A Software Toolchain for Real-Time Testing of Synchrophasor Algorithms in MATLAB

Lalit Kumar,
CEMSE, KAUST,
23955, Saudi Arabia,
lalitnb@gmail.com

Shehab Ahmed,
CEMSE, KAUST,
23955, Saudi Arabia,
shehab.ahmed@kaust.edu.sa

Luigi Vanfretti,
ECSE, RPI,
12180, USA
luigi.vanfretti@gmail.com

Nand Kishor,
FCSEE, Østfold
University College, 1757,
nand.kishor@hiof.no

Abstract—This article develops and demonstrates the software toolchain for real-time testing of synchrophasor based algorithms. Real-time testing procedure being the need of smart grid, requires to be hassle-free and easily accessible to the researchers. The developed software toolchain combines both MATLAB and open-source software. The toolchain requires recorded phasor measurement unit (PMU) or phasor data concentrator (PDC) signals, which are then played-back in real-time in the same computer using local sockets. The data is replayed using transmission control protocol/internet protocol (TCP/IP) sockets and the IEEE C37.111-2013 data transfer standard. The data is then retrieved and processed in real-time by any synchrophasor based algorithm in MATLAB. The toolchain is demonstrated with two examples, one that shows the main functionality by testing the connection with a PMU/PDC and another testing of a wide-area forced oscillation (FO) monitoring algorithm.

Index Terms—Wide-Area Monitoring System (WAMS), Phasor Measurement Unit (PMU), real-time testing, North American Synchrophasor Initiative (NASPI)

Abbreviations: PMU, phasor measurement unit; PDC, phasor data concentrator; TCP/IP, transmission control protocol/internet protocol; FO, forced oscillation; WAMS, wide-area monitoring system; NASPI, North American synchrophasor initiative; COMTRADE, common format for transient data exchange; RIAPS, resilient information architecture platform for smart systems; SADF, synchrophasor application development framework; PUPO, A PMU-PDC-Stream Simulator; GUI, graphical user interface; MSC, magnitude squared coherence.

I. INTRODUCTION

The advancement of the technology to exploit phasor measurement unit (PMU) data in the last two decades has greatly facilitated new means for power system monitoring. Synchrophasor/PMU technology aids in delivering fast synchronous data streams from broad geographical spans, that are used in software applications within “wide area monitoring systems” (WAMS) [1]–[4]. The power industry has found that this technology has an important role for power system resilience [5], [6], and thus has become an important research topic. However, attaining confidential PMU data from power companies is challenging and may limit the advance of research endeavors. As an alternative, power system simulations can be used, e.g., PMU-like signals [7], to develop new algorithms when real-world data is not available, albeit such data is stored in data files and not streamed. In addition, realizing the need of PMU signal based tools, some companies and research institutes have publicly released the PMU data and invited the researchers around the globe to utilize it for the research [5], [8], [9]. Generally, the format of these few-minute data (.csv file) follows the common format for transient data exchange (COMTRADE) standard or IEEE/IEC Std. C37.111-2013 [10]. Using such data sets, applications for power system monitoring and stability-assessment [11], [12] have been developed. However, studying how such applications would perform in a real-time environment is usually not addressed.

The increased complexity and vulnerability in the power system, requires utilizing the monitoring potential of PMUs in real-time monitoring environments, so that stability issues can be detected and counteracted as fast as possible. Hence, it is necessary to develop and test new monitoring algorithms under real-time conditions. This requires real-time access to PMU or phasor data concentrator (PDC) signals, which becomes nearly impossible to access due to security protocols and confidentiality issues. Another alternative is to create the real-time PMU signals within a laboratory setting, [13], which is time consuming and costly. Therefore, it would be attractive to develop a simple and cost-effective approach to develop real-time applications. Moreover, because of the prevalence of specific signal analysis tools, e.g. MATLAB, it is ideal to provide means to quickly prototype applications exploiting the functionalities provided by such tools. This can be attractive to support companies’ plans for large scale deployment of low cost micro-PMU/open-PMU [14], [15] and the ‘resilient information architecture platform for smart systems’ (RIAPS) [16] to develop or upgrade their WAMS system.

This article demonstrates a software-toolchain to test a WAMS algorithm in real-time. It only requires open-source software and MATLAB, and thus it makes the real-time testing of WAMS algorithms simpler. The developed toolchain is demonstrated with two examples.

The reminder of this article is organized as follows; in Section II, available open-source software for real-time synchrophasor communication are discussed briefly. In Section III, two specific open-source software, “synchro-measurement application development framework (SADF)” and “PUPO: a PMU-PDC stream simulator” are reviewed. Section IV demonstrates the proposed software toolchain and lastly, the conclusions are drawn in Section V.

II. OPEN SOURCE SOFTWARE FOR REAL-TIME SYNCHROPHASOR COMMUNICATION

Developing the software toolchains, requires communication technologies between a client computer (e.g., where applications run) and a server (e.g., PDC, where the data is acquired). In the recent past, different efforts have developed and released open-source tools [13], [17]–[21] for synchrophasor communication. These tools can be categorized into two types: (1) PMU/PDC simulator, and (2) signal retriever (i.e. a parser that makes data available for use in other software environments). The first
allows to publish PMU signals similarly as performed by commercial PMU/PDC (i.e., acting as a server), and the latter retrieve the signals within a client computer. Table I lists some open source software which are freely available online.

<table>
<thead>
<tr>
<th>PMU/PDC simulator (Server)</th>
<th>Signal retriever (Client)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMU-PDC Stream Simulator (C++) [21]</td>
<td>SADF (Matlab) [13]</td>
</tr>
<tr>
<td>pyPMU (Python) [17]</td>
<td>SIDK (Labview) [23]</td>
</tr>
<tr>
<td>jPDC (Linux) [19]</td>
<td>Babelfish (Labview) [20]</td>
</tr>
<tr>
<td>openPDC (Java) [22]</td>
<td>PhasorToolBox (Python) [18]</td>
</tr>
</tbody>
</table>

A comparative study between these software tools is out of scope of this article. However, the authors in [13] have briefly reviewed some of them and pointed out their differences. Apart from these, a small software called PMU connection tester [22] is also available, which only allows to check/view the small stream of the signal in real-time and create configuration files. PMU connection tester and OpenPDC [22] are the most widely used open-source software by the researchers and industry in the synchrophasor community. However, [13] argues for the need of user-friendly and easily accessible tool in MATLAB for synchrophasor communication, leading to the development of SADF. As indicated in the Table I, SADF is a client tool, or signal retriever, which can perform real-time computations while ingesting real-time data. To the knowledge of the authors, SADF is the only open source tool available for MATLAB for such purposes, and thus, it is used for the client-side tool in this paper. In the category of “PMU/PDC simulator”, the C++ based software, ‘PUPO: a PMU-PDC-StreamSimulatoR’ [21] is used in this article, due to its user-friendly graphical user interface (GUI).

III. REVIEW ON PUPO AND SADF

A. PUPO

Commercial PDCs exist both in dedicated hardware (industrial computers) or software that can run in off-the-shelf computers. A local PDC is usually a hardware PDC that is installed at a substation for the collection of all PMUs measurement within that substation. To aggregate data from multiple substations, a centralized software PDC is installed in the utility’s control center to collect measurements from all the PMUs/PDCs of individual substations. The receiving data packets from multiple dispersed PMUs may encounter transmission delays with respect to each other, in reaching to the centralized PDC [24]. The key objective of the commercial software PDC is to resynchronize the data packets in real-time with reference to a GPS clock [25].

Being a simulator of a software PDC, the functionality of PUPO should not be confused with the commercial hardware or software PDCs. Like other PMU/PDC/simulators, PUPO is a software to publish previously recorded synchrophasor streams or data files representing signals from a single or multiple PMUs. Unlike commercial software PDCs, it does not receive real-time streams from multiple PMUs. Instead, PUPO emulates parts of a PDCs behavior, publishing the recorded signal according to IEEE C.37.118.2 standard. The software runs on MS Windows and is of light size and easy to install. To use PUPO to publish phasor data, it needs to be provided recorded data stored in a similar format to Comma Separated Values (CSV), called “phcsv” (i.e. phasor CSV), which can be opened by any text editor, such as Notepad/Notepad++. Fig. 1 shows the GUI of the PUPO. The phasor data needs to be provided in rectangular format and organized in row arrays.

B. SADF

SADF [13] is a MATLAB-based toolbox that receives TCP, UDP, or TCP/UDP synchro-measurement data. SADF not only serves the purpose of retrieving and storing the signal, but it enables parallel computation for WAMS applications, taking advantage of the parallelization tools provided by MATLAB. To use the SADF framework it only needs to be added to the MATLAB path. The function “SADF_seting” allows to configure the PMU/PDC connection settings such as TCP/IP, port, device ID and retrieval time. The main function “SADF_run” allows to embed any designed WAMS algorithm (function) as shown in Fig. 2.

For the user, it is necessary to develop some basic understanding of the default function, “demo_WAMS” to implement a new WAMS algorithm having a compatibility with SADF. The default function is given to plot the retrieved signal in real time, with its specifications mentioned on the plot, which is also being saved in MATLAB workspace. Even though the developer has suggested to embed a new algorithm in the “SADF_run” file, but, per the authors’ experience, the new
function will need to define a few variables globally, i.e. “demo” and “CFG_2_3”. These variables are already declared by the function “demo_WM3S” for plotting the signal of time-length ‘demo.window’ secs. It is observed that, for visualizing the signal and monitoring results both, it is better to embed the algorithm inside the “demo_WM3S” function instead of “SADF_run”.

In this paper, the software-toolchain demonstrated below uses PUPO and SADF, and is referred as ‘PUPO-SADF software-toolchain’, which is demonstrated next.

IV. ‘PUPO-SADF SOFTWARE-TOOLCHAIN’

This section demonstrates the ‘PUPO-SADF software-toolchain’ as a simple framework for real-time synchrophasor application development and testing. The toolchain is illustrated by the flow chart shown in Fig. 3, comprising the PUPO and SADF parts. The working of both have already been discussed briefly in Section III.A and III.B. The toolchain is demonstrated using two examples: (1) testing of software PMU/PDC simulator, and (2) real-time forced oscillation (FO) monitoring.

The recorded PMU data from north American synchrophasor initiative (NASPI) [5] is considered for the examples, which was collected in 2014. The data file “NASPI-2014-Workshop-Oscillation-Case2.csv” which belongs to “Oscillation detection: test case 2”, consists of 30 signals which are of 10-minute time-length sampled at 60 Hz. These 30 signals include 5 voltage magnitude signals, 5 voltage angle signals, 10 current magnitude signals and 10 current phase signals. The single line diagram for the interconnected NASPI’s subsystem corresponding to this data, is given in Ref [26]. The recorded voltage-phasor signal from bus-06 and bus-01 are selected to be published using PUPO which allows to emulate the broadcasting of the original signals as if they were streaming in 2014 from the commercial PDC.

A. Testing of PUPO

The testing of commercial PDC requires a rigorous approach [13, 27], in which several compliance aspects have to be considered. For example, one test requires to check whether the processing time taken by the PDC, to resynchronize the receiving data packets, is within the defined limit or not [13, 24]. However, this is not necessary in the case of PUPO’s testing, since the data which is to be consumed by PUPO is already synchronized. The key aspect in the testing of PUPO is to ensure that the performance of the real-time monitoring algorithm should not be adversely affected.

The published signals are retrieved by SADF using host (PDC) TCP/IP and port settings, defined in “SADF_settings” file. The retrieved 10-minute voltage-magnitude signals along with timestamps, are stored in the MATLAB workspace in the variable, “DATA_Magnitude” and “DATA_TimeStamp”. For field-recorded real-time data, the length of the retrieved signals to be stored in workspace can be managed in the SADF-MATLAB code, depending upon the individuals’ computer capacity. Here, the testing of PUPO includes two important checks that ensures that the monitoring algorithm will produce authentic real-time results.

1) Check for synchronized overlapping

The check for synchronized overlapping can be done by plotting the retrieved and published signals in one figure to check overlapping. PUPO broadcasts the imported signals (.phcsv) of fixed sample-length, in a loop without encountering any time-delay in jumping from end-sample to first one. When, both the SADF and PUPO are run at the same time, SADF takes few seconds, to connect to the PDC, and in those few seconds, some data has already been published by PUPO. Thus, it is very unlikely that the first sample of the retrieved signal by SADF in the MATLAB workspace is exactly the same as that of the published signal. Therefore, there is a need of ‘array-alignment’ (array-shift) in retrieved data so that it could overlap the published data. Whatever ‘array-alignment’ rule is followed for one array (signal) to have the overlapping, must also be followed by rest of the arrays/signals. And, if the same rule can overlap rest of the arrays with their corresponding published signals, then ‘synchronized overlapping check’ can be passed. It is to be noted that the significance of ‘array-alignment’ exists only in this check. Since the recorded NASPI signals are already synchronized in its data file (.csv) [5], it is necessary to ensure that these signals, if published by PUPO, would also be synchronized or not. Referring to Fig. 3, ‘array-alignment’ rule would not be needed if both part of the toolchain, i.e. PUPO and SADF, could start functioning at the same instant after pressing their respective start/run buttons. Also, the ‘array-alignment’ rule changes every time the software-toolchain is run depending upon the time-gap between PUPO’s start and SADF’s run. Moreover, PUPO once started need not to stop as it broadcast the imported signals in a loop. However, SADF may require frequent run/stop for code modifications.

Figure 4 shows the plot of both published and both retrieved signals after ‘array-alignment’. Both the retrieved voltage magnitude signals overlap synchronously with the respective published signals by following single ‘array-alignment rule’ and thus ‘synchronized overlapping check’ is passed for PUPO.

![Flow chart of the PUPO-SADF software-toolchain](image-url)
2) **Check for sampling rate**

In this check, it is to be verified that the PUPO has published the recorded signals at the desired data rate or not. It is a very important check, as the monitoring algorithm may require a specific data rate of the signal to produce the results. For example, the spectral tools generally require a constant sampling rate at which signal is sampled, e.g. for frequency estimation of power oscillations [3]. Also, this sampling/data rate should be constant throughout the retrieved signal, to intact the precision in monitoring results. This check can be performed by plotting the retrieved variable, “DATA.TimeStamp”. Fig. 5 shows the plot of retrieved time stamps versus the sample number, which allows to illustrate a typical data preparation issue.

The PUPO has published the desired at 60 Hz sampling/data rate, which is apparent in Fig. 5, as 60 samples (x-axis) have been retrieved in every one second (y-axis). But, the retrieved time stamps sent by the PUPO follows a step pattern, which is supposed to be linear. This indicates that the PUPO is indeed broadcasting the 60 samples of the signals in one second, but is streaming only one timestamp 60 times. This is because the time stamp provided to PUPO was limited to second (HH:mm:ss) and not up to milliseconds (HH:mm:ss.sss). This simple check allows to verify that the time stamps provided have not been corrupted (which in this example, they have been corrupted) and allow also to illustrate how this can be resolved by SADF if necessary, as described next.

![Fig. 4. Streamed out and retrieved voltage signals](image)

![Fig. 5. Plot of retrieved time array (DATA.TimeStamp)](image)

![Fig. 6. Retrieval of PMU stream and FO monitoring in real-time](image)
As can be noticed in Fig. 5, the published sampling/data rate is still 60 Hz and constant throughout. In the case of an incorrect timestamp, it can be corrected in real-time using the “linspace” command in the SADF WAMS function, which will convert the timestamp array from step to linear. In this way, if the ‘sampling rate’ check fails in PUPO, e.g. due to data entry errors, it can be rectified within the WAMS function, while retrieving by SADF.

B. Testing of wide-area FO monitoring algorithm

Two kinds of power oscillations are very severe in power system i.e. natural (electromechanical) and FO. Both of them have been defined, differentiated and vastly studied over the last decade [3], [26]. It is desired to have real-time monitoring of these oscillations in power systems. In this section the ‘PUPO-SADF’ software-toolchain is demonstrated for the testing of an FO monitoring algorithm.

The algorithm is implemented in the “demo WAMS” function which utilizes the “mscohere” function [28]. This function calculates the magnitude-squared coherence (MSC) estimate [28] between two signals. The value of MSC estimate lies in between 0 to 1, which indicate how coherent the first signal is to the second signal at each frequency. However, unlike in Ref. [28], the monitoring illustrated here, is in real-time, which requires the ‘PUPO-SADF’ software-toolchain. To this end, the default function “SADF_run” was edited so that the SADF could plot the retrieved signals in real-time and also plot the real-time MSC estimate and segmented-MSC estimate [3] between two signals, as can be seen in Fig. 6. The display-window size (“demo.window”) is kept to 30 sec. It can be seen that the FO at frequency 13.33 Hz is the major FO which is widely spread in the NASPI test power system. The presence of this FO is also confirmed in [29], [30]. Figure 6 might seem as unorthodox means of representation, but is inspired from the stacked screen-palette view used in the monitoring and control centers. The recorded screen-video for this real-time retrieval and monitoring can also be seen in Ref. [31].

V. Conclusions

With the help of open-source software, a software-toolchain for real-time testing of synchrophasor based algorithms in MATLAB was presented. The proposed toolchain was demonstrated using two examples, including a synchrophasor application for oscillation monitoring. The developed ‘PUPO-SADF’ software-toolchain is easy to set-up and use, it is hardware-free, and ideal time saving solution for real-time development and testing of WAMS algorithms for researchers familiar with MATLAB. Real-time testing of a FO monitoring algorithm was presented, and found a FO at frequency 13.33 Hz, which has been reported by other authors in the literature, indicating the successful testing of FO monitoring algorithm.

Acknowledgements

This work is 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).

References


