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

Lalit Kumar,
CEMSE, KAUST,
23955, Saudi Arabia,
lalitnbd@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,
Østfold University College,
1757, Norway
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 with a transmission control protocol/internet protocol (TCP/IP) socket 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, synchro-measurement application development framework; PUPO, A PMU-PDC-StreamSimulator; GUI, graphical user interface; MSC, magnitude squared coherence.

I. Introduction

The advancement in the technology of phasor measurement unit (PMU) in the last two decades, has enormously 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 the 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 the IEEE/IEC Std. C37.111-2013 protocol [10]. Using such these data sets, applications for power system monitoring and stability-assessment [11], [12] have been developed. However, addressing how such applications would perform in a real-time environment is usually not addressed.

The increased complexity and vulnerability in the power system, requires to utilize the monitoring-potential of PMUs in real-time 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 the real-time access to PMU or phasor data concentrator (PDC) signals, which becomes nearly impossible 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 is 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' plan 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 develops 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 rest of the article is organized as follows; in Section II, available open-source software for real-time synchrophasor communication are discussed in brief. 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 developed software-toolchain and lastly, the conclusions are drawn in Section V.

II. Open Source Software for Real-Time Synchrophasor Communications

Developing the software-toolchains, requires communication technologies between a client computer (e.g., where applications run) and a server (e.g., PDC). In the recent past, different efforts have developed and released open-source tools [13], [17]–[21] for synchrophasor communication. These software 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 as similar as by commercial PMU/PDC (i.e., acting as a server), and later is to retrieve the signals within a client

computer. Table I lists some open source software which are freely available online.

TABLE I. OPEN SOURCE SOFTWARE FOR REAL-TIME COMMUNICATION

PMU/PDC simulator (Server)	Signal retriever (Client)
• PMU-PDC Stream Simulator (C++) [21]	• SADF (Matlab) [13]
• pyPMU (Python) [17]	• S3DK (Labview) [23]
• iPDC (Linux) [19]	• BabelFish (Labview) [20]
• openPDC (Java) [22]	• PhasorToolBox (Python) [18]

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 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 setup/tool in MATLAB for synchrophasor communication, leading to the development of the Synchro-measurement Application Development Framework (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**-**P**DC-Stream Simulat**O**r' [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 exists 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 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 a PMU signal. Unlike commercial software PDC, it does not receive real-time streams from multiple PMUs. Instead, PUPO emulates a part of a PDCs behavior, which publishes 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. The method used to prepare input files, is described on the website [21]. When preparing input files using typical editing tools, e.g. Microsoft Excel, and convert it to CSV for PUPO (CSV), it is necessary to recall issues with column limits, in editing tools. In cases when of "oversized" PMU data files need to be created (i.e. exceeding > Excel's column limit of 16384 columns), or to deal with MATLAB's limitations to handle oversized data in "double" format, the simplest way to create input files, is to paste the column array in a text editor and then replace all '^p' by nil (blank). This step will delete all "Enter" in the file opened in the text editor and the data will come in row format, which then can be used in PUPO.

By default, the PUPO has one PMU station in which three phasor signals can be imported. However, the users can increase/decrease the number of stations by clicking on 'folder'/'cross' icon as can be seen in Fig. 1. The recorded data files are then published to the localhost using the port chosen, i.e. 4712, in Fig. 1.

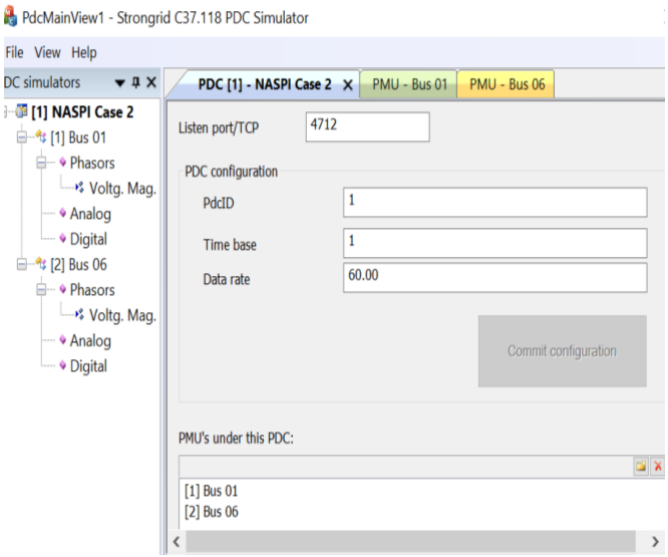

Fig. 1. GUI of PUPO

B. SADF

SADF [13] is a MATLAB-based framework that enables receiving of TCP, UDP, or TCP/UDP synchro-measurement data. SADF not only serve for the purpose of retrieving and storing the signal, but it has a broad application through 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_setting`" 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.

```
68  %%%                                                    %%%
69  % Build your synchro-measurement supported application here %
70  %%%                                                    %%%
```

Fig. 2. Adding a WAMS function in SADF

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_WAMS`" 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_WAMS`" function instead of "`SADF_run`".

In this paper, the software-toolchain developed by utilizing PUPO and SADF is referred as 'PUPO-SADF software-toolchain', which is demonstrated next.

IV. Development and Demonstration of 'PUPO-SADF Software-Toolchain'

This section develops and demonstrates the 'PUPO-SADF software-toolchain' as a simple framework for real-time synchrophasor application development and testing. The development 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 infinite data such as field-recorded real-time data, the length of the retrieved signals to be stored in workspace can be managed in 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 NAPSI signals are already synchronized in its data file (.csv) [5], now we need 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 stopped 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 one single 'array-alignment rule' and thus 'synchronized overlapping check' is passed for PUPO.

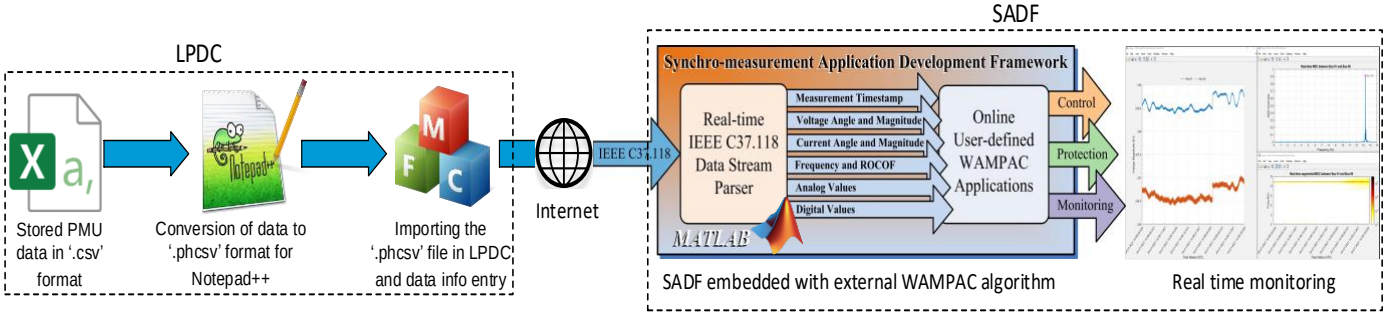

Fig. 3. Flow chart of the PUPO-SADF software-toolchain

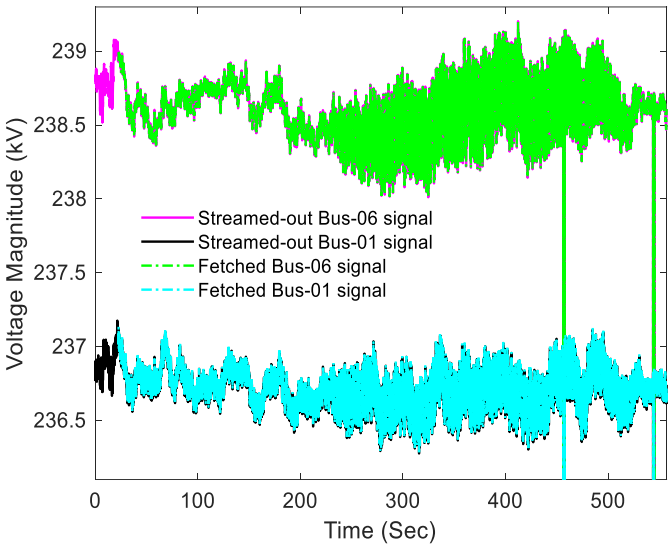

Fig. 4. Streamed out and retrieved voltage signals

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 requires 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 in 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 at the desired 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.

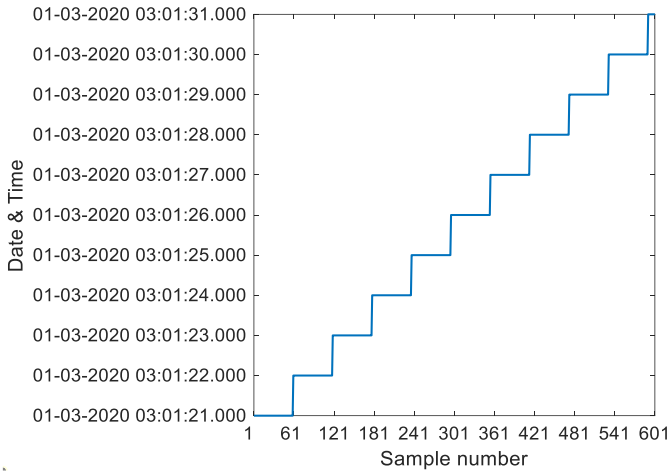

Fig. 5. Plot of retrieved time array (`DATA.TimeStamp`)

Fig. 6. Retrieval of PMU stream and FO monitoring in real-time

As can be noticed in Fig. 5, the published sampling/data rate is still 60 Hz and constant throughout. The incorrect timestamp can be corrected in real-time retrieval using "`linspace`" command in the embedding WAMS function, which will convert the timestamp array from step to linear. In this way, 'sampling rate' check is failed for PUPO, but the data entry error can be rectified, 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 well the first signal corresponds to the second signal at each frequency. However, unlike in Ref. [28], the monitoring here, is in real-time, which requires meticulous encoding for compatibility with '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]. Fig. 6 might seems as unorthodox means of representation, but is inspired from the stacked screen-palette view in the monitoring and control center. 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 was developed for real-time testing of synchrophasor based algorithms in MATLAB. The proposed toolchain was demonstrated using two examples, including a synchrophasor application for oscillation monitoring. The developed 'PUPO-SADF software-toolchain' is easy to build and use, and is hardware-free and 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.

ACKNOWLEDGMENTS

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

[1] L. Vanfretti, S. Bengtsson, and J. O. Gjerde, "Preprocessing synchronized phasor measurement data for spectral analysis of electromechanical oscillations in the Nordic Grid," Int. Trans. Electr. Energy Syst., vol. 25, no. 2, pp. 348–558, 2015, doi: 10.1002/etep.

[2] L. Kumar and N. Kishor, "Determination of mode shapes in PMU signals using two-stage mode decomposition and spectral analysis," IET Gener. Transm. Distrib., vol. 11, pp. 4422–4429, 2017, doi: 10.1049/iet-gtd.2017.0316.

[3] L. Kumar and N. Kishor, "Wide area monitoring of sustained oscillations using double-stage mode decomposition," Int. Trans. Electr. Energy Syst., vol. 28, no. 6, pp. 1–18, 2018, doi: 10.1002/etep.2553.

[4] S. Li, L. Zhang, J. N. Paquin, J. Belanger, and L. Vanfretti, "Hardware-in-the-loop use cases for synchrophasor applications," in 2019 International Conference on Smart Grid Synchronized Measurements and Analytics, SGSMA 2019, 2019, doi: 10.1109/SGSMA.2019.8784526.

[5] "NASPI Oscillation Detection and Voltage Stability Tools Technical Workshop - Houston, TX | North American SynchroPhasor Initiative." [Online]. Available: https://www.naspi.org/node/440. [Accessed: 01-Mar-2020].

[6] "Wide Area Monitoring Protection and Control | Statnett." [Online]. Available: https://www.statnett.no/en/about-statnett/research-and-development/our-prioritised-projects/wide-area-monitoring-protection-and-control/. [Accessed: 01-Mar-2020].

[7] N. Zhou, "A cross-coherence method for detecting oscillations," IEEE Trans. Power Syst., vol. 31, no. 1, pp. 623–631, Jan. 2016, doi: 10.1109/TPWRS.2015.2404804.

[8] "ISO-NE Test Cases | Kaggle." [Online]. Available: https://www.kaggle.com/jacklewis0221/isone-test-cases/data. [Accessed: 01-Mar-2020].

[9] "Power-grid frequency database." [Online]. Available: https://power-grid-frequency.org/. [Accessed: 15-Mar-2022].

[10] Power System Relay Committee of the IEEE, IEEE Standard Common Format for Transient Data Exchange (COMTRADE) for Power Systems, vol. 1999, no. November. 1999.

[11] A. Singh et al., "Report on Low Frequency Oscillation in Indian Power System," Task Force Report, Power System Operation Corporation Limited, New Delhi, 2016.

[12] "Report on frequency quality for the year 2018 has been published - Fingrid." [Online]. Available: https://www.fingrid.fi/en/pages/news/news/2019/report-on-frequency-quality-for-the-year-2018-has-been-published/. [Accessed: 02-Mar-2020].

[13] 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 Trans. Instrum. Meas., vol. 67, no. 8, pp. 1804–1814, Aug. 2018, doi: 10.1109/TIM.2018.2807000.

[14] 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 Trans. Instrum. Meas., vol. 62, no. 4, pp. 701–709, 2013, doi: 10.1109/TIM.2013.2240920.

[15] E. Dusabimana and S.-G. Yoon, "A Survey on the Micro-Phasor Measurement Unit in Distribution Networks," Electronics, vol. 9, no. 2, p. 305, Feb. 2020, doi: 10.3390/electronics9020305.

[16] S. Eisele, I. Mardari, A. Dubey, and G. Karsai, "RIAPS: Resilient information architecture platform for decentralized smart systems," in Proceedings - 2017 IEEE 20th International Symposium on Real-Time Distributed Computing, ISORC 2017, 2017, pp. 125–132, doi: 10.1109/ISORC.2017.22.

[17] S. Sandi, B. Krstajic, and T. Popovic, "PyPMU - Open source python package for synchrophasor data transfer," in 24th Telecommunications Forum, TELFOR 2016, IEEE, 2017, pp. 1–3, doi: 10.1109/TELFOR.2016.7818916.

[18] X. Zhong, P. Arunagirinathan, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks, "PhasorToolBox-A Python Package for Synchrophasor Application Prototyping," in Clemson University Power Systems Conference, PSC 2018, 2019, doi: 10.1109/PSC.2018.8664020.

[19] 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," 2012. [Online]. Available: http://www.iitk.ac.in/npsc/Papers/NPSC2012/papers/12223.pdf. [Accessed: 02-Apr-2020].

[20] M. S. Almas, L. Vanfretti, and M. Baudette, "BabelFish—Tools for IEEE C37.118.2-compliant real-time synchrophasor data mediation," SoftwareX, vol. 6, pp. 209–216, Jan. 2017, doi: 10.1016/j.softx.2017.08.002.

[21] "GitHub - ALSETLab/PMU-PDC-StreamSimulator: A C++ PMU and/or PDC Stream Simulator for IEEE C37.118.2." [Online]. Available: https://github.com/ALSETLab/PMU-PDC-StreamSimulator. [Accessed: 02-Mar-2020].

[22] "Grid Protection Alliance - Home." [Online]. Available: https://www.gridprotectionalliance.org/. [Accessed: 02-Apr-2020].

[23] 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, Jan. 2018, doi: 10.1016/j.softx.2018.08.001.
[24] H. Retty, J. Delport, and V. Centeno, "Development of tests and procedures for evaluating phasor data concentrators," in 2013 IEEE Grenoble Conference PowerTech, POWERTECH 2013, 2013, doi: 10.1109/PTC.2013.6652210.
[25] E. Farantatos, "Let's Talk About Synchrophasors, PMUs & Applications." [Online]. Available: https://www.naspi.org/node/809. [Accessed: 02-Apr-2020].
[26] S. Maslennikov et al., "A test cases library for methods locating the sources of sustained oscillations," in IEEE Power and Energy Society General Meeting, 2016, vol. 2016-Novem, doi: 10.1109/PESGM.2016.7741772.
[27] M. Kezunovic et al., "NASPI Task Force: Report on PMU testing and certification, North American Synchrophasor Initiative Report of Task Force on Testing and Certification," 2013. [Online]. Available: https://www.naspi.org/sites/default/files/reference_documents/43.pdf?fileID=1149. [Accessed: 02-Apr-2020].
[28] N. Zhou and J. Dagle, "Initial results in using a self-coherence method for detecting sustained oscillations," IEEE Trans. Power Syst., vol. 30, no. 1, pp. 522–530, 2015, doi: 10.1109/TPWRS.2014.2321225.
[29] Space-Time Insight, "NASPI Oscillation Detection and Voltage Stability Workshop, Houston, TX, October 22, 2014 - YouTube." [Online]. Available: https://www.youtube.com/watch?v=2WYEj4Qz0Hc&feature=youtu.be. [Accessed: 02-Apr-2020].
[30] A. Silverstein, "NASPI Technical Report: Diagnosing Equipment Health and Mis-operations with PMU Data," 2015. [Online]. Available: https://www.naspi.org/sites/default/files/reference_documents/14.pdf. [Accessed: 02-Apr-2020].
[31] "Video: Real-time FO monitoring." [Online]. Available: https://drive.google.com/file/d/1On1Gd5DV2SO9lkMokjiAezgf8V6RtEaa/view?usp=sharing. [Accessed: 02-Mar-2020].