ABSTRACT: We combined adaptive filters associated with power estimators to characterize the knock signal, obtained from a knock sensor, in an internal combustion engine. The filters were implemented using the automotive model-based design methodology, and the resulting software was embedded into hardware for a real-time evaluation of the proposed solution. The knock signals could be qualitatively identified in real-time, and thus have the potential to aid in the management of flexible-fuel engines. This approach has an extensive range of applications within the automotive industry, since it can be implemented within any model-based control strategy. For example, this methodology can be applied in commercial ECUs, currently used in most vehicles for knock detection, by simply eliminating an internal, dedicated, integrated circuit for knock identification, or by serving as a redundancy device (i.e.: for safety purposes). Finally, this methodology can identify the knock signal, obtained from a knock sensor, just by using an algorithm implemented in the ECU’s processor, which we show is identifiable in real-time.

INDEX TERMS: Adaptive filters, automotive electronics, model-based development, knock sensor.

I. INTRODUCTION

Internal combustion engines are complex machines present in a large number of modern systems, such as automobiles, aircraft, heavy-duty machinery, and power generators. They require a control system that guarantees proper operating conditions in order to provide optimized power output, safety, fuel efficiency, and compliance with strict safety and environmental regulations. Internal combustion engines are generally managed by an electronic control unit (ECU), which must manage, in real-time, several input and output parameters.

In automotive applications, one of the key parameters in engine control is the spark instant, which ignites the air/fuel mixture. Optimization of the appropriate instant for the spark provides a maximum pressure over the piston, leading to a maximum torque output. Advancing the instant of the spark, with respect to the one in which the piston reaches the top position, i.e.: top dead center (TDC) inside the cylinder, is generally used to obtain maximum power output, (which occurs within 8 to 10 degrees after TDC).

There is a limitation to that strategy, however. If the spark occurs too early, the flame front of the ignited air/fuel mixture will reach the piston too early, (i.e.: before TDC, as shown by [1]), it can be within 23 to 30 degrees before TDC depending on the engine and its conditions) and some parts of the mixture may self ignite generating a new flame front, which collides with the original one. This causes shock waves that could reach the natural frequency of the engine chamber, which would produce a strong vibration. This phenomenon, known as “knock”, generates a characteristic sound. The occurrence of knock is an indicator that the optimal spark instant was not found, and it is necessary to reduce the spark advance to achieve optimal performance. The knock can also occur when there is a spontaneous self-combustion inside the chamber, which may be the result of an occasional spark due to sediments on the spark plugs [2]. The knock, with a characteristic strong mechanical vibration in a certain frequency range, is generally identified in automobiles with a knock sensor [3], which is a piezoelectric device that converts the knock vibration into electrical signals. Many techniques have been used to identify this phenomenon with a measurement of a noisy mechanical signal [4]–[6].
When the knock occurs, the electrical signal of the sensor is received and identified by an integrated circuit (IC) that forwards an analog or digital signal to a processing unit—either a microcontroller or a microprocessor. Then, that unit actuates to reduce the spark advance angle until the knock disappears. One of the problems with this detection technique (using a knock sensor) is that other noisy signals, that are uncorrelated to the knock but that have similar frequencies, could be present in the engine, which may cause a false-positive knock identification by the knock sensor. For this reason, only when the engine is under certain working conditions, which were prespecified during the development phase, does the ECU microprocessor consider the incoming signal from the IC for knock detection.

Several studies have been carried out to identify an optimal ignition advance to increase engine efficiency, and the knock characterization and detection is an important component to achieve it. Many studies have shown that the occurrence of knock depends on engine thermodynamics and the chemical composition of the fuel [7]–[10]. For example, one study found that the inlet air temperature combined with direct ethanol injection can be used to reduce knock occurrence and hence enable further advance of the spark timing [11]. Other studies have explored how such information could be introduced in a control framework by using integrated signal detection [12] based on either knock distribution estimation [13], [14], the distribution of the phenomenon’s maximum amplitude of the pressure oscillation (MAPO) [15], or even integrating a model of knock occurrence using probabilistic approaches to tune a control framework [16]. The knock events have been modeled as probability density functions of the spark angle advance to determine the best threshold for knock avoidance. In this way, determination of the optimal threshold for the crank angle advance can be integrated into a control strategy [17].

The proposed threshold has been tested in a mock-up setup using model-based tools (ETAS INTECRIOS) and Simulink, and a similar approach has been applied to pressure sensor signals by [18]. Probabilistic modeling of knock occurrence has also been applied by [19]. The authors used different input parameters related to the engine such as mixture fraction, enthalpy and temperature of unburned gases. Time-frequency analysis of the knock has also been extensively explored. It may be used as either a pre-processing tool [4], [15], [20], or in combination with pattern recognition techniques [21], given that the knock phenomenon presents a typical frequency range, although this range varies for different engine manufacturers [5], [22]. In a frequency-related analysis, an experiment [23] has utilized wavelets as a tool for denoising the knock signal and the empirical mode decomposition, which essentially extracts intrinsic mode functions of the signal of interest (i.e.: they consist of oscillation modes of time varying frequencies that can represent local characteristics of non-stationary signals. They can also be interpreted as an exploratory basis expansion of a signal) for knock detection in a mock-up gasoline engine. Frequency-based decomposition has also been used to clean up and characterize knock signals using a mock-up engine setup [24]. Additionally, some of the aforementioned methods have been used in embedded real-time framework [13], [14], [17]–[19], [23]–[25].

Since the knock detection is more reliable using an in-cylinder pressure sensor, most authors have used the knock sensor as the only source to identify it [14]. Those investigations focused on extracting metrics for knock detection based on in-cylinder pressure sensors [26]. Time-frequency has also been utilized to develop new metrics of knock detection, providing a binary output after analysis [27].

Here, we propose an alternative real-time software-based method to distinguish different characteristics of the knock occurrence. The knock phenomenon is characterized by signal processing techniques (adaptive and nonlinear filters) that assume signal acquisition from a knock sensor: a piezoelectric accelerometer, with one degree of freedom, which converts the mechanical vibrations of the engine’s block to an electric signal. This software was implemented in an embedded hardware, in order to demonstrate its feasibility in real-time. If feasible, our approach could allow the elimination of the dedicated IC especially used for such function, in currently commercialized ECUs. This methodology also offers a redundancy device for the ECU (if the dedicated IC is maintained), which could be used for safety purposes. The proposed method can also distinguish between different intensities, which allows one to characterize the duration of the knock signal. The method was fine-tuned and validated in simulated knock scenarios using a modeled knock signal with additive white Gaussian noise (AWGN) [28]. We achieved good real-time characterization in such scenarios by combining adaptive filters (AF) with power estimators. The adaptive filter can emphasize knock occurrence by distinguishing knock statistics from background AWGN ones. The power estimator can smooth the filter output, providing clear evidence of the knock occurrence.

The main application is for engines with a flexible-fuel technology that works with any ethanol and gasoline mixture. In a vehicle with such technology, the relevance of knock characterization is even higher than in vehicles with pure gasoline, since the maximum power output (i.e.: torque) is highly dependent on the fuel mixture, especially when submitted to higher compression rates. A proper knock characterization eases the signal detection, enabling the ECU to manage precisely the proper ignition timing, which leads to an optimized power output for any fuel mixture. The knock characterization/detection strategy is only a small part of the whole engine management algorithm, which is comprised of several steps of modeling, integration, code development, calibration, and validation.

This paper is organized as follows: Section II presents the background used in this investigation, which includes the definition of the knock phenomenon and an introduction to the signal processing algorithms. Section III presents the methodologies used to implement the software pipeline; Section IV presents some details of this implementation;
Section V presents simulations to evaluate the proposed algorithms under several scenarios; and Section VI presents the results in an embedded real-time system.

II. BACKGROUND

A. THE KNOCK PHENOMENON

The knock phenomenon is an abnormal combustion and represents an intrinsic limitation on engine performance. Therefore, the knock represents a limiter of torque exploration, which could allow further or minor torque request depending on several factors, such as the chamber geometry and fuel quality.

Several factors can cause the knock, which leads to a non-stationary electrical signal. When the knock occurs, it has an energy peak followed by a damping effect until it reaches the background noise level of the sensor. Therefore, we modeled the knock signal as the impulsive response of a second-order under-damped system with a background white noise. That damped sinusoid has a statistical pattern that is considerably different from white noise, which justifies the use of statistical filters for the knock characterization. In order to characterize this signal, we used three types of estimators: the power estimator, the Least Mean Square (LMS), and the Normalized Least Mean Square (NLMS). The last two estimators are stochastic estimators, also referred to as adaptive filters.

B. ADAPTIVE FILTERS

The adaptive filters used here are recursive stochastic methods derived from gradient descent and Newton’s method. These algorithms find, at each iteration $i$, a vector of coefficients $\hat{w}_i$ that minimizes the estimation error (also referred to as cost function) defined as:

$$ J(\hat{w}) = E[\hat{d} - \hat{d}]^2 = E[\hat{y} - y]^2, $$

where $\hat{d}$ is the measured value (scalar random variable), $E$ is the expected value operator, and $J(\hat{w})$ is the cost function. Here, random variables are represented as bold letters. The term $\hat{d}$ is an estimation of the measured value, defined as:

$$ \hat{d} = \sum_i u_i \hat{w}_i $$

in which $\hat{w}_i$ is a column vector and $u_i$ is a random row vector, both of size $M$.

All estimation problem aim to get a cost function to its minimal possible value. It can be shown that the minimum value of $J(\hat{w})$ can be obtained with [29]:

$$ J_{\text{min}} = \sigma^2 - R_{ud} R_u^{-1} R_{du}, $$

where $R_{ud} = E\hat{u}\hat{d}^*$ is a row vector, $R_{du} = E\hat{d}\hat{u}^*$ is a column vector representing the cross-covariance between the observation (or filter output) vector $\hat{u}$ and the input filter input $\hat{d}$. Finally, $R_u^{-1}$ is the inverse covariance matrix, which is a square matrix of the observed values. The notation “$*” refers to the complex-conjugate transposition and $\sigma^2$ is the variance of the measured value.

The cost function minimal value, Eq. (3), is obtained by applying an optimal coefficient vector, denoted by $\hat{w}_i$, in the cost function, Eq. (1). The optimal coefficient $\hat{w}_i$ is the solution to the linear estimator model, also known as the normal equation:

$$ R_u \hat{w} = R_{du}. $$

If there exists an inverse to $R_u$, the optimal coefficient vector can be obtained by:

$$ \hat{w} = R_u^{-1} R_{du}. $$

In an iterative/recursive form, and for a given cost function, the optimal coefficient vector can be approximated by using the steepest descent algorithm, defined as:

$$ w_i = w_{i-1} + \mu[R_{du} - R_u w_{i-1}], $$

where $\mu$ is the step size. The term “approximate” was used since the steepest descent convergence to $\hat{w}$ is only achieved as $i \to \infty$ and under certain choices of the step size $\mu$, as discussed later. Therefore, for the algorithm in Eq. (6), $w_i$ is “approximately” converging to the closest possible value to $\hat{w}$, under a finite number of iterations and, consequently, $w_i$ is within a certain range from the minimum, which could be either local or global. By setting $\tilde{w}_i = \hat{w} - w_i$, which is the error distance between $\hat{w}_i$ and the optimal vector of coefficients $\hat{w}_i$ at instant $i$, and by assuming that $R_{du}$ can be represented by $R_u \hat{w}$, as presented in Eq. (4), it can be shown [29] that:

$$ \tilde{w}_i = [I - \mu R_u] \tilde{w}_{i-1}, $$

where $I$ is the identity matrix (of appropriate dimensions) and the equation inside the brackets serves to illustrate its connection with the convergence speed of the algorithm given in Eq. (6)– either from $w_1$ to $\hat{w}$ or $J(w)$ to $J_{\text{min}}$. Each eigenvalue of $I - \mu R_u$ determines a convergence mode of Eq. (6), such that each convergence mode can be represented by $1 - \mu \lambda_k$, with $k > 0$ and $k \in \mathbb{N}$. The smaller the convergence mode, the faster the convergence of $w_i$ towards $\hat{w}$ and, consequently, from $J(w)$ towards $J_{\text{min}}$. The step size, $\mu$, is intrinsically connected to the algorithm convergence speed and is bounded to $R_u$ eigenvalues because, as shown in [29], $\mu$ must ensure convergence, with:

$$ 0 < \mu < 2/\lambda_{\text{max}}. $$

An indication of the existence of Eq. (8) is that the eigenvalues of $I - \mu \lambda_k$ are within the unitary circle.

The vector of coefficients $w$ can also be obtained from the Newton’s method [29], which is derived from the steepest descent algorithm, defined as:

$$ w_i = w_{i-1} + \mu[I + R_u]^{-1} [R_{du} - R_u w_{i-1}], $$

where $\epsilon$ is a small positive value for adjustments. The Newton’s method, unlike the gradient descent, does not depend on values of $R_u$ to make the error, $\tilde{w}$, converge to zero, where $\tilde{w} = \hat{w} - w_i$. The method only depends on the step
size value, $\mu$. As shown in [29], if the condition $0 < \mu < 2$ is satisfied, $\tilde{w} \to 0$ is guaranteed for this algorithm, but again under the premise that $i \to \infty$. Newton’s algorithm also differs from gradient descent in the number of convergence modes. Unlike gradient descent, Newton’s method has only one convergence mode, which is for a quadratic cost function. The convergence mode depends on the step size and is defined by:

$$1 - \mu$$

for the given cost functions. Since Newton’s method reaches the optimal value of $w^\circ$ faster than gradient descent, the minimal estimated error is also reached faster. In real applications, the statistics of the signal of interest are rarely known, even though they are required by the steepest descent algorithm (because $R_{du}$ and $R_u$ are deterministic). Therefore, the solution presented in Eq. (4) cannot be reached in practice. Nonetheless, a few stochastic algorithms were created based on the steepest descent algorithm. Such algorithms were conceived under stochastic approximations of $R_{du}$ and $R_u$ to outline no knowledge on signal statistics. These approximations are given by [29]:

$$\tilde{R}_{du} = d(i)u_i^2, \text{ and}$$

$$\tilde{R}_u = u_i^* u_i,$$

(11)

where $u_i$ represents a vector of $M$ measurements prior to the $i$ instant of $u(i)$, where $u_i = \{u(i-1), u(i-2), u(i-3), \ldots u(i-M+1)\}$ is a regressor vector and $d(i)$ is a scalar, which is the value of $d$ at the instant $i$. The approaches based on (11) generate the algorithms used in this paper, namely the LMS and NLMS.

1) LEAST MEAN SQUARE (LMS) ALGORITHM
The LMS algorithm is based on the gradient descent algorithm presented in Eq. (6). After the application of stochastic approximations, defined in Eq. (11), the following algorithm is obtained:

$$w_i = w_{i-1} + \mu u_i^*(d(i) - u_i w_{i-1}).$$

(12)

The LMS inherits all of the characteristics of the gradient descent algorithm, such as a high dependency on input values for convergence. On the other hand, this algorithm is computationally simple and easy to implement.

2) NORMALIZED LEAST MEAN SQUARE (NLMS) ALGORITHM
The NLMS algorithm is derived by applying Eq. (11) to Newton’s method from Eq. (9). With a few algebraic manipulations [29], one obtains:

$$w_i = w_{i-1} + \mu \frac{u_i}{|u_i|^2} (d(i) - u_i w_{i-1}).$$

(13)

Just as the LMS inherits the gradient descent characteristics, the NLMS inherits the characteristics from Newton’s method such as the fast tracking of the signal to be estimated, which is a result of the convergence rate depending only on the step size.

C. ADAPTIVE PREDICTION
When there is a narrow band signal, $s(i)$, embedded in a broadband signal, $v$ (for example, white noise), where $i$ is the iteration or temporal realizations, adaptive predictors can be used to separate them [30]. An adaptive predictor (AP) is comprised of a filtering structure, usually a transverse finite impulse response (FIR) filter, and a learning algorithm. At instant $i$, a FIR filter is represented by its response to the impulse (truncated) captured in the column vector $w_i$.

In order to separate $s(i)$ from $v(i)$, the AP tries to predict the trend of the reference signal, represented by $u(i)$, through a linear combination of previous samples, i.e.:

$$\hat{u}(i) = u_{i-1} w_{i-1}.$$  

More explicitly, the linear combination of past samples is represented by:

$$u_{i-1} = [u(i-1) \ldots u(i-M+1)].$$

which is the line regression vector, and $w_{i-1}$ is the response to the predictor filter impulse at instant $i - 1$.

The vector $w_{i-1}$ is designed to minimize the prediction error:

$$e(i) = u(i) - \hat{u}(i).$$

The optimal predictor, $w^\circ$, can be found using the normal Eq. (4) [31]:

$$R_{du} w^\circ = R_{du i},$$

where $R_u \cong E u^* u$ and $R_{du} \cong E d^* u$ are the signal statistics, as shown in the previous section. The index $i$ indicates an iterative calculation of $w^\circ$ for each $R_u, R_{du}$ at each time $i$.

In the prediction case, $d \leftarrow u(i)$ and $u \leftarrow u_{i-1}$.

D. POWER ESTIMATOR
It is well known that the energy $E$ of a signal $x(i)$ can be defined as [32]:

$$E = \sum_{-\infty}^{\infty} x(i)^2$$

(14)

and the definition of the instantaneous power $p(i)$ is given by [32]:

$$p(i) = x(i)^2.$$  

(15)

The application of the formula in Eq. (14) is not possible because the sum is infinite. A more practical algorithm would be $y(i) = y(i-1) + x(i)^2$, which calculates the energy produced since $i = 0$, but this algorithm is unstable. A better choice would be:

$$y(i) = \alpha y(i-1) + \beta x(i)^2,$$

(16)

in which $0 < \alpha < 1$, and $\alpha = 1 - \beta$. If $\beta$ is close to one, $y(i)$ is approximately the instantaneous power. If $\alpha$ is larger than $\beta$ (which should not be zero), the main pattern shown in the realization of $y$ will be the developed attenuated energy of
the applied signal \( x(i) \). That is, the signal \( y(i) \) is more energy-like. The non-linearity (power two) attenuates relatively small values of \( x(i) \). Nevertheless, one can realize that this algorithm, in principle, may not be so useful for knock events with amplitude values close to the sensor background values or its noise. Lastly, from Eq. (16), the higher the \( \alpha \) the more energy emphasis is given to the estimator and, consequently, the lower the \( \alpha \) the more emphasis is given to instantaneous power. Therefore, the higher the \( \alpha \), the slower the response of the power estimator to changes in the signal \( x(i) \). So, for a faster response, \( \alpha \) has to be small. We propose a method in which the adaptive filtering is combined with power estimation, in order to distinguish the different knock levels as well as the different time duration of knock events.

### III. THE V-MODEL DESIGN AND IMPLEMENTATION SCHEME

In order to better explain how the contributions of this paper were obtained and validated, we explain in this section the design paradigms, known as the “V”-model and model-based development, and in section IV we explain how they were used in this research.

The design and implementation of a complex system, such as the one developed here, can be divided in many ways. Typically, after the specification of requirements, a logical architecture design, composed of function blocks that interact with each other, is done. Then, the technical system architecture (hardware and communication infrastructure) is proposed. Those steps form the system development phase. After that, the software development phase begins, in which the requirements are translated from the system development phase, the architecture is proposed, and software components are designed and coded (see the central part of Fig. 1).

Integration of components, acceptance tests, and verification and validation must be performed. The V-Model system development methodology, shown in Fig. 1, represents a way to organize the development cycle such that the entire system life cycle can be more efficient and controllable. The left side of the “V” corresponds to the design process, in different abstraction levels, ranging from the requirements to the software implementation. The right side corresponds to the integration and test phase, which can consist of a more detailed phase such as testing of software components or more abstract ones that employ integration of components.

On the left side of the “V”, after defining the specifications and requirements of the system, models of the plant and controllers are obtained and simulated. Maybe some corrections could be executed at that level. All this is executed on a virtual level. The appropriate control hardware is selected, and the code is generated and uploaded. All of the parts are integrated and tested up to the acceptance tests (on the right side). This methodology is adequate for the development of complex systems and complies with the ASAM MCD-2 MC automotive standard used to define the description formats of ECU’s internal variables utilized for measurement and calibration [34].

#### A. MODEL-BASED DEVELOPMENT

The Model-Based Development (MBD) is a method of developing software in a modular form, making the application level of software independent of the hardware. This development is only possible because the automotive software is divided into a stack of layers, such that each layer provides services to the level above by standard function calls. In this sense, the ECU’s RTOS (real-time operating system, such as RTA-OSEK) provides system calls to the applications, as well as services of basic software (BSW) that has the software drivers of input and output, which includes complex drivers (such as the injector). In the following we describe, using the V-model, the steps that would be followed by this kind of project, which is based on the model-based development (see section IV, and Fig. 2).

#### B. FUNCTION GENERATION

Function generation refers to the development of the knock detection function to be embedded in the ECU, corresponding to the red part of Fig. 2. Initially, taking into account only the logic behavior, this could be executed using block diagrams, programming languages, embedded system description language (ESDL), or even state machines. Those blocks
communicate with virtual inputs and outputs that are simulated in real or virtual time (generally at the same computer). Examples of such function blocks can be found in Figs. 3 and 14.

C. SOFTWARE DEVELOPMENT

In the software development phase, the automatic code generation is performed. This consists of compiling the block diagrams created in the previous phase in order to generate the A2L file (an ECU description file) and a HEX file (an ECU program), which is often used by the automotive industry. This corresponds to the blue part of the “V” in Fig. 2.

D. VIRTUAL TEST AND VALIDATION

In the virtual test and validation phase, the detection algorithm (function) is generated in the ECU, which is connected to a computer running a real-time operational system (RTOS). It works along with the ECU in a closed-loop configuration, running a virtual plant (in this case, the knock signal generator) that allows for a complete simulation of the real environment in which the ECU will be applied. The computer must be equipped with an analog-to-digital (and digital-to-analog) converter, as well as other signal conditioners, in order to generate and receive analog signals that can be directly exchanged with the ECU as if it was reading sensors in the real car. This type of computer is generally called a HIL (hardware in the loop) plant, and it can also simulate sensors and actuators models, as well as set-points from the car user.

E. MEASUREMENT AND CALIBRATION

The last phase of the V-Model corresponds to the test and validation phase using the real plant, i.e. the ECU is connected to a vehicle, where the final task of calibration can be executed. The calibration is executed by accessing the ECU through automotive protocols for measurement and calibration, such as CAN, CCP, or ETK.

Those protocols allow access to the command set, in order to: obtain values from the measurement variables, write values in the maps and calibration parameters online, and change these maps while the real plant (e.g.: engine and transmission) operates without needing ECU reprogramming. In this phase adjustments related to fuel, sensors, drivability, mileage, comfort, and others, are performed with no need for compiling another code.

IV. IMPLEMENTATION OF THE ADAPTIVE FILTER USING V-MODEL

The primary goals of this project are:
- Generate the knock phenomenon in a test vehicle;
- Collect the knock signals accordingly; and
- Process the acquired signal using a specific hardware with the implemented solution in order to characterize the knock phenomenon.

A. SELECTION AND USE OF SIGNAL PROCESSING ALGORITHMS

In Section II-B, the proposed algorithms were introduced to solve the problem discussed in Section II-A. The use of adaptive filters must be justified, since they may carry higher computational complexity, as compared to power estimators. The use of adaptive filters is explained by the AP setting, as described in Section II-B.

Considering Eq. (5), one can establish two operation scenarios for the proposed algorithms in the context of knock occurrence: (a) normal operation and (b) knock.

- (a) Normal Operation: \( u(i) = v(i) \).
  In this scenario, since \( v(i) \) is white noise and \( u(i) \) is the resulting sensor signal, \( R_u = E u^*u = \sigma_v^2 I \) and \( R_{du} = E u(i)u_{i-1} = 0 \). Therefore, \( w_e^0 = 0_{M \times 1} \), i.e. is numerically zero.

- (b) Knock: \( u(i) = s(i) + v(i) \).
  In this scenario, damped oscillation measured in the sensor due to knocking is assumed to be a correlated

FIGURE 3. NLMS execution block, considering Eq. (13).
signal that is embedded in AWGN, such that:

\[ R_u = E u^* u \]
\[ = s_{i-1}^* s_{i-1} + \sigma_v^2 I \]
\[ R_{du} = E u(i) u_{i-1}^* \]
\[ = s(i) s_{i-1}^*, \]

(17)

with \( s_{i-1} = [s(i-1) \ldots s(i-M)] \) being a regressor and \( v_{i-1} = [v(i-1) \ldots v(i-M)] \). Equation (5) requires the inversion of \( R_u \) and by applying the matrix inversion lemma \[29, 31\] in \( R_u \):

\[ R_u^{-1} = \frac{1}{\sigma_v^2} I - \frac{1}{\sigma_v^2} s_{i-1}^* \left( \frac{1}{\sigma_v^2 + s_{i-1} s_{i-1}^*} \right) s_{i-1}. \]

(18)

Applying such results in Eq. (5), as previously used in \[31\], and combining it with Eq. (17), implies that:

\[ w_o^* = \frac{s_{i-1} s_{i-1}^*}{\sigma_v^2} \left( 1 - \frac{s_{i-1} s_{i-1}^*}{\sigma_v^2 - s_{i-1} s_{i-1}^*} \right). \]

(19)

Let \( c = s_{i-1} s_{i-1}^* \) so that for each measurement:

\[ w_i^o = \frac{[c]}{\sigma_v^2} \left( 1 - \frac{c}{\sigma_v^2 - c} \right) \neq 0 \]

(20)

i.e.: the obtained \( w_i^o \) is numerically different from zero.

This property can be efficiently explored to detect the knock phenomenon by establishing a certain threshold (THR) at which the optimal predictor components \( w_i^o \) must be checked at each iteration \( i \) and compared to the threshold. Considering that \([w_i^o]_n\) is the \( n^{th} \) vector component of \( w_i^o \), if \([w_i^o]_n \geq (\text{THR})\), a knock occurred, while if \([w_i^o]_n < (\text{THR} > 0)\) the engine is said to be operating normally. In order to better characterize the phenomenon, the entry \([w_i^o]_1\) is chosen for this comparison (that is, \( n = 1 \)), since the first entry of \( w_i^o \) tends to be the largest among all \( n \) components because of the \( s(i) \) correlation to itself \[35\]. Here, the threshold THR was determined empirically, but other investigations have explored finding it statistically \[5, 12, 15, 20, 21, 25\].

The next step is to efficiently design \( w_i^o \). This design is complex since the signal parameters \( s(i) \) are not known a priori, as discussed in Section II-B. Therefore, the normal equations must be solved in an approximate form to adaptive algorithms (section II-B) \[29\]. With an initial estimate \( w_{i-1} \), \( w_i^o \) can be updated (optimized) in an adaptive manner, as:

\[ w_i = w_{i-1} + \mu p_i, \]

(21)

where \( p_i \) is a directional update vector. Eq. (21) is the reference to the gradient descent algorithm, shown in Section II-B. If \( p_i \) is chosen according to the stochastic law of gradient descent algorithm, the LMS and NLMS algorithms can be obtained, as shown in section II-B \[29\].

Both algorithms were successfully tested. If \( \mu \) is correctly chosen, after some iterations, \( w_i \) will be statistically close to \( w_i^o \). If a certain threshold, THR, is carefully chosen, the adaptive algorithms can identify the knock. The only difference is that the tests were conducted with \( w_i \), instead of \( w_i^o \), as \( w_i^o \) is unknown. Once the knock peak is detected, the time taken to return to THR can be interpreted as an indirect way to measure the knocking signal length, or its duration.

\section*{B. SOFTWARE WORKFLOW}

The software workflow is the sequence of algorithms that are applied to the signal. The definition of this workflow is contingent to the analysis that is presented in the next section. After presenting such analysis, the software workflow is discussed in further detail along with the physical experimental results.

\section*{V. SIMULATIONS}

This section presents several evaluations (scenarios) in a simulated environment using synthetic generated signals. This section shows that the selected algorithms can characterize the signals according to their duration and intensity. Several scenarios of knock occurrence were proposed, and their respective simulations were performed in MATLAB® code is available at https://github.com/rafa-coding-projects/RTKnockChar. It is important to highlight that since the signals are simulated, the time axis in all plots under this section is fictitious and intentionally scaled to be representative of a signal having a knock event of around 7 KHz. Since the knock signal is observed from the knock sensor, it is measured in Volts. The output of the proposed digital filters is also measured in Volts.

The fast-tracking characteristic of NLMS, explained by the definition arising from Newton’s method, could be combined with the characterization obtained by the power estimator for better identification of the phenomenon. The reliability of LMS and its computational simplicity, combined with the power estimator, can result in an easy and affordable solution to an embedded system. The properties mentioned regarding adaptive algorithms have been comprehensively explored by \[29\].

The phenomenon characterization needs to reduce or filter the background noise to the smallest possible value and emphasize the behavior due to the knock as much as possible. A combination of the algorithms can even promote an enhanced response at the end. For example, adaptive predictors can remove the sensor background noise from the knock signal. Then, this property may be used with the power estimator, taking into account that the latter alone does not present a good performance when filtering such noise \[28\].

In addition to these characteristics, the adaptive algorithms offer a softened response of the knock signal in the coefficient vector \( w_i \). Therefore, both softened responses of LMS and NLMS algorithms could be inputs to the power estimator.

The use of adaptive algorithms for the initial stage of the combination is explained by the fact that they are sensitive to the signal statistics, as shown in section IV-A, contrary to the power estimator, which is a signal power indicator. The power estimator does not present a satisfactory performance in identifying the signal in a scenario in which the knock
amplitude is close to that of the background noise. However, the signal statistics change drastically due to the correlation arising from the knock. According to section IV-A, the first position of the vector \( w_i \) is the one that should be used in the power estimator [28].

Then, distinct scenarios of knock signal, applied to different combinations of the LMS, NLMS, and power estimator, are evaluated. The knock profiles are a damped sinusoid with AWGN [28]. Random intensities and occurrences of the phenomenon are generated to evaluate the algorithms in a variety of scenarios, which are:

1) Normal knock: random knock occurrence with random amplitude;

2) Critical case: subsequent occurrence of a knock with each occurrence having an amplitude higher than the previous one; and

3) Knock and background noise with near amplitudes: similar to the normal knock scenario, but with amplitude necessarily close to the background noise.

The critical case has received this name since it is a scenario in which the signal is highly correlated to itself for the most part, making the filtering more challenging.

In order to choose the parameters or tune the filter response, it is important to follow the relationship established in [29]. Under non-stationary conditions, the step size \( \mu \) must comply with the following relation for mean square convergence and stability:

\[
\sum_{k=1}^{M} \frac{\lambda_k \mu}{2 - \lambda_k \mu} < 1 \quad (22)
\]

for real values and:

\[
\frac{1}{2} \sum_{k=1}^{M} \frac{\lambda_k \mu}{1 - \lambda_k \mu} < 1 \quad (23)
\]

for complex values. Assuming \( \lambda_k \), an eigenvalue of the eigen decomposition of \( R_u \), it implies that \( \lambda_k = \sigma_u^2 \). Therefore, replacing in Eq. (22) and Eq. (23): \( M \sigma_u^2 \mu < 1 - \sigma_u^2 \mu \) (real) and \( M \sigma_u^2 \mu < 2 - 2\sigma_u^2 \mu \) (complex). Then,

\[
\mu < \frac{2}{(M+2)\sigma_u^2} \quad (24)
\]

for real values and,

\[
\mu < \frac{2}{(M+1)\sigma_u^2} \quad (25)
\]

for complex values. The Eqs. (24) and (25) show that the highest the order \( M \), the lowest the step size \( \mu \) should be.

**A. SCENARIO 1: NORMAL KNOCK OCCURRENCE**

1) LMS WITH POWER ESTIMATOR

In order to combine the power estimator with LMS using the coefficient vector \( w_i \), it has been shown that the combination of algorithms make the phenomenon characterization more efficient when compared to the individual performance [28]. The combination enabled a change in the LMS parameter setting, where the best result was obtained with \( \mu = 0.3 \) and \( M = 3 \).

Figure 4 shows that the combination of these algorithms using the LMS output practically missed some occurrences of the phenomenon with low intensity. This occurred even when increasing \( \mu \) to 1 in order to obtain faster tracking. Therefore, the best setting to scenario 1 for the LMS and power estimator is using \( w_i \) of LMS as the input of the power estimator. The result of the second attempted combination for this scenario is shown in Fig. 5.

**TABLE 1. Scenario 1, best parameter values for LMS.**

| Algorithm | Parameters |
|----------|------------|
| LMS \((w_i)\) | \( \mu = 0.3 \) and \( M = 3 \) |
| LMS \((d)\) | \( \mu = 1 \) and \( M = 4 \) |

2) NLMS WITH POWER ESTIMATOR

The best performance of NLMS was obtained with \( \mu = 0.25 \) and \( M = 2 \). The values of \( \mu = 1 \), as used before, and with \( M > 2 \) presented good performance in signal tracking, but with higher variations.

**FIGURE 4.** Scenario 1 with combined algorithms. (Bottom) The generated knock signal according to the model proposed by [28], (middle) the output \( \hat{d} \) of LMS after filtering the signal, (top) the power estimator output, with \( \hat{d} \) of LMS being its input.

**FIGURE 5.** Scenario 1 with combined algorithms. (Bottom) The generated knock signal according to the model proposed by [28], (middle) the vector of coefficients \( w_i \) of LMS after filtering the signal, (top) the power estimator output, with \( w_i \) of LMS being its input.
Table 2. Scenario 1, best parameter values for NLMS.

| Algorithm | Parameters       |
|-----------|-----------------|
| NLMS ($w_i$) | $\mu = 0.25, M = 2$ and $\varepsilon = 0.651$ |
| NLMS ($d$) | $\mu = 1.1, M = 5$ and $0.651$ |

Figure 6 shows that the characterization, using $w_i$ of NLMS as the input of the power estimator, resulted in an output with the knock signal emphasized in a softened way and also reduced the background noise to nearly zero, as initially intended. By making a comparison between the mentioned responses, and those obtained individually from the LMS and the power estimator, it is possible to notice how the combination of algorithms is efficient. Therefore, the best setting for scenario 1 between both algorithms is the one using the output $\hat{d}$ of NLMS applied into the power estimator input. The result of the second attempted combination for this scenario is shown in Fig. 7.

Table 3. Scenario 2, best parameter values for NLMS.

| Algorithm | Parameters       |
|-----------|-----------------|
| NLMS ($w_i$) | $\mu = 0.4, M = 9$ and $\varepsilon = 0.6$ |
| NLMS ($d$) | $\mu = 0.1, M = 4$ and $0.7$ |

2) NLMS with Power Estimator

In this scenario, the same values obtained in the individual experiment of NLMS were used, $\mu = 0.32, \varepsilon = 0.3$ and $M = 10$. The combination among algorithms enabled a reduction of order to $M = 9$ and an increase of step to $\mu = 0.4$. The parameter $\varepsilon$ helped to improve the responses observed in the coefficients with $\varepsilon = 0.6$. The combination with the power estimator presented an efficient softening of variations obtained either in the coefficients or in the estimated output of NLMS.

Using $\hat{d}$ in the power estimator and with $\mu = 0.049$ and $M = 15$, all amplitude levels were accurately reproduced in a signal envelope manner, obeying the magnitude order between the peaks, i.e., if the second original knock peak was a little smaller than the third knock peak, the combination would reproduce such relationship. Higher values of $\mu$ and lower values of $M$ were tried (i.e.: $\mu = 0.1$ and $M = 4$), leading to a better knock representation in the adaptive filter output. However, results similar to the ones in the original setting ($\mu = 0.049$ and $M = 15$) were obtained by combining the algorithms. This observation is important because smaller values of order and higher step sizes ease the implementations of systems in real-time. The results are presented in Fig. 9.

To combine with NLMS, the most accurate response to the input signal was obtained using $\hat{d}$. The result of the second attempted combination for this scenario is shown in Fig. 8, where it is clear the reason why this scenario is called critical. The amplitude levels and the relationship between them of the input signal are not reproduced in the final output when using $w_i$ as the input of the power estimator, i.e.: the 4th peak.
is higher than the second and third ones in the input, but this is not reflected at the output of the filter.

### C. SCENARIO 3: KNOCK AND WHITE NOISE WITH SIMILAR AMPLITUDE

1) LMS WITH POWER ESTIMATOR

As shown in Fig. 10, a noise of higher intensity was observed in the output of the combined algorithm utilizing \( w_i \) as the power estimator input. This can be taken as a spurious knock because it happened before the simulated set of knock.

The combination in this scenario did not allow much flexibility in changing parameters for better performance. Only a small increase in \( \mu \) was possible without lowering the LMS individual performance. However, no improvements were obtained comparing the combined algorithm against the individual performance of LMS for the same signal.

#### TABLE 4. Scenario 3 combined - Configurations of LMS.

| Algorithm | Parameters |
|-----------|------------|
| LMS \( (w_i) \) | \( \mu = 0.7, M = 1 \) |
| LMS \( (\hat{d}) \) | \( \mu = 0.5 \) and \( M = 15 \) |

Using \( \hat{d} \), a combination with the power estimator softened variations obtained in its input, enabled an increase in the step size and, consequently, a better traceable performance. However, increasing the step size caused an increase in variations in the combined output. By increasing it to \( \mu = 1 \) and the order to \( M = 10 \), a better characterization in terms of less variation in the power estimator output was observed.

In terms of signal intensity, the combination using \( w_i \) presented a better performance, once the results obtained had a magnitude of around \( 10^{-1} \) while the results of combinations using \( \hat{d} \) presented magnitudes around \( 10^{-3} \). In terms of characterization, the use of \( w_i \) resulted in a softened response in the output, but with an irregularity in the second occurrence, a while before the peak. The use of \( \hat{d} \) was more accurate in characterizing the original signal, but presented several variations. The remaining of the system determined which combination should be used, considering their respective features. The results are presented in Fig. 10 and Fig. 11.

2) NLMS WITH POWER ESTIMATOR

Utilizing \( w_i \) with \( \mu = 0.1, M = 3, \epsilon = 0.05 \) provided the best results. The combination of algorithms also enabled the change of parameters to use \( \hat{d} \), where the setting \( \mu = 1 \) and \( M = 10 \) provided a better result if compared to their individual performance [28]. The results are presented in Figures 12 and 13.

#### VI. IMPLEMENTATION, PHYSICAL EXPERIMENTS AND RESULTS

Based on the simulations presented previously, we discuss next the combination of algorithms chosen, as well as its implementation and real-time observation.
A. REAL-TIME IMPLEMENTATION AND SOFTWARE WORKFLOW

The NLMS adaptive filter was implemented using MBD, given that the LMS did not present satisfactory results as the first algorithm. The LMS was sensitive to correlated data since its convergence depended on input values, as shown in Eq. (8). As a result, the LMS characterization of knock phenomenon was poor, given that the knock is a highly correlated signal. Therefore, the NLMS was the only adaptive filter considered for experimental validation. Although Fig. 10 shows a less noisy result utilizing $w_i$ as the adaptive filter output, the required order for that configuration was $M = 9$, which we believed a bit expensive for real-time calculation, not to mention the result obtained at scenario 2, shown in Fig. 8. Therefore, following the premises and analysis of section V, we chose the parameters for NLMS that were found in scenario 1, which are $\mu = 0.25$, $\epsilon = 0.651$ and order $M = 2$.

For the power estimator, we used $\beta = 0.01$, $\alpha = 0.99$.

The NLMS was connected to the power estimator, as shown in Fig. 14. This model was tested against others, and it provided the best characterization of the phenomenon. The output of the NLMS, which is connected to the power estimator input, is the estimated output $d(i)$, which is the denoised version of the input. Finally, the characterization of knock phenomenon is observed at the output of the power estimator. The software execution workflow can be described by the steps shown in Algorithm 1.

![Algorithm 1 Real Time Knock Characterization](image)

This workflow is also represented with the actual state machine software in Figs. 3, 14, 15 and 16.

B. EXPERIMENTS AND RESULTS

We proceeded to the experimental part with the model and the parameters chosen in section VI-A. The model-based software was developed using the model-based function generator ASCET (from ETAS) and then downloaded into the HIL ES630, also from ETAS. The results were measured in...
FIGURE 16. Power estimator execution block. Also, please refer to Eq. (16).

a Volkswagen Polo 2010 2.0, flex-fuel engine, placed on a passive dynamometer roll.

Fig.17 shows the experimental setup, with a laptop working as a Human-Machine Interface, the HIL ES630 (blue cases at the right hand side), an oscilloscope to measure the knock signal, and the automobile.

The vehicle was also equipped with an ECU developed by our research group (see Fig. 18 in the upper right side), which allowed the control of ignition timing. The spark angle in this ECU is advanced with respect to the vehicle nominal calibration point, in order to cause the knock for the experiments.

With that, the following procedure was executed:
1) Car parked placed in the dynamometer roll;
2) Start the car;
3) Start throttle with first gear engaged;
4) Wait the vehicle reaches no more than 10 km/h in initial gears, for example:
   1st gear shift: Speed up to 10 km/h;
   2nd gear shift: Speed up to 25 km/h;
   3rd gear shift: Speed up to 35 km/h;
   4th gear shift: Speed up to 45 km/h;
   5th gear shift: Speed up to 55 km/h;
5) Collect the signal arising from the knock sensor using the HIL.

During the experiment, the throttle pedal was completely pressed. This provided a higher torque request to the engine via ECU because the engine speed is low, as well as the speed developed by the roll to the selected gear. This increased the pressure inside the chamber and, therefore, the temperature.

Fig.19 shows why scenario 3 was created: high levels of noise can be observed at the sensor in either engine conditions off and on, respectively. Observing the noise levels also helped us to choose the appropriate threshold value used in practice for knock detection. The highest observed amplitudes with the engine working with no knock was of 0.25V for
this vehicle, whereas for knock event, we considered amplitudes higher than 0.5V as the knock occurrence threshold (THR), as shown in Fig. 20. Such threshold corresponded to the instant when the phenomenon was audibly detected. As observed in Fig. 20, using the established threshold, one can see that the filtered signal shows an envelope pattern that allows one to calculate the knock duration, as well as informing all distinct intensities with a smooth signal profile.

VII. CONCLUSION
The algorithms proposed here have shown the viability of characterizing the knock signal by combining adaptive filters and power estimators. The simulations helped to assess the algorithm performance and finding parameters that allow an efficient embedded real-time computation in a real flex-fuel engine, and this strategy could be generalized to gasoline engines as well. Combined to power estimator, an NLMS engine, and this strategy could be generalized to gasoline engines as well. Further software development, could provide reliable information for knock characterization that could replace the current integrated circuit of knock treatment or work as a redundant device for safety purposes. Validation with more test signals would also be needed, to fully develop an integrated solution with the appropriate threshold for detection.

Three output variables could be easily obtained from the implementation of this study: one could inform the binary occurrence of a knock while the other two could provide the duration and amplitude of the detected knock. Integration with a control pipeline is probably needed, in order to assess the real impact in vehicle dynamics and engine management strategy, hence, allowing a comparison to the current state-of-the-art integrated control methods.

REFERENCES
[1] A. V. O. Maggio, “Simulação hardware in the loop de um motor de combustão Flex,” M.S. thesis, Dept. Electron. Syst. Eng. Escola Politécnica, Universidade de São Paulo, Brasilia, Brazil, 2019.
[2] J. B. Heywood, Internal Combustion Engine Fundamentals. New York, NY, USA: McGraw-Hill, 1988.
[3] Murata. (Jun. 2018). Vibration Modes of Piezoelectric Materials. [Online]. Available: www.murata.com/products/resonator/basic/cerarock/vibration.html
[4] M. M. Ettefagh, M. H. Sadeghi, V. Pirouzpanah, and H. Arjmandi Tash, “Knock detection in spark ignition engines by vibration analysis of cylinder block: A parametric modeling approach,” Mech. Syst. Signal Process., vol. 22, no. 6, pp. 1495–1514, Aug. 2008.
[5] A. Stotsky, “Statistical algorithms for engine knock detection,” Automot. Technol., vol. 8, p. 10, 2007.
[6] M. Urlaub and J. F. Bohme, “Evaluation of knock in spark-ignition engines by least-squares,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2005, p. 681.
[7] J. A. Caton, “The interactions between IC engine thermodynamics and knock,” Energy Convers. Manage., vol. 143, pp. 162–172, Jul. 2017.
[8] T. Li, T. Yin, and B. Wang, “A phenomenological model of knock intensity in spark-ignition engines,” Energy Convers. Manage., vol. 148, pp. 1233–1247, Sep. 2017.
[9] M. Božić, A. Vucetic, M. Sjeric, D. Kozarac, and Z. Lulić, “Experimental study on knock sources in spark ignition engine with exhaust gas recirculation,” Energy Convers. Manage., vol. 165, pp. 35–44, Jun. 2018.
[10] G. Li, C. Zhang, and Z. Zhou, “Study on the knock tendency and cyclical variations of a HCCI engine fueled with n-butanol/n-heptane blends,” Energy Convers. Manage., vol. 133, pp. 548–557, Feb. 2017.
[11] Y. Zhuang, Y. Qian, and G. Hong, “The effect of ethanol direct injection on knock mitigation in a gasoline port injection engine,” Fuel, vol. 210, pp. 187–197, Dec. 2017.
[12] P. Bares, D. Selmanaj, C. Guardiola, and C. Onder, “A new knock event definition for knock detection and control optimization,” Appl. Thermal Eng., vol. 131, Dec. 2017.
[13] X. Shen and T. Shen, “Real-time statistical learning-based stochastic knock limit control for spark-ignition engines,” Appl. Thermal Eng., vol. 127, pp. 1518–1529, Dec. 2017.
[14] X. Shen and T. Shen, “Knock limit controller based on exponential moving average of knock intensity,” IFAC-PapersOnLine, vol. 49, no. 11, pp. 691–695, 2016.
[15] E. Galloni, “Dynamic knock detection and quantification in a spark ignition engine by means of a pressure based method,” Energy Convers. Manage., vol. 64, pp. 256–262, Dec. 2012.
[16] P. Bares, D. Selmanaj, C. Guardiola, and C. Onder, “Knock probability estimation through an in-cylinder temperature model with exogenous noise,” Mech. Syst. Signal Process., vol. 98, pp. 756–769, Jan. 2018.
[17] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. M. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner, “Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles,” PLoS Biol., vol. 5, pp. 511–513, 2007.
[18] Z. Yang, S. Yao, W. Yang, J. Harsulkar, E. Ansari, N. Miganakallu Narasimhamurthy, P. Dice, J. Naber, Y. Lonari, and S. Szwaja, “Investigation of combustion knock distribution in a boosted methane-gasoline blended fueled Si engine,” in Proc. SAE Tech. Paper Ser., Apr. 2018, pp. 1–8.
[19] A. d’Adamo, S. Breda, F. Berni, and S. Fontanesi, “The potential of statistical RANS to predict knock tendency: Comparison with LES and experiments on a spark-ignition engine,” Appl. Energy, vol. 249, pp. 126–142, Sep. 2019.
[20] D. Siano and D. D’Agostino, “Knock detection in Si engines by using the discrete wavelet transform of the engine block vibrational signals,” Energy Procedia, vol. 81, pp. 673–688, Dec. 2015.

[21] K. Akimoto, H. Komatsu, and A. Kurauchi, “Development of pattern recognition knock detection system using short-time Fourier transform,” IFAC Proc. Volumes, vol. 46, no. 21, pp. 366–371, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016383987

[22] A. A. Stotsky, “Statistical engine knock modelling and adaptive control,” Proc. Inst. Mech. Eng., D, J. Automobile Eng., vol. 222, no. 3, pp. 429–439, Mar. 2008.

[23] F. Bi, T. Ma, and X. Wang, “Development of a novel knock characteristic detection method for gasoline engines based on wavelet-denoising and EMD decomposition.” Mech. Syst. Signal Process., vol. 117, pp. 517–536, Feb. 2019.

[24] F. Bi, X. Li, C. Liu, C. Tian, T. Ma, and X. Yang, “Knock detection based on the optimized variational mode decomposition,” Measurement, vol. 140, pp. 1–13, Jul. 2019.

[25] X. Shen, Y. Zhang, T. Shen, and C. Khajorntraidet, “Spark advance self-optimization with knock probability threshold for lean-burn operation mode of Si engine,” Energy, vol. 122, pp. 1–10, Mar. 2017.

[26] X. Shen, Y. Zhang, and T. Shen, “Cylinder pressure resonant frequency cyclic estimation-based knock intensity metric in combustion engines,” Appl. Thermal Eng., vol. 158, Jul. 2019, Art. no. 113756.

[27] P. Bares, D. Selmanaj, C. Guardiola, and C. H. Onder, “A new knock event definition for knock detection and control optimization,” Appl. Energy, vol. 131, pp. 80–88, Feb. 2018.

[28] R. L. Silva, A. Lagana, and A. C. Seabra, “Characterization of detonation phenomenon signal using adaptive filtering and power estimation,” in Proc. SAE Tech. Paper Ser., Oct. 2013, pp. 2–8.

[29] A. H. Sayed, Adaptive Filters. Salt Lake City, Utah, United States of America: IEEE Press, 2008.

[30] P. S. R. Diniz, Adaptive Filtering Algorithms and Practical Implementation, 3rd ed. New York, NY, USA: Springer, 2008.

[31] C. G. Lopes, E. H. Satorius, P. Estabrook, and A. H. Sayed, “Adaptive carrier tracking for mars to Earth communications during entry, descent, and landing,” IEEE Trans. Aerosp. Electron. Syst., vol. 46, no. 4, pp. 1865–1879, Oct. 2010.

[32] S. Haykin and B. Van Veen, Signals and Systems. Hoboken, NJ, USA: Wiley, 1999.

[33] T. Zurawka and J. Schäuffele, Automotive Software Engineering, 2nd ed. Warrendale, PA, USA: SAE International, 2016. [Feb. 2020]. ASAM MCD-2 MC. [Online]. Available: https://www.asam.net/standards/detail/mcd-2-mc/

[34] A. Papoulis and S. U. Pillai, Probability, Random Variables, and Stochastic Processes, 4th ed. Singapore: McGraw-Hill, 2004.

RAFAEL LUIZ DA SILVA (Student Member, IEEE) was born in São Paulo, Brazil. He received the bachelor’s degree in technology of industrial automation from the Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Brazil, in 2008, and the M.Sc. degree in electrical engineering from the Escola Politécnica, University of São Paulo, Brazil, in 2013. He is currently pursuing the Ph.D. degree in electrical engineering with North Carolina State University, Raleigh, NC, USA.

PEDRO ROSSETTI received the B.Sc. degree in electrical engineering from the Escola Politécnica, University of São Paulo, in 2015. Since 2013, he has been an Embedded Software Engineer in developing emulation boards for automotive electronic control units with ETAS Inc., Ann Arbor, MI, USA.

JOÃO PEDRO F. SANTOS engaged the Exchange Program in industrial engineering with Aalto University, Finland, in 2015. He received the B.Sc. degree in electrical engineering from the Escola Politécnica, University of São Paulo, in 2016. He was with Siemens, from 2014 to 2016, the first two years with the Technical Sales Team, Brazil, and the part of the Business Development Team, Finland. Since 2016, he has been with Alvarez and Marsal, focusing in consulting activities for distressed companies.

ARMANDO ANTONIO MARIA LAGANÁ received the degree in electrical engineering from the Maua School of Engineering, in 1975, and the Ph.D. degree in electrical engineering from the Escola Politécnica, University of São Paulo, in 1994. He has experience in materials and processes of microelectronics. He is currently a Professor with the Department of Electronic Systems Engineering, Escola Politécnica, University of São Paulo. He works in automotive electronics field with an emphasis on engine management.

DIEGO COLÓN received the B.S. degree in electrical engineering and the master’s and Ph.D. degrees in control engineering from the University of São Paulo (USP), in 1997, 1999, and 2003, respectively. He worked in projects to the Brazilian Navy, from 2003 to 2006. He was an Assistant Professor with São Paulo State University, from 2006 to 2013. He is currently an Assistant professor in electrical engineering with the University of São Paulo. His research interests include control of mechanical and automotive systems, robotics, applications of Lie groups, and differential geometry.

JOÃO FRANCISCO JUSTO received the B.Sc. and M.Sc. degrees in physics from the University of São Paulo, in 1988 and 1991, respectively, and the Ph.D. degree in nuclear engineering from the Massachusetts Institute of Technology, in 1997. He was a Visiting Associate Professor with the University of Minnesota, from 2007 to 2008. He has experience in computational modeling of nanomaterials and embedded electronics. He is currently a Full Professor with the Escola Politécnica, University of São Paulo.