Hindawi
International Transactions on Electrical Energy Systems
Volume 2022, Article ID 6215040, 13 pages
https://doi.org/10.1155/2022/6215040

Research Article

Real-Time Testing of Synchrophasor-Based Wide-Area Monitoring System Applications Acknowledging the Potential Use of a Prototyping Software Toolchain

Lalit Kumar,[1] **Shehab Ahmed**,[1] **Luigi Vanfretti**,[2] **and Nand Kishor**[3]

[1]*CEMSE, KAUST, Thuwal, Saudi Arabia*
[2]*ECSE, RPI, Troy, USA*
[3]*FCSEE, Østfold University College, Halden, Norway*

Correspondence should be addressed to Lalit Kumar; lalitnbd@gmail.com

Received 13 December 2021; Revised 18 June 2022; Accepted 29 June 2022; Published 30 July 2022

Academic Editor: Sobhy Abdelkader

This article presents a study on real-time testing of synchrophasor-based "wide-area monitoring system's applications (WAMS application)." Considering the growing demand of real-time testing of "wide-area monitoring, protection, and control (WAMPAC)" applications, a systematic real-time testing methodology is formulated and delineated in diagrams. The diagrams propose several stages through which an application needs to be assessed (sequentially) for its acceptance prior to implementation into a production system. However, only one stage is demonstrated in this article which comprises the use of a prototyping software toolchain and whose potential is assessed as sufficient for preliminary real-time testing (PRTT) of WAMS applications. The software toolchain is composed of two components: the MATLAB software for application prototyping and other open-source software that allows ingesting prerecorded phasor measurement unit (PMU) signals. With this software toolchain, a PRTT study is presented for two WAMS applications: "testing of the PMU/phasor data concentrator (PDC)" and "testing of wide-area forced oscillation (FO) monitoring application."

1. Introduction

The main use of synchrophasor technology has been in the field of "wide-area monitoring, protection and control" (WAMPAC) [1–4]. However, the full potential of "phasor measurement units (PMUs)-enabled wide-area network (WAN)" has not been fully realized in the wide-area monitoring and control (WAMC) center, while many novel and advanced WAMPAC applications are proposed yearly, and the rate of adoption of new applications in WAMC centers is much lower. In other words, the adoption of new applications in WAMC centers is slow despite their availability in the literature. This is because the transmission system operator (TSO) in WAMC demands the application to be tested and approved based on a certain eligibility criterion before its real-time implementation [5, 6]. This eligibility criterion differs from application to application based on how each specific application operates. However, there is one of the foremost eligibility criteria that must be met by all WAMPAC applications, that is, "real-time testing." The demand for real-time testing continues to grow, and this trend is unlikely to change with technological advancement and the deployment of low-cost PMUs [7, 8] at a large scale. The other reason behind the slow adoption of WAMPAC applications is the lack of a clear generalized and systematic real-time testing methodology in the literature.

Real-time testing poses several challenges, and it is not only needed for application testing but also for power system component testing as well [9]. Encoding an algorithm into a software application in a real-time environment requires additional expertise than what is required for offline software applications. Figure 1 depicts the gap between the TSO's demand and researchers' limitations which is the major roadblock in real-time testing of WAMPAC

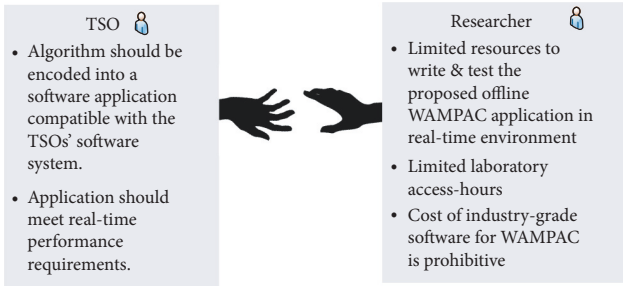

Figure 1: Depiction of the gap between TSO's demands and researchers' limitations.

applications and thus is a roadblock in its frequent upgradation in WAMC. Indeed, the WAMPAC applications should meet real-time performance requirements, but more importantly, the approach to test applications should also be systematic while at the same time considering the researchers' limitations as depicted in Figure 1. However, staff from the National Institute of Standards and Technology (NIST) of the United States have proposed a requirement test framework for applications that has not yet gained adoption, and it does not formalize a testing methodology [10]. To the best knowledge of the authors, there is no clear systematic testing methodology reported yet that could be used for real-time testing of WAMPAC applications in a time-efficient manner so that the gap depicted in Figure 1 can be reduced.

One successful method for thorough testing requires a laboratory testbed consisting communication-enabled "real-time-simulator" which may be coupled with hardware-in-loop (HIL), software-in-loop (SIL), power HIL (PHIL), and control HIL (CHIL) setup [11–15]. In [15], a synchrophasor-based voltage stability monitoring application was tested with (1) positive-sequence based (PSB) simulations, (2) PMU data from real-time-HIL, and (3) PMU data from a real Nordic grid transmission system, sequentially. The authors shared their learned lessons that any synchrophasor-based application must be tested thoroughly with actual PMU measurements so that the developed application with PSB simulations can be modified to become robust for more realistic features such as noise and outliers. However, a more common approach is to ignore all aspects of data communications, i.e., a communication-less approach and perform an application's testing only by playing back simulation data or measurement records [16].

The major issue to incorporate communication-based real-time testing in 2nd and 3rd stages lies in the fact that such testing requires all-time access to signal steam/broadcast until the application is tested satisfactorily. Unlike prerecorded data, such all-time access to signal steam/broadcast is difficult to attain in 2nd and 3rd stages due to limited laboratory access hours and security-protocol issues, etc.

With the hype of advanced laboratories, the potential of exploiting a software toolchain remained unacknowledged in the literature which can be very helpful in communication-based preliminary real-time testing (PRTT) before attempting testing in an advanced laboratory. It allows to prototype "wide-area monitoring system's applications (WAMS applications)," enhance, and test them (preliminarily) without the need of a laboratory. The PRTT stage is proposed herein as a part of a real-time testing methodology as will be discussed in detail later. The main contributions of this article are as follows:

(i) Formulation of a real-time testing methodology for synchrophasor-based WAMPAC applications

(ii) Establishing an approach to exploit a software-only toolchain for the PRTT stage of the proposed real-time testing methodology

(iii) Demonstration of the software toolchain for PRTT of WAMS/WAMS applications

The remainder of the paper is organized as follows: in Section 2, with brief discussion on the WAMPAC system, a methodology is formulated for real-time testing of WAMPAC applications. In Section 3, open-source software for real-time synchrophasor communication is discussed. Section 4 presents the software toolchain. Section 5 demonstrates the software toolchain for PRTT of WAMS/WAMS application, and lastly, the conclusions are drawn in Section 6.

2. WAMPAC System and Methodology Formulation for Its Real-Time Testing

2.1. WAMPAC System. Typical deployment of the WAN-enabled WAMPAC system is shown in Figure 2. Generally, the WAMPAC system and involved applications are divided into three categories: (1) wide-area monitoring system (WAMS), (2) wide-area protection system (WAPS), and (3) wide-area control system (WACS). Classifying how synchrophasor-based applications are tested results in two categories: (1) WAMS and (2) WAPS/WACS as depicted in Figure 2 in the red oblong label as "WAMPAC." The differences between these categories will be discussed in the sequel. In many cases, system operators may apply protection and control actions after assessing the WAMS application's results. Such an assessment based on the operator's own experience is generally referred 'situational awareness' [17], as depicted in Figure 2.

Among the two categorizations previously mentioned, the latter one, i.e., the synchrophasor-based WAPS/WACS is still in its infancy as far as real-time grid operation is concerned, and there are relatively few applications deployed in wide-area systems [18]. On the other hand, the first categorization, i.e., the synchrophasor-based real-time WAMS is a widely studied part of the WAMPAC system and is also deployed in many power systems, majorly for situational awareness purposes. The developed application under both categorizations should be tested systematically, and thus, a real-time testing methodology is formulated next.

2.2. Formulation of the Real-Time Testing Methodology. This section provides a testing methodology to reduce the gap depicted in Figure 1 by formulating a systematic real-

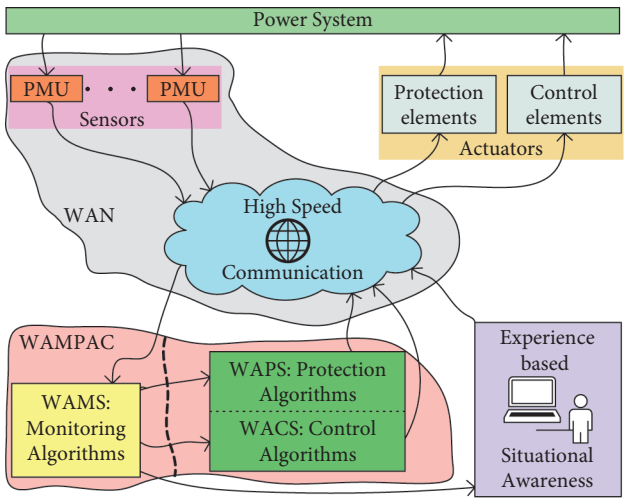

Figure 2: Typical depiction of the WAN-enabled WAMPAC system.

time testing methodology. If $\mathscr{T}$ is the specified domain of a performance metric that must be achieved by a WAMPAC application $\mathscr{A}$ in its testing, then it can be deemed successfully tested if the following condition holds true:

$$T \supset \mathscr{T}, \quad \ni \mathscr{T} = \mathbb{T}(\mathscr{A}), \tag{1}$$

where $\mathbb{T}$ is the operator that is operated on $\mathscr{A}$ to achieve its performance metrics $\mathscr{T}$ and T is the actually achieved performance. In practice, $\mathbb{T}$ can be referred as a testing methodology and $\mathscr{T}$ could be the set of performance criteria, e.g., execution time $<$ specified time, application's output index $<$ pre-defined threshold, etc. Achieving equation (1) can be regarded equivalent to the stamp of 'tested and approved' on $\mathscr{A}$.

For generalized testing of any $\mathscr{A}$, the existing literature does not report any method to formulate $\mathbb{T}$. Conventionally, this testing methodology $\mathbb{T}$ requires a costly laboratory testbed. Indeed, such a testbed is needed to test whether meets the performance criteria or not, under multiple realistic power grid conditions, before it could be deemed successfully tested. However, performing laboratory tests with naively implemented WAMPAC applications might be difficult to justify especially at initial development stages. However, a communication-enabled software toolchain can greatly help in PRTT and debugging of WAMS/WAMS applications, and thus, a significant amount of time and effort can be reduced prior to laboratory tests. Figure 3 shows the formulated methodology for real-time testing of WAMPAC applications involved in WAMPAC systems, which will be described next in detail.

2.2.1. Proposed Testing Methodology. The legend shown at the top left corner of Figure 3 should be perused first before following the remainder of Figure 3. This legend specifies a "green box," a red asterisk, and three different arrows, bold-red, dotted-red, and bold-blue. These arrows describe the paths on which methodology is applied from the one testing stage to another after passing the preceding stage. As mentioned earlier, WAMPAC applications $\mathscr{A}$ are categorized into two categories based on the involved testing methodology, i.e., $\mathscr{A}_1$: WAMS applications and $\mathscr{A}_2$: WAPS/WACS applications and $\mathscr{A} = [\mathscr{A}_1, \mathscr{A}_2]$. Similarly, the testing methodology $\mathbb{T}$ can also be segregated into two parts, i.e., $\mathbb{T}_1$: testing methodology for $\mathscr{A}_1$ and $\mathbb{T}_2$: testing methodology for $\mathscr{A}_2$ and $\mathbb{T} = [\mathbb{T}_1, \mathbb{T}_2]$. By virtue of this categorization, Figure 3 is segregated into two parts where the light-orange colored part indicates $\mathbb{T}_1$ and the light-blue colored part indicates $\mathbb{T}_2$. Both parts comprise multiple testing stages grouped in to "offline," "pseudo-online," and "online" stages and these groups are represented by gray colored boxes. Now, $\mathscr{A}_1, \mathscr{A}_2$ needs to be passed/moved along the shown path for their approval by their respective parts $\mathbb{T}_1, \mathbb{T}_2$ comprised their respective grouped-stages. Formulated methodology can be understood by thorough observation shown in Figure 3 in conjunction with the following noteworthy points:

(i) The "testing" and "enhancement" of $\mathscr{A}$ are interlinked to each other. Testing results may call for enhancements to the application and enhancement would call for testing. Therefore "testing" and "enhancement" are a recursive process that ends when testing results yield specified performance metrics or until equation. (1) is achieved.

(ii) As depicted in the column's head in Figure 3, the application not only is being tested in the offline/online stages but will also go through debugging (enhancements). How fast an application performs to meet real-time requirements, depends on how $\mathscr{A}$ is implemented in the software prototype.

(iii) Before moving to laboratory testing (red asterisk boxes in Figure 3) with $\mathscr{A}$, it is desirable to apply the testing stages (unasterisk boxes in Figure 3) to debug $\mathscr{A}$, which will minimize the cost and effort of performing laboratory tests

(iv) Differences in $\mathbb{T}_1$ and $\mathbb{T}_2$ are as follows: (1) as apparent from Figures 2 and 3, the online stages of $\mathbb{T}_1$ require PMU signals to be broadcasted as input, whereas the stages of $\mathbb{T}_2$ require the ascertained monitoring results by $\mathbb{T}_1$ as input. (2) $\mathbb{T}_2$ essentially requires the power system modelling data in all its stages, whereas it is not required in all stages of $\mathbb{T}_1$, as apparent from Figure 3 because synchrophasor-based monitoring not necessarily needs the power system modelling data.

(v) The findings of WAPS/WACS depend on the monitoring results that are the output of the WAMS applications. These results are important as the performance metrics of $\mathscr{A}_1$ aid in bounding those of $\mathscr{A}_2$, as can be observed in Figures 2 and 3.

(vi) For $\mathscr{A}_1$ to be tested in $\mathbb{T}_1$, the attributes to be monitored in the PMU signals must be known a priori by ancillary studies, e.g., eigenvalue analysis, online test case libraries [19], or self-created attributes via simulation.

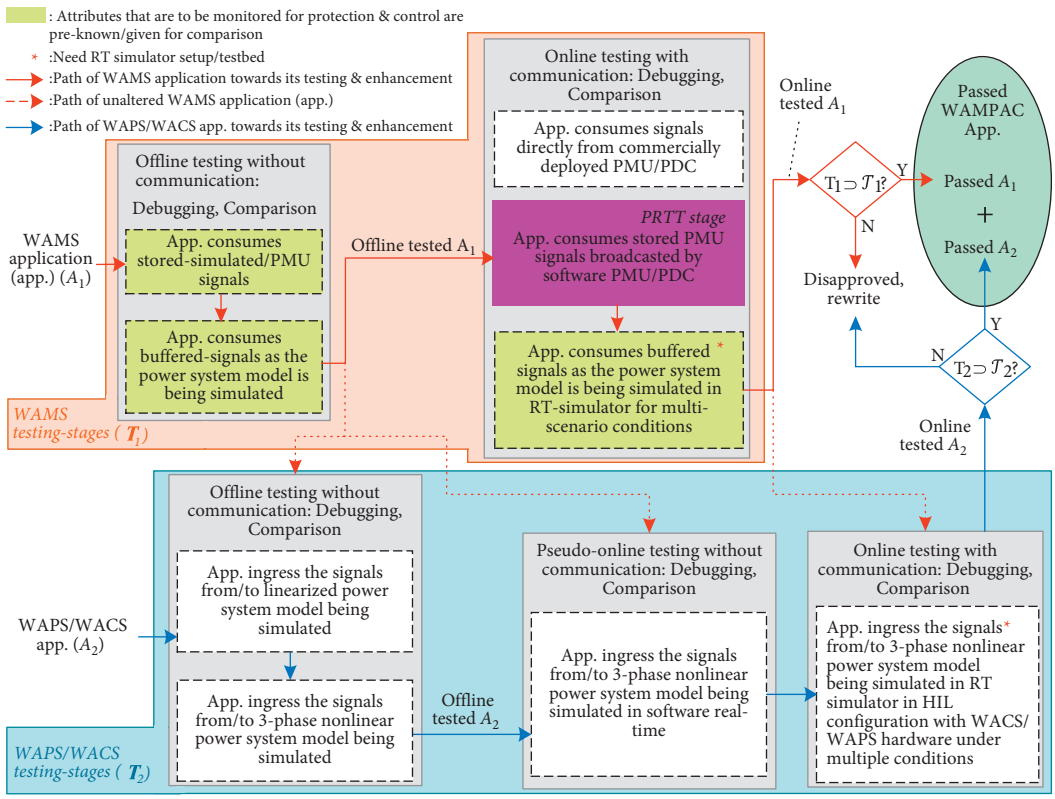

Figure 3: Testing methodology ($\mathbb{T}$) for WAMPAC applications ($\mathscr{A}$).

(vii) Condition in equation (1) can be divided into $T_1 \supset \mathcal{T}_1$ (corresponding to $\mathscr{A}_1$) and $T_2 \supset \mathcal{T}_2$ (corresponding to $\mathscr{A}_2$). Meeting such conditions implies that requirements of $\mathscr{A}_1$ and $\mathscr{A}_2$ have been fulfilled in their PRTT, as depicted in the upper right corner of Figure 3.

The stage highlighted by the box with a pink outline in $\mathbb{T}_1$ (see Figure 3) is identified as the most important stage (PRTT stage) in the formulated testing methodology which avails the facility of the pseudo online environment. This stage utilizes a software toolchain to test $\mathscr{A}_1$ that can broadcast the prerecorded PMU data similar to how commercial PMUs stream data and where the data can be retrieved by an application $\mathscr{A}$ in real time. Even though the PRTT stage is not sufficient for commercial real-time testing because it uses the prerecorded data, still, it helps in preliminary testing and enhancement of $\mathscr{A}_1$ before entering the final testing stage (red asterisk box in $\mathbb{T}_1$). The remainder of this article presents a study that is focused only on the PRTT stage.

3. Open-Source Software for Real-Time Synchrophasor Communication

Real-time synchrophasor communication requires industry-specific communication protocols (IEEE C37.118.2) between PMUs, PDCs, and synchrophasor applications. In the recent years, several efforts have been made to develop and release open-source software [12, 20–26] tools that support synchrophasor communications and application development. These software tools can be categorized into two types: (1) signal broadcaster (publisher and server) and (2) signal retriever (subscriber and client). The former type serves to broadcast prerecorded/stored PMU signals as similarly performed by commercial PMUs/PDCs, and the latter type helps retrieve the broadcasted signal. Table 1 lists such pieces of open-source software, some of which were briefly reviewed by the authors of [12], which could be followed there if required. The detailed comparative study between these listed pieces of software is out of the scope of this article. A software toolchain can be formed by combining one signal broadcaster and one signal retriever, and one of each type is discussed next.

3.1. PUPO: Signal Broadcaster. Among the signal broadcasters listed in Table 1, 'PMU-PDC stream simulator Or' (PUPO) [24] is found to be a suitable signal broadcaster to form the software toolchain, which is developed in C++. To briefly describe PUPO, the main GUI is shown in Figure 4. The merits of PUPO over others are summarized as follows:

(i) Easy distribution and deployment (by an '.exe' file or small size).

(ii) A simple and intuitive user-friendly graphical user interface (GUI).

(iii) No complex installation is required, and the main GUI opens directly on clicking the 'StrongridSimulatorGUI_MFC.exe' file.

Table 1: Open-source software for real-time communication.

Signal broadcaster	Signal retriever
(i) PMU-PDC stream simulator (C++) [24]	(i) SADF (MATLAB) [12]
(ii) PyPMU (Python) [20]	(ii) S3DK (Labview) [25]
(iii) iPDC (Linux) [22]	(iii) Babelfish (Labview) [23]
(iv) OpenPDC (Java) [26]	(iv) Phasor toolbox (Python) [21]
	(v) OpenPDC (Java) [26]

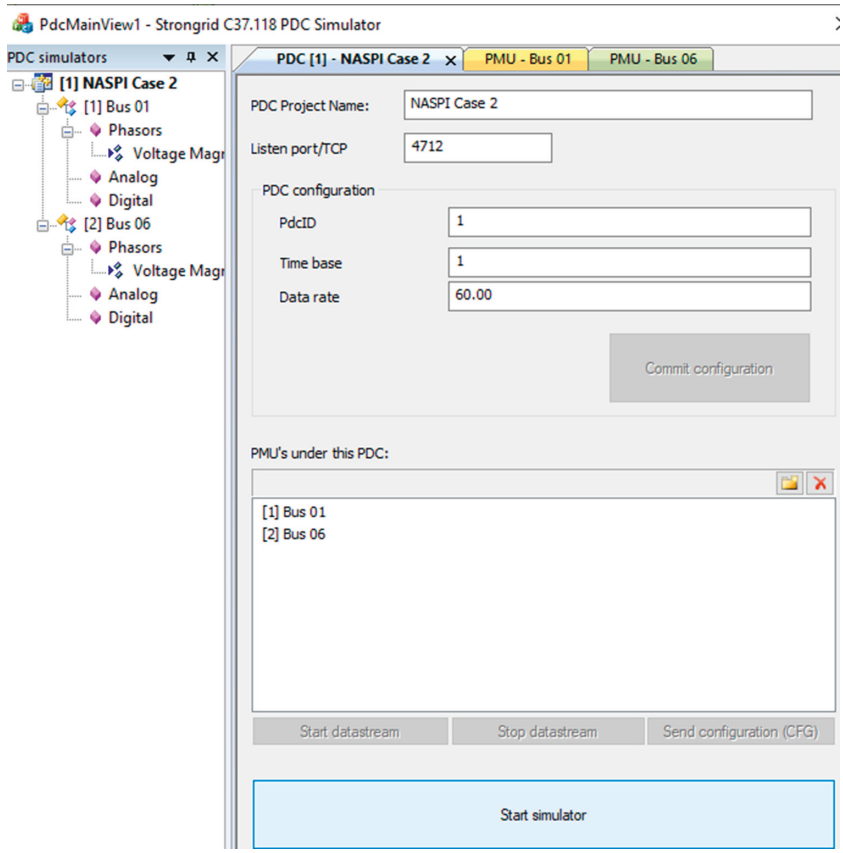

Figure 4: GUI of PUPO.

Being a PMU/PDC emulator, the inner workings of PUPO should not be mistaken with a conventional software PDC. PUPO was designed to emulate the PMU/PDC data streaming functionalities by broadcasting the prerecorded signals by following the same protocols used by commercial PMUs/PDCs. The current version of PUPO consumes the prerecorded data in a.csv file (editable in any tool, e.g., Notepad/Notepad++) and uses an extension name, ".phcsv". The prerecorded data file can be imported by choosing the option 'file-based input' given under each phasor tab. The polar form of the PMU data given in the "csv" file has to be first converted in a rectangular form before entering it into the input file. The method of preparing the input file and other necessary steps are described in [24]. By default, the PUPO has one PMU station in which three phasor signals can be imported. However, both of these numbers can be increased/decreased by clicking on the 'folder/cross' icon as depicted in Figure 4.

3.2. SADF: Signal Retriever. The recently developed "synchro-measurement application development framework (SADF)" [12] is the MATLAB-based software that enables the retrieval of TCP, UDP, or mixed TCP/UDP synchro-measurement data. To the best of the authors' knowledge, SADF is the only MATLAB-based open-source toolbox available for such a purpose to date. It enables MATLAB to retrieve the PMU data in real time with additional facility to perform parallel computations leveraging the parallelization tools within MATLAB. SADF can be used for real-time prototyping of WAMPAC applications [12]. Due to the promising potential of SADF and vast functionalities available with the widely used MATLAB, SADF was selected from the signal retrievers given in Table 1 to form the software toolchain.

The script, "SADF_setting" allows us to configure the PMU/PDC connection settings such as the TCP/IP, port, device ID, and maximum time of retrieval. The main script,

'SADF_run' allows us to embed a custom code for a WAMPAC application [12]. The user is required to develop an understanding of the default script, 'demo_WAMS' to restructure and encode their WAMPAC application as per the compatibility with SADF. This default script is given to plot the retrieved signal in real time with its specifications mentioned on the plot. The retrieved signal is being stored in the MATLAB workspace which can also be utilized later for the offline analysis.

4. PUPO-SADF Software Toolchain

As indicated before, PUPO and SADF are the simplest software in their respective categories (Table 1) that can communicate with each other through the TCP/IP protocol (Internet) on a single computer. By the combination of these, a software toolchain is formed named, 'PUPO-SADF toolchain,' which is found suitable for the PRTT stage (pink-outlined box in Figure 3). This software toolchain provides a real-time environment for testing and enhancement of any $\mathscr{A}_1$ at the PRTT stage.

The flowchart for the complete PUPO-SADF toolchain is shown in Figure 5. Firstly, the prerecorded PMU data given in the standard "csv" file need to be edited in any text editor (e.g., Notepad/Notepad++). This file is then imported into the PUPO, and then, the broadcast can be started as per IEEE C.37.118.2 compliance by pressing the 'start simulator' button as depicted in Figure 5. Then, SADF can retrieve the data in real time and can execute $\mathscr{A}_1$ (with parallelization, if needed) consuming the retrieved data segment/window $w(n)$. The time length of this window T_w can be defined by the user as per their choice at the beginning of the encoded application $\mathscr{A}_1$. The computed results can be monitored in real time through the MATLAB plot of an individual's choice as written in the encoded application $\mathscr{A}_1$.

4.1. Monitoring Resolution: Speed in Real-Time Monitoring. For any application, $\mathscr{A}_1$ to be executed in real-time, the computational time, $T_c^{\mathscr{A}_1}$ must be lower than T_w i.e. $T_c^{\mathscr{A}_1} < T_w$ [27]. The authors [28] have used the term 'near real-time' instead of 'real-time.' Indeed, one application can achieve real-time requirements better than another based on $T_c^{\mathscr{A}_1}$. Therefore, a new index, i.e., "monitoring resolution" Ψ is defined herein as follows:

$$\Psi_{\mathscr{A}_1} = \left(1 - \frac{T_c^{\mathscr{A}_1}}{T_w}\right) \times 100. \quad (2)$$

Ψ relates to the speed of the $\mathscr{A}_1$ in real time and $\Psi = 100\,\%$ is practically impossible because no application can produce the results in zero seconds. Therefore, the value of Ψ lies in the range of $0 < \Psi < 100$, and a high value of Ψ signifies that real-time performance is being met for $\mathscr{A}_1$. However, the index Ψ should not be mistaken for the overall performance of any application. Ψ is just one index among several indices required to govern overall performance. For example, the application developed in [29] for oscillation monitoring utilized a mode decomposition technique, and thus, it will give low Ψ due to the heavy computation burden involved in comparison to a simple power spectral density (PSD) operation. On the other hand, the application developed by authors of [29] would give better oscillation detection results subjected to mode mixing problems [29]. In other words, if there exists any detection accuracy index $\mathfrak{D}$, then the application described in [29] would have a better score than the PSD operation, but Ψ would be compromised. It is obvious that the application's overall performance would be a function of Ψ and $\mathfrak{D}$, as well as other indices. There are no sets of indices defined in the literature to govern the global performance of a particular synchrophasor application yet, and further discussion on this aspect is also omitted here.

As mentioned before, the proposed software toolchain provides a real-time environment for testing and enhancement of any $\mathscr{A}_1$ at the PRTT stage shown in Figure 3. After qualifying this stage, the application moves to the next and final testing stage (red asterisk box) where a real-time-HIL laboratory testbed is required. If any $\mathscr{A}_1$ is qualified/approved in the PRTT stage, then it is expected to meet the performance requirements of the final stage, i.e., the red asterisk box/stage of $\mathbb{T}_1$ in Figure 3 or at least it will assist in the final testing stage. Next, the PUPO-SADF toolchain for the PRTT is demonstrated.

5. Real-Time Testing of Synchrophasor-Based WAMS Using the PUPO-SADF Toolchain

Before continuing, it is important to mention here that the PUPO-SADF toolchain is exclusively utilized in the PRTT stage; however, the SADF can be utilized in another online testing also where the signal stream is accessible in real time. A demonstration of the PUPO-SADF toolchain is presented in two WAMS applications (1) testing/validation of PDC emulation and (2) testing of wide-area forced oscillation (FO) monitoring application. Even though the first one is not related to application testing, it helps to verify that PUPO can be used to develop monitoring applications with it.

The prerecorded PMU data from North American synchrophasor initiative (NASPI) [30] are considered for the analysis. The data file "NASPI-2014-Workshop-Oscillation-Case2.csv," which belongs to "oscillation detection: test Case 2", consists of 30 signals of 10 min sampled at 60 Hz. The prerecorded voltage phasor signals from Bus-06 and Bus-01 are broadcasted using PUPO which can now be treated as if they were wide-area PMU signals streaming in real time from a commercial PDC.

5.1. Testing of PUPO for Authentic WAMS Results. Testing of PMU/PDC compliance is one of the important parts of WAMPAC application's testing [31]. The testing of commercial PDC requires a rigorous approach considering several compliance aspects; however, in the case of PUPO's testing in this article, the objective is to ensure validity of results for real-time WAMS applications. The broadcasted signals are retrieved by the SADF by inputting TCP/IP and port settings of PUPO. Testing of PUPO is a postretrieval process. The retrieved 10 min voltage magnitude signals

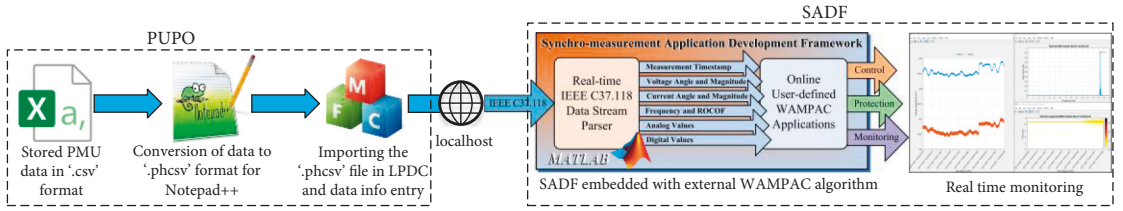

Figure 5: Flowchart of the PUPO-SADF toolchain.

along with time stamps are stored in the MATLAB workspace in the variable, "DATA.Magnitude' and 'DATA.-Timestamp." Now, the testing of PUPO includes two aspects to validate real-time monitoring results as follows.

5.1.1. Check for Synchronized Overlapping. The recorded NAPSI signals are already synchronized in its data file (.csv) [30], and it is necessary to ensure that the signals while being broadcasted by PUPO should also remain synchronized. This check can be performed by plotting retrieved and broadcasted signals in one figure. If the retrieved signals perfectly overlap the respective broadcasted signals, then it can be accepted that the PUPO had broadcasted the signals correctly. To this end, 'array alignment' (time-frame shifting) of the retrieved signals with prerecorded (PUPO-broadcasted) signals needs to be assessed. This is because the PUPO broadcasts the imported array/signals (".phcsv") in a loop without encountering any time delay in streaming from the end sample to the first one, and thus, it is unlikely that the first sample of the retrieved array/signal in the MATLAB workspace is exactly the same as that of the inputted signal in PUPO. Whatever array-alignment rule is followed for one array (signal) should also be followed by all other arrays. If the same array-alignment rule can overlap all retrieved signals with respective broadcasted signals, then the 'synchronized overlapping check' can be passed. Figure 6(a) shows the plot of both prerecorded ('.csv') and both retrieved signals by following the one single 'array-alignment' rule. It can be seen that both the retrieved signals overlap with the respective broadcasted signals; therefore, this check can be passed for PUPO.

5.1.2. Check for the Sampling Rate. The purpose of this check is to verify that the PUPO has broadcasted the prerecorded signals at its original sampling rate, F_s, and if it is uniform throughout. This is an important test as $\mathscr{A}_1$ may require F_s to compute the monitoring results, e.g., $\mathscr{A}_1$ comprising the PSD operation will require F_s. This check can be performed by plotting the retrieved timestamps stored in the workspace variable, "DATA.TimeStamp." Figure 6(b) shows the plot of retrieved timestamps versus the sample number in which it is apparent that the PUPO had indeed broadcasted at the original sampling/data rate $F_s = 60\,Hz$ as 60 samples (x-axis) have been retrieved every one second (y-axis), and also, this F_s remained uniform. Therefore, this check is also passed for PUPO. Thus, PUPO has been tested "OK" for the testing objective defined earlier. However, Figure 6(b) suggests one bug in the current version of PUPO that does not affect the defined testing objective, but it will create a problem in plotting the retrieving signal. The identified bug is further discussed and rectified next.

5.1.3. Discussion on an Identified Bug and Its Rectification. Looking at the x-axis and y-axis in Figure 6(b), it can be ascertained that the retrieved timestamps are following the staircase pattern instead of a linear pattern because the PUPO is streaming the same timestamp 60 times in one second. This is because the GPS-clock quality of PUPO is limited to seconds (HH:mm:ss) and not up to milliseconds (HH:mm:ss:sss). This corresponds with PUPO's own repository [24] which states that the "commit button does not gray out for setting PMU station settings." This is also shown here with the help of Figure 7 where it can be seen that due to nonfunctioning of the highlighted commit button, the clock quality of PUPO cannot be more granular. To further understand the problem with this bug, notice that the bug has caused the repeated entries at the y-axis/array in Figure 6(b) which correspond to the timestamps at the x-axis in the real-time signal. Therefore, the real-time plot would not be possible unless the repeated timestamps, i.e., staircase patterns in Figure 6(b) are corrected with a linear pattern.

Notice that this rectification is equivalent to improving the clock quality of PUPO. This is achieved by using the 'linspace' command on retrieving the data window $w(n)$ in the embedding $\mathscr{A}_1$, which is as follows:

$$w_{\text{timestamps}}(n) = \text{linspace}\left(w_{\text{timestamps}}(1), w_{\text{timestamps}}(\text{end}), T_w \times F_s\right). \tag{3}$$

Readers may follow next to next figure before reading further to observe how the real-time signal retrieving plot is made possible by rectifying the bug as described above. Notice that the PUPO's bug is not rectified in PUPO but is rectified in the SADF. Next, the PUPO-SADF toolchain as a whole provides a suitable mean for PRTT of any $\mathscr{A}_1$.

5.2. PRTT of Wide-Area FO Monitoring Application

5.2.1. FO Monitoring Application. In this section, the FO monitoring application utilizes the magnitude-squared coherence (MSC) tool [16] for spectral analysis. The MSC estimate is a powerful spectral tool for oscillation monitoring that has been shown to yield better results as compared to the PSD [16]. The authors in [27] indicated that without having any prior information it is very challenging to distinguish oscillations from ambient responses by only

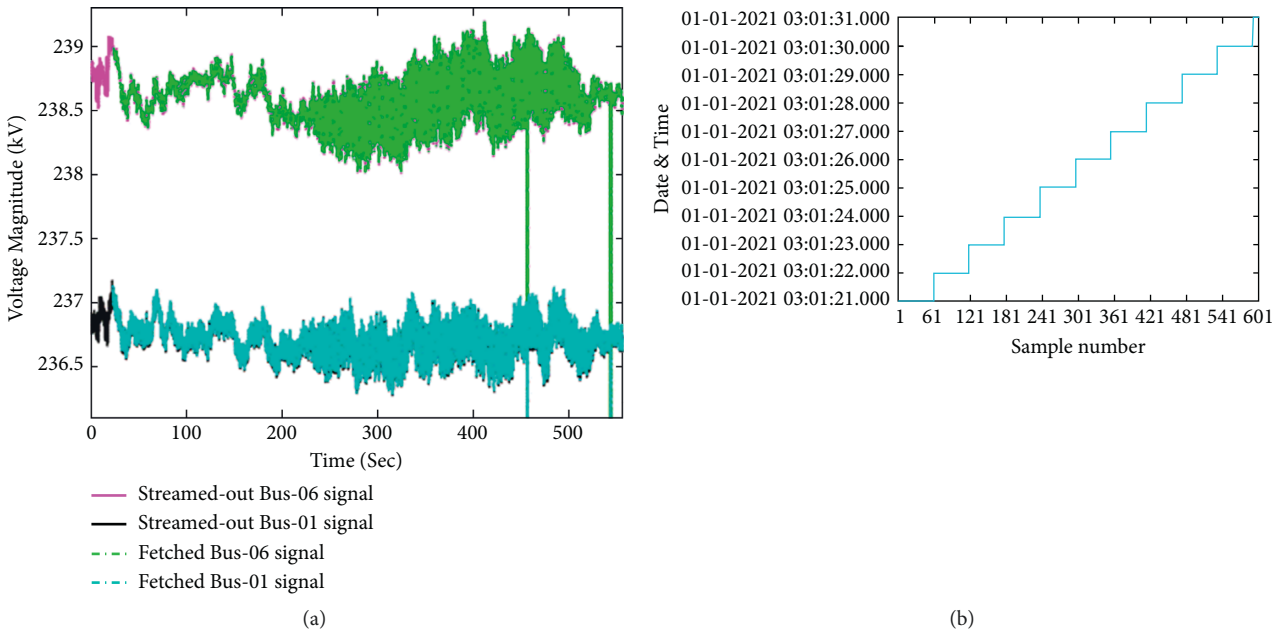

Figure 6: (a) Prerecorded and retrieved voltage signals. (b) Retrieved timestamps.

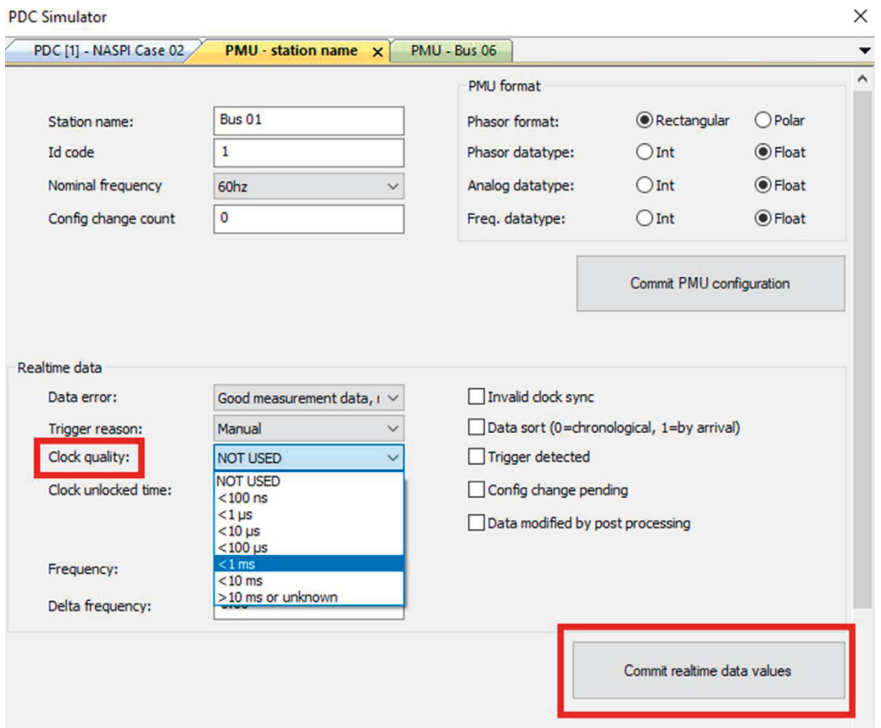

Figure 7: Depiction of the disabled commit button in the PUPO.

using PSDs and spectrums. The MSC estimate C_{xy}, is a function of frequency f which reflects how well the sequence $x(n)$ correlates to another sequence $y(n)$ at any frequency f where $f \in \mathrm{R}$. The region of space R maps the space $[0(F_s/2)]$ where F_s is the common sampling frequency of both the sequences $x(n)$and$y(n)$. The mathematical expression of MSC is written in the following equations [27, 32]:

$$C_{xy}(f) \triangleq \frac{\left|P_{xy}(f)\right|^2}{P_{xx}(f)P_{yy}(f)}, \tag{4}$$

$$0 \le C_{xy}(f) \le 1, \quad \forall f \in R, \tag{5}$$

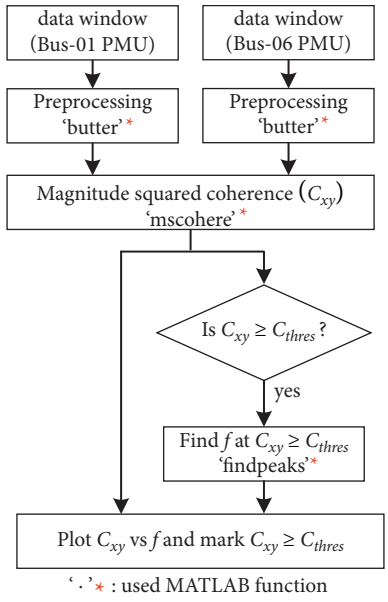

Figure 8: Main steps in the "smart-WAMS" application (A1).

where $P_{xx}(f)$ and $P_{yy}(f)$ are the PSDs of $x(n)an\ dy(n)$, respectively, and $P_{xy}(f)$, is the cross PSD (CPSD) between $x(n)an\ dy(n)$. The values of the MSC estimate are real and lie between 0 and 1 as indicated in equation. (5). If $x(n)an\ dy(n)$ are two sinusoidal signals of frequency f_x and f_y, respectively, then the following equation holds true:

$$\left.\begin{aligned} C_{xy}(f_x) = C_{xy}(f_y) = 1if, \quad f_x = f_y \\ C_{xy}(f_x) = C_{xy}(f_y) = 0\,if, \quad f_x \neq f_y \end{aligned}\right\}. \tag{6}$$

The application utilizing the MSC estimate is encoded in the script, "smart_WAMS" ($\mathscr{A}_1$) using the 'mscohere' function of the MATLAB's 'signal processing toolbox' that computes the MSC estimate between two signals. Before MSC, the application first preprocesses the signals by using the 1st order high-pass Butterworth filter with a cut-off frequency of 0.01 Hz. Increasing the window length T_w increases the frequency resolution but reduces the time resolution [27]. The time length of the data window T_w is considered 30 sec herein. A Hamming window is chosen to minimize leakage noise [27].

The application also incorporates an automatic peak detector with the help of the function "findpeaks". This function detects the peaks in the MSC estimate in real time and shows it on the MSC plot to provide real-time visualization of the FO frequency. The threshold $C_{xy}^{\text{thres}}(f)$ for peak detection is set to 0.1; i.e., if $C_{xy}(f) \geq C_{xy}^{\text{thres}}(f)$,then the peak will be detected in MSC indicating the FO detection at its corresponding frequency. The major steps in the encoded 'smart_WAMS' application are delineated in the flowchart shown in Figure 9.

5.2.2. Real-Time FO Monitoring Using the PUPO-SADF Toolchain. The script "smart_WAMS" is embedded in the main script of SADF, i.e., "SADF_run". The script "SADF_run" is edited to compute the MSC and plot it which is being updated in real time. The script 'smart_WAMS' also plots the segmented MSC in real time. The method to plot the segmented MSC is the same as for the segmented PSD and segmented self-coherence described in [32]. Here, the 30 sec window $w(n)$ is slid 30 times for the next retrieving 30 sec. As a result, 30 MSC estimates are obtained and plotted as a segmented MSC. The recorded screen capture for this real-time retrieval and monitoring can be viewed in [33]. As the SADF begins retrieving and plotting the signals, the MSC plot (colormap) begins showing the results only after the retrieval of the predefined length of the window T_w. The retrieving signals and their MSC estimate are also shown in Figures 8(a) and 8(b), respectively, for one instant in the real-time monitoring.

As the embedded script "smart_WAMS" runs and updates the figures in real-time [33], it is also made to save the colormap (segmented MSC) every 30 secs by using the "saveas" command placed in the script. The purpose of performing this is to show the real-time monitoring [33] by a set of relevant figures (video frames), as shown in Figure 10.

5.2.3. Summary of Figure 10 Results. From Figures 6(a) and 10 and the recorded screen [33], it can be noticed that the first frame (Figure 10) is associated with the early sec of the signals (Figure 6(a)) wherein the 13.3 Hz FO is observable. The second frame (Figure 10) shows the monitoring of pre-event seconds (Figure 6(a)) wherein it can be noticed that the spectrum band of the 13.3 Hz FO has widened. The occurrence of an event at 272 seconds (Figure 6(a)) triggered another 1.25 Hz FO which was sustained over the event's duration. The third frame (Figure 10) shows the monitoring of the signals during the period when the first outliers appear in the data (see Figure 6(a)). The presence of outliers can be

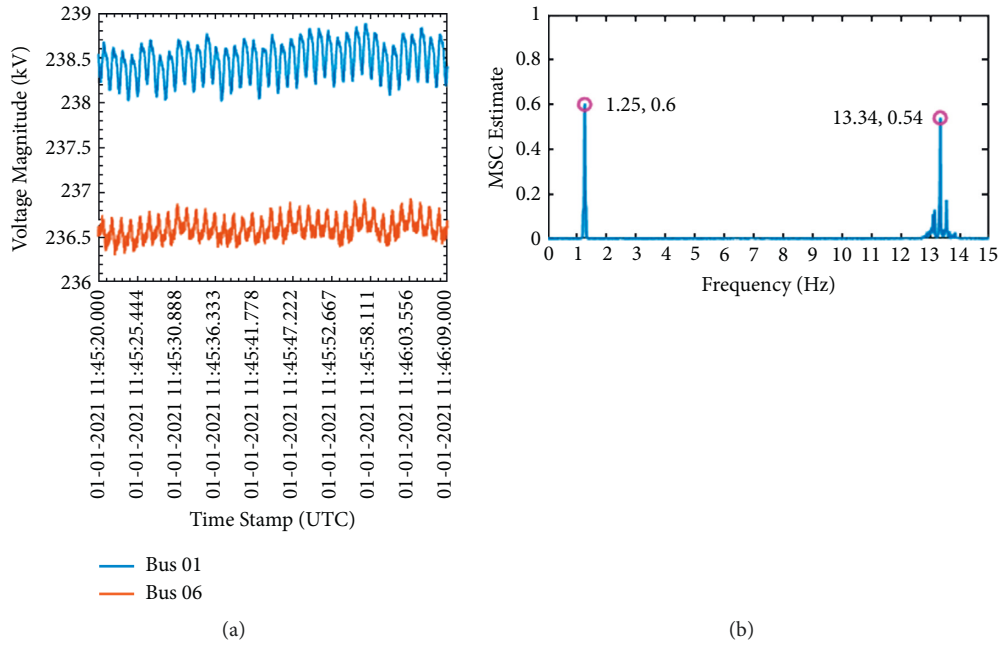

Figure 9: (a) Real-time retrieval of broadcasted PMU signals. (b) Real-time FO monitoring using MSC estimate.

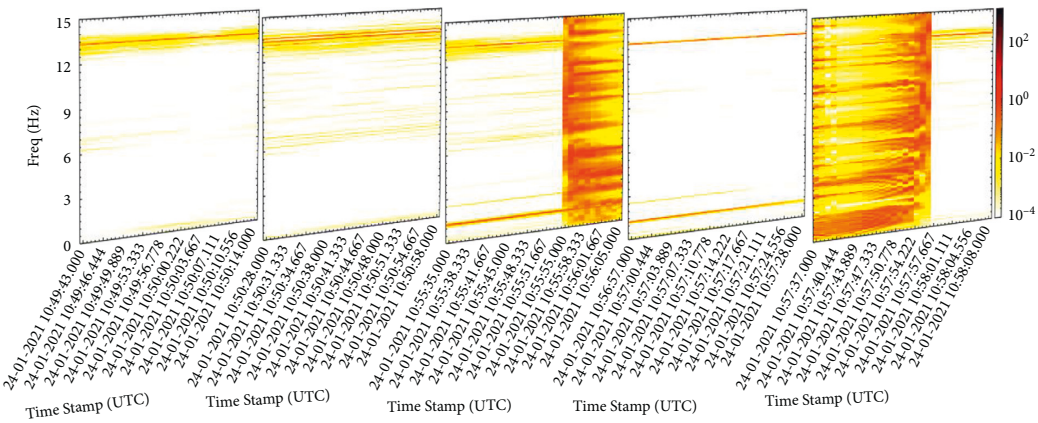

Figure 10: Real-time FO monitoring using thePUPO-SADF toolchain.

noticed along with the newly triggered 1.25 Hz FO. When outliers are present, the MSC estimate does not provide useful information. It can be noticed in the fourth frame (Figure 10) that after the first outlier is passed, both FOs sustained, and the widened spectrum band of 13.3 Hz FO has reverted to narrow. The last frame is the monitoring results during the period when additional/second outliers (Figure 6(a)) finished passing through the window $w(n)$. It can be noticed that after the outliers are passed and after the event stopped at around 587 seconds (Figure 6(a)), the 1.25 Hz FO disappears and 13.3 Hz FO still sustains. The discussed analysis can be verified from the video described in [33]. The presence of these FOs is also confirmed by the authors of [34] in an offline study.

5.2.4. Speed of "Smart_WAMS": Discussion on Ψ. It is observed that the computation time of "smart_WAMS" $T_c^{\text{smart\_WAMS}}$ varies. On a computer with characteristics, Windows 10 OS, Intel(R) Core(TM) i7-8550U CPU @ 1.80 GHz, 16 GB RAM, the computation times $T_c^{\text{smart\_WAMS}}$ are measured for 709 retrieved windows, $w(n)$ with T_w = 30secs, $F_s = 60\,Hz$. The measurements are presented in a form of the normalized histogram shown in Figure 11. The average $T_c^{\text{smart\_WAMS}}$ and average $\Psi_{\text{smart\_WAMS}}$ are obtained as $T_{c_{avg}}^{\text{smart\_WAMS}} = 0.8961$ sec and $\Psi_{\text{smart\_WAMS}_{\text{avg}}} = 97.01\,\%a$, respectively, which indicates that this particular application has acceptable performance and thus can pass the PRTT stage to move on to the final testing stage (red asterisk box in Figure 3).

5.2.5. Discussion. Next, the additional potential of the software toolchain to be used in the final testing stage is discussed. As mentioned before, the PRTT stage is the only stage that is demonstrated in this article; however, there are potential uses in the final testing stage, i.e., the red asterisk

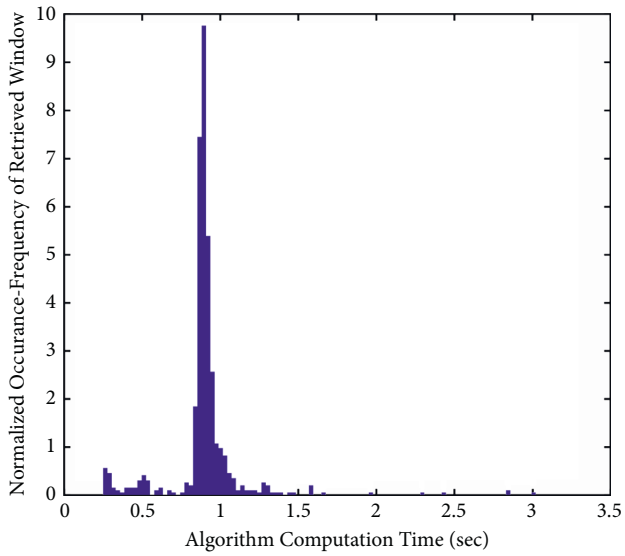

Figure 11: Real-time FO monitoring using the segmented MSC estimate.

box/stage of $\mathbb{T}_1$ in Figure 3. After being approved in the PRTT stage, the designed 'smart_WAMS' application is ready to move to its final testing stage, where the only modification required is to change the connection IDs in the SADF's script from PUPO's IDs to laboratory PMU/PDC's IDs. No further efforts have to be made in a laboratory in rewriting the application unless it fails tests when subjected to multiple realistic power grid conditions. The software toolchain can help here too by storing the PMU/PDC signals recorded from the failed tests and replaying them using the toolchain to debug the application while avoiding the cumbersomeness and limited access time at the laboratory. In this way, the PRTT stage incorporating the PUPO-SADF toolchain is highly beneficial in professional testing of WAMPAC applications.

6. Conclusions

This article presents a study on real-time testing of synchrophasor-based WAMSPAC applications. A real-time testing methodology is formulated and specified in multiple stages. A few offline testing stages are incorporated along with online testing stages, and it is expected that the application passes (for approval) all the stages sequentially, making the testing process more efficient and less cumbersome. The potential uses of a software toolchain to assist in the testing process were acknowledged and found sufficient for PRTT of WAMPAC applications. The software toolchain is formed using MATLAB and the PUPO open-source software which is referred as the PUPO-SADF toolchain. With this software toolchain, the PRTT is demonstrated in two WAMS applications, i.e., 'testing of PMU/PDC' and 'testing of wide-area FO monitoring application'. In the first one, a minor bug was found and rectified in custom encoding within the SADF, while in the latter one, FOs at frequencies 13.33 Hz and 1.25 Hz are detected and monitored in real time using the recorded PMU data provided by NASPI.

Abbreviations:

WAMPAC: Wide-area monitoring, protection, and control
PMU: Phasor measurement unit
PDC: Phasor data concentrator
FO: Forced oscillation
PRTT: Preliminary real-time testing
WAN: Wide-area network
WAMC: Wide-area monitoring and control
TSO: Transmission system operator
HIL: Hardware-in-loop
SIL: Software-in-loop
PHIL: Power HIL
CHIL: Control HIL
WAPS: Wide-area protection system
WACS: Wide-area control system
PUPO: PMU-PDC-stream simulator
SADF: Synchro-measurement application development framework
NASPI: North American synchrophasor initiative
MSC: Magnitude-squared coherence
PSD: Power spectral density
CPSD: Cross PSD
T, achieved: Performance
$\mathscr{A}$: WAMPAC application
$\mathbb{T}$: Testing methodology
$\mathscr{T}$: Specified domain of performance-eligibilities
$\mathscr{A}_1$: WAMS-applications
$\mathscr{A}_2$: WAPS/WACS applications
$\mathbb{T}_1$: Testing methodology for $\mathscr{A}_1$
$\mathbb{T}_2$: Testing methodology for $\mathscr{A}_2$
C_{xy}: MSC estimate
$P_{xx}(f)$: PSD
T_w: Window length
T_c: Computation time
Ψ: Monitoring resolution.
$\lim_{x \longrightarrow \infty}$

Data Availability

The utilized data can be sought by approaching NASPI in the link https://www.naspi.org/.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was carried out under the project, 'Gridx: The Autonomous Digital Grid' funded by the 'King Abdullah University of Science and Technology, Saudi Arabia' under grant OSR-2019-CoE-NEOM-4178.12 as a part of the Kingdom's vision, "New Future," and "New Enterprise Operating Model" (NEOM-2030). The fourth co-author acknowledges the funding support received by the Department of Science and Technology (DST), New Delhi,

Govt. of India, granted with a SERB No: CRG/2019/000951.

References

[1] H. Haugdal, K. Uhlen, D. Muller, and H. Johannsson, "Estimation of oscillatory mode activity from PMU measurements," in *Proceedings of the IEEE PES Innovative Smart Grid Technologies Conference Europe*, pp. 201–205, Institute of Electrical and Electronics Engineers (IEEE), Piscataway, NJ, USA, 2020.

[2] N. Kishor, K. Uhlen, and S. R. Mohanty, "Identification of coherency and critical generators set in real-time signal," *IET Generation, Transmission and Distribution*, vol. 11, no. 18, pp. 4456–4464, 2017.

[3] I. Kamwa, A. K. Pradhan, and G. Joos, "Adaptive phasor and frequency-tracking schemes for wide-area protection and control," *IEEE Transactions on Power Delivery*, vol. 26, no. 2, pp. 744–753, 2011.

[4] A. Ashrafi and S. M. Shahrtash, "Dynamic wide area voltage control strategy based on organized multi-agent system," *IEEE Transactions on Power Systems*, vol. 29, no. 6, pp. 2590–2601, 2014.

[5] S. Hofsmo Jakobsen and K. Uhlen, "Testing of a hydropower plant's stability and performance using PMU and control system data in closed loop," *IET Generation, Transmission and Distribution*, vol. 13, no. 23, pp. 5339–5348, 2019.

[6] S. M. Hur Rizvi, P. Kundu, and A. K. Srivastava, "Hybrid voltage stability and security assessment using synchrophasors with consideration of generator Q-limits," *IET Generation, Transmission and Distribution*, vol. 14, no. 19, pp. 4042–4051, 2020.

[7] D. M. Laverty, R. J. Best, P. Brogan, I. Al Khatib, L. Vanfretti, and D. J. Morrow, "The OpenPMU platform for open-source phasor measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 4, pp. 701–709, 2013.

[8] S. Chauhan and R. Dahiya, "Multiple μPMU placement solutions in active distribution networks using nonlinear programming approach," *International Transactions on Electrical Energy Systems*, vol. 31, no. 11, Article ID e13116, 2021.

[9] S. Chatterjee, P. K. Ghosh, and B. K. Saha Roy, "PMU-based power system component monitoring scheme satisfying complete observability with multicriteria decision support," *International Transactions on Electrical Energy Systems*, vol. 30, no. 2, Article ID e12223, 2020.

[10] Nist-Usa, "PMU Application Requirements Test Framework (PARTF)," 2022, https://github.com/usnistgov/PARTF.

[11] A. C. Adewole and R. Tzoneva, "Co-simulation platform for integrated real-time power system emulation and wide area communication," *IET Generation, Transmission and Distribution*, vol. 11, no. 12, pp. 3019–3029, 2017.

[12] M. Naglic, M. Popov, M. A. M. M. van der Meijden, and V. Terzija, "Synchro-measurement application development framework: an IEEE standard C37.118.2-2011 supported MATLAB library," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 8, pp. 1804–1814, 2018.

[13] A. S. Musleh, S. M. Muyeen, A. Al-Durra, and I. Kamwa, "Testing and validation of wide-area control of STATCOM using real-time digital simulator with hybrid HIL–SIL configuration," *IET Generation, Transmission and Distribution*, vol. 11, no. 12, pp. 3039–3049, 2017.

[14] E. Rebello, L. Vanfretti, and M. S. Almas, "Experimental testing of a real-time implementation of a PMU-based wide-area damping control system," *IEEE Access*, vol. 8, pp. 25800–25810, 2020.

[15] R. Leelaruji, L. Vanfretti, K. Uhlen, and J. O. Gjerde, "Computing sensitivities from synchrophasor data for voltage stability monitoring and visualization," *International Transactions on Electrical Energy Systems*, vol. 25, no. 6, pp. 933–947, 2015.

[16] N. Zhou and J. Dagle, "Initial results in using a self-coherence method for detecting sustained oscillations," *IEEE Transactions on Power Systems*, vol. 30, no. 1, pp. 522–530, 2015.

[17] D. R. Shrivastava, S. A. Siddiqui, and K. Verma, "A new synchronized data-driven-based comprehensive approach to enhance real-time situational awareness of power system," *International Transactions on Electrical Energy Systems*, vol. 31, no. 5, Article ID e12887, 2021.

[18] A. Ashok, A. Hahn, and M. Govindarasu, "Cyber-physical security of wide-area monitoring, protection and control in a smart grid environment," *Journal of Advanced Research*, vol. 5, no. 4, pp. 481–489, 2014.

[19] K. Sun, B. Wang, and J. Ivan, "Locating the source of sustained oscillation," 2021, http://web.eecs.utk.edu/~kaisun/TF/Tutorial_2016IEEEPESGM/Synchrophasor_8_Kai.pdf.

[20] S. Sandi, B. Krstajic, and T. Popovic, "PyPMU - open source python package for synchrophasor data transfer," in *Proceedings of the 24th Telecommunications Forum, TELFOR 2016*, pp. 1–3, IEEE, Piscataway, NJ, USA, 2017.

[21] X. Zhong, P. Arunagirinathan, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks, "Phasortoolbox-a python package for synchrophasor application prototyping," in *Proceedings of the Clemson University Power Systems Conference, PSC 2018*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, USA, 2019.

[22] K. V. Khandeparkar, N. Pandit, A. M. Kulkarni, V. Z. Attar, and S. U. Ghumbre, "Design of a phasor data concentrator for wide area measurement system," 2020, http://www.iitk.ac.in/npsc/Papers/NPSC2012/papers/12223.pdf.

[23] M. S. Almas, L. Vanfretti, and M. Baudette, "Babelfish—tools for IEEE C37.118.2-compliant real-time synchrophasor data mediation," *Compliant Real-Time Synchrophasor Data Mediation'SoftwareX*, vol. 6, pp. 209–216, 2017.

[24] ALSETLab/Pmu-Pdc-StreamSimulator, "A C++ PMU And/or PDC stream simulator for IEEE C37.118.2," 2020, https://github.com/ALSETLab/PMU-PDC-StreamSimulator.

[25] M. Baudette, S. R. Firouzi, and L. Vanfretti, "The STRONgrid library: A modular and extensible software library for IEEE C37.118.2," *Compliant Synchrophasor Data Mediation'SoftwareX*, vol. 7, pp. 281–286, 2018.

[26] OpenPDC-A Open Source Software & Services for Electric Utilities, https://www.gridprotectionalliance.org/, 2020.

[27] N. Zhou, "A cross-coherence method for detecting oscillations," *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 623–631, 2016.

[28] M. Naglic, M. Popov, M. A. M. M. van der Meijden, and V. Terzija, "Synchronized measurement technology supported online generator slow coherency identification and adaptive tracking," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3405–3417, 2020.

[29] L. Kumar and N. Kishor, "Wide area monitoring of sustained oscillations using double-stage mode decomposition," *International Transactions on Electrical Energy Systems*, vol. 28, no. 6, Article ID e2553, 2018.

[30] "NASPI oscillation detection and voltage stability tools technical workshop - houston," 2020, https://www.naspi.org/node/440.

[31] P. Kaliappan, K. S. Meera, and M. P. Selvan, "Assessment of compliance of phasor measurement units (PMUs) for smart grid applications," *International Transactions on Electrical Energy Systems*, vol. 31, no. 4, Article ID e12835, 2021.

[32] L. Kumar and N. Kishor, "Spectral identification of forced oscillation in PMU signal using mode decomposition," in *Proceedings of the First International Colloquium on Smart Grid Metrology (SmaGriMet)*, pp. 1–6, IEEE, Piscataway, NJ, USA, 2018.

[33] L. Kumar, "Video: real-time FO monitoring through LPDC-SADF real-time framework," 2021, https://drive.google.com/file/d/1L_KI4FF-u4Q2C74rzmtLOwi9yFN6LdVh/view.

[34] A. Silverstein, "NASPI technical report: diagnosing equipment health and mis-operations with PMU data," 2020, https://www.naspi.org/sites/default/files/reference_documents/14.pdf.