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], Nand Kishor[3]

[1] CEMSE, KAUST, Saudi Arabia

[2] ECSE, RPI, USA

[3] FCSEE, Østfold University College, Norway

Correspondence

Lalit Kumar, CEMSE, KAUST, Saudi Arabia

Email: lalitnbd@gmail.com

Abstract: 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 pre-recorded phasor measurement unit (PMU) signals. With this software toolchain, a PRTT study is presented for two WAMS applications; 'testing of PMU/phasor data concentrator (PDC)' and 'testing of wide-area forced oscillation (FO) monitoring application'.

KEYWORDS: wide-area monitoring protection and control (WAMPAC), forced oscillation, phasor measurement unit (PMU), phasor data concentrators (PDC), north American synchrophasor initiative (NASPI), synchro-measurement application development framework (SADF)

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; WAMS, wide-area monitoring system; 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

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, 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 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 forth most eligibility which must be met by all WAMPAC-applications, and that is 'real-time testing'. The demand of real-time testing continues to grow and this trend is unlikely to change with technology advancement and the deployment of low cost PMUs [7, 8] at 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 literature.

Real-time testing poses several challenges and not only is needed for application testing but 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 researcher's limitations which is the major roadblock in real-time testing of WAMPAC-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 Fig. 1. While staff from the National Institute of Standards and Technology (NIST) of the United States have proposed a requirements test framework for applications, this has not yet gained adoption and it does not formalize a testing methodology [10]. To the 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 Fig. 1 can be reduced.

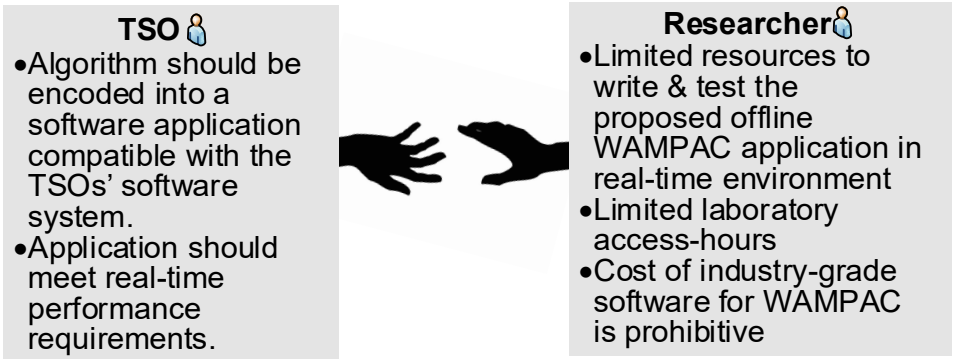

Fig. 1. Depiction of gap between TSO's demands and researcher's limitations

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, outliers etc. 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 playing back simulation data or measurement records [16].

The major issue to incorporate communication-based real-time testing in 2nd and 3rd stages numbered above lies in the fact that such testing requires all-time access to signal steam/broadcast until the application is tested satisfactorily. Unlike pre-recorded data, such all-time access to signal steam/broadcast is difficult to attain in 2nd and 3rd stages numbered above, 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. 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 real-time testing methodology for synchrophasor-based WAMPAC-applications.

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

iii. Demonstration of a software toolchain for PRTT of WAMS/WAMS-applications.

The reminder of the paper is organized as follows; in section 2, with brief discussion on WAMPAC system, a methodology is formulated for real-time testing of WAMPAC-applications. In section 3, open-source software for real-time synchrophasor communication are discussed. Section 4 presents the software toolchain. Section 5 demonstrates 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 WAN-enabled WAMPAC system is shown in Fig. 2. Generally, the WAMPAC system and involved-applications are divided in 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 Fig. 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 assessment based on operator's own experience is generally referred as 'situational awareness' [17], as depicted in Fig. 2.

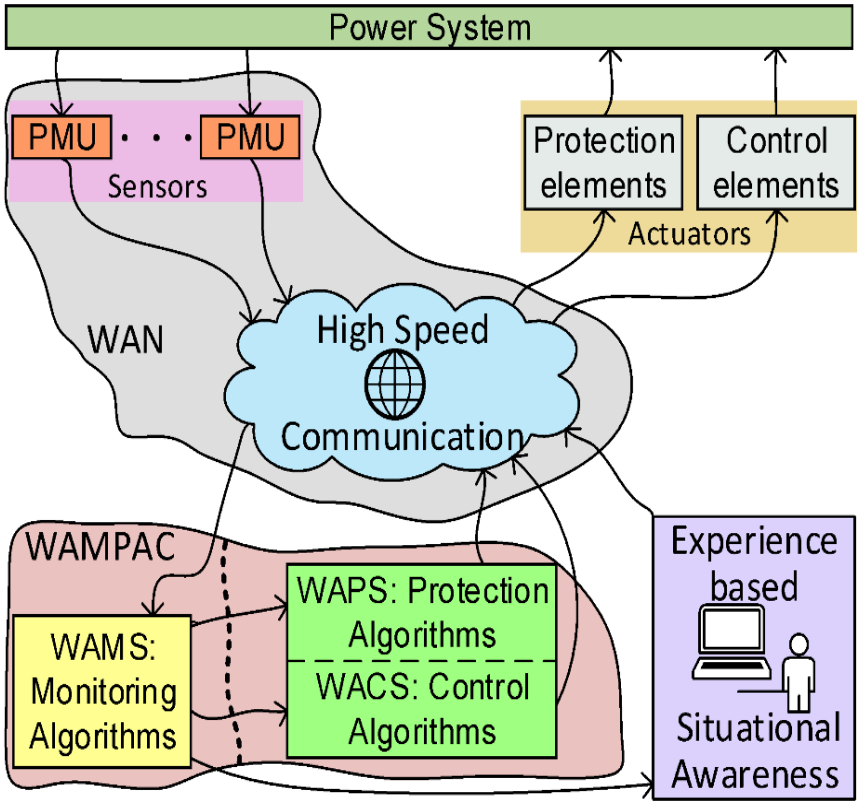

Fig. 2. Typical depiction of WAN-enabled WAMPAC system

Among two categorization listed above, the later one i.e. 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. synchrophasor based real-time WAMS is widely studied part of the WAMPAC system and also deployed in many power system, 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 real-time testing methodology

This section provides a testing methodology to reduce the gap depicted in Fig. 1, by formulating a systematic real-time testing methodology. If $\mathcal{T}$ is the specified domain of performance metric that must be achieved by a WAMPAC-application, $\mathcal{A}$ in its testing, then it can be deemed successfully tested if the following condition holds true;

$$T \supset \mathcal{T} \qquad \ni \quad \mathcal{T} = \mathbb{T}(\mathcal{A}) \qquad (1)$$

where $\mathbb{T}$ is the operator that is operated on $\mathcal{A}$ to achieve its performance metrics, $\mathcal{T}$ and T is the actually achieved performance. In practice, $\mathbb{T}$ can be referred as testing

methodology and $\mathcal{T}$ could be the set of performance criteria e.g. execution time < specified time, application's output index < pre-defined threshold, etc. Achieving the Eq. (1) can be regarded equivalent to the stamp of 'tested and approved' on $\mathcal{A}$.

For generalized testing of any $\mathcal{A}$, 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 testbed is needed to test whether $\mathcal{A}$ meets the performance criteria or not, under multiple realistic power grid conditions, before it could be deemed successfully tested. However, performing laboratory tests with a naively implemented WAMPAC-application 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 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.

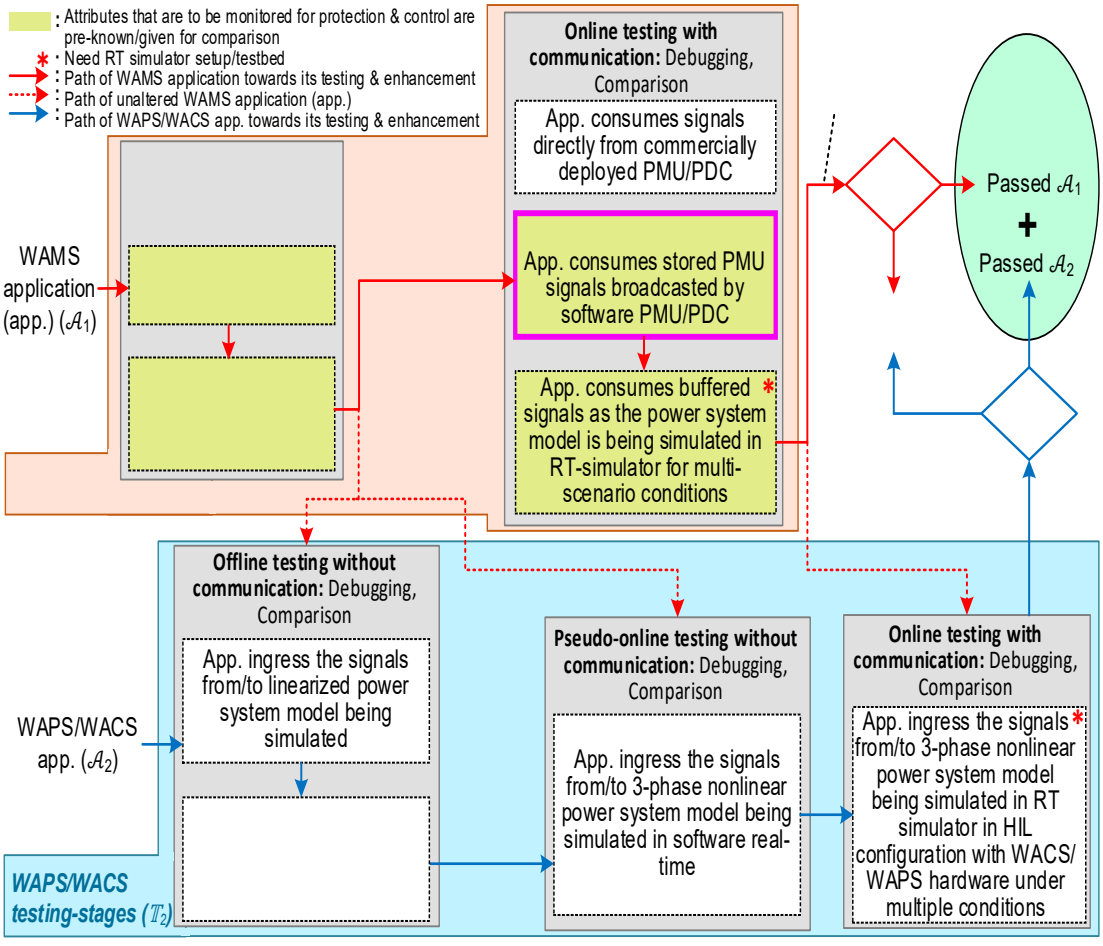

Fig. 3. Testing methodology ($\mathbb{T}$) for WAMPAC-applications ($\mathcal{A}$)

2.2.1 Proposed testing methodology

The legend shown at the top left corner of Fig. 3 should be perused first before following of the reminder of Fig. 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 one testing stage to another after passing the preceding stage. As mentioned earlier, WAMPAC-applications, $\mathcal{A}$ are categorized in two categories based on the involved testing methodology i.e. $\mathcal{A}_1$: WAMS-applications and $\mathcal{A}_2$: WAPS/WACS-applications and $\mathcal{A} = [\mathcal{A}_1, \mathcal{A}_2]$. Similarly, the testing methodology, $\mathbb{T}$ can also be segregated in two parts i.e. $\mathbb{T}_1$: testing methodology for $\mathcal{A}_1$ and $\mathbb{T}_2$: testing methodology for $\mathcal{A}_2$, and $\mathbb{T} = [\mathbb{T}_1, \mathbb{T}_2]$. By virtue of this categorization, the Fig. 3 is segregated in 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, $\mathcal{A}_1, \mathcal{A}_2$ needs to be passed/moved along the shown path for their approval by their respective parts $\mathbb{T}_1, \mathbb{T}_2$ comprised of their respective grouped-stages. Formulated methodology can be understood by thorough observation of Fig. 3 in conjunction with the following noteworthy points;

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

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

iii. Before moving to laboratory testing (red asterisk boxes in Fig. 3) with $\mathcal{A}$, it is desirable to apply the testing stages (un-asterisk boxes in Fig. 3) to debug $\mathcal{A}$, 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 Figs. 2-3, the online stages of $\mathbb{T}_1$ requires PMU signals to be broadcasted as input whereas the stages of $\mathbb{T}_2$ requires the ascertained monitoring results by $\mathbb{T}_1$ as input. (2) $\mathbb{T}_2$ essentially requires the power system modelling data in its all stages whereas it

is not required in all stages of $\mathbb{T}_1$, as apparent from Fig. 3 because synchrophasor-based monitoring not necessarily needs the power system modelling data.

v. The findings of WAPS/WACS depends on the monitoring results that are the output of the WAMS applications. These results are important as the performance metrics of $\mathcal{A}_1$ aids in bounding those of $\mathcal{A}_2$, as can be observed in Figs. 2-3.

vi. For $\mathcal{A}_1$ 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.

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

The stage highlighted by the box with pink outline in $\mathbb{T}_1$ (see Fig. 3) is identified as a most important stage (PRTT stage) in the formulated testing methodology which avails the facility of pseudo on-line environment. This stage utilizes a software toolchain to test $\mathcal{A}_1$ that can broadcast the pre-recorded PMU data similar to how commercial PMUs stream data, and where the data can be retrieved by an application $\mathcal{A}$ in real-time. Even though, the PRTT stage is not sufficient for commercial real-time testing because it uses the pre-recorded data, still, it helps in preliminary testing and enhancement of $\mathcal{A}_1$ before entering the final testing-stage (red asterisk box in $\mathbb{T}_1$). The reminder 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 past, 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), (2) Signal-retriever (subscriber and client). The former type serves to broadcast pre-recorded/stored PMU signals as similarly done by commercial PMUs/PDCs and the latter type help to retrieve the broadcasted signal. Table I lists such open-source software, some of which were briefly reviewed

by the authors in Ref. [12] which could be followed there if required. The detailed comparative study between these listed software is out of the scope of this article. A software toolchain can be formed by combining one signal-broadcaster and one signal-retriever, one of each type are discussed next.

Table I. Open-source software for real-time communication

Signal-broadcaster	Signal-retriever
• PMU-PDC Stream Simulator (C++) [24] • pyPMU (Python) [20] • iPDC (Linux) [22] • openPDC (Java) [26]	• SADF (Matlab) [12] • S3DK (Labview) [25] • BabelFish (Labview) [23] • PhasorToolBox (Python) [21] • openPDC (Java) [26]

3.1 PUPO: Signal Broadcaster

Among the signal broadcasters listed in the Table I, '**PMU**-**PDC** stream simulat**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 Fig. 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).

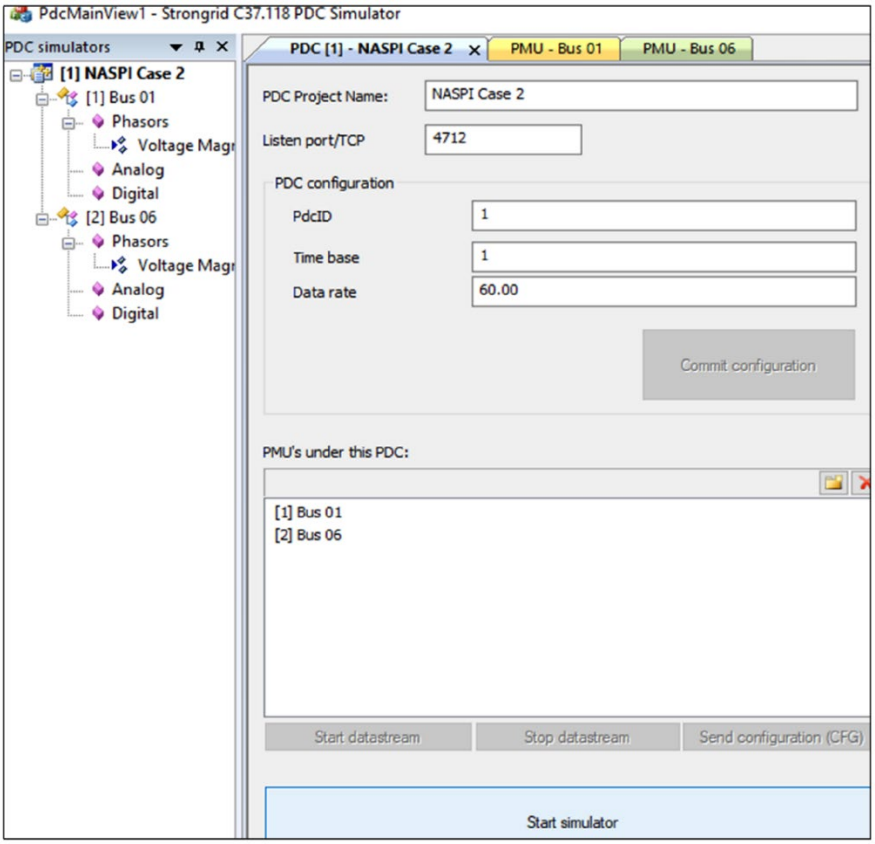

Fig. 4.GUI of PUPO

iii. No complex installation required, the main GUI opens directly on clicking '`StrongridSimulatorGUI_MFC.exe`' file.

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 pre-recorded signals following the same protocols used by commercial PMUs/PDCs. The current version of PUPO consumes the pre-recorded data in a `.csv` file (editable in any tool, e.g. Notepad/Notepad++) and use an extension name, '`.phcsv`'. The pre-recorded 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 converted in rectangular form first, before entering it into the input file. The method of preparing the input file and other necessary steps are descripted in Ref. [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 'folder/cross' icon as depicted in Fig. 4.

3.2 SADF: Signal Retriever

The recently developed 'synchro-measurement application development framework (SADF)' [12] is a MATLAB-based software that enables the retrieval of TCP, UDP, or mixed TCP/UDP synchro-measurement data. To the knowledge of the authors, SADF is the only MATLAB-based open-source toolbox available for such purpose to the 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 promising potential of SADF and vast functionalities available with the widely used MATLAB, SADF was selected from the signal retrievers given in the Table I, to form the software toolchain.

The script, '`SADF_setting`' allows to configure the PMU/PDC connection settings such as TCP/IP, port, device ID and maximum time of retrieval. The main script, '`SADF_run`' allows to embed 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 I) that can communicate to each other through 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 Fig. 3). This software toolchain provides a real-time environment for testing and enhancement of any $\mathcal{A}_1$ at PRTT stage.

The flowchart for the complete PUPO-SADF toolchain is shown in the Fig. 5. Firstly, the pre-recorded PMU data given in the standard '.csv' file needs to be edited in any text editor (e.g. Notepad/Notepad++). This file is than imported in the PUPO and then the broadcast can be started as per IEEE C.37.118.2 compliance by pressing 'start simulator' button as depicted in Fig. 5. Then SADF can retrieve the data in real-time and can execute $\mathcal{A}_1$ (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, $\mathcal{A}_1$. The computed results can be monitored in real-time through the MATLAB plot of individual's choice as written in the encoded application, $\mathcal{A}_1$.

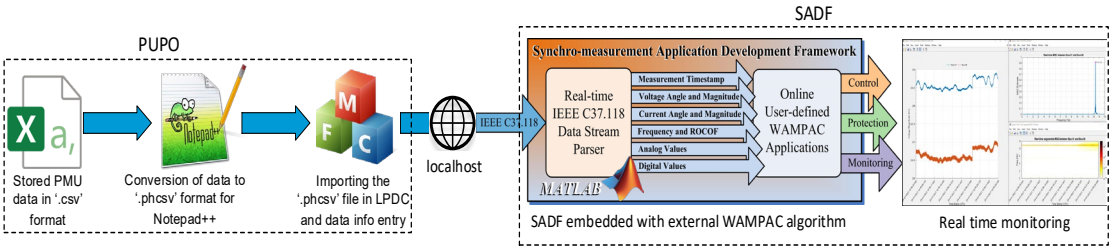

Fig. 5. Flow chart of PUPO-SADF toolchain

4.1 Monitoring Resolution: Speed in real-time monitoring

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

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

Ψ relates to the speed of the $\mathcal{A}_1$ 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 $\mathcal{A}_1$. However, the index Ψ should not be mistaken with the overall performance of any application. Ψ is just a one index among several indices required to govern overall performance. For example, the application developed in [29] for oscillation monitoring, utilized mode decomposition technique and thus it will be giving low Ψ due to the heavy computation burden involved in comparison to a simple power spectral density (PSD) operation. On the other hand, [29]'s application 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 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 the 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 $\mathcal{A}_1$ at PRTT stage shown in Fig. 3. After qualifying this stage, the application moves to next and final testing stage (red-asterisk box) where a real-time-HIL laboratory testbed is required. If any $\mathcal{A}_1$ is qualified/approved in the PRTT stage then it is expected to meet the performance requirements of the final stage i.e. red asterisk box/stage of $\mathbb{T}_1$ in Fig. 3 or at the 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 BAED WAMS USING PUPO-SADF TOOLCHAIN

Before continuing, it is to mention here that PUPO-SADF toolchain is exclusively utilized in PRTT stage, however, SADF can be utilized in other 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 with application testing, it helps to verify that PUPO can be used to develop monitoring applications with it.

The pre-recorded 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 pre-recorded 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 SADF by inputting TCP/IP and port settings of PUPO. Testing of PUPO is a post-retrieval process. The retrieved 10-min voltage-magnitude signals 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], it is necessary to ensure that the signals, while being broadcasted by PUPO, would also remain synchronized. This check can be done 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 pre-recorded (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 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. Fig. 6 (a) shows the plot of both pre-recorded ('`.csv`') and both retrieved signals 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 sampling rate

The purpose of this check is to be verify that PUPO has broadcasted the pre-recorded signals at its original sampling rate, F_s, and if it is uniform throughout. This is an important test as $\mathcal{A}_1$ may require F_s to compute the monitoring results e.g. $\mathcal{A}_1$ 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`'. Fig. 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 in 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, Fig. 6 (b) suggests one bug in 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

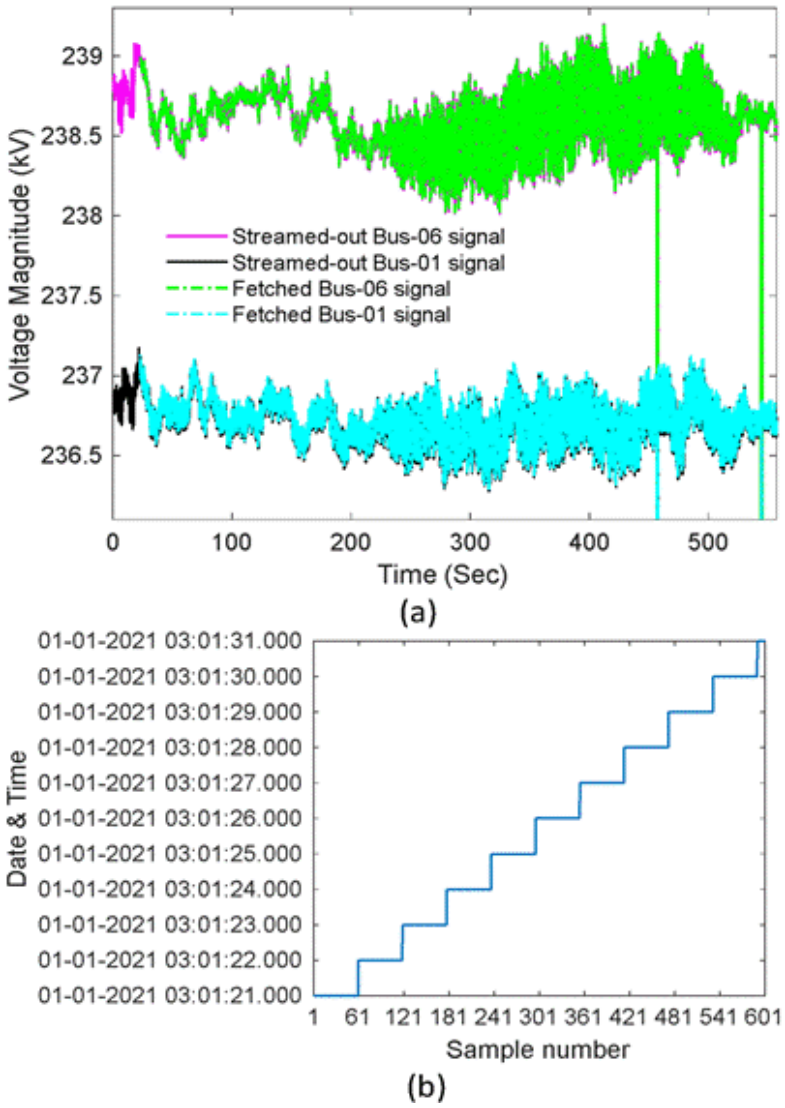

Fig. 6. (a) Pre-recorded and retrieved voltage signals (b) Retrieved timestamps

bug is further discussed and rectified next.

5.1.3 Discussion on identified bug and its rectification

Looking to the x-axis and y-axis in Fig. 6 (b), it can be ascertained that the retrieved timestamps are following stair case pattern instead of a linear pattern because the PUPO is streaming one same timestamp 60 times in one second. This is because the GPS-clock quality of PUPO is limited to second (HH:mm:ss) and not up to milliseconds (HH:mm:ss.sss). This corresponds with PUPO's own repository [24] which states that "*commit button doesn't gray out for setting PMU station settings*". This is also shown here with the help of Fig. 7 where it can be seen that due to non-functioning of 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 y-axis/array in Fig. 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. stair case pattern in Fig. 6 (b) are corrected with a linear

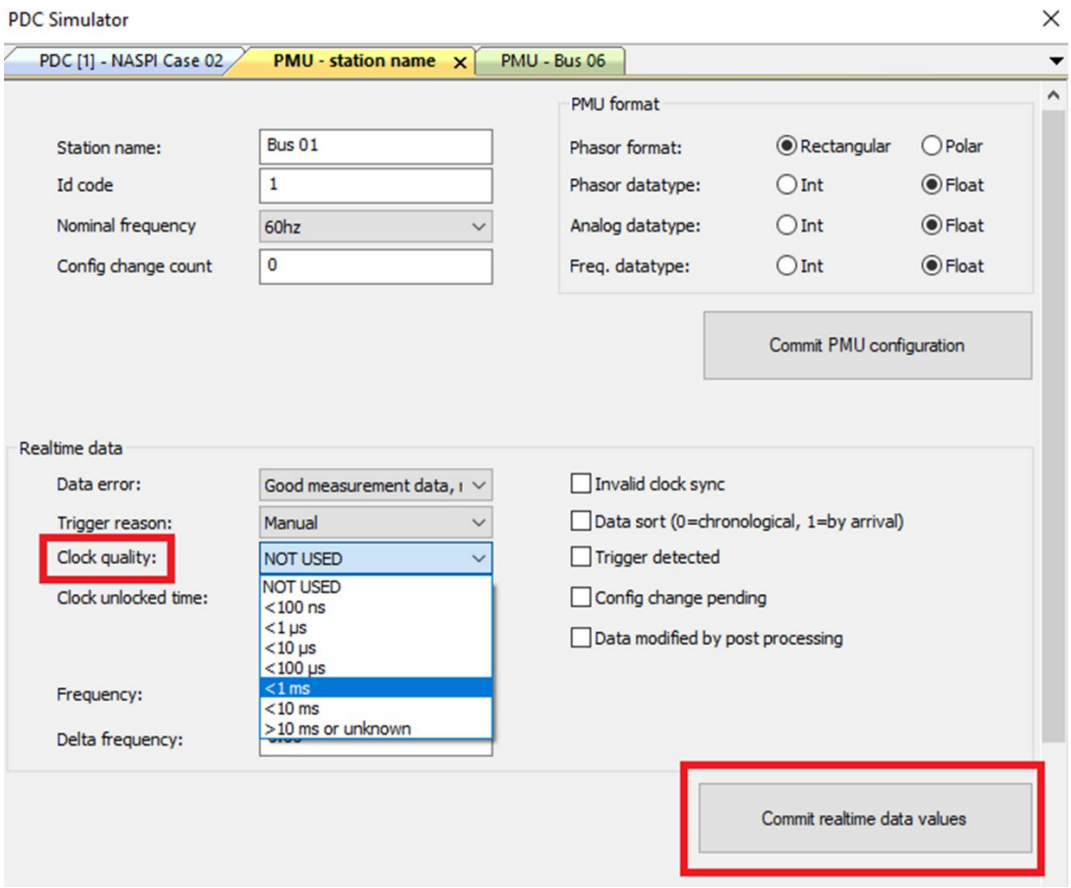

Fig. 7. Depiction of the disabled commit button in PUPO

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

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

Readers may follow Fig. 9 (a) 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 SADF. Next, the PUPO-SADF toolchain as a whole provide a suitable mean for PRTT of any $\mathcal{A}_1$.

5.2 PRTT of wide-area FO monitoring application

5.2.1 FO monitoring application

In this section the FO monitoring application utilizes magnitude squared coherence (MSC) tool [16] for spectral analysis. 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] indicate that without having any prior information, it is very challenging to distinguish oscillations from ambient responses by only 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 $f \in$ R. The region of space, R, maps the space $[0\ \frac{F_s}{2}]$ where F_s is the common sampling frequency of both the sequences, $x(n)$ and $y(n)$. The mathematical expression of MSC is written as [27, 32];

$$C_{xy}(f) \triangleq \frac{|P_{xy}(f)|^2}{P_{xx}(f)P_{yy}(f)} \quad (4)$$

$$0 \leq C_{xy}(f) \leq 1 \quad \forall\ f \in \mathrm{R} \quad (5)$$

where $P_{xx}(f)$ and $P_{yy}(f)$ are the PSDs of $x(n)$ and $y(n)$ respectively and, $P_{xy}(f)$, is the cross PSD (CPSD) between $x(n)$ and $y(n)$. Values of MSC estimate are real and lie between 0 and 1 as indicated in Eq. 5. If $x(n)$ and $y(n)$ are two sinusoidal signals of frequency, f_x and f_y respectively than following holds true;

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

The application utilizing the MSC estimate is encoded in the script, '`smart_WAMS`' ($\mathcal{A}_1$) using the '`mscohere`' function of MATLAB's 'signal processing toolbox' that computes the MSC estimate between two signals. Before MSC, the application first

pre-processes the signals by 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 as 30 sec herein. A Hamming window is chosen to minimize leakage noise [27].

The application also incorporates automatic peak-detector with the help of function '`findpeaks`'. This function detects the peaks in MSC estimate in real-time and shows it on the MSC plot to provide real-time visualization of the FO frequency. Threshold, $C_{xy}^{thres}(f)$ for peak-detection is set to 0.1 i.e. if $C_{xy}(f) \geq C_{xy}^{thres}(f)$ 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 Fig. 8's flow chart.

5.2.2 Real-time FO monitoring using 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 same as for segmented-PSD and segmented-self-coherence described in Ref. [32]. Here, the 30 secs window, $w(n)$ is slid 30 times for next retrieving 30 secs. 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 SADF begins retrieving and plotting the signals, the MSC plot (colormap) begins showing the results only after the retrieval of predefined length of the window, T_w. The retrieving signals and their MSC estimate are also shown in Fig. 9 (a) and (b) respectively for one instant in the real-time monitoring.

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

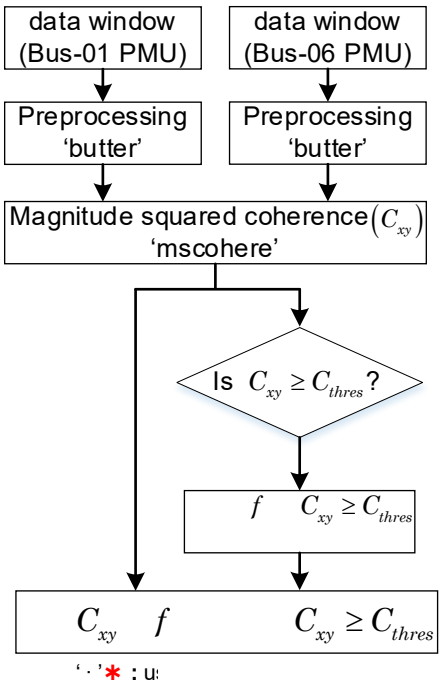

Fig. 8. Main steps in the '`smart-WAMS`' application ($\mathcal{A}_1$)

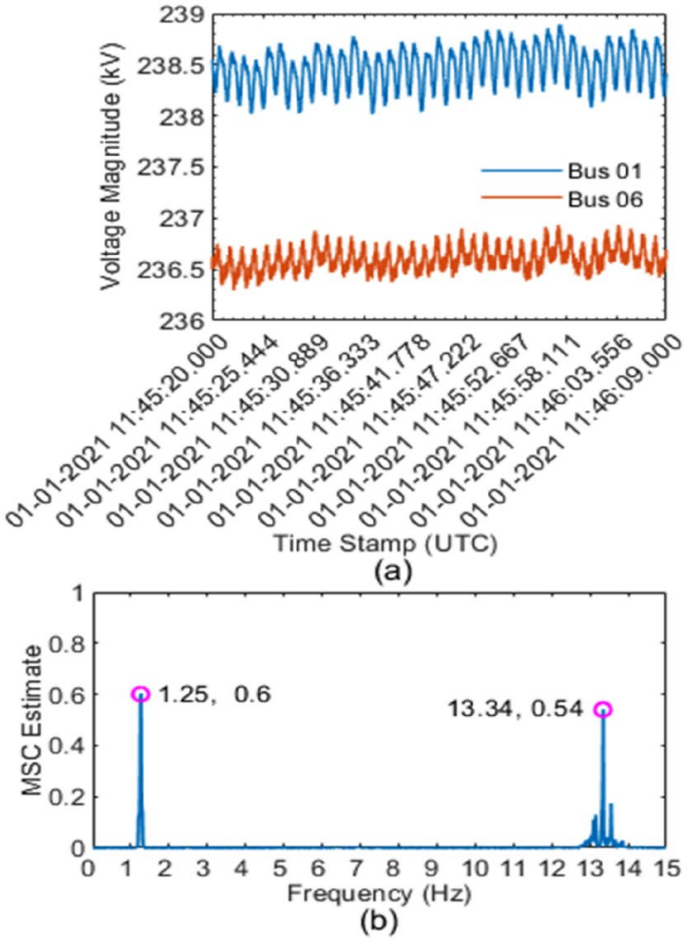

Fig. 9. (a) Real-time retrieval of broadcasted PMU signals (b) Real-time FO monitoring using MSC estimate

5.2.3 Summary of Fig. 10 results

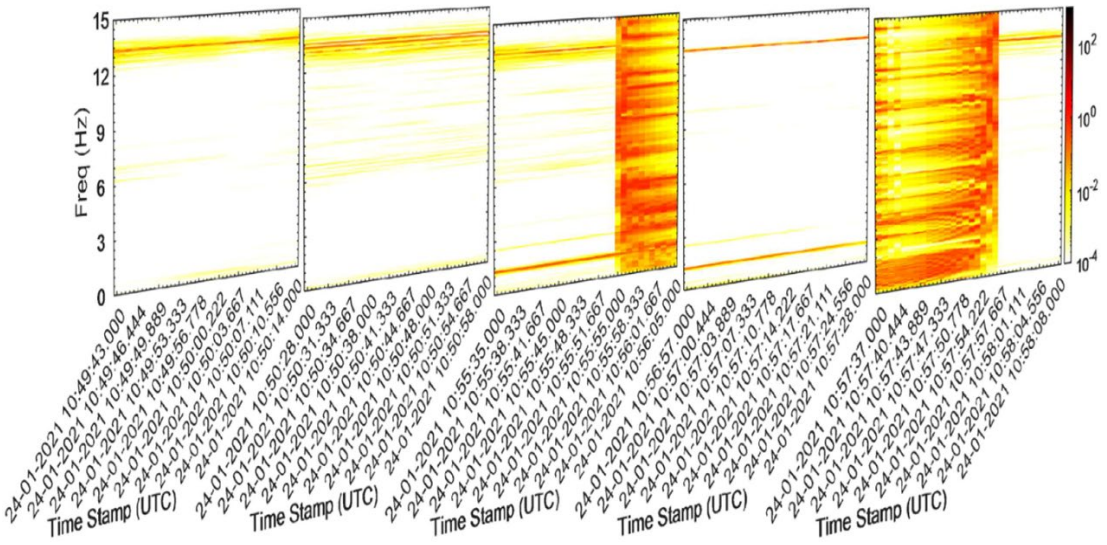

Fig. 10. Real-time FO monitoring using PUPO-SADF toolchain

From Fig. 6 (a), Fig. 10 and recorded screen [33], it can be noticed that first frame (Fig. 10) is associated with the early secs of the signals (Fig. 6 (a)) wherein the 13.3 Hz FO is observable. The second frame (Fig. 10) shows the monitoring of pre-event secs (Fig. 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 secs (Fig. 6 (a)) triggered another 1.25 Hz FO which sustained over the event's duration. The third frame (Fig. 10) shows the monitoring of the signals during the period when first outliers appears in the data (see Fig. 6 (a)). The presence of outliers can be 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 (Fig. 10) that after first outlier is passed, both FOs sustained, and the widened spectrum-band of 13.3 Hz FO has reverted back to narrow. The last frame is the monitoring results during the period when additional/second outliers (Fig. 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 secs (Fig. 6 (a)), the 1.25 Hz FO disappears and 13.3 Hz FO still sustains. The discussed analysis can be verified from the video in Ref.[33] .The presence of these FOs were also confirmed in the Ref. [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`', $T_c^{smart\_WAMS}$ varies. On a computer with characteristics, *Windows 10 OS, Intel(R) Core(TM) i7-8550U CPU @*

1.80GHz, 16 GB RAM, the computation times, $T_c^{smart\_WAMS}$ are measured for 709 retrieved window, $w(n)$ with $T_w = 30\ secs$, $F_s = 60\ Hz$. The measurements are presented in a form of normalized histogram shown in Fig. 11. The average $T_c^{smart\_WAMS}$ and average Ψ_{smart_WAMS} are obtained as, $T_{c_{avg}}^{smart\_WAMS} = 0.8961\ sec$ and $\Psi_{smart\_WAMS_{avg}} = 97.01\ \%$ respectively, which indicates that this particular application has acceptable performance and thus can pass the PRTT stage to move on to final testing stage (red-asterisk box in Fig. 3).

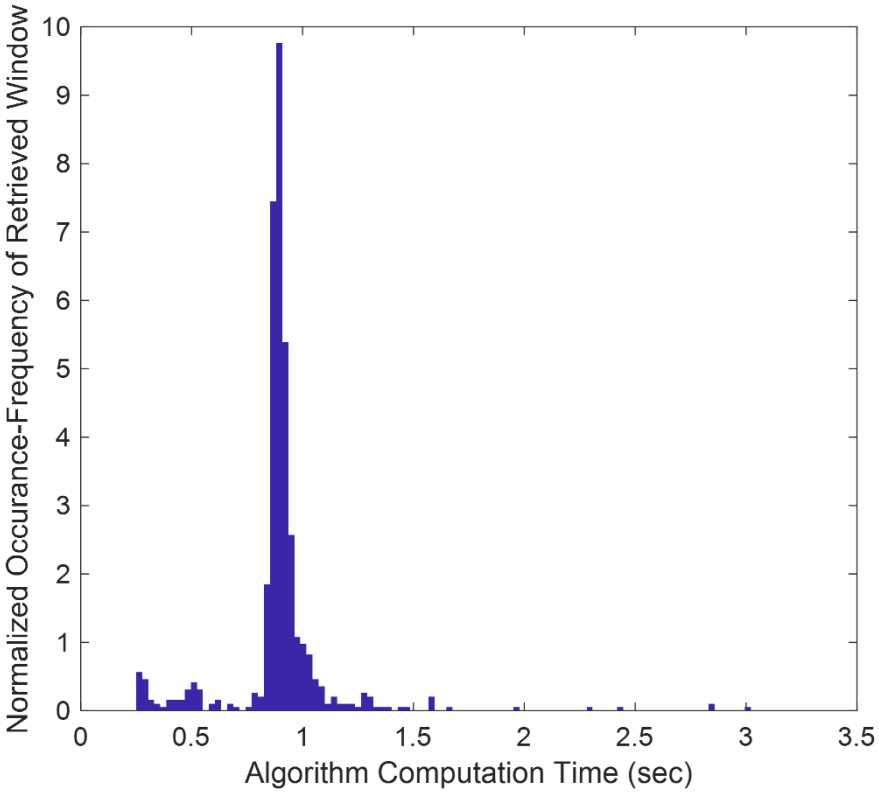

Fig. 11. Real-time FO monitoring using segmented-MSC estimate

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. red asterisk box/stage of $\mathbb{T}_1$ in Fig. 3. After being approved in 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 laboratory in rewriting of 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 was 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 first one, a minor bug was found and rectified in custom-encoding within SADF while in later one, FOs at frequencies 13.33 Hz and 1.25 Hz are detected and monitored in real-time using recorded PMU data provided by NASPI.

ACKNOWLEDGMENTS

This work was carried out under the project, 'Gridx: The Autonomous Digital Grid' funded by '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 department of science and technology (DST), New Delhi, Govt. of India, granted with a SERB No: CRG/2019/000951.

Symbols: $T, achieved$ performance; $\mathcal{A}$, WAMPAC- $application$; $\mathbb{T}$, Testing methodology; $\mathcal{T}$, specified domain of performance-eligibilities; $\mathcal{A}_1$, WAMS-applications; $\mathcal{A}_2$, WAPS/WACS applications; $\mathbb{T}_1$, testing methodology for $\mathcal{A}_1$; $\mathbb{T}_2$, testing methodology for $\mathcal{A}_2$; C_{xy}, MSC estimate; $P_{xx}(f)$, PSD; T_w, window length; T_c, computation time; Ψ, monitoring resolution

REFERENCES

1 Haugdal, H., Uhlen, K., Muller, D., Johannsson, H.: 'Estimation of Oscillatory Mode Activity from PMU Measurements', in 'IEEE PES Innovative Smart Grid Technologies Conference

Europe' (Institute of Electrical and Electronics Engineers (IEEE), 2020), pp. 201–205

2 Shweta, Kishor, N., Uhlen, K., Mohanty, S.R.: 'Identification of coherency and critical generators set in real-time signal'*IET Gener. Transm. Distrib.*, 2017, **11**, (18), pp. 4456–4464.

3 Kamwa, I., Pradhan, A.K., Joos, G.: 'Adaptive phasor and frequency-tracking schemes for wide-area protection and control'*IEEE Trans. Power Deliv.*, 2011, **26**, (2), pp. 744–753.

4 Ashrafi, A., Shahrtash, S.M.: 'Dynamic wide area voltage control strategy based on organized multi-agent system'*IEEE Trans. Power Syst.*, 2014, **29**, (6), pp. 2590–2601.

5 Jakobsen, S.H., Uhlen, K.: 'Testing of a hydropower plant's stability and performance using PMU and control system data in closed loop'*IET Gener. Transm. Distrib.*, 2019, **13**, (23), pp. 5339–5348.

6 Hur Rizvi, S.M., Kundu, P., Srivastava, A.K.: 'Hybrid voltage stability and security assessment using synchrophasors with consideration of generator Q-limits'*IET Gener. Transm. Distrib.*, 2020, **14**, (19), pp. 4042–4051.

7 Laverty, D.M., Best, R.J., Brogan, P., Al Khatib, I., Vanfretti, L., Morrow, D.J.: 'The OpenPMU platform for open-source phasor measurements'*IEEE Trans. Instrum. Meas.*, 2013, **62**, (4), pp. 701–709.

8 Chauhan, S., Dahiya, R.: 'Multiple μPMU placement solutions in active distribution networks using nonlinear programming approach'*Int. Trans. Electr. Energy Syst.*, 2021, **31**, (11), p. e13116.

9 Chatterjee, S., Ghosh, P.K., Saha Roy, B.K.: 'PMU-based power system component monitoring scheme satisfying complete observability with multicriteria decision support'*Int. Trans. Electr. Energy Syst.*, 2020, **30**, (2), p. e12223.

10 'NIST-USA: "PMU Application Requirements Test Framework (PARTF)"', https://github.com/usnistgov/PARTF, accessed May 2022

11 Adewole, A.C., Tzoneva, R.: 'Co-simulation platform for integrated real-time power system emulation and wide area communication'*IET Gener. Transm. Distrib.*, 2017, **11**, (12), pp. 3019–3029.

12 Naglic, M., Popov, M., Meijden, M.A.M.M. Van Der, Terzija, V.: 'Synchro-Measurement Application Development Framework: An IEEE Standard C37.118.2-2011 Supported MATLAB Library'*IEEE Trans. Instrum. Meas.*, 2018, **67**, (8), pp. 1804–1814.

13 Musleh, A.S., Muyeen, S.M., Al-Durra, A., Kamwa, I.: 'Testing and validation of wide-area control of STATCOM using real-time digital simulator with hybrid HIL–SIL configuration'*IET Gener. Transm. Distrib.*, 2017, **11**, (12), pp. 3039–3049.

14 Rebello, E., Vanfretti, L., Almas, M.S.: 'Experimental Testing of a Real-Time Implementation of a PMU-Based Wide-Area Damping Control System'*IEEE Access*, 2020, **8**, pp. 25800–25810.

15 Leelaruji, R., Vanfretti, L., Uhlen, K., Gjerde, J.O.: 'Computing sensitivities from synchrophasor data for voltage stability monitoring and visualization'*Int. Trans. Electr. Energy Syst.*, 2015, **25**, (6), pp. 933–947.

16 Zhou, N., Dagle, J.: 'Initial results in using a self-coherence method for detecting sustained

oscillations'*IEEE Trans. Power Syst.*, 2015, **30**, (1), pp. 522–530.

17 Shrivastava, D.R., Siddiqui, S.A., Verma, K.: 'A new synchronized data-driven-based comprehensive approach to enhance real-time situational awareness of power system'*Int. Trans. Electr. Energy Syst.*, 2021, **31**, (5), p. e12887.

18 Ashok, A., Hahn, A., Govindarasu, M.: 'Cyber-physical security of wide-area monitoring, protection and control in a smart grid environment'*J. Adv. Res.*, 2014, **5**, (4), pp. 481–489.

19 Sun, K., Wang, B., Ivan, J., *et al.*: 'Locating the Source of Sustained Oscillation', http://web.eecs.utk.edu/~kaisun/TF/Tutorial_2016IEEEPESGM/Synchrophasor_8_Kai.pdf, accessed January 2021

20 Sandi, S., Krstajic, B., Popovic, T.: 'PyPMU - Open source python package for synchrophasor data transfer', in '24th Telecommunications Forum, TELFOR 2016, IEEE' (IEEE, 2017), pp. 1–3

21 Zhong, X., Arunagirinathan, P., Jayawardene, I., Venayagamoorthy, G.K., Brooks, R.: 'PhasorToolBox-A Python Package for Synchrophasor Application Prototyping', in 'Clemson University Power Systems Conference, PSC 2018' (Institute of Electrical and Electronics Engineers Inc., 2019)

22 Khandeparkar, K. V, Pandit, N., Kulkarni, A.M., Attar, V.Z., Ghumbre, S.U.: 'Design of a Phasor Data Concentrator for Wide Area Measurement System', http://www.iitk.ac.in/npsc/Papers/NPSC2012/papers/12223.pdf, accessed April 2020

23 Almas, M.S., Vanfretti, L., Baudette, M.: 'BabelFish—Tools for IEEE C37.118.2-compliant real-time synchrophasor data mediation'*SoftwareX*, 2017, **6**, pp. 209–216.

24 'GitHub - ALSETLab/PMU-PDC-StreamSimulator: A C++ PMU and/or PDC Stream Simulator for IEEE C37.118.2', https://github.com/ALSETLab/PMU-PDC-StreamSimulator, accessed March 2020

25 Baudette, M., Firouzi, S.R., Vanfretti, L.: 'The STRONgrid library: A modular and extensible software library for IEEE C37.118.2 compliant synchrophasor data mediation'*SoftwareX*, 2018, **7**, pp. 281–286.

26 'Grid Protection Alliance - Home', https://www.gridprotectionalliance.org/, accessed April 2020

27 Zhou, N.: 'A cross-coherence method for detecting oscillations'*IEEE Trans. Power Syst.*, 2016, **31**, (1), pp. 623–631.

28 Naglic, M., Popov, M., Meijden, M.A.M.M. Van Der, Terzija, V.: 'Synchronized Measurement Technology Supported Online Generator Slow Coherency Identification and Adaptive Tracking'*IEEE Trans. Smart Grid*, 2020, **11**, (4), pp. 3405–3417.

29 Kumar, L., Kishor, N.: 'Wide area monitoring of sustained oscillations using double-stage mode decomposition'*Int. Trans. Electr. Energy Syst.*, 2018, **28**, (6), pp. 1–18.

30 'NASPI Oscillation Detection and Voltage Stability Tools Technical Workshop - Houston, TX | North American SynchroPhasor Initiative', https://www.naspi.org/node/440, accessed March 2020

31 Kaliappan, P., Meera, K.S., Selvan, M.P.: 'Assessment of compliance of phasor measurement units (PMUs) for smart grid applications' *Int. Trans. Electr. Energy Syst.*, 2021, **31**, (4), p. e12835.

32 Kumar, L., Kishor, N.: 'Spectral identification of forced oscillation in PMU signal using mode decomposition', in '2018 First International Colloquium on Smart Grid Metrology (SmaGriMet)' (IEEE, 2018), pp. 1–6

33 Lalit Kumar: 'Video: Real-time FO monitoring through LPDC-SADF real-time framework', https://drive.google.com/file/d/1L_KI4FF-u4Q2C74rzmtLOwi9yFN6LdVh/view, accessed January 2021

34 Silverstein, A.: 'NASPI Technical Report: Diagnosing Equipment Health and Mis-operations with PMU Data', https://www.naspi.org/sites/default/files/reference_documents/14.pdf, accessed April 2020