sciendo
Power Electronics and Drives
Volume 9(44), 2024 DOI: 10.2478/pead-2024-0026
Power Electronics and Drives

Contribution to a New Algorithm to Perform an Automatic Self-Calibration of Current Sensors

Research paper

Djamel Eddine Beladjine[1,*], Djamel Boudana[2], Abdelhafidh Moualdia[3], Patrice Wira[4]

[1]Electrical Engineering Department, University Yahia Fares, Medea, Algeria
[2]Automatic Department, National Polytechnic School, Algiers, Algeria
[3]Electrical Engineering Department, University Yahia Fares, Medea, Algeria
[4]Electrical Engineering Department, University of Haute Alsace, Mulhouse, France

Received: 18 March 2024; Accepted: 23 June 2024

Abstract: Sensors calibration plays a crucial role in controlling systems and achieving fault-tolerant control by ensuring accuracy, performance, safety, energy efficiency, and compliance with standards. It is an essential to maintain the reliability and effectiveness of modern control systems across various applications. In this paper, we represent a new algorithm that processes a set of raw data collected by a sensor to find the mapping function that relates the raw data to the real value of the measured signal by the sensor. Working on sensors with an unknown mapping function, unknown parameters, or with external disturbances, that affects their behaviour, represents a problem; moreover, it takes a lot of time and effort to calibrate the sensor before each use. Several techniques were used to overcome these aspects mostly by recording the output of the sensor for different input values that change manually, to calibrate the sensor. However, the represented technique in this paper can easily provide us with the input/output model of a specific sensor by doing only one experiment; it also improves the accuracy of the measurements as it is a self-calibrating technique that reduces the nonlinearity and noise problems to deliver a better estimation of the measured signal, which is validated in this paper experimentally using a low-cost current sensor by comparing the obtained results from this algorithm with the results using the extracted input/output model illustrated in the datasheet. Furthermore, if the sensor is pretty poor, and if the application requires more precision, the provided estimation by the mapping function can be mixed with other sensor/s readings using sensor fusion algorithms to find a more precise value of the input. The represented algorithm can also perform self-calibration while evaluating the functionality of the application and the variations of the temperature and other external disturbances that affect the sensor.

Keywords: *algorithm • self-calibration • disturbances • estimation • mapping function*

1. Introduction

Mainly, all sensors, especially low-cost sensors, are sensitive to different disturbances (i.e. temperature changes, magnetic interferences, etc.). They are noisy and suffer from a nonlinearity behaviour; moreover, many of them require calibration before and even during the functioning of the sensor by doing self-calibration, depending on the application that is used for it; in addition, some of the cheap sensors that are available in the market are provided without defined input-to-output characteristics (i.e., the ZMPT101B voltage sensor; Kurniawan et al., 2022). Therefore, it is necessary to find out these characteristics by doing some experiments on the sensor, to use it efficiently.

Calibration of current sensors is of paramount importance in the realm of controlling systems and fault-tolerant control methodologies (Hong et al., 2023; Teler and Orłowska-Kowalska, 2023). Accurate measurements of electric current are indispensable for upholding control performance, making fault detection easier (Adamczyk and Orlowska-Kowalska, 2023), ensuring system safety, optimizing energy utilization, enhancing robustness, and adhering to stringent industry standards. Through meticulous calibration procedures (Djokic and So, 2005), current sensors enable control systems to make judicious decisions, promptly identify deviations, mitigate potential risks,

* Email: *djameleddine.beladjine@yahoo.fr*

and optimize energy efficiency. Consequently, calibration serves as a cornerstone in guaranteeing the precision, reliability, and safety of contemporary control systems, thereby laying a solid foundation for further investigation and advancement within scientific and engineering domains.

Different calibration techniques were developed starting with the classical techniques as assumed and used such as the multivariable regression technique, which required a precise current source; also, there are potential power dissipation issues in high current scenarios (Badura et al., 2019; Lifton and Liu, 2020), that in most cases necessitate specific materials, consume time and require effort, as they are mostly based on minimizing the all-known measurement errors. Other advanced techniques were developed, such as compensation with artificial networks and statistical methods that require specialized equipment, software, and sophisticated simulation tools (Cordero et al., 2018; Khan et al., 2003; Pertijs, 2014). To efficiently approximate the characteristics of the sensor, however, despite the efficient results obtained from the above-mentioned techniques, they still consume time when training the model and collecting the training data points and choosing the optimal parameters, and require knowledge of the statistical characteristics of the sensor, specifying whether they require only a specific type of sensors (smart sensors) or multiple sensors (Hwang et al., 2021).

In opposite, the represented algorithm in this paper has been able to solve and reduce those problems, as it performs automatic self-calibration and does not require precise, known input values; in addition, it saves time and effort. It uses a set of collected raw data by the sensor for a well-known-amplitude, sinusoidal input signal that we refer to as the reference signal to calibrate the sensor (Wu et al., 2020), allowing increasing the accuracy of the measurements and calibrating the sensor in less time and effort, and without the need of changing the parameters' input source.

The technique is based on collecting a set of measurements by the sensor for a sinusoidal- input reference signal with well-known amplitude and then using the provided algorithm to converge to the exact frequency and the phase values of the reference signal, and, finally, finding the best fit of the measured data versus the estimated sinusoidal reference signal by using the algorithm. The output of this algorithm is a mapping function that can be a nonlinear function, which can be also used later to convert any raw output of the sensor to an estimation of the real, measured signal.

The present paper is organized as follows: first, we present the description of the proposed algorithm used to extract the mapping function, including all the parameters and data to develop a mathematical model, and the entire approach through sequential steps, thereafter; this algorithm has been implemented by using an ACS712T 20 Amp, low-cost current sensor via the Atmel SAM3X8E ARM (Advanced RISC Machine) microcontroller board to obtain the results presented and discussed in the final section.

2. Algorithm Description

The algorithm sequentially executes a series of tasks aimed at estimating the input-to-output model of a sensor as shown in Figure 1, relying solely on the well-established amplitude of a sinusoidal input signal and a set of recorded raw data from the sensor corresponding to that sinusoidal input. This algorithm comprises 10 distinct steps, outlined as follows:

The algorithm consists of 10 steps that are represented as follows.

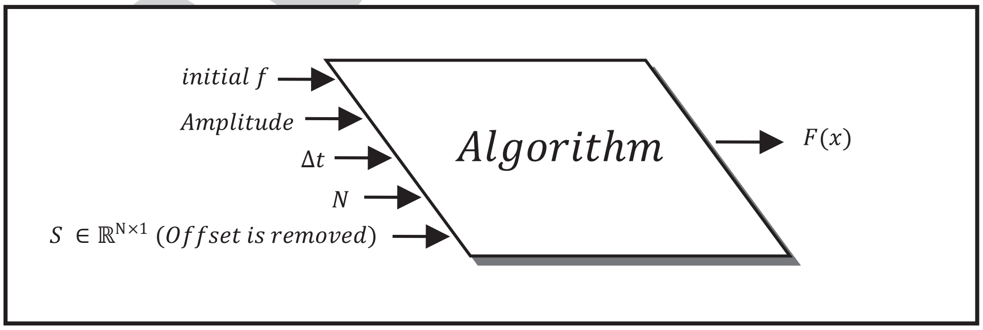

Figure 1. The algorithm inputs/output.

2.1. Collect the raw data

The initial step involves the collection of raw data from the sensor. *N* samples of raw data, denoted as $S \in R^{(N \times 1)}$, are gathered utilizing the Analogue-to-Digital Converter (ADC) while the sensor is exposed to a sinusoidal reference signal with a known amplitude. It is essential to ensure that *N* significantly exceeds T/Δt, where T represents the predetermined period of the reference input signal and Δt denotes the sampling time. Subsequently, the collected data is stored after eliminating any existing DC offset, facilitating subsequent processing. Additionally, the algorithm requires an initial frequency value for the sinusoidal reference input signal, which need not be precise and can be arbitrarily selected within defined constraints, to prevent the system from diverging.

2.2. Estimating the zero-crossing point (ZCP) time in the collected data

Estimating the ZCPs involves identifying each successive sign change in the collected sample data *S* and computing the zero-crossing time through linear interpolation between these points. Following the linear interpolation (Hu et al., 2022), the linear estimation around each ZCP is represented by the following sub-algorithm:

For *n* ranging from 1 to *N*−1.

$$\text{if: } \left(S(n) \times S(n+1) < 0\right)$$

and

$$a^S(k^S) = \frac{S(n+1) - S(n)}{(n+1)\Delta t - n\Delta t}, \tag{1}$$

Here, $a^S \in \mathbb{R}^{K^S \times 1}$ is an array of slopes representing the linear interpolation around each pair of successive sign change points, with k^S indexing each zero crossing, and K^S is the total number of detected ZCPs. The other parameter b^S is calculated as follows.

$$\varphi_{t0}\left(k^S\right) \begin{cases} -2 \times \pi \times f \times t_0^S\left(k^S\right) \text{if } a^S\left(k^S\right) > 0 \\ -2 \times \pi \times f \times t_0^S\left(k^S\right) - \pi \text{ otherwise} \end{cases},$$

crossing, and K^S is the total number of detected ZCPs. The other parameter b^S is calculated as follows.

$$b^2\left(k^2\right) = \frac{S(n) + S(n+1) - a^S(k^S)(n\,\Delta t + (n+1)\,\Delta t)}{2}, \tag{2}$$

The vector $b^S \in \mathbb{R}^{K^S \times 1}$ contains the parameters of the linear interpolation for each ZCP.

These parameters facilitate the precise estimation of zero crossing times in the sampled data.

To determine the precise zero crossing time $t_0{}^S\left(k^S\right)$ for each crossing point k^S, the function $y^S\left(k^S\right)$ is set to zero. This is achieved through the following computation:

$$\begin{gathered} y^S\left(k^S\right) = a^S\left(k^S\right)t + b^S\left(k^S\right), \\ t \in [n\Delta y, (n+1)\Delta t] \end{gathered} \tag{3}$$

where $t_0{}^S$ is a K^S-dimensional vector containing the estimated zero crossing times. These calculated times provide precise references for the occurrence of zero crossings in the sampled data.

y^S is an K^S dimensional function vector containing the linear interpolation around the detected ZCPs.

For each ZCP k^S, we calculate the zero-crossing time $t_0{}^S\left(k^S\right)$ by putting $y^S\left(k^S\right) = 0$ as follows.

$$\begin{gathered} \text{For}: k^S = 1:1:K^S \\ t_0^S(k^S) = -\frac{b^S(k^S)}{a^S(k^S)}, t_0^S \in R^{k^S \times 1} \end{gathered} \tag{4}$$

2.3. Estimating the phase

To estimate the phase, the following procedure is employed. Utilizing the estimated crossing time points $t_0^S(k^S)$ along with the frequency f, the phase value at each crossing point is computed using the subsequent formula:

$$\varphi_{t0}(k^s)\begin{cases}-2\times\pi\times f\times t_0^s(k^s) & \text{if } a^s(k^s)>0\\ -2\times\pi\times f\times t_0^s(k^s)-\pi & \text{otherwise}\end{cases}, \tag{5}$$

Here, $\varphi_{t0}(k^S)$ denotes the estimated phase at each detected ZCP k^S. Consequently, an array is generated, containing all the estimated phase values from each crossing point, denoted as $\varphi_{t0}\in\mathbb{R}^{K^S\times 1}$.

The estimated phase value is then computed by calculating the mean of φ_{t0}, represented as

$$\varphi=\frac{\sum_{i=1}^{K^S}\varphi_{t0}(i)}{K^S}, \tag{6}$$

Here, φ signifies the total estimated phase. This approach yields an array containing the estimated phase values at each crossing point, from which the overall phase value is determined, by averaging these individual estimates.

2.4. Generating a discrete, sinusoidal reference signal

To generate a discrete sinusoidal reference signal, a reference signal denoted as R is generated and stored, represented as a vector in $\mathbb{R}^{N\times 1}$. This signal serves to represent the input signal and is generated utilizing the sample time Δt, the frequency f, and the value of the estimated phase φ. The generation of the reference signal is described by the following Eq. (7):

$$R(n)=\sqrt{2}A_{\text{rms}}\sin(2.\pi.\text{f.n}.\Delta t+\varphi), \tag{7}$$

where n ranges from 1 to N, and $\sqrt{2}A_{rms}$ denotes the amplitude of the signal. This equation depicts the discrete sinusoidal signal R, wherein each sample $R(n)$ is determined by the given parameters, including the sample time, frequency, and estimated phase.

$$\begin{aligned}&\text{for } k^R=1:1:K^R\\ &t_0^R(k^R)=-\frac{b^R(k^R)}{a^R(k^R)},\, t_0^R\in R^{K^{R\times 1}}\end{aligned} \tag{8}$$

where K^R is the total number of detected ZCPs of the generated samples R.

a^R and b^R are both $\mathbb{R}^{K^R\times 1}$ dimensional arrays and are the parameters of the linear interpolation y^R around each ZCP in the $R(n)$ signal.

Zero-crossing time arrays' dimension correction

Depending on the value of f,N and φ, the dimensions of the vector t_0^R might differ from the dimension of the vector t_0^S:

In case the t_0^R dimension is higher than t_0^S dimension, we reconstruct the vector t_0^R by taking only the first elements that lead to equal dimensions.

Otherwise, if the dimensions of t_0^S are greater than the dimensions of t_0^R we reconstruct the vector t_0^S by taking only the first elements that make the dimensions equal.

If one of those cases exist, either K^R or K^S will be changed to get

$$K^R=K^S=K, \tag{9}$$

This operation can also be done in another way (which is the one used to validate the result) by removing several excess terms from the start and end of the largest dimension array; taking into account the case of the old value of dimension, this method allows the dimension correction to be made in the middle.

2.5. Calculate the zero-crossing-time error

To assess the accuracy of the estimated zero crossing times, the time difference between each estimated zero crossing time of the generated discrete sinusoidal reference signal *R*(*n*) and the collected data *S*(*n*) is computed. This calculation is expressed as

$$t_0^{\mathrm{error}} = t_0^S - t_0^R, \tag{10}$$

Here, t_0^{error} represents the time error, indicating the deviation between the estimated zero crossing times from the collected data $S(n)$ and the generated reference signal $R(n)$. It is noteworthy that this time difference, t_0^{error} may exhibit noise due to inherent sensor noise.

2.6. Fitting the difference time vector t_0^{error}

To fit the difference time vector t_0^{error} using a first-order polynomial function, we express it as

$$y^{t_0^{error}}(k) = P_1 k + P_2, \tag{11}$$

Here, P_1 and P_2 represent the coefficients of the linear function $y^{t_0^{error}}$, which is the best fit in a least-squares sense of the t_0^{error} data.

The calculation of P_1 and P_2 involves forming the Vandermonde matrix V with the given system

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ . & 1 \\ . & 1 \\ . & 1 \\ K-1 & 1 \\ K & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} t_0^{\mathrm{error}}(1) \\ t_0^{\mathrm{error}}(2) \\ . \\ . \\ . \\ t_0^{\mathrm{error}}(K-1) \\ t_0^{\mathrm{error}}(K) \end{bmatrix}, \tag{12}$$

$$VP = y^{t_0^{\mathrm{error}}}.$$

This system can be expressed as $VP = y^{t_0^{error}}$, where V is the Vandermonde matrix.

The next step involves applying the QR factorization to the matrix V, which can be achieved using various methods (Anderson et al., 1992; Demeure and Scharf, 1989), including the Gram–Schmidt process (Leon et al., 2013). This process decomposes the matrix V into an orthogonal matrix Q and an upper triangular matrix R_{QR}.

To perform the Gram–Schmidt process, we first form the vectors a_1, e_1, a_2, and e_2.

Vector a_1 is defined as

Vector a_1:

$$a_1 = [1\,2 \ldots K-1 K]^T, a_1 \in R^{K\times 1} \tag{13}$$

And vector e_1 is obtained by normalizing a_1 as follows:

Vector e_1:

$$\mathrm{e}_1 = \frac{a_1}{\sqrt{1^2 + 2^2 + \ldots + (\mathrm{K}-1)^2 + \mathrm{K}^2}} \tag{14}$$

$e_1 \in R^{K\times 1}$

$$\sqrt{1^2 + 2^2 + ... + (K-1)^2 + K^2} = \sqrt{\frac{K(K+1)(2K+1)}{6}}, \tag{15}$$

By substituting Eq. (14) in Eq. (15) we get:

$$e_1 = \frac{\sqrt{6}\, a_1}{\sqrt{K(K+1)(2K+1)}} \tag{16}$$

Vector a_2 is defined as

Vector a_2:

$$a_2 = [11\ldots11]^T,\ a_2 \in R^{K\times1} \tag{17}$$

Vector e_2:

And vector e_2 is computed using the formula

$$e_2 = \frac{a_2 - a_2.e_1 \times e_1}{\sqrt{\Sigma_{i=1}^{K}\left(a_2(i) - a_2.e_1 \times e_1(\mathrm{i})\right)^2}},\ \mathrm{e}_2 \in R^{K\times1} \tag{18}$$

Simplifying the expression for e_2, we get:

$$e_2 = \sqrt{\frac{2(2K+1)}{K(K-1)}}\left(a_2 - \frac{\sqrt{3K(K+1)}}{\sqrt{2(2K+1)}} \times e_1\right), \tag{19}$$

2.7. Calculating the *Q* and *R* matrix

After forming the vectors e_1 and e_2, the matrices Q and R_{QR} are calculated using the following formulas:

$$Q = [e_1\ \ e_2]$$
$$R = Q^T V = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \begin{bmatrix} t_0^{\text{error}}(1) \\ t_0^{\text{error}}(2) \\ . \\ . \\ . \\ t_0^{\text{error}}(K-1) \\ t_0^{\text{error}}(K) \end{bmatrix}, \tag{20}$$

2.8. Calculating the *P* vector

Finally, the coefficient of best linear fitting of the system is given by

$$P = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = R^{-1}\left(Q^T y^{t}{}_0^{\text{error}\,T}\right), \tag{21}$$

where P_1 represents the estimated frequency f. P_2 represents the calculated phase φ, which depends on the value of frequency f; hence, P_2 is indirectly affected by the frequency.

2.9. Convergence test and updating the frequency

After computing the coefficient vector $P = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$ from the QR factorization, a convergence test is conducted to assess whether the conditions $|P_1| < \varepsilon_{P_1}$ and $|P_2| < \varepsilon_{P_2}$ are satisfied. If the convergence condition is not met, indicating that the error is not within the acceptable range, the frequency f is updated using the provided formula, and the algorithm iterates back to step C. The updated frequency is determined as

$$f_{\text{new}} = f_{\text{old}} + K_{\text{gain}} \times P_1, \tag{22}$$

Here, ε_{P_1} and ε_{P_2} are very small constants defining the acceptable error region to confirm convergence. These values need to be selected carefully. K_{gain} represents a gain controlling the convergence speed, and its selection is critical to ensure algorithm convergence.

Note that it is also possible to use other correction forms. The correction process can be observed graphically by visualizing the rotation of the $y^{t_0^{error}}$ line around the center. Thus, the objective of steps *H* and *I* is to minimize t_0^{error}, ensuring improved accuracy in the estimation of zero-crossing times.

2.10. Fit the collected raw data with the generated reference signal

After achieving convergence in **step 9**, a fitting process is conducted between the newly adjusted signal $R(n)$ and the collected data $S(n)$. This fitting is accomplished by selecting a fitting function F and employing an algorithm to minimize the error between $S(n)$ and $R(n)$ by adjusting the parameters of the chosen function F.

The Vandermonde matrix technique, previously utilized for coefficient estimation, can be adapted to minimize the error between $S(n)$ and $R(n)$ when selecting a polynomial fitting function with a specific degree.

The objective is to minimize the following system:

$$F(S(n)) = R(n), \tag{23}$$

The outcome of this algorithm is a function *F* that accepts the raw values from the sensor as inputs and converts them into estimations of the actual, measured input clarifies in Figure 2. However, the precision of the function *F* may still not be sufficient, especially when using a poor sensor. Therefore, for applications demanding higher precision, it may be necessary to combine the readings of other sensor(s) using a sensor fusion algorithm (Elmenreich, 2002; Yeong et al., 2021).

3. Experimental results

The experimental section shows that the effectiveness of the proposed algorithm was validated using a current sensor, in particular the ACS712T 20 Amp current sensor. The experimental setup involved connecting the current sensor to the Atmel SAM3X8E ARM microcontroller to make it easier for the computer to collect data. The current sensor measured the current flowing to a power resistor connected to an AC source. To ensure compatibility with the ADC of the ARM SAM3X8E chip and avoid potential damage, a voltage divider was used to reduce the current sensor output range from 0–5 *V* to 0–3.3 *V*, cause the analogue input of the ARM SAM3X8E chip should not exceed 3.3 *V*, then reading the measured voltage using the 12-bit resolution ADC and dividing the measured value by the sensitivity characteristic of the sensor to find the final values of current in real time and collect the raw data to print the current evolution and apply the algorithm developed below in Figure 3.

The amplitude of the measured sinusoidal AC current was determined using the root mean square (RMS)arm value provided by a precise mustimeter. The collected data, denoted as *S*, underwent preprocessing to remove the DC offset by calculating the mean of several samples collected over a certain number of periods. Subsequently, the processed data *S* was input into a MATLAB (**MAT**rix **LAB**oratory) software script that executed the described algorithm.

In Figure 4, the amplitude values of the blue samples represent the readings from the ADC after removing the DC offset. The algorithm processes the AC input exclusively, which conforms to the form $\sqrt{2}I_{rms}\sin(2\pi ft + \varphi)$. Due to the rapid changes in the I_{rms} value caused by the grid AC source; it was necessary to collect data for a short period. The estimated crossing points were determined by the intersection of the linear interpolations of each successive opposite sign point with the x-axis, providing a clear indication of the algorithm's functioning.

Throughout the iterative process, the algorithm dynamically evaluates the constructed signal R based on the calculated frequency and phase values. At each iteration, the ZCPs are estimated in exactly the same way as with

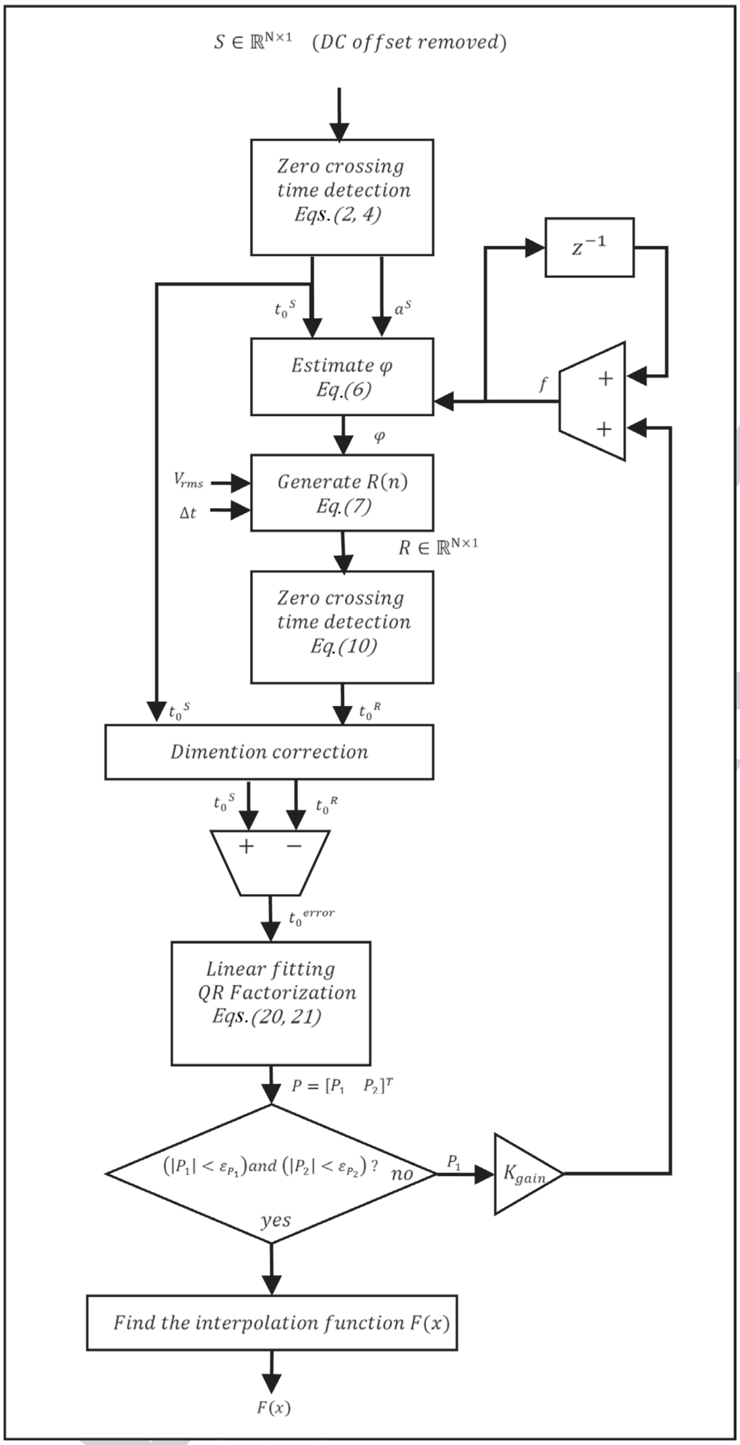
$S \in \mathbb{R}^{N\times1}$ $(DC\ offset\ removed)$
Zero crossing time detection Eqs.(2, 4)
t_0^S
a^S
z^{-1}
Estimate φ Eq.(6)
f
φ
V_{rms}
Δt
Generate $R(n)$ Eq.(7)
$R \in \mathbb{R}^{N\times1}$
Zero crossing time detection Eq.(10)
t_0^R
Dimention correction
t_0^{error}
Linear fitting QR Factorization Eqs.(20, 21)
$P = [P_1 \quad P_2]^T$
$(|P_1| < \varepsilon_{P_1})and\ (|P_2| < \varepsilon_{P_2})$?
no
yes
P_1
K_{gain}
Find the interpolation function $F(x)$
$F(x)$

Figure 2. The algorithm flow chart description.

the input signal *S*, as illustrated in Figure 5. This process ensures the consistency and accuracy of the estimation procedure.

The graphical representation in Figure 6 depicts the iterative refinement of the ZCP time error correction between the signals S and R across various iterations, where the slope of the linear fitting of t_0^{error} serves as the parameter P_1.

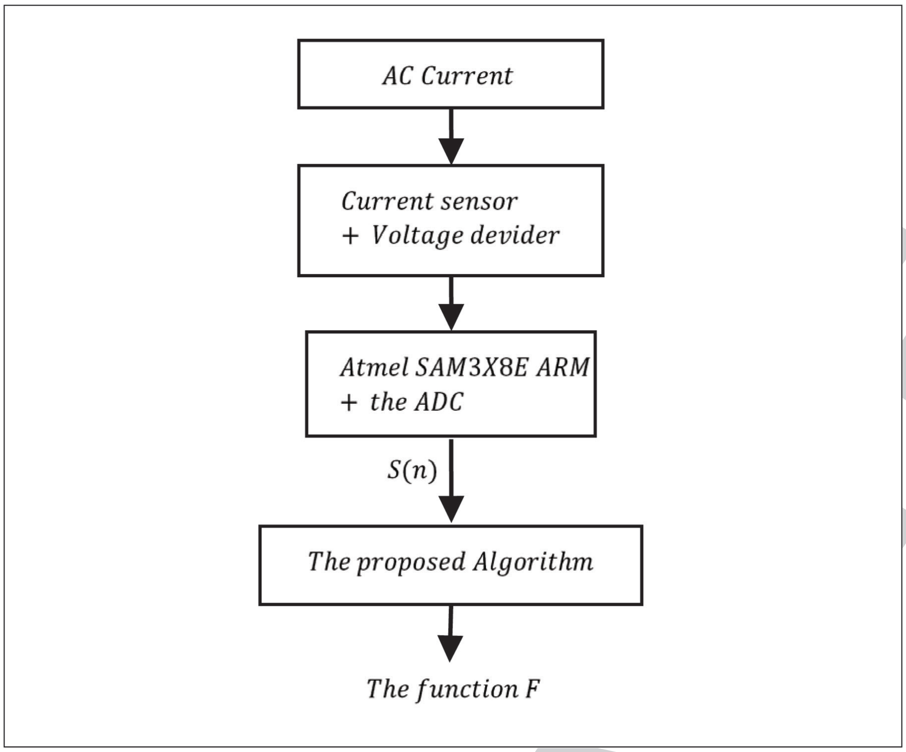

Figure 3. The implementation process. ADC, analogue-to-digital converter.

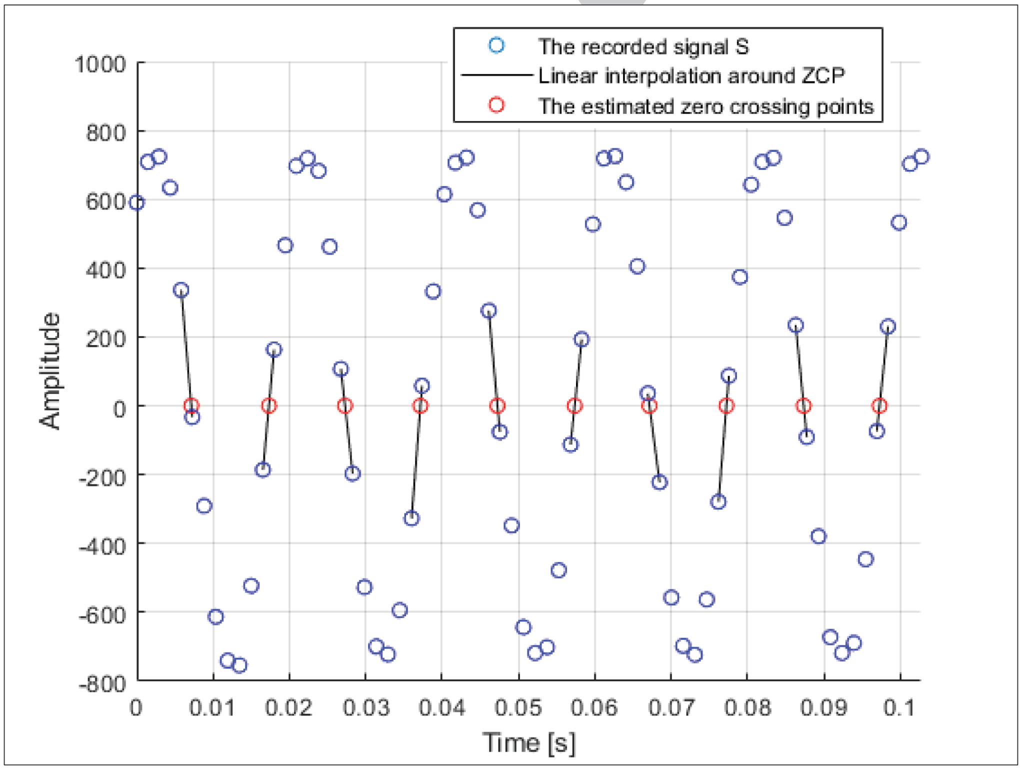

Figure 4. The collected *S* input (*DC offset* removed) and the detected ZCPs in the *S* signal. ZCPs, zero-crossing points.

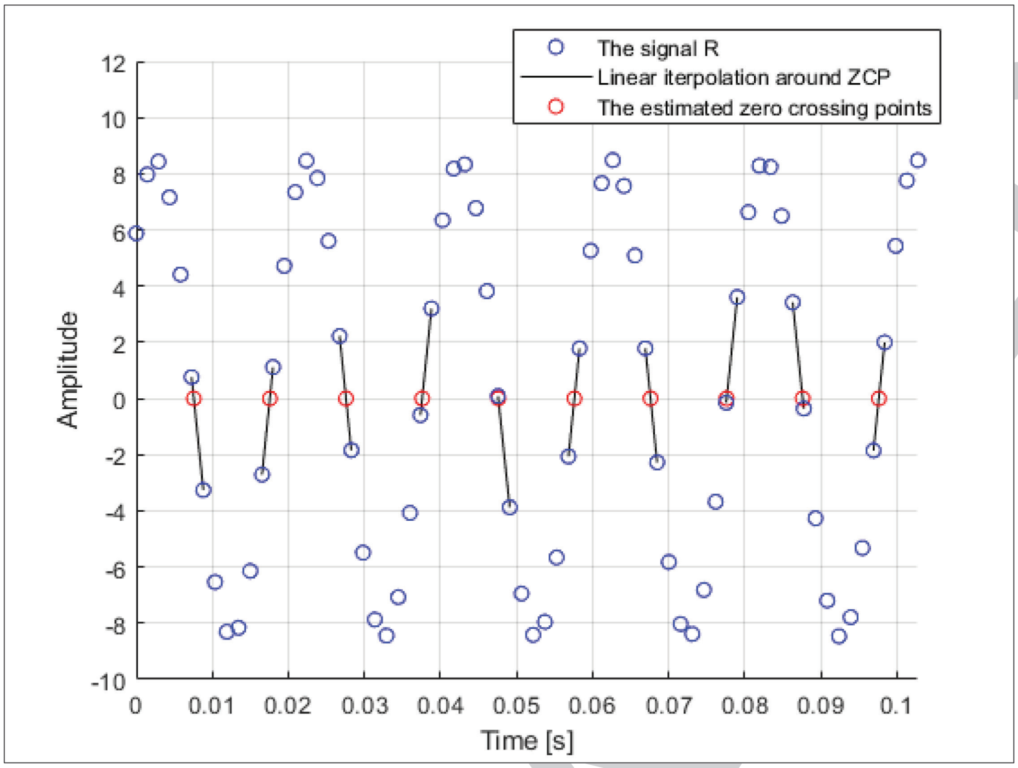

Figure 5. The generated signal *R* and the detected ZCPs in it at a random iteration (iteration = 60). ZCP, zero-crossing point.

Furthermore, the process of frequency and phase correction in signal R relative to input signal S is illustrated across different iterations in Figure 7. Notably, signal R is amplified by a factor of 50 in the figure to enhance the visibility of the signal in the figure, to notice the frequency and phase correction effects compared to S.

The convergence of the algorithm towards the estimation of frequency and phase is evident across iterations, particularly when initialized with different initial values of f. Notably, the algorithm rapidly converges to these values, especially when a suitable initial value of f is chosen. Achieving precision in the converged value of f is crucial for obtaining accurate results (Figure 8); also the selection of small values for ε_{P_1} and ε_{P_2} to improve results.

The scatter plot of R values against S values shown in Figure 9 exhibits a non-linear pattern due to the presence of noise in the measurement process. This characteristic highlights an advantage of the algorithm, as it effectively fits all values, accommodating the noise inherent in the measurements. This capability is facilitated by the periodic input, enabling the measurement of various sensor outputs at identical input values. Additionally, users have the flexibility to select from an unlimited number of fitting functions, albeit with the consideration of the execution time when implementing the algorithm in microcontrollers for diverse applications.

The comparison is made between the application of the estimated mapping function F generated by the algorithm to the input S in Figure 10, and the application of the extracted input/output formula derived from the ACS712T datasheet (Lazarević et al., 2022) to the same input S.

The extracted input to output formula from the graph in the *ACS712T* datasheet (Leon et al., 2013) can be written as follows in our case.

$$\left(\frac{3.3}{2^{12}-1}\right)\times\left(\frac{3}{2}\right)\times 10\times x = 1.2087\times 10^{-2}\times x, \tag{24}$$

Here, x represents the DC-removed readings from the ADC.

Comparatively, the estimated input-to-output formula produced by the algorithm is represented as a polynomial function, as illustrated in Table A2 in Appendix, with the general form

$$a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0, \tag{25}$$

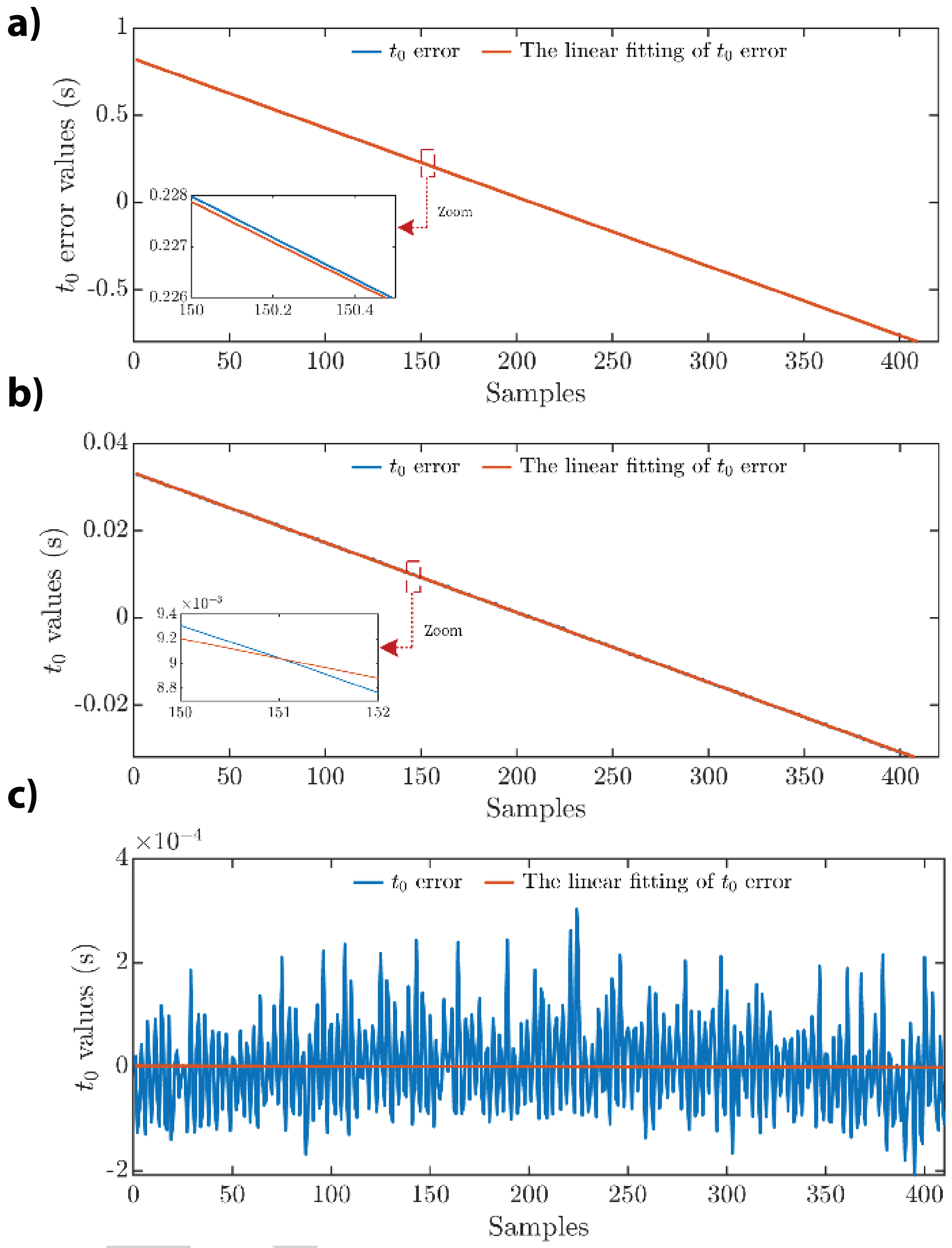

Figure 6. The variation of t_0^{error} and the linear interpolation function $y^{t_0^{error}}$ at different iterations: (a) at iteration = 20, (b) at iteration = 40 and (c) at iteration = 85.

By ignoring the parameters $a_0, a_2, a_3, a_4,$ and a_5, we obtain

$$a_1 x = 1.2552 \times 10^{-2} x, \tag{26}$$

This outcome closely resembles the expression derived from the datasheet formula, validating that the algorithm is capable of estimating the formula for any sensor through a single experimental test. Additionally, the estimated

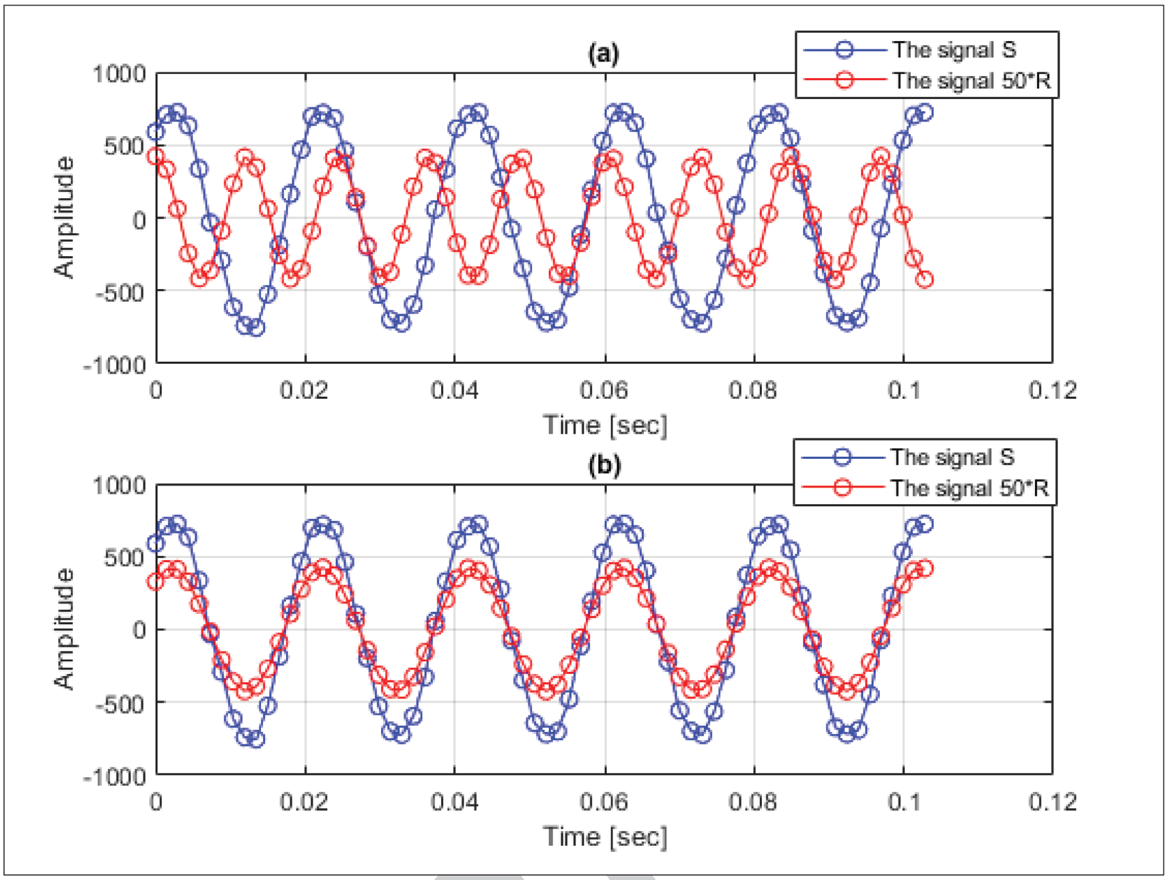

Figure 7. The signals *S* and *R* during the process: (a) at a random iteration (Iteration = 20) and (b) at the end of the frequency and the phase correction.

function includes the neglected terms, enhancing result accuracy compared to the datasheet. It is noteworthy that a one-degree polynomial function could have been chosen as a fitting function, solely to estimate the datasheet formula. Therefore, the advantages of the presented algorithm can be summarized as follows:

1. *Efficiency in formula estimation*: The algorithm streamlines the process of estimating input/output formulas for unknown or novel sensors, significantly reducing the time and effort required, by performing just one simple experiment.
2. *Enhanced accuracy*: By offering flexibility in selecting any mapping function and optimizing its parameters using a dataset as mentioned in Table A1, the algorithm enhances accuracy. It considers the nonlinearity behaviour and existing noise, allowing for more precise calibration.

An invaluable application of the algorithm lies in its implementation within the processing unit (Gao, 2018; Xu, 2014) to calibrate current sensor arrays. Leveraging the sinusoidal power source and connected processing units, it offers significant advantages in sensor calibration (Shalamov, 2016).

4. Optimizing the Algorithm

The first step to optimize the elapsed time of the algorithm is to reduce the number of samples N, but keeping it satisfying theoretically $N > \frac{T}{\Delta t}$.

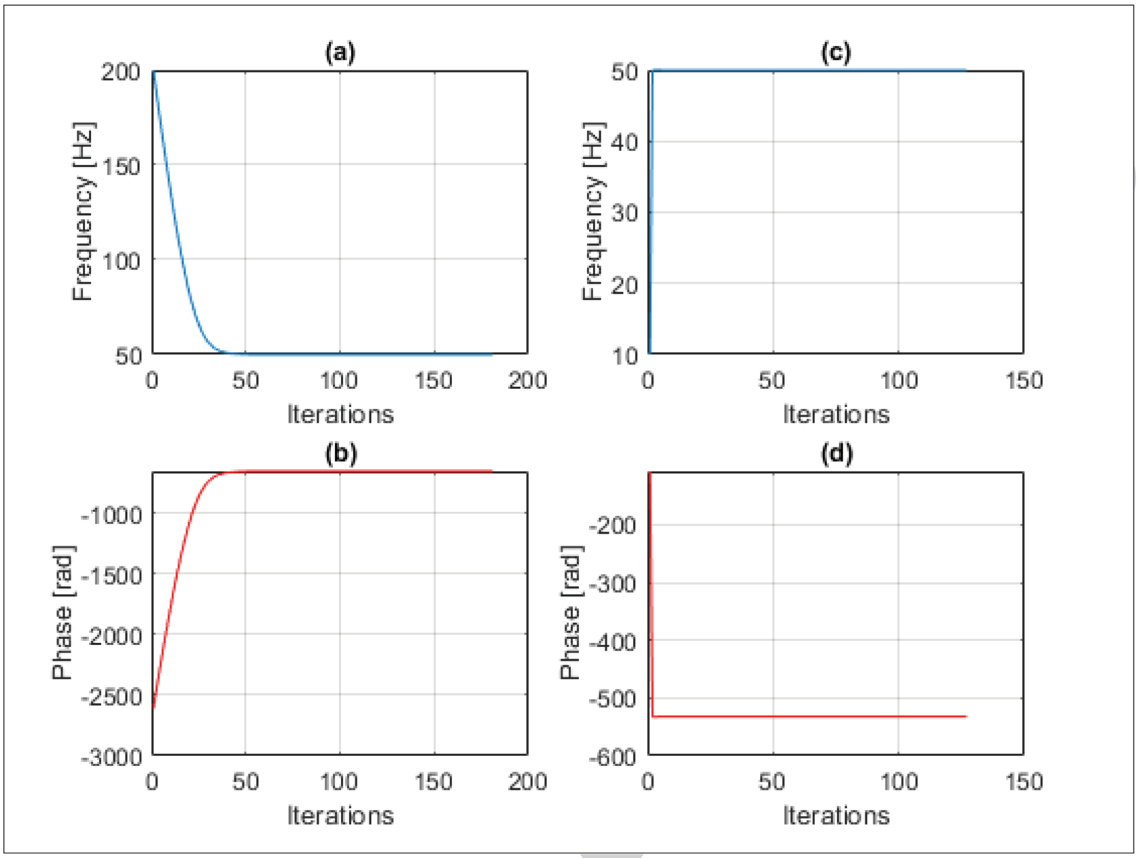

Figure 8. The convergence of *f* and *φ* for diffirent initial values of *f*: (a) initial frequency of 200 Hz, (b) initial phase of −2600 rad, (c) initial frequency of 50 Hz and (d) initial phase of −100 rad.

However, practically, it was proved that reducing the number of the samples highly affects the performances; therefore, the number of samples must be chosen carefully and kept as high as possible (advised $N \gg \frac{T}{\Delta t}$). The higher the *N* is, the better the algorithm results are, but it is also limited by the storage capacity of the microcontroller unit in the case of applying the algorithm in a microcontroller instead of a to a computer, wherein a tradeoff needs to be done between the desired performance and the storage capacity of the device.

As the phase is calculated by taking the mean and since the dimension correction is made at the middle, the error t_0^{error} can be considered to be symmetrical, and that allows us to reduce the size of the Vandermonde matrix V and re-define the system in the following way:

$$V_{\text{opt}} P = y_{\text{opt}}^{t_0^{\text{error}}}, \qquad (27)$$
$$0 < d < \frac{K}{2},$$

$$\begin{bmatrix} d & 1 \\ d+1 & 1 \\ . & 1 \\ . & 1 \\ . & 1 \\ K-d-1 & 1 \\ K-d & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} t_0^{\text{error}}(d) \\ t_0^{\text{error}}(d+1) \\ . \\ . \\ . \\ t_0^{\text{error}}(K-d-1) \\ t_0^{\text{error}}(K-d) \end{bmatrix},$$

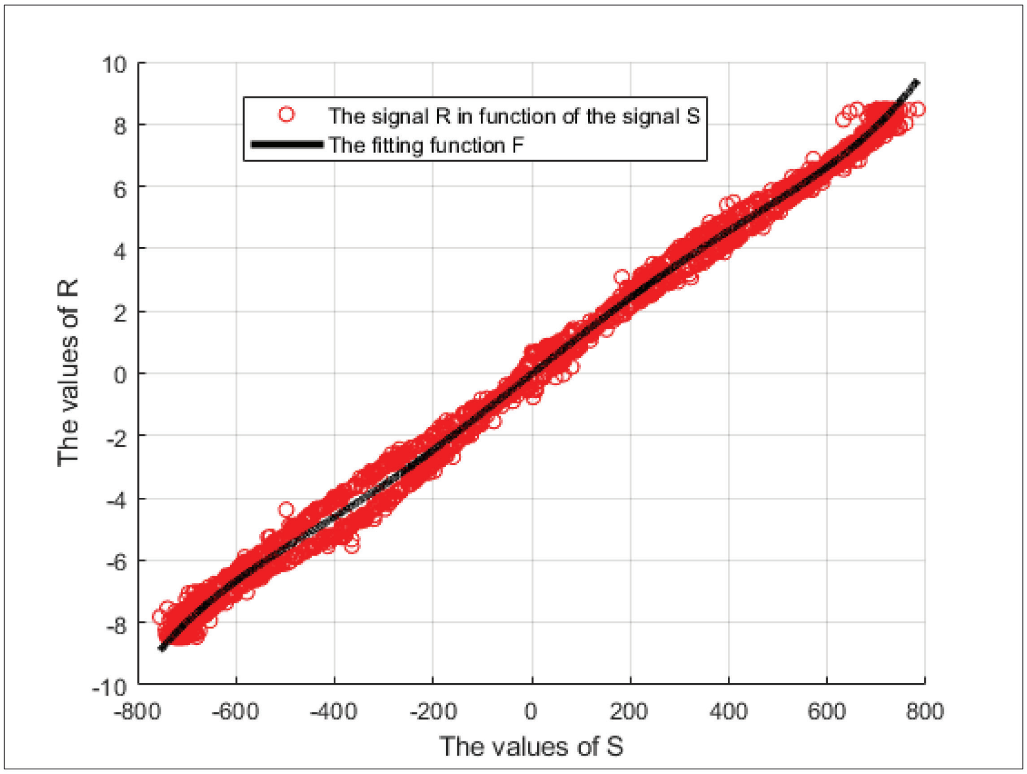

Figure 9. The scatter plot of *R*(*S*) and the estimated fitting function *F*.

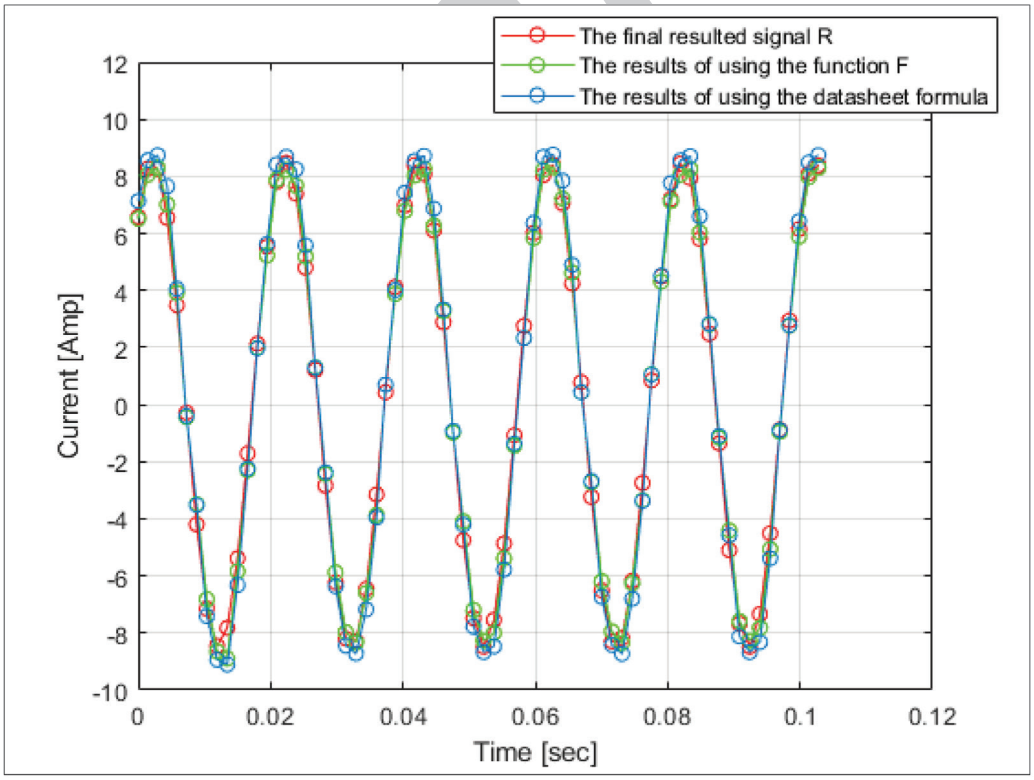

Figure 10. The results of applying *F* to *S* compared to the results of using the *ACS712T* datasheet formula and the estimated reference input *R*.

The less *d* is, the better the results of the algorithm are, the higher time execution will be. By solving the new system using the same technique in step 8, we can find the vector $\begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$. Reducing the system's dimension causes a loss of pieces of information and decreases the performance of the results, and it is not advised if it is not needed, as it can even lead to a non-convergence problem for a desired ε_{P_1} and ε_{P_2}.

It is also possible to low-pass-filter the data t_0^{error} and then select only few points to form the Vandermonde matrix. The chosen points can be done randomly or can be the mean of t_0^{error} through a number of chosen sub-intervals.

This optimization decreases the execution time of the algorithm; however, it highly affects the performances of it and can lead to undesirable results if it was not performed carefully.

5. Conclusion

To conclude, the represented algorithm provides an efficient solution for calibrating a sensor, without requiring variable, precise input values, and it saves a lot of time and effort than when using the classical techniques to calibrate the sensor for each use. It provides us with an input/output mapping function that maps the raw values of the sensor to estimated values of the output, as the results shows. The algorithm can be used during each start of the measuring device to find the best mapping function. It can also be used during the functionality of the application, where in each loop, a part of the algorithm can be executed, to minimize the execution time needed to not highly affect the elapsed time of the device; in this way, it will perform a self-correction in the measurements and, therefore, rejecting any error that can be caused by external disturbances, making the measurements robust, and that makes it very useful for any number of applications.

The accuracy of the final result of this algorithm depends on the poor degree of the used sensor. However, when choosing an appropriate fitting function, the noise in the final estimation in most of cases is a white Gaussian noise, which gives us the benefit to mix the measured values with the results from other sensors, using fusion sensor algorithms to estimate the exact value of the input.

Further work can be done by developing techniques to minimize the calculations of the algorithm, to increase its convergence speed and the performances, or as discussed before, it can also be tested in real applications and doing self-tuning during the application functionality, to increase the performance of the system, and even mixed with other sensor reading, by taking the advantage of white noise as a convenience.

References

Anderson, E., Bai, Z. and Dongarra, J. (1992). Generalized QR Factorization and Its Applications. *Linear Algebra and Its Applications*, 162, pp. 243–271. doi: 10.1016/0024-3795(92)90379-O.

Badura, M., Batog, P., Drzeniecka-Osiadacz, A. and Modzel, P. (2019). Regression Methods in the Calibration of Low-Cost Sensors for Ambient Particulate Matter Measurements. *SN Applied Sciences*, 1, p. 622.

Cordero, J. M., Borge, R. and Narros, A. (2018) Using Statistical Methods to Carry Out in Field Calibrations of Low Cost Air Quality Sensors. *Sensors and Actuators B: Chemical*, 267, pp. 245–254. doi: 10.1016/j.snb.2018.04.021.

Demeure, C. J. and Scharf, L. L. (May, 1989). Fast least squares solution of vandermonde systems of equations. In: *International Conference on Acoustics, Speech, and Signal Processing*, IEEE. pp. 2198–2210.

Djokic, B. and So, E., (2005). Calibration system for electronic instrument transformers with digital output. In: *IEEE Transactions on instrumentation and Measurement*, 54(2), pp. 479–482.

Elmenreich, W., (2002). Sensor fusion in time-triggered systems (*Doctoral dissertation, Technische*). Universität Wien.

Gao, P. (2018). Power System Current Measurement using Sensor Array Techniques. University of Alberta doctorat thesis.

Hong, S., Luo, M. and Wang, X. (2023). Control of BLDC Motor Drive with Single Hall Sensor Considering Angle Compensation. *Power Electronics and Drives*, 8(1), pp. 299–309.

Hu, Z., Chen, D., Kallel, A. Y., Wang, S. and Kanoun, O. (2022). Self-Calibrated AC Zero Potential Circuit for Two-Dimensional Impedimetric Sensor Matrices. *IEEE Sensors Journal*, 22(6), pp. 6002–6009. doi: 10.1109/JSEN.2022.3147038.

Hwang, J. Y., Park, J. H., Choi, J. H., Uhm, J. I., Lee, G. H. and Lim, H. S. (2021). A Precise Current Detection Method using a Single Shunt and FET Rds (on) of a Low-Voltage Three-Phase Inverter. *Electronics*, 11(1), p. 9. doi: 10.3390/electronics11010009.

Khan, S. A., Shahani, D. T. and Agarwala, A. K. (2003). Sensor Calibration and Compensation using Artificial Neural Network. *ISA Transactions*, 42(3), pp. 337–352.

Kurniawan, I. H., Hayat, L. and Pratama, D. K. (2022). IoT Based Electrical Energy Monitoring System. In: *AIP Conference Proceedings*, AIP Publishing, Purwokerto, Indonesia, 2578, p. 040005.

Lazarević, Đ., Živković, M., Kocić, Đ. and Ćirić, J. (2022). The Utilizing Hall Effect-Based Current Sensor ACS712 for True RMS Current Measurement in Power Electronic Systems. *Scientific Technical Review*, 72(1), pp. 27–32.

Leon, S. J., Björck, Å. and Gander, W. (2013). Gram-Schmidt Orthogonalization: 100 Years and More. *Numerical Linear Algebra with Applications*, 20(3), pp. 492–532.

Lifton, J. J. and Liu, T. (2020). Evaluation of the Standard Measurement Uncertainty due to the ISO50 Surface Determination Method for Dimensional Computed Tomography. *Precision Engineering*, 61, pp. 82–92.

Pertijs, M., (2014). Calibration and Self-Calibration of Smart Sensors. Smart Sensor Systems. In: *Emerging Technologies and Applications*, pp.17–41.

Shalamov, S. P. (2016). An induction sensor for measuring currents of nanosecond range. *Электротехника и электромеханика*, 5 (eng), pp. 57–60.

Teler, K. and Orłowska-Kowalska, T. (2023). Analysis of the Stator Current Prediction Capabilities in Induction Motor Drive using the LSTM Network. *Power Electronics and Drives*, 8(1), pp. 31–52. doi: 10.2478/pead-2023-0003.

Wu, G., Zhang, M. and Guo, F. (2020). Self-Calibration Direct Position Determination using a Single Moving Array with Sensor Gain and Phase Errors. *Signal Processing*, 173, p. 107587. doi: 10.1016/j.sigpro.2020.107587.

Xu, L., Xu, W., Cao, Z., Liu, X. and Hu, J., (2014). Multiple parameters′ estimation in horizontal well logging using a conductance-probe array. *Flow Measurement and Instrumentation*, 40, pp.192–198.

Yeong, D. J., Velasco-Hernandez, G., Barry, J. and Walsh, J. (2021). Sensor and Sensor Fusion Technology in Autonomous Vehicles: A review. *Sensors*, 21(6), p. 2140. doi: 10.3390/s21062140.

Appendix

Table A1 represents the values of the parameters used in the experiment.

Table A1. The used parameters in the experiment.

ADC resolution	12 bits
N	2,412
∇t	0.0014 s
I_{rms}	6 Amp
Initial f	50 Hz
K_{gain}	1,000
ε_{P_1}	3.2691e-18
ε_{P_2}	5.7960e-07
The degree of the chosen polynomial interpolation mapping function	5

ADC, analogue-to-digital converter.

Table A2 shows the values calculated by the algorithm.

Table A2. The algorithm's calculated values.

P_1	−2.183 e-18
P_2	4.042 e-7
f	49.9739 Hz
φ	−532.4465 rad
$F(x)$	$1.3\times10^{-14}x^5 - 9\times10^{-14}x^4 - 8.76\times10^{-9}x^3 + 3.78\times10^{-8}x^2 + 1.2552\times10^{-2}x - 2.3965\times10^{-2}$