K., & Poolla, K. (2009). The filter design from data (FD2) problem: Nonlinear set membership approach. *Automatica, 45*, 2350–2357.

Novara, C., Fagiano, L., & Milanese, M. (2013). Direct feedback control design for nonlinear systems. *Automatica, 49*, 849–860.

Novara, C., Ruiz, F., & Milanese, M. (2011). Direct identification of optimal SM-LPV filters and application to vehicle yaw rate estimation. *IEEE Transactions on Control Systems Technology, 19*, 5–17.

Novara, C., Ruiz, F., & Milanese, M. (2013). Direct filtering: A new approach to optimal filter design for nonlinear systems. *IEEE Transactions on Automatic Control, 58*, 86–99.

Oliveri, A., Barcelli, D., Bemporad, A., Genuit, B.A.G., Heemels, W.P.M.H., Poggi, T., … Storace, M. (2012). MOBY-DIC: A Matlab toolbox for the circuit design of explicit MPC. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control* (pp. 218–225). Noordwijkerhout, the Netherlands.

Poggi, T., Rubagotti, M., Bemporad, A., & Storace, M. (2012). High-speed affine virtual sensors. *IEEE Transactions on Industrial Electronics, 56*, 1228–1237.

Repetto, L., Storace, M., & Parodi, M. (2003). A method for the approximate synthesis of cellular non-linear networks – Part 2: Circuit reduction. *International Journal of Circuit Theory and Applications, 31*, 299–313.

Rovatti, R., Fantuzzi, C., & Simani, S. (2000). High-speed DSP-based implementation of piecewise-affine and piecewise-quadratic fuzzy systems. *Signal Processing, 80*, 951–963.

Rubagotti, M., Poggi, T., Bemporad, A., & Storace, M. (2012). Piecewise-affine direct virtual sensors with reduced complexity. In *51st IEEE Conference on Decision and Control* (pp. 656–661). Maui, HI.

Ruiz, F., Taragna, M., & Milanese, M. (2009). Direct data-driven filter design for automotive controlled suspensions. In *Proceedings of the European Control Conference* (pp. 4416–4421). Budapest, Hungary.

Storace, M., Repetto, L., & Parodi, M. (2003). A method for the approximate synthesis of cellular non-linear networks – Part 1: Circuit definition. *International Journal of Circuit Theory and Applications, 31*, 277–297.

Storace, M., & Poggi, T. (2011). Digital architectures realising piecewise-linear multivariate functions: Two FPGA implementations. *International Journal of Circuit Theory and Applications, 39*, 1–15.