World Academy of Science, Engineering and Technology
International Journal of Educational and Pedagogical Sciences
Vol:16, No:3, 2022

Elegant: An Intuitive Software Tool for Interactive Learning of Power System Analysis

Eduardo N. Velloso, Fernando M. N. Dantas, Luciano S. Barros

***Abstract*—**A common complaint from power system analysis students lies in the overly complex tools they need to learn and use just to simulate very basic systems or just to check the answers to power system calculations. The most basic power system studies are power-flow solutions and short-circuit calculations. This paper presents a simple tool with an intuitive interface to perform both these studies and assess its performance in comparison with existent commercial solutions. With this in mind, Elegant is a pure Python software tool for learning power system analysis developed for undergraduate and graduate students. It solves the power-flow problem by iterative numerical methods and calculates bolted short-circuit fault currents by modeling the network in the domain of symmetrical components. Elegant can be used with a user-friendly Graphical User Interface (GUI) and automatically generates human-readable reports of the simulation results. The tool is exemplified using a typical Brazilian regional system with 18 buses. This study performs a comparative experiment with 1 undergraduate and 4 graduate students who attempted the same problem using both Elegant and a commercial tool. It was found that Elegant significantly reduces the time and labor involved in basic power system simulations while still providing some insights into real power system designs.

***Keywords*—**Free- and open-source software, power-flow, power system analysis, Python, short-circuit.

I. Introduction

THE successful planning, design, and operation of industrial and commercial power systems often require a number of studies to help in evaluating their initial and future performance, as well as their reliability, safety, and ability to scale up with production or operational demands. Simulations are therefore ubiquitous in modern power system analysis and design [1]. Most applications in the electrical sector require efficient and powerful software able to deal with the ever-increasing size of interconnected power grids. However, when engineering students are introduced to the concepts and techniques behind power system calculations, these commercial tools are usually the ones to which they turn to for learning aid. This is generally less than ideal given that, in this specific context, systems are quite simple and most programs have a steep learning curve and require putting in some effort even for the most basic tasks. As noted by [2] positive, immediate feedback is an important part of a rapid, efficient learning process for a large majority of students.

With the increased access to personal computers in the nineties, some educational packages were presented as alternatives to closed commercial software [3]-[5]. With the popularity of high-level languages for designing educational programs, many MATLAB power system tools have been proposed (e.g., PSAT [6]). Recent efforts have been directed toward developing open-source power system tools in Python such as pandapower [7], which aims at automating optimization problems, and PyPSA [8], aimed at time-series simulation of optimal power flow. A key feature that favors Python-based products is the well-developed set of libraries for scientific computation that this language has. Other notable Python power system analysis tools include GridCal [9], which comes with a GUI, and PYPOWER [10], the Python port to well-known MATLAB tool MATPOWER [11]. For a more thorough review of software packages in power system analysis, see [12], [13].

In this paper, we present Elegant (Electrical Grid Analysis Tool), a software system that analyzes basic electrical networks in the most intuitive fashion possible. It is open-source and implemented in pure Python under the GNU General Public License. Elegant allows students to draw their own networks, to calculate bus voltages and power flows, and to estimate prospective short-circuit currents. It can also be used to develop their own tools based on the extensible framework provided.

In the design of the proposed tool, we have tried to include the following aspects:

- a minimalist, user-friendly, and clean GUI, so even the most inexperienced users can simulate their networks quite easily;
- real-time updates, encouraging users to experiment with their data;
- generation of clear, human-readable documents reporting the simulation results with detailed, informative tables.

The trade-off is naturally that the user will not have as many different electric elements available as other open-source libraries. While this does mean that Elegant is not particularly suitable for commercial or research purposes, it can still be used to simulate real electrical grids and to bring valuable insight into their design, as we intend to demonstrate in this paper.

The remaining part of this paper is organized as follows: The next section focuses on the description of the software operation and interface. After that, in Section III we demonstrate a sample usage of Elegant with a case study and in Section IV we discuss the simple experiment we performed to validate our proposed

E. N. Velloso and F. M. N. Dantas are with the Department of Electrical Engineering, Federal University of Rio Grande do Norte, Natal, Brazil (e-mail: dioph@pm.me).

L. S. Barros is with the Department of Computer Systems, Federal University of Paraíba, João Pessoa, Brazil.

software. Finally, Section V discusses the conclusions and perspectives.

II. Software Description

We outline below a number of features related to the use of Elegant and describe each one in detail.

A. Using Elegant as a Regular Python Package

Elegant is a regular package developed in Python 3 and made freely available to download from the Python Package Index (PyPI). Consequently, all its functionalities can be imported within user-defined programs in order to script computations.

The system is modeled by the Elegant package using bus-branch representation, in which the network topology is translated to a set of buses interconnected by generic branches. The *PowerSystem* class is its main data structure, representing the system to be simulated. It allows the user to add buses and create connections between them using *TransmissionLine* or *Transformer* objects. Currently, the power base is hardcoded to be $S_{\text{base}} = 100$ MVA.

A *PowerSystem* object has convenient functions to solve the power-flow problem for the bus voltages V and complex power S as well as to derive the fault currents for each of the previously mentioned four fault types considering each of the nodes as the faulted one. Alternatively, all these steps can be automatically performed with its *update* function. To achieve this, every system state can be translated to matrices Y, $Y^{(0)}$ and $Y^{(1)}$, corresponding to the admittance matrix of the network and its zero sequence and positive sequence equivalents.

For the power-flow analysis, there are implementations available for both Gauss-Seidel and Newton-Raphson methods, with the possibility to tune the number of iterations or the tolerance ε.

B. Graphical User Interface

The proposed interface was developed using *PyQt5*, a Python binding for the well-known *Qt* GUI framework. The Elegant interface was designed to handle possible user mistakes, i.e., existing circuit data are continually verified in order to avoid invalid inputs and posterior misleading results.

The application is based on a grid-like canvas. When interacting with the canvas, there are three movements that the user can perform: single-clicking, double-clicking, and mouse-dragging.

A *single-click* simply selects the network element in the cursor position. Whenever the user clicks at any point inside the canvas, a yellow square highlights the selected grid position, as can be seen in Fig. 1. More importantly, if the selected position contains an element, the corresponding *inspector menu* will open in the sidebar as a convenient way to access the relevant data, as shown by Fig. 2.

A *double-click* inserts a new bus in the grid at the cursor location, as long as there are no other elements already placed there. We have chosen to represent the buses as circles instead of vertical or horizontal bars in order to improve visual clarity, as recommended by [14]. Increasing integers are automatically assigned to the buses as identifiers, changing accordingly when buses are removed. The first bus to be inserted is always the slack bus, and if it gets removed the next insertion will be a new slack bus. This guarantees the presence of one (and no more than one) angular reference in the grid. The simulations are always performed solely with the connected graph component containing the slack, which means no calculations are made without the presence of a slack bus.

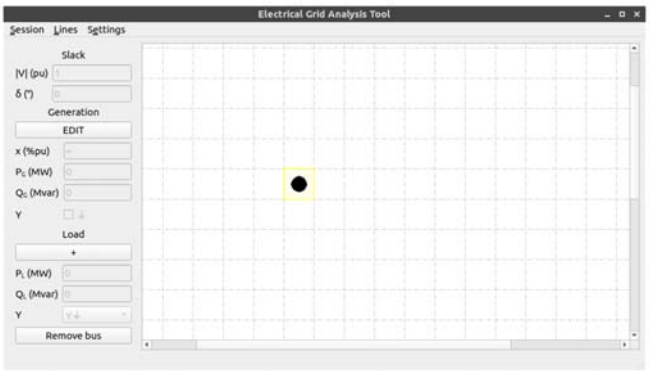

Fig. 1 Highlighted slack bus and its sidebar inspector menu

The *bus inspector*, shown in Fig. 2 (a), allows for insertion of power generators and three-phase loads to the bus. When connecting a generator to a bus, the user supplies constant values of voltage magnitude and generated active power as well as the machine model internal reactance, which is relevant when determining the admittance matrices of sequence networks. It is also possible to toggle neutral point grounding. Loads are specified through their real and reactive powers and their connection scheme. The push-buttons allow to easily insert, edit, save or remove data from those fields. The slack bus is, by design, always connected to a power generation, which can be edited but not removed. The buses themselves can be removed through the *Remove bus* button, with subsequent removal of any branches linked to them.

Finally, *mouse-dragging* draws a branch, which can be either a transmission line or a transformer. A branch will be successfully drawn and inserted in the system after releasing the mouse button if the branch ends at a different bus from where it started and does not cross over itself or any other branches. These criteria guarantee that the drawn network elements can coexist and all single-clicks will select unique elements. The drawn branches are transmission lines by default (color-coded in blue), but at any time they can be easily converted into transformers (color-coded in red).

The *line inspector*, shown in Fig. 2 (b), allows the user to edit the line parameters, either by directly specifying the series impedance and shunt admittance or by means of a previously saved line model. In the former case, the user supplies values in per-unit for the resistance R, the inductive reactance X_L and the capacitive susceptance B_C, confirming with the button *Submit by impedance*. Alternatively, a line model can be selected from a drop-down list in order to derive distributed parameters, in which case the line length must also be specified and the user should choose *Submit by model*. The voltage base is necessary

in order to convert current flow from per-unit to ampères.

Line models may be defined and saved through the menu shown in Fig. 2 (c), in order to be more easily reused. The models are unambiguously associated with a model name and are defined by: the resistivity (ρ), the conductor radius (r), the number of conductors per phase (m), the distance between conductors in each phase (d), the three distances between phases (d_{12}, d_{23}, d_{31}), and the ampacity ($I_{\max}$).

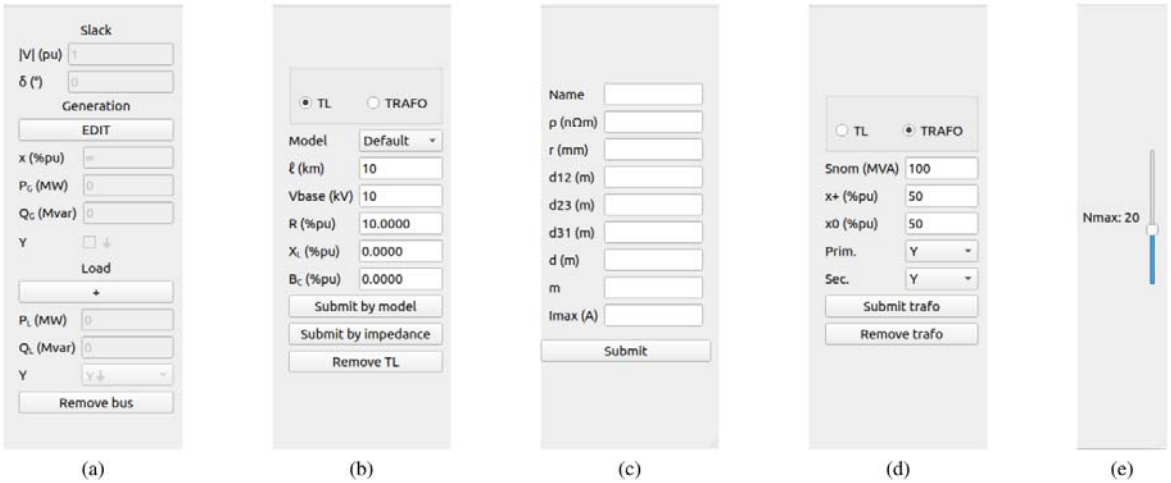

Fig. 2 Sidebar menus: (a) Bus inspector. (b) Line inspector. (c) Line models. (d) Trafo inspector. (e) Configure simulation

At the top of the line inspector, there is also an option to convert the branch into a transformer. Fig. 2 (d) shows the *trafo inspector*, which models a trafo with a given nominal power S_{nom} and equivalent sequence reactances x^+ and x^0 in per-unit with respect to its nominal base. The connection scheme can be chosen from delta, wye, or grounded wye through the drop-down lists for both primary and secondary windings.

The *Configure simulation* sidebar, shown in Fig. 2 (e), can be accessed from the *Settings* tab and allows to set the maximum number of iterations for the power-flow solution through the $N_{\max}$ slider. If the methods do not converge within this upper iteration limit, nothing is updated until further system modifications are made.

A new empty system can be conveniently loaded by clicking *Start new system* in the *Session* tab, so that the graphical and mathematical representations of the current system are completely erased, leaving the canvas ready for a new system input.

Finally, the current session can be saved in a file which keeps track of all drawn elements as well as all defined line models by clicking *Save current session* in the *Session* tab, and recovered anytime with *Open session* in the same tab.

C. Auto-Generated Reports

After experimenting with the system parameters, the user might want to generate a report with the simulation results. Although some of the resulting data are already visible in the sidebar (e.g. bus voltages in per unit, active and reactive power), in order to maintain the interface minimalism Elegant generates a PDF file with all resulting data from simulations. In this document, the user can find tables detailing bus voltages, power flow and losses in each branch, and short-circuit fault currents in each bus. There is also an embedded image representing the system schematic with annotated bus IDs and power-flow (see Section III *C*).

It is important to emphasize that the short-circuit fault currents can only be obtained from the reports, otherwise the main interface would be excessively cluttered with information.

III. Case Study

In order to illustrate the usage of Elegant, a complete example is presented here. The developed system is the implementation of a typical Brazilian regional system with 18 buses, presented schematically in Fig. 3.

A. System Information

The leftmost bus is actually a representation of the whole National Interconnected System (*Sistema Interligado Nacional*, SIN), which, for all practical purposes, works as a slack bus. The rightmost buses represent cities with static loads.

As indicated in Fig. 3, there are three different types of transmission lines being used here. Their topologies are shown on the right, while Table I details the relevant parameters for each conductor. The resistivity values presented are the equivalent values obtained if the conductors were built from a single wire. Each type of transmission line will thus be referred to by combining the code word that identifies its conductor with its topology (for instance, Linnet/A means a transmission line using Linnet conductors and topology A).

TABLE I
Conductor Parameters

Name	Conductor	Strands	ρ (nΩm)	r (mm)	$I_{\max}$ (A)
Linnet	336.4 ACSR	26/7	53.45	9.155	510
Tulip	336.4 AAC	19	46.21	8.450	495
Daisy	266.8 AAC	7	45.21	7.440	420

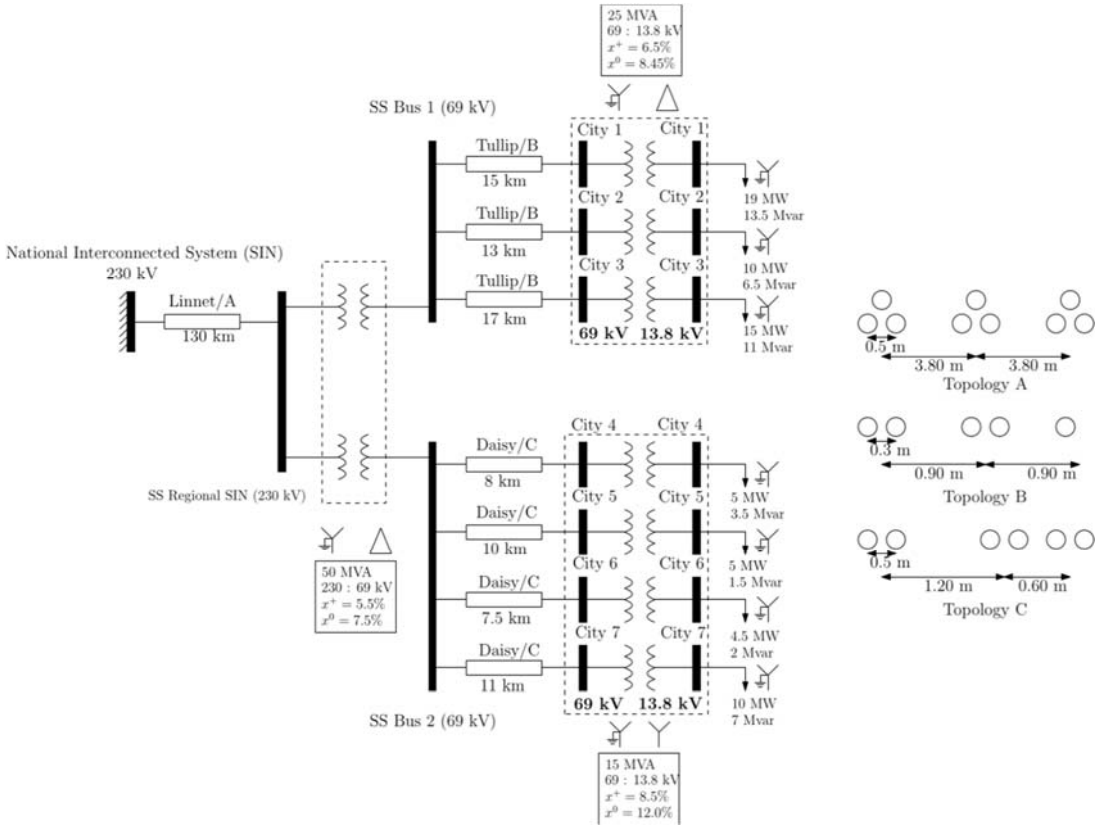

Fig. 3 Representation of a typical Brazilian regional system

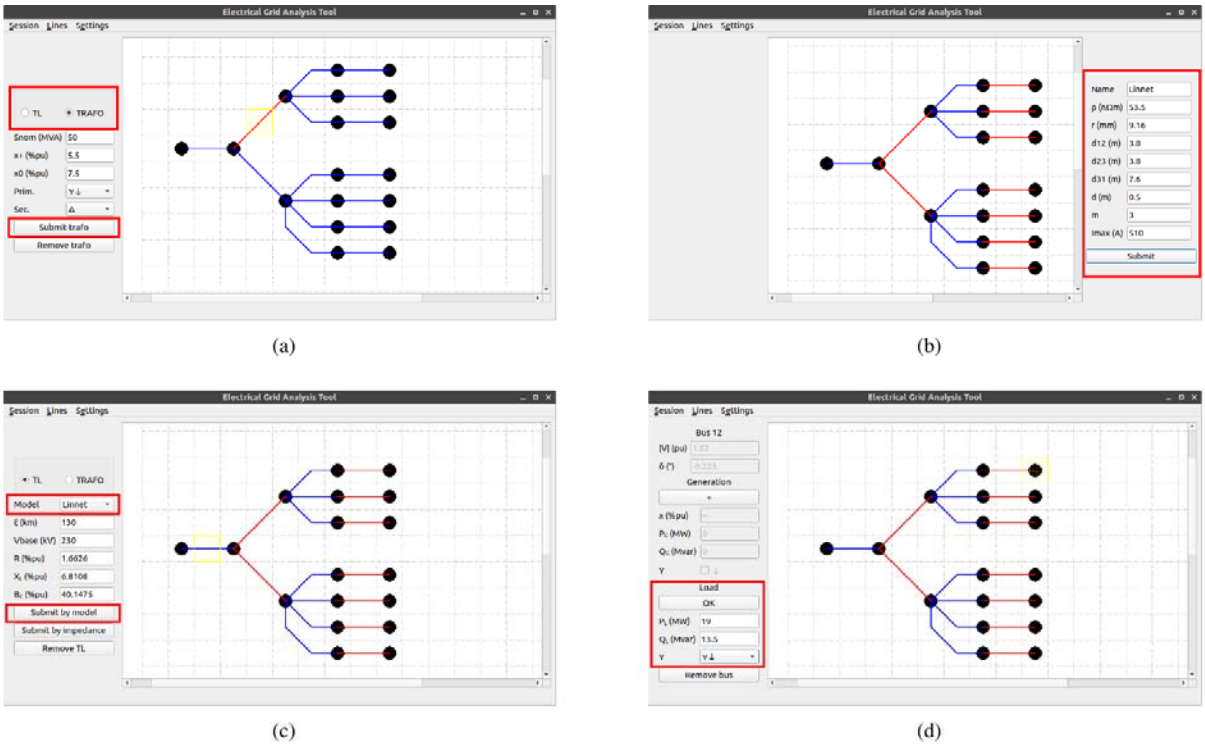

Fig. 4 Step-by-step simulation process: (a) Insertion of buses and lines. (b) Linnet/A line model. (c) Line model selection and updated line parameters. (d) Insertion of loads

B. User Input

The steps involved in the system input are very straightforward. As mentioned in Section II, a single click is used to point out the cursor position, and a double click is used

to put a generic bus into the pointed position. Note that the first inserted bus must be the slack bus and the remaining buses will be numbered according to the order of insertion.

Fig. 4 (a) shows the system after the insertion of the 18 buses as well as the branches connecting them. The blue lines represent transmission lines; however, some of these lines should be converted into transformers, which can be achieved simply by clicking on the transmission line, selecting the toggle button *TRAFO* at the top and, after inserting its data, clicking *Submit trafo*. Fig. 4 (a) shows that the selected branch is now red, representing a trafo. The same procedure will be repeated for all transformers in the system. Subsequently, the transmission lines should be updated with their physical and geometrical parameters presented in Table I and Fig. 3. A new transmission line model can be stored by clicking *Add new line types* in the *Lines* tab. Fig. 4 (b) shows the interaction with the sidebar for the Linnet/A line model; this model will be stored by clicking *Submit*. The same procedure is repeated for Tulip/B and Daisy/C transmission lines. By selecting the pre-stored model in the *Model* drop-down list and filling in its length and voltage base, as shown in Fig. 4 (c), the transmission line parameters can be immediately updated after clicking *Submit by model*.

Finally, we need to add the active and reactive loads to the rightmost buses. This is accomplished, as highlighted in Fig. 4 (d), by clicking the + button on the *Load* section of the corresponding bus inspector, inputting the data for P_L and Q_L in the respective fields, and clicking *OK*.

This sequence of steps could have been performed in any arbitrary order, but we decided to present it this way for clarity.

C. Data Visualization

The quickest way to view bus data is with the bus inspector, where only power-flow solutions for voltages and power are shown. For the full system data, there is the auto-generated report, available with *Generate system report* in the *Session* tab. One only needs to choose the path for the generated PDF file. The report's tables are clear and human-readable, as shown by Fig. 5.

D. Analysis of the Results

The table presented in Fig. 5 summarizes the bus voltages, generated power, and loads, while Fig. 6 shows the calculated short-circuit currents. Table II details the load-flow results obtained for each branch, where the "Load" column represents the percentual current with respect to the ampacity for the transmission lines and the percentual power-flow with respect to the nominal power for the transformers.

At first glance, the results from Table II indicate an overload at the transformer connecting buses 2 and 3. This is to be expected, since there is no power generator in this network besides the slack bus, so that the high demand coming from cities 1, 2, and 3 need to be entirely supplied via this single transformer. Another related observation is the decrease in per-unit voltage magnitude and increase in phase angle lag as the distance from that single generation gets larger, as can be seen in Fig. 5. If the limits tolerated by the regulating agency are $\pm 5\%$ of the nominal voltage [15], then most cities are operating well below this threshold.

TABLE II
LOAD-FLOW RESULTS FOR THE SYSTEM BRANCHES

	Branch	Loss		Flow		
		MW	Mvar	MW	Mvar	Load (%)
Transmission Lines	1 – 2	1.09	-34.20	69.03	54.26	77.82
	3 – 5	0.24	0.17	19.00	15.43	79.69
	3 – 6	0.05	-0.23	10.00	6.97	39.16
	3 – 7	0.17	-0.06	15.00	12.20	62.79
	4 – 8	0.01	-0.23	5.00	3.75	22.90
	4 – 9	0.01	-0.29	5.00	1.68	19.32
	4 – 10	0.01	-0.22	4.50	2.16	18.27
	4 – 11	0.06	-0.24	10.00	8.07	47.33
Transformers	2 – 3	0.00	4.12	44.45	34.47	112.50
	2 – 4	0.00	1.01	24.58	14.67	57.25
	5 – 12	0.00	1.93	19.00	13.50	93.23
	6 – 13	0.00	0.47	10.00	6.50	47.71
	7 – 14	0.00	1.20	15.00	11.00	74.40
	8 – 15	0.00	0.25	5.00	3.50	40.69
	9 – 16	0.00	0.18	5.00	1.50	34.80
	10 – 17	0.00	0.16	4.50	2.00	32.83
	11 – 18	0.00	1.07	10.00	7.00	81.38

By experimenting with the data, a user might find that, by adding distributed generations to the buses with the largest loads (12 and 14, for instance), both mentioned problems can be solved or at least mitigated. In addition to that, the prospective short circuit fault currents increase on average when using this technique. One might argue that increased currents offer more risk of safety implications in maintenance procedures, equipment damage, or power interruptions. This is just a simple example of the type of insight that can be provided to students once they start simulating and experimenting with real design problems.

IV. Group Experiment

To evaluate the overall experience and perception of potential users, a very simple experiment was devised. Five students were selected: one undergraduate and four graduate students. They were tasked with solving the 18-bus example presented in this paper using both a commercial software from Brazilian Electrical Engineering Research Centre (CEPEL) and the Elegant GUI. There was only one of them with prior experience with the commercial software, and none of them with Elegant. They were instructed on the basics of each program usage and measured the time taken to complete the task using each software.

The average time taken to complete the task was 37m03s $\pm$ 16m17s with Elegant and 1h53m40s $\pm$ 39m46s with the software from CEPEL. This shows a significant improvement of near three-fold reduction on the time involved in the simulations.

V. Conclusion

This paper presented Elegant, an educational tool to simulate electrical grids with a gentle learning curve and providing real-

time feedback. While Elegant is mainly intended for first-time learners of power system analysis, it is also suited for presentation and software development purposes.

Report auto-generated by Elegant at 11/05/2020 08:13:38

1 Buses

1.1 Load-Flow Solution

Bus	$\|V\|$ (pu)	δ (deg)	P_G (MW)	Q_G (Mvar)	P_L (MW)	Q_L (Mvar)	Z_L (pu)
1	1.0000	0.00	70.12	20.07	0.00	0.00	∞
2	0.9619	−2.45	0.00	0.00	0.00	0.00	∞
3	0.9192	−5.62	0.00	0.00	0.00	0.00	∞
4	0.9444	−4.15	0.00	0.00	0.00	0.00	∞
5	0.8992	−6.29	0.00	0.00	0.00	0.00	∞
6	0.9111	−5.94	0.00	0.00	0.00	0.00	∞
7	0.9014	−6.22	0.00	0.00	0.00	0.00	∞
8	0.9417	−4.23	0.00	0.00	0.00	0.00	∞
9	0.9421	−4.28	0.00	0.00	0.00	0.00	∞
10	0.9426	−4.23	0.00	0.00	0.00	0.00	∞
11	0.9367	−4.34	0.00	0.00	0.00	0.00	∞
12	0.8564	−9.97	0.00	0.00	19.00	13.50	3.15∠35.39
13	0.8917	−7.78	0.00	0.00	10.00	6.50	6.67∠33.02
14	0.8673	−9.08	0.00	0.00	15.00	11.00	4.04∠36.25
15	0.9197	−6.10	0.00	0.00	5.00	3.50	13.86∠34.99
16	0.9325	−6.13	0.00	0.00	5.00	1.50	16.66∠16.70
17	0.9300	−5.90	0.00	0.00	4.50	2.00	17.56∠23.96
18	0.8900	−8.24	0.00	0.00	10.00	7.00	6.49∠34.99

Fig. 5 Partial screenshot of the auto-generated report

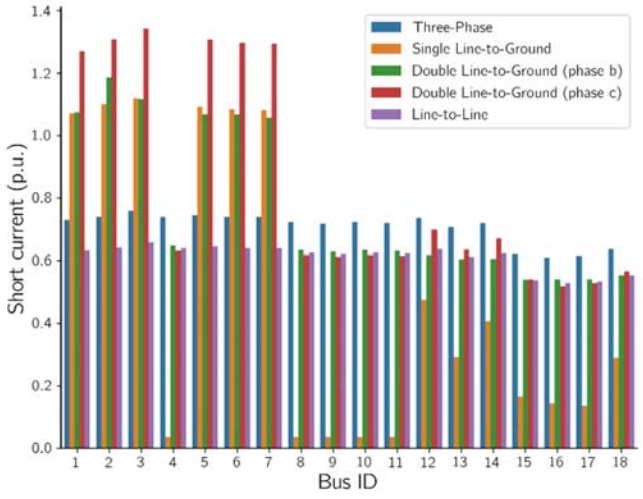

Fig. 6 Short current values for each fault type in each bus

We evidenced that Elegant significantly reduces the time and labor involved in basic power systems simulations. It can thus be a useful tool to help students learn the basic concepts in power system analysis and help them explore different design strategies.

Future software enhancements will include a modified grid, enabling a larger number of connections per bus, as well as the possibility to add shunt capacitive compensations to buses.

References

[1] "IEEE Recommended Practice for Industrial and Commercial Power System Analysis (Brown Book)," *IEEE Standard 399-1997*, Aug 1998.
[2] R. S. Lysakowski and H. J. Walberg, "Classroom reinforcement and learning: A quantitative synthesis," *The Journal of Educational Research*, vol. 75, no. 2, pp. 69-77, 1981.
[3] B. Jeyasurya, "An educational interactive program for direct analysis of power system transient stability," *IEEE Transactions on Education*, vol. 38, no. 1, pp. 90-94, 1995.
[4] M. Songur and B. Ercan, "POWERHU-a PC-based electric power system analysis software package for electric power system courses," *IEEE Transactions on Education,* vol. 40, no. 4, 1997.
[5] J. Yang and M. D. Anderson, "PowerGraf: an educational software package for power systems analysis and design," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1205-1210, 1998.
[6] F. Milano, "An open source power system analysis toolbox," *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1199-1206, 2005.
[7] 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 Transactions on Power Systems*, vol. 33, no. 6, pp. 6510-6521, 2018.
[8] T. Brown, J. Hörsch, and D. Schlachtberger, "PyPSA: Python for Power System Analysis," *Jounal of Open Research Software*, vol. 6, no. 4, 2018 (Online). Available: https://doi.org/10.5334/jors.188
[9] S. Vera, "GridCal – research oriented power systems software," 2018.
[10] R. W. Lincoln, "Learning to trade power," Ph.D. dissertation, University of Strathclyde, 2011.
[11] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12-19, 2010.
[12] L. Bam and W. Jewell, "Power system analysis software tools," in *IEEE Power Engineering Society General Meeting, 2005*. IEEE, 2005, pp. 139-144.
[13] S. S. Chauhan and P. Khamparia, "A Survey of Software Packages in Power System Analysis," *International Journal of Electrical Engineering Education*, vol. 51, no. 2, pp. 134-145, 2014.
[14] G. P. de Azevedo, C. S. de Souza, and B. Feijó, "Enhancing the human-computer interface of power system applications," *IEEE Transactions on Power Systems*, vol. 11, no. 2, pp. 646-653, 1996.
[15] ANEEL, "Resolution No 505," Nov 2001.