2022 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA) | 978-1-6654-9823-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/SGSMA51733.2022.9805843

A Python-based Ringdown Analysis Toolbox for Electromechanical Modes Identification

R. D. Reyes and J. A. de la O
UANL
Monterrey, Mexico
rodrigo.reyesd@uanl.edu.mx,
jdelao@ieee.org

M.R.A. Paternina
UNAM
Mexico City, Mexico
mra.paternina@fi-b.unam.mx

J. H. Chow
RPI
Troy, NY, USA
chowj@rpi.edu

A. Zamora and J. Ortiz
U. Michoacana
Michoacan, Mexico
azamoram@umich.mx

***Abstract*—This paper presents a user-friendly graphical user interface (GUI) that is embedded in a Python toolbox to identify electromechanical modes emerging after large power system disturbances. This GUI is able to read xslx or csv files that contain the information of several phasor measurement units (PMUs) or a data file with a proper read format. Besides this GUI incorporates three well-known methods that stand out to perform ringdown analysis in power systems such as Prony's method (PM), eigensystem realization algorithm (ERA), and matrix pencil (MP). A straightforward implementation is adopted to capture dynamic parameters by processing single or multiple channels. Numerical and graphical results demonstrate the usability of the GUI, even in real system events[1].**

***Index Terms*—Electromechanical modes, ringdown analysis, Python toolbox, Prony, Eigensystem Realization, Matrix Pencil.**

I. Introduction

The identification of modal information in power systems after large disturbances provides crucial information for the situational awareness and decision-making process in control centers, since the existence of poor damped modes, or even worse, negatively damped modes may lead to the system separation or complete collapse.

To ensure the correct monitoring of power systems around the world, wide-area monitoring systems (WAMS) [1]- [2] have been implemented with the purpose of collecting electric parameters such as frequency, voltage, current, phase angle and powers. All these parameters are monitored through phasor measurement units (PMUs) installed at different electrical substations [3], achieving in this way, the more PMU installed, the better observability of the system. The main feature of the PMUs is that all measurements have their own time-stamp, allowing to be compared or analyzed with other PMU measurements [4]. With this synchronized information, most WAMS implement different digital signal processing techniques to assess the current status of the system.

Up-to-date, different mathematical methods can capture the necessary information for an adequate system monitoring. In this paper, we incorporate three methods such as Prony, Eigensystem Realization Algorithm (ERA) and matrix pencil (MP), to analyze single and multiple channels [5]. The goal of this paper is focused on proposing a python-based open source platform that extracts the main modal patterns stemming from PMU signals whose information is a valuable tool in both real-time and offline operations to tune controllers and provide reduced order models [6].

Considering the great development in computing, it is easier and necessary to implement the most accurate, reliable and robust mathematical algorithms for monitoring the system performance. Currently, the development of open source tools that adequately calculate the dynamic patterns of the system allows to the operators and analysts at the control centers to know the current status of the system.

A. Related Works

In [7], the authors present an application called Power System Toolbox (PST) to perform power system simulations for small and medium size power systems. This toolbox is developed in Matlab, designed mainly for graduate students. The basic purpose of PST is to provide models for performing power system simulation. PST implements network functions programming a load flow solution using a Newton-Raphson algorithm, a reduced Y matrix computation and a non-conforming load network solution; regarding dynamic models, PST possesses routines to model generators, excitation systems, static var control, power system stabilizers (PSS), HVDC system, wind power plant, battery energy storage, solar PV system, and so on.

In [8], several analysis techniques are implemented such as Fourier transform, Prony's method, MP, S-transform, global wavelet spectrum and Hilbert Huang transform for the identification and analysis of low frequency oscillations in power systems, mainly inter-area electromechanical oscillations. The application was also developed in Matlab. The example consists of analyzing the earthquake that happened on August 23, 2011 in the US East Coast, provoking a 1600 MW generator trip. Frequency measurements obtained from FNET/GridEye were used for analysis and comparing purposes of the aforementioned methods which exhibited good performance to identify electromechanical dynamics.

The common factor of the above-mentioned applications, as well as in [9]-[11], is that those are developed in Matlab which is well known by being powerful, relatively easy to program, equipped with many functions already implemented,

[1]Authors gratefully thank to the Power Information Technology Lab and the FNET/GridEye project led by Prof. Yilu Lu for the real network data provided. M.R.A. Paternina acknowledges financial support from the Project Support Program for Research and Technological Innovation of UNAM (DGAPA, PAPIIT-2021) through the project TA101421.

and popular in student and research communities. Nonetheless, some industries and even some academia that do not have software licensing, are greatly limited to the universality of the software. Despite Matlab-based applications may be converted to executable files (*.exe), this fact forbids the access and edition of the source code. To overcome this problem, Python represents a good alternative, since it is free to use.

Recently, other applications have been developed for power system analysis purposes, they possess open source code and have the advantage that can be run on any PC without licensing [12]-[14].

B. Contribution

This paper implements a Python-based development toolbox for ringdown analysis of oscillating signals recorded by multiple PMUs which arise after large disturbances. All signals are time-synchronized and contain their own time stamp. This application embeds the Prony method together with the ERA and MP methods to analyze single and multiple signals belonging to the same event. The GUI configuration allows to select all signals to be analyzed by defining a time range. The general structure of the program, in a modulated way, is intended to be be extended by including other signal processing methods. Besides a user manual is provided along with the toolbox; where all mathematical formulations, application configuration, execution, results and the export of graphic and numerical data, as well as, criteria and scope of use, are described in detail.

II. Linear Ringdown Analysis Methods

The basic principles of the linear ringdown analysis methods embedded in this toolbox such as Prony, ERA, and MP are enclosed in this section.

A. Prony analysis for single channels

The Prony method is described as a sum of exponentially damped sinusoidal signals, whose signal model is given by [5],

$$\hat{y}(k) = \sum_{i=1}^{n} A_i e^{\sigma_i * n\Delta t} \cos(2\pi f_i n\Delta t + \theta_i), \tag{1}$$

where modal parameters A_i, σ_i, f_i, θ_i respectively stand for amplitude, damping factor, frequency, and phase. Δt is a sample time for N samples in $\hat{y}(t)$. Likewise, the model in (1) can be represented by

$$\hat{y}(k) = \sum_{i=1}^{n} B_i z_i^k, \tag{2}$$

where n is the number of the estimated eigenvalues corresponding to the Prony's model order and $z_i = e^{\lambda_i \Delta t}$ denotes the roots of the n-th characteristic polynomial function of the system. A pseudo-code for the Prony analysis is listed in **Algorithm 1**.

B. Prony analysis for multiple channels

The use of multiple measurement data has been introduced in [15]. Considering a set of M signals $\mathbf{Y}^{(m)}$, $m = 1, ..., M$ that share a common set of eigenvalues obtained from the same event and the same sampling time. Toeplitz matrix $\mathbf{T}$ and vector $\mathbf{Y}$ in (3) can be formulated for every channel and notated as $\mathbf{T}^{(m)}$ and $\mathbf{Y}^{(m)}$. Thus, the LPM in Step 2 of the pseudo-code in **Algorithm 1** can be extended as follows:

$$[\mathbf{Y}^{(1)}\ \mathbf{Y}^{(2)}\ \cdots\ \mathbf{Y}^{(m)}] = [\mathbf{T}^{(1)}\ \mathbf{T}^{(2)}\ \cdots\ \mathbf{T}^{(m)}]\mathbf{a}. \tag{6}$$

and the next steps are the same to estimate the roots z_i using multiple channels.

C. Eigensystem Realization Algorithm: single-channel

The eigensystem realization algorithm is a system identification technique that is widely used to identify and reduce linear systems. The ERA is based on the singular value decomposition (SVD) of the Hankel matrix $\mathbf{H}_0$ associated with the ringdown behavior of a linear system. A pseudo-code for the ERA method is detailed in **Algorithm 2**.

D. Eigensystem Realization Algorithm: multiple-channel

For multiple outputs channels, a set of M signals is shaped in matrix form such that $\mathbf{Y}_m \in \Re^{N\times m}$, that is, m-column arrays corresponding to single channels, as follows:

$$\mathbf{Y}_m = [\mathbf{y}^{\{1\}}\ \mathbf{y}^{\{2\}}\ \cdots\ \mathbf{y}^{\{q\}}\ \cdots\ \mathbf{y}^{\{m\}}] \tag{11}$$

with $\mathbf{y}^{\{q\}} = [y(0)\ y(1)\ \cdots\ y(N-1)]^T$.

The Hankel matrix can be also derived for multiple output channels by using submatrices that are constructed by each signal. This is called a block Hankel matrix and is given by

$$\tilde{\mathbf{H}}_0 = [\mathbf{H}^1 \quad \mathbf{H}^2 \quad \cdots \quad \mathbf{H}^m]^T \tag{12}$$

Thus, this block Hankel matrix and its shifted Hankel matrix $\tilde{\mathbf{H}}_1$ are replaced in Step 1 of the pseudo-code in **Algorithm 2**,

Algorithm 1 Prony analysis.

```
Require: Time-series data of recorded signal $y(k)$ with $N$ samples
Ensure: $n \leq N$ is the subset of modes to be determined
1: To construct a Toeplitz matrix $\mathbf{T}$ using the elements of the recorded signal
2: To fit the data with a discrete linear prediction model (LPM) of order N  ▷
       $\mathbf{Y} = \mathbf{Ta}$    (3)
   ▷ where $\mathbf{Y} = [y(n) \quad y(n+1) \quad \cdots \quad y(N-1)]^T$ is of size $N$.
3: To find the polynomial coefficients  ▷
       $\mathbf{a} = (\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T\mathbf{Y}$    (4)
4: To find the $z_i$ roots of the polynomial  ▷
       $z^n - (a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_{n-1} z^0) = 0$    (5)
return $z_i$
```

enabling to extend for multiple channels. Hence, the next steps are the same to estimate the roots z_i using multiple channels.

E. Matrix Pencil: single-channel

The matrix pencil method produces a matrix whose roots (z_i) facilitate the extraction of the eigenvalues [16]-[18]. The number of significant modes (n) in the system is determined from the SVD of the Hankel matrix $\mathbf{H}$. A pseudo-code for the MP method is presented in **Algorithm 3**.

F. Matrix Pencil: multiple-channel

This implementation was introduced in [18]. For each channel, two shifted Hankel matrices are formed. Note that $\mathbf{H}_1^{(m)}$ and $\mathbf{H}_2^{(m)}$ are the Hankel matrices for the m-th channel [19]. The aggregated Hankel matrices are, as follows:

$$\mathbf{H}_1 = [\mathbf{H}_1^{(1)}, \mathbf{H}_1^{(2)}, ..., \mathbf{H}_1^{(m)}] \tag{18}$$

$$\mathbf{H}_2 = [\mathbf{H}_2^{(1)}, \mathbf{H}_2^{(2)}, ..., \mathbf{H}_2^{(m)}] \tag{19}$$

These Hankel matrices in (18)-(19) are decomposed in the same way as in (13). The complete process to get z_i, it is the same as the single-channel from (14)-(17) in **Algorithm 3**. The last steps are the same to estimate the roots z_i using multiple channels.

G. Estimation of modal parameters

Once the roots z_i are computed by all three methods according the pseudo-codes in **Algorithms 1-3**, the eigenvalues of the discrete system z_i can be found as $\lambda_i = \frac{\ln(z_i)}{\Delta t}$ and the modal parameters can be estimated by

$$\hat{f}_i = \mathrm{Im}\left(\frac{\lambda_i}{2\pi}\right) \tag{20}$$

$$\hat{\sigma}_i = \mathrm{Re}(\lambda_i) \tag{21}$$

The damping ratio is obtained as

$$\hat{\zeta}_i = \frac{\hat{\sigma}_i}{\hat{\omega}_i} \tag{22}$$

where $\hat{\omega}_i = 2\pi\hat{f}_i$. Likewise, the modal energy can be estimated by

$$\hat{E}_i = \frac{1}{2}\hat{\omega}_i^2\hat{A}_i^2 \tag{23}$$

where $\hat{A}_i$ corresponds to the amplitude estimates, $\hat{A}_i = 2|B_i|$.

The residues B_i are calculated solving the Vandermonde problem using the z_i roots as

$$\mathbf{B} = (\mathbf{Z}^H\mathbf{Z})^{-1}\mathbf{Z}^H\mathbf{y} \tag{24}$$

where the complex matrix $\mathbf{Z}$ is formed using the roots z_i, and $\mathbf{y}$ is the recorded signal of size N.

Algorithm 2 Eigensystem Realization Algorithm.

```
Require: Time-series data of recorded signal $y(k)$ with $N$ samples
Ensure: $r = \frac{N}{2} - 1$.    ▷ Where $r$ is the size of the Hankel matrix $\mathbf{H}$, this choice assumes that the number of data points is sufficient such that $r > n$
1: To construct the Hankel matrix $\mathbf{H}_0$ and its shifted Hankel matrix $\mathbf{H}_1$ using the elements of the recorded data signal
2: To apply SVD to    ▷
       $\mathbf{H}_0 = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$    (7)
3: To separate $\mathbf{H}_0$ into two components, a $n$ large (nonzero in the case of noiseless measurements) and $s$ small (zero in the case of noiseless measurements) singular values    ▷
       $\mathbf{H}_0 = \begin{bmatrix}\mathbf{U}_n & \mathbf{U}_s\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_n & 0 \\ 0 & \boldsymbol{\Sigma}_s\end{bmatrix}\begin{bmatrix}\mathbf{V}_n^T \\ \mathbf{V}_s^T\end{bmatrix}$    (8)
4: To approximate the high-rank Hankel matrix $\mathbf{H}_0$ by a reduced-rank $n$ matrix    ▷
       $\mathbf{H}_0 \approx \mathbf{U}_n\boldsymbol{\Sigma}_n\mathbf{V}_n^T$    (9)
5: To compute the discrete system matrix $\mathbf{A}$    ▷
       $\mathbf{A} = \boldsymbol{\Sigma}_n^{-\frac{1}{2}}\mathbf{U}_n\mathbf{H}_1\mathbf{V}_n^T\boldsymbol{\Sigma}_n^{-\frac{1}{2}}$    (10)
6: To compute $z_i = eig(\mathbf{A})$
return $z_i$
```

Algorithm 3 Matrix Pencil method.

```
Require: Time-series data of recorded signal $y(k)$ with $N$ samples
Ensure: $r = \frac{N}{2} - 1$.    ▷ Where $r$ is the size of the Hankel matrix $\mathbf{H}$, this choice assumes that the number of data points is sufficient such that $r > n$
1: To construct $\mathbf{H}$ using the elements of the recorded data signal    ▷ This is similar as in the ERA method, $\mathbf{H}_0$
2: To apply SVD to    ▷
       $\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$    (13)
3: To define matrices $\mathbf{V}_1$ and $\mathbf{V}_2$    ▷
       $\mathbf{V}_1 = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_{n-1}]$    (14)
       $\mathbf{V}_2 = [\mathbf{v}_2 \quad \mathbf{v}_3 \quad ... \quad \mathbf{v}_n]$    (15)
4: To compute matrices $\mathbf{Y}_1$ and $\mathbf{Y}_2$    ▷
       $\mathbf{Y}_1 = \mathbf{V}_1^T\mathbf{V}_1$    (16)
       $\mathbf{Y}_2 = \mathbf{V}_2^T\mathbf{V}_1$    (17)
5: To compute $z_i = eig(\mathbf{Y}_1^{\dagger}\mathbf{Y}_2)$.
return $z_i$
```

III. Graphical User Interface Structure

This section encloses the implementation via Python.

A. Python programming language

Python is an interpreted high-level general-purpose programming language [20]. Its design philosophy emphasizes code readability with its use of significant indentation. Its language allows to construct logical code for small and large-scale projects [21], [22].

Due to the Python inherent features (simple, intuitive, free and open source, portability, expansibility, embeddability, large and comprehensive standard libraries), this GUI is implemented considering an object-oriented programming (OOP) methodology, which it takes the following advantages:

- To provide a clear and ordered program structure.
- Easy maintenance and modification of existing code.
- To enhance the program modularity because each object exists independently and new features can be added easily without disturbing the existing ones.
- To present a good framework for code libraries where all supplied components can be easily adapted and modified.
- To impart code reusability.

B. Application

The goal of this toolbox is the analysis of a single signal or a set of signals obtained from events measured in electrical power systems, which can be synchrophasor data containing the same time-stamp. When a set of signals is considered, this must be obtained from the same event in the same system. The analysis methods embedded in this platform are: Prony, Matrix Pencil and Eigensystem Realization. These are integrated in the GUI allowing their execution in a simple way, enabling the provision of graphical and numerical results [23].

C. General structure and libraries

This application contains three main classes: *Window1*, *Window2* and *Window3*. In turn, these classes correspond to three windows (objects) of the GUI, such as: *Main Window*, *Edit Window* and *View Window*, respectively. Classes 2 and 3 exchange information with *class 1* which manage the data reception, settings, analysis and results.

This GUI is developed using mainly the Tkinter library for the graphical interface, Numpy and Scipy are employed as mathematical tools, and Matplotlib is used to plot all results.

The flowchart of the general schematization for this GUI is enclosed in Fig. 1. The first step is to read the file with data samples from PMU or other devices providing the same time-stamp for all samples. These files must meet certain criteria in order to be read. Then, the configuration according to the criteria of each user needs to be established; if some errors or inconsistencies appear, then the application notifies the error being necessary to correct them for continuing with the execution of the GUI by clicking the button *RUN*. Afterwards, the application will collect all data verifying that everything is correct and it will show all numerical results in the results section.

D. Requirements for plugging new methods

Since the code implementation is structured using interconnected modules, the division of variables is easy to manipulate, allowing to correct, improve, and implement new actions to the application.

The requirements for the incorporation of new analysis methods are a few, but they must respect the pre-established criteria for the correct read and operation of the application.

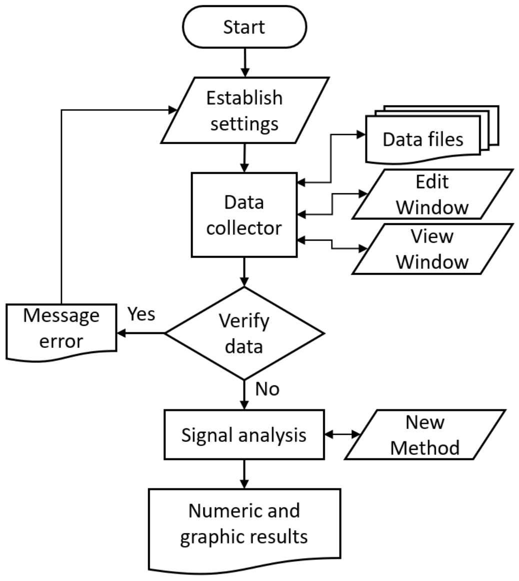


Fig. 1. Graphical user interface's flow diagram.

The new function must have a different name to: *Prony*, *MP*, *ERA*, *Multi-Prony*, *Multi-MP*, *Multi-ERA*. These names are already used in the current application. The order and data type must be respected as *def* func_name(*self*, *Y*, *N*, *dt*, *fct_method*), whose input arguments are listed below:

- *Y* (float), numpy vector with n samples of the signal.
- *N* (integer), scalar value with the vector Y length.
- *dt* (float), scalar value with the sampling period.
- *fct_method* (float), numerical scalar value that may be provided from the interface.

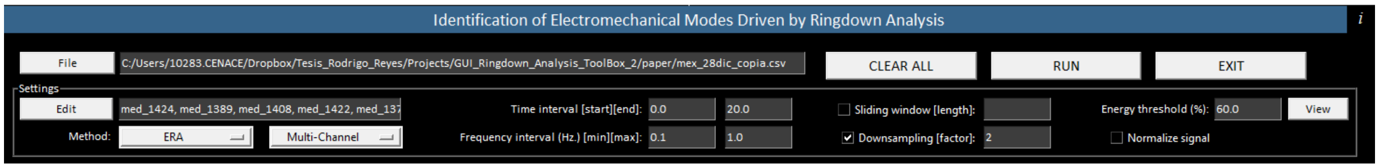
The return of function values must be provided with the following syntax:

- *modes* (integer, scalar), calculated modes.
- *mag* (float, numpy matrix), calculated magnitudes, which depend on the input signal).
- *ang* (float, numpy matrix) calculated phase angles (deg).
- *damp.* (float, numpy vector) calculated damping $(1/s)$.
- *freq.* (float, numpy vector) calculated frequency (Hz).
- *damprat* (float, numpy vector) damping ratio (%).
- *energy* (float, numpy vector) calculated energy (%).
- *roots* (float, numpy complex vector) calculated poles and zeros.

E. Settings

At the top of the main window is located the settings which are shown in Fig. 2. In this section, from left to right are the following elements:

- *File button* is the browser, allowed extensions **.csv* or **.xlsx*.
- *Manual entry 1* (*string*), it allows to provide the path, name and extension of file to be read.
- *Clear button*, it clears all data and graphics of the main window.
- *RUN button*, it executes the analysis of the signal(s).
- *Exit button*, it exits of the application.
- *Edit button*, it displays graphical and numerical input data.


Identification of Electromechanical Modes Driven by Ringdown Analysis

File: C:/Users/10283.CENACE/Dropbox/Tesis_Rodrigo_Reyes/Projects/GUI_Ringdown_Analysis_ToolBox_2/paper/mex_28dic_copia.csv

Settings

Edit: med_1424, med_1389, med_1408, med_1422, med_137

Method: ERA | Multi-Channel

Time interval [start][end]: 0.0 | 20.0

Frequency interval (Hz.) [min][max]: 0.1 | 1.0

☐ Sliding window [length]:

✓ Downsampling [factor]: 2

Energy threshold (%): 60.0

☐ Normalize signal

Fig. 2. Settings of the main window of the GUI.

- *Manual entry 2* (*text string*), it allows to edit the signals' name.
- *Menu button 1*, it allows to choose the solution method: Prony, MP or ERA.
- *Menu button 2*, it enables to select single or multiple channels.
- *Manual entry 3/4* (*float*), it enables to provide the initial and final time of the analysis window.
- *Manual entry 5/6* (*float*), it enables to provide the initial and final time of the Fourier spectrum window.
- *Box 1* (*Boolean*), it activates or deactivates sliding window analysis.
- *Manual Entry 7* (*float*), it enables to adjust the size of the sliding window.
- *Box 2* (*Boolean*), it activates or deactivates the downsampling.
- *Manual entry 8* (*integer*), it enables to provide a downsampling factor.
- *Manual entry 9*, this has two options depending on the selection of the analysis method *Prony* or any of *ERA* or *MP*.
  - If *Prony* (*integer*) is selected, this leads to ask for the number of *Modes*, then the *View* button is disabled.
  - If *ERA or MP* (*float*) are selected, this lead to ask for the *Energy threshold (%)* to get the singular values, then the *View button* is enabled.
- *View button*, it allows to choose the number of singular values from the Hankel matrix for ERA o MP.
- *Box 3* (*Boolean*), it activates or deactivates the signal normalization.

F. Application example for a real system event in Mexico

In Mexico, there are four electrical systems that operate in isolation, Muleje, Baja California (BCA), Baja California Sur (BCS), and the National Interconnected System (NIS) which is the largest one. The total installed capacity is approximately 83,120MW with a maximum demand about 50GW in 2019. The NIS is a mesh system with radial connections towards the northwest and southeast. Its transmission level is mainly composed of transmission lines (TL) with operating voltages of 400kV, 230 kV and 130-69kV; and it covers from Quintana Roo to Sonora state, making up the main electrical power grid in Mexico. The NIS is split into 7 regions: Central, Eastern, Western, Northwestern, North, Northeastern, and Peninsular. For the sake of brevity, public official documents are available at [23].

On Dec 28th, 2020, two of the main TLs that interconnect the Northwestern, North and Northeastern regions with the Central, Eastern, Western and Peninsular regions were tripped provoking a cascade event. Due to the loading capacity of the TLs and the presence of power oscillations, the Northwestern, North and Northeastern regions remained connected with the Central, Eastern, Western and Peninsular by just one TL, despite they were out of step and they rotated against each other. Thus, this sequence of events yielded that the power generation surpassed the load demand in the Northwestern, North and Northeastern causing a frequency excursion up to 61.14Hz that activated the stages of remedial action schemes; as consequence, several generators were tripped. Meanwhile the load demand surpassed the power generation in the Central, Eastern, Western, and Peninsular regions, dropping down the frequency up to 58.76Hz and activating the load shedding stages, as consequence of the low frequency. The system total affectation of power generation and load was of 8,696MW, representing 26% of the demand .

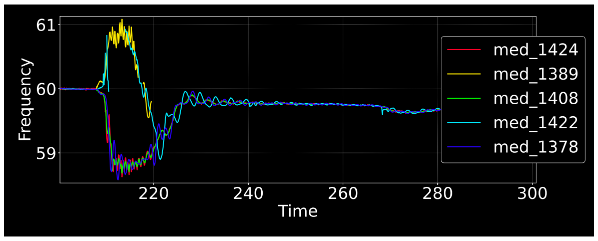


Fig. 3. Frequency measurements of 5 FDRs located at the NIS.

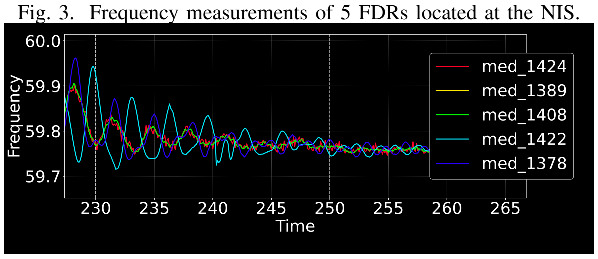


Fig. 4. Ringdown window analysis.

This example case consists of the recorded measurements in the mentioned event, which are collected by five time-synchronized frequency disturbance recorders (FDRs) located at the following locations: 1389-Monterrey (Northeastern), 1422-Mazatlan (Northwestern), 1378-Guadalajara and 1408-Morelia (both Western), and 1424-Chetumal (Peninsular). Numbers denote the ID code for the FDRs in the FNET/GridEye project[2]. The collected measurements from five regions are illustrated in Fig. 3, where the measurements 1389, 1422 exhibited a frequency increment; whereas the measurements 1378, 1408 and 1424 presented a frequency drop. For this real event, the length of the analysis window is adopted of 20s, as shows in Fig. 4.

G. Results

Once the configuration and inputs are completed, the results are displayed in the tabs of the results section. This section

[2]http://fnetpublic.utk.edu/index.html

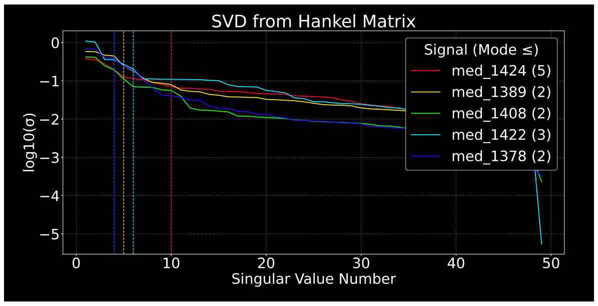


Fig. 5. Singular values - View window.

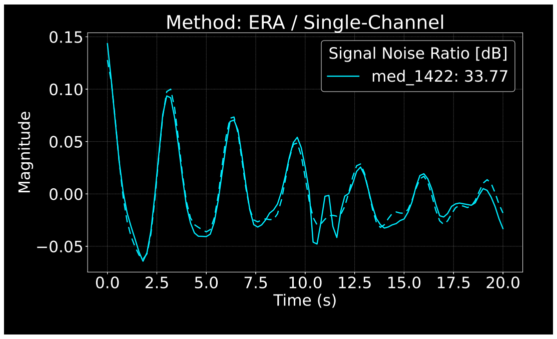


Fig. 6. Results - Signal tab.

summarizes all numerical results in a table and a graphical book with four tabs. The table shows all numerical results from the selected method, giving the option to select or deselect any of the calculated mode. In turn, this change will be represented in the graphic results book.

For this example, an ERA single-channel method is considered which is setup with a 60% of energy threshold. A visualization of the singular values associated with the Hankel matrix for the selected signals and the established configuration, is possible, as displays in Fig. 5. Where five signals are shown for all analyzed signals. The legend of Fig. 5 indicates the maximum number of modes to obtain, this can be reconfigured whether it is necessary to get a better approximation for known modes.

The result section is described into the following: the first tab depicts the original signal (continuous line) and its approximation (dashed line) in Fig. 6. After adding all calculated modes, the reconstructed signal matches very well to the original signal. The legend of Fig. 6 shows the signal-to-noise ratio (SNR) of the signal *vs* the approximation. This is useful to quantify the miss-match of the calculus and the capacity of the signal reconstruction.

The second tab of the graphic book is the Fourier spectrum. This graph is the only one that does not depend on the analysis method or any other configuration in the application. It is only to show the frequencies obtained after applying the Fourier analysis, and it provides a reference for our analysis. Continuing with the previous example, the Fourier spectrum is shown in Fig. 7, where all signals are shown without considering the signal selector.

The third tab of the graphic book is the mode shapes. To plot this feature, it is necessary to select *Multi-channel* analysis, since the computation of the mode shapes works with several signals at a time. This tab shows the forms of the calculated modes which are the vectors for all signals

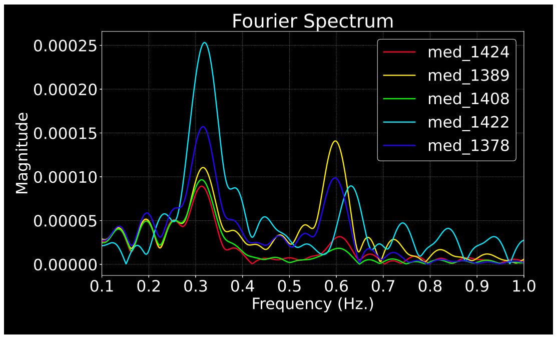


Fig. 7. Results - Fourier Spectrum tab.

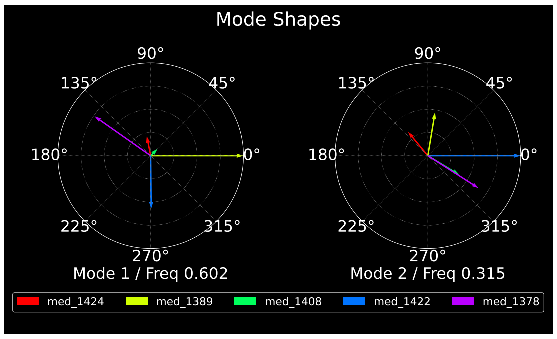


Fig. 8. Results - Mode shape tab.

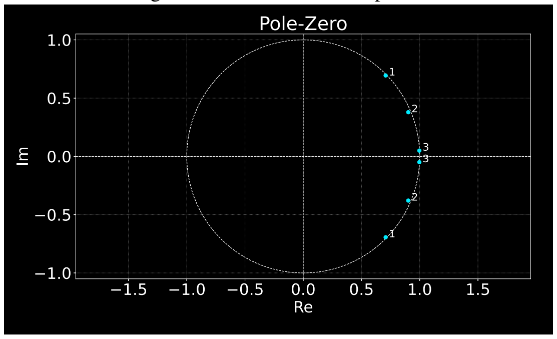


Fig. 9. Results - Pole Zero in the z-plane for tab.

in the i-th mode. Mode shapes are tipically exhibited in a polar plane, considering the signal with the higher magnitude as reference in magnitude and angle to normalize between 0 and 1. Fig. 8 depicts the result for the signal approximation modes consisting of two modes, where two frequencies at 0.315 and 0.602Hz can be observed, these match with those of the Fourier spectrum shown in Fig. 7.

The fourth tab of the graphical result book shows the calculated poles and zeros. Figure 9 depicts the result of the poles and zeros. Three dominant modes are obtained considering a 60.0% of the energy threshold; they are stable modes since are within the unit circle, unstable modes are symbolized by X.

The export of the attained results is done through the *Export* button in the main window. This button creates an Excel workbook in xlsx format with the latest solution information obtained from the application. The file is created in the same address as the application path, being named as *results_ddmmyyyy_hhmmss.xlsx*, substituting day, month, year, hour, minute and second, respectively, in which the file was generated. The results' file shows the numerical information of the analysis of the signal(s), where the first part indicates the settings and criteria for the analysis; whereas the second part

TABLE I
NUMERICAL RESULTS AFTER ANALYZING FIVE SIGNALS COLLECTED FROM A NIS REAL EVENT AT DEC 28^{th}, 2020.

Signal name	Mode number	Type	Frequency (Hz)	Amplitude	Damping $(1/s)$	Damping ratio (%)	Phase (rad)	Pole-Zero (Re + jIm)		Energy (%)
med_1424	1	Intra-Plant	2.4523	0.0061	-0.0951	-0.6174	-0.7260	-0.9794	$j0.0588$	0.4360
med_1424	2	Inter-Area	0.5987	0.0091	-0.0918	-2.4398	2.4182	0.7168	$j0.6709$	0.0592
med_1424	3	Inter-Area	0.3103	0.0350	-0.1096	-5.6186	3.1234	0.9049	$j0.3719$	0.2335
med_1389	1	Inter-Area	0.5959	0.0469	-0.0870	-2.3238	-0.4091	0.7198	$j0.6690$	1.5447
med_1389	2	Inter-Area	0.2997	0.0437	-0.1112	-5.9047	-2.7662	0.9095	$j0.3596$	0.3380
med_1408	1	Inter-Area	0.3052	0.0374	-0.1069	-5.5738	-2.8171	0.9077	$j0.3662$	0.2568
med_1422	1	Inter-Area	0.6175	0.0210	-0.0433	-1.1160	0.6164	0.7076	$j0.6943$	0.3329
med_1422	2	Inter-Area	0.3165	0.1025	-0.1096	-5.5100	-0.0587	0.9019	$j.3789$	2.0775
med_1378	1	Inter-Area	0.3049	0.0636	-0.1136	-5.9317	-2.7873	0.9067	$j0.3653$	0.7410
med_1378	2	Inter-Area	0.5921	0.0353	-0.0948	-2.5485	-0.0641	0.7219	$j0.6645$	0.8615

exhibits all dynamic parameters. Table I depicts the numerical results for the NIS real event, this can be exported in xlsx format; meanwhile graphical results can be saved in several formats (eps, pdf, png, etc).

For this example, the execution time was of 0.01454s, meanwhile the total execution time was 0.5492s using an average time over 10 runs. These computations were done using Python 3.9 environment in a Windows 10 desktop with Intel(R) Xeon(R) CPU E5-1620 v4, 3.5 GHz, 32 GB RAM, and 64-bit.

IV. CONCLUSION

This article presents the development of an open source platform to conduct ringdown analysis in power systems. The GUI application can effectively process signals stemming from multiple sensors or PMUs. This encloses a graphical user interface that has a very simple and intuitive interface, capable of analyzing a single-signal or multi-signals from the same event, with three different methods. It is also demonstrated that the GUI solves and presents results in short time. In turn, results can be numerically and graphically presented and exported in an easy way.

The modular way conceived to build this application is also shown, enabling the inclusion of other or new analysis methods by considering the guidelines and criteria necessary to add subroutines to the graphical interface. The major advantage of this application is its developed environment, since it allows any user that has the programming language installed on its PC, to run this application, do studies and implement other electromechanical identification methods in the same GUI (Python does not require licensing).

REFERENCES

[1] T. U. of Tennessee Knoxville, "Fnet/grideye frequency display," http://fnetpublic.utk.edu/tabledisplay.html, May 2018.

[2] B. Indian Institute of Technology, "Wide area frequency measurement system," http://103.7.128.82/rwafms/wafms/, May 2018.

[3] A. Zamora, V. M. Venkatasubramanian, J. A. de la O Serna, J. M. Ramirez, and M. Paternina, "Multi-dimensional ringdown modal analysis by filtering," *Electric Power Systems Research*, vol. 143, pp. 748–759, 2017.

[4] J. Y. Cai, Z. Huang, J. Hauer, and K. Martin, "Current status and experience of wams implementation in north america," in *IEEE/PES Transm. & Distrib.: Asia and Pacific*. IEEE, 2005, pp. 1–7.

[5] J. Sanchez-Gasca and D. Trudnowski, "Identification of electromechanical modes in power system," IEEE Task Force on Identification of Electromechanical Modes of the Power System Stability, Power & Energy Society, Tech. Rep., June 2012.

[6] J. H. Chow, *Power system coherency and model reduction*. Springer, 2013.

[7] J. H. Chow and K. W. Cheung, "A toolbox for power system dynamics and control engineering education and research," *IEEE Trans. Power Systems*, vol. 7, no. 4, pp. 1559–1564, 1992.

[8] M. M. Dedović, S. Avdaković, A. Mujezinović, and N. Dautbašić, "Comparison of different methods for identification of dominant oscillation mode komparacija različitih metoda za identifikaciju dominanontnih oscilatornih modova," *B&H Electrical*, p. 43.

[9] K. Schoder, A. Hasanovic, and A. Feliachi, "Pat: a power analysis toolbox for matlab/simulink," *IEEE Trans. Power Systems*, vol. 18, no. 1, pp. 42–47, 2003.

[10] E. Zamora-Cárdenas, A. Pizano-Martínez, J. Lozano-García, V. Gutiérrez-Martínez, and R. Cisneros-Magaña, "Computational development of a practical educational tool for state estimation of power systems using the matlab optimization toolbox," *Intern. J. of Elect. Eng. & Educ.*, vol. 56, no. 2, pp. 105–123, 2019.

[11] S. Cole and R. Belmans, "Matdyn, a new matlab-based toolbox for power system dynamic simulation," *IEEE Trans. Power systems*, vol. 26, no. 3, pp. 1129–1136, 2010.

[12] W. Bukhsh, C. Edmunds, and K. Bell, "Oats: Optimisation and analysis toolbox for power systems," *IEEE Trans. Power Systems*, vol. 35, no. 5, pp. 3552–3561, 2020.

[13] L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Trans. Power Systems*, vol. 33, no. 6, pp. 6510–6521, 2018.

[14] T. Phongtrakul, Y. Kongjeen, and K. Bhumkittipich, "Analysis of power load flow for power distribution system based on pypsa toolbox," in *15th Intern. Conf. on Electr., Computer, Tel. and Inf. Tech.* IEEE, 2018, pp. 764–767.

[15] D. J. Trudnowski, J. Johnson, and J. F. Hauer, "Making prony analysis more accurate using multiple signals," *IEEE Trans. Power Systems*, vol. 14, no. 1, pp. 226–231, 1999.

[16] A. Almunif, L. Fan, and Z. Miao, "A tutorial on data-driven eigenvalue identification: Prony analysis, matrix pencil, and eigensystem realization algorithm," *Int. Trans. Electr. Energy Syst.*, vol. 30, no. 4, p. e12283, 2020.

[17] N. Yilmazer, J. Koh, and T. Sarkar, "Utilization of a unitary transform for efficient computation in the matrix pencil method to find the direction of arrival," *IEEE Trans. Antennas and Propagation*, vol. 54, no. 1, pp. 175–181, 2006.

[18] T. K. Sarkar, S. Park, J. Koh, and S. M. Rao, "Application of the matrix pencil method for estimating the sem (singularity expansion method) poles of source-free transient responses from multiple look directions," *IEEE Trans. Antennas and Propagation*, vol. 48, no. 4, pp. 612–618, 2000.

[19] A. Almunif, L. Fan, and Z. Miao, "A tutorial on data-driven eigenvalue identification: Prony analysis, matrix pencil, and eigensystem realization algorithm," *Intern. Trans. Electrical Energy Systems*, vol. 30, no. 4, p. e12283, 2020.

[20] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[21] D. Kuhlman, *A python book: Beginning python, advanced python, and python exercises*. Dave Kuhlman Lutz, 2009.

[22] S. Tosi, *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[23] R. Reyes, M. Paternina, A. Zamora, J. de la O, J. Chow, and J. Bejar, *A Python-based Ringdown AnalysisToolbox for Electromechanical Modes Identification*, 2021 (accessed Nov 10, 2021), https://github.com/rreyesdeluna/Python-based-Ringdown-AnalysisToolbox.git.