Received: 8 November 2019 | Revised: 17 July 2020 | Accepted: 24 July 2020

DOI: 10.1002/eng2.12269

RESEARCH ARTICLE

Graphical user interface editor for Octave applications

Enrique S. Burgos[1] | **Eduardo J. Adam**[2]

[1]Facultad Regional Paraná, Universidad Tecnológica Nacional, Av. Almafuerte 1033 Paraná, Entre Ríos, E3102SLK, Argentina

[2]Facultad de Ingeniería Química, Universidad Nacional del Litoral, Santiago del Estero 2654 Santa Fe, Santa Fe, S3000AOJ, Argentina

Correspondence
Enrique S. Burgos, Universidad Tecnológica Nacional - Facultad Regional Paraná, Almafuerte 1033, Paraná, Entre Ríos CP 3100, Argentina.
Email: sergioburgos@frp.utn.edu.ar

Abstract

The use of calculation software for pedagogical purposes in degree courses is a central tool for addressing disciplinary knowledge in engineering careers. Particularly, open source and free license tools generally have lower associated costs than their propriety equivalents while offering a similarly rich set of functionalities. This document presents a software tool called guiEditor designed to create script applications based on graphical interfaces for GNU Octave. guiEditor is a visual editor, which incorporates different functionalities aimed at allowing the development of graphical applications based on basic programming knowledge. Within the development environment it incorporates, in addition to the typical controls of the graphical interfaces, new controls that extend the script language used. In addition, as an example of the development environment potential, two applications created with guiEditor are presented. The assessment made by the students about a third application, called ltitool, which has been used in degree courses to address Process Control topics, is also presented.

KEYWORDS

calculation software, control system, GNU Octave, GUI

1 | INTRODUCTION

In classrooms, when topics involving complex calculations are addressed, it is sometimes necessary to use software that allows is to verify and interpret different aspects of a system against changes in its parameters.

A reasonable solution is the implementation of software applications dedicated to modeling a particular system using classical programming languages (Pascal, C, C ++, Java, Python). In this case, specific knowledge associated with the development of applications is necessary, occasionally moving the focus of the degree courses.

In recent years, the use of calculation tools incorporating high-level interpreted programming languages has been found as a solution to this problem. These usually have libraries that implement typical algorithms of each discipline and offer flexibility in the development of calculation applications. In their simplest working mode, these calculation engines allow direct interaction with the user, generating responses to commands entered by keyboard or grouping sentences into text files, which when evaluated they produce a response, either on the screen or by generating new files. Within this category the most popular tools that can be found are Matlab, SciLab, Spyder IDE for Python language, and GNU Octave.

This work has taken into account backgrounds of educational applications using free software. Particularly the works of Milano[1] and Vanfretti and Milano,[2] among others available in the literature, present a toolbox for teaching, compatible with GNU Octave and that uses a graphical interface are a good example. However, there are no free software tools that allow the visual development of this type of interface, applied to GNU Octave, as is the case of guiEditor. Following in the same direction as the mentioned works, an implementation of a toolbox was carried out and it was evaluated from the user's perspective.

Clearly, in order to further facilitate user-computer interaction, graphical user interfaces (GUI) play a key role in that interaction. The main advantage of GUI is that they allow to users, through dialog windows and functions, to interact in a very simple way with a software. This represents an important advantage in teaching-learning processes for engineering careers, where both of them can focus on the predetermined objectives for a certain concept. As a disadvantage, a software that includes GUI allows that the user can only interact with situations that are preset by the programmer.

The evolution observed in electronics and information technology have generated new possibilities that can be exploited due to an increment of calculation speed. For example, the use of simulators developed with a specific calculation software can provide very good results in an educational context as it is shown below and other works[3,4] among others. For this reason, the incorporation of GUI in a software allows the professor to approach the presentation of complex concepts in a simple way, thus facilitating improved understanding of a student.[5-9]

Although this approach is attractive, an associated problem is the cost of the licensed software and its limitations in terms of knowledge of the implementations included. In addition, it should be noted that there are other solutions for the scientific-educational market (such as Quansar[a], among many others) that include nonfree licenses that in some cases are expensive and that offer off-the-shelf, completely, self-contained, and require no modification by the educator, but at the same time in some cases they do not cover everything that educator needs because this is a closed package. An alternative is free software, such as GNU Octave,[10] which is very suitable for calculations in engineering problems. This software provides functionalities that go beyond a specific theme, since it incorporates a toolbox set, such as signal processing, linear control system, linear algebra, among others, making this software applicable to different knowledge areas.

The implementation of GNU Octave in education is a growing activity, finding in the literature a great variety of works in different fields of engineering (is recommended to see the reference in the footnotes[b]).

It stands out among many works to the contributions of Eaton[11] regarding the GNU Octave in the scientific field. Rawaling et al[12] who developed predictive control applications for both research and postgraduate teaching and, together a Risbeck, develop the mpc toolbox[13] for Matlab and GNU Octave. Milano,[1] Vanfretti and Milano[2] who developed a toolbox for teaching power lines, Richter[14] who developed a toolbox for Modeling and Simulation in Materials Science and Engineering. Čisar et al,[15] in same way, develop a toolbox for telecommunications teaching.

There are more benefits than the economic savings when free and open software is chosen.[16,17] It is possible to know the way of implementations, allowing the students to have a software, that they can use with lower hardware resources than the commercial counterparts and from a scientific point of view, it gives them the freedom of recreating environments to verify their results.

These ideas have motivated researchers to develop GUI editors, particularly in this work, an open source environment that allows interaction with different calculation functionalities incorporated in GNU Octave is presented.[18] The most significant feature that incorporates, is the possibility of developing projects that generate script applications that are totally independent of the development environment, using invocations to native GNU Octave functions. Consequently, this editor (here designated as guiEditor) allows to build interactive graphical interfaces and to use prototyping platforms through a visual mode.

Particularly, this tool was applied to develop toolboxes for linear control systems for undergraduate and graduate studies, as it is shown in this article. However, as mentioned earlier, guiEditor allows its application in different areas of knowledge.

This article is organized as follows. In Section 2 a brief discussion about advantages and disadvantages of calculation software for engineering careers currently available. Then, in Section 3, a brief description of guiEditor here development is included. In Section 4, in order to show the potential of the guiEditor two demonstrative examples are included. Then, experiences and results are discussed in Section 5. Finally, concluding remarks are made in the last section.

2 | CALCULATION SOFTWARE FOR ENGINEERING CAREERS

In the teaching of engineering careers, for several years, different software tools has been incorporated. These are often used to address complex issues by carrying out mechanical calculation procedures. In this aspect, taking into account the

[a]https://www.quanser.com/digital/quanser-interactive-labs/
[b]http://wiki.octave.org/Publications_using_Octave

large number of available tools, Matlab, GNU Octave, and SciLab have comparable characteristics[19] and are widely used in an academic setting.

Matlab is a commercial tool, developed by the company MathWorks, that incorporates not only an interpreter of language calculation, but also in numerous libraries (designed as toolbox) and tools applicable to different knowledge areas. It has tools that from GUI allow interacting with models and systems without the need for deep programming knowledge. In addition to the already developed graphical tools that it incorporates, it has the possibility of developing new graphical applications through the guide tool, allowing to build GUI and invoke different functions from them and implement algorithms using the MATLAB programming language.

As a counterpart to the pedagogical benefits of this tool, both for teachers and students, some aspects can be considered a disadvantage. Namely,

- Being a commercial software, its use requires the acquisition of licenses that in many cases can be expensive. This could be a limitation since student access to licenses significantly increases costs.
- It is a tool developed with *closed* software models. From the didactic point of view, it becomes convenient to have tools that allow focusing on the topics of interest, at a high level, obviating the specific problems of implementation. However, as the learning process progresses, it is important to check and understand the implementation of algorithms and linking them with the theoretical aspects.
- Significant hardware resources is necessary, for example, for the version 2019b according to the page[c]a processor with 4 cores is recommended, with 64-bit architecture, 8 GB of RAM, solid state disk with 3.1 GB of space for a minimum installation and, between 5 and 8 GB for a typical installation. Although the above-mentioned features correspond to a typical desktop PC, when performing intensive calculations, the hardware features may need to be expanded.

These weaknesses are resolved with the other two mentioned tools (SciLab and GNU Octave). Both are open and free code environments, so licensing costs are not applicable. With regard to the hardware requirements to use these environments, experiences show that they can be run with less resource equipment. Consequently, the hardware characteristics are not a limitation for its use.

A significant feature is that the three tools keep certain points in common with respect to their operation and characteristics, and it is important to remark that the smallest differences are found between GNU Octave and Matlab.[19]

3 | GUI-EDITOR FOR GNU OCTAVE

The GNU Octave project started around 1988 [d], and since then it has evolved, becoming an alternative calculation tool for scientists, professors, students and industry professionals.

From version 4 onward, a graphical interface that facilitates interaction with the user is incorporated. This was developed using the Qt graphic toolkit. In addition, as of version 4, has been included the possibility of use the same graphic toolkit as an alternative for the representation of user controls in script applications, similarly to Matlab, allowing script applications to incorporate GUI. Although the compatibility between the graphic controls available in Matlab is not complete, in the new versions of GNU Octave there has been a significant advance in the increase in compatibility.

Taking into account the fact that GNU Octave does not incorporate a visual tool for the construction of user graphical interfaces, a project called guiEditor was created to develop a tool for this purpose.

3.1 | guiEditor

The project was initiated by the authors of this document as a joint work initiative between the Laboratory of Instrumentation and Process Control of the Chemical Engineering Faculty at Universidad Nacional del Litoral and the Laboratory of Computer Science of the Electronic Engineer of the Facultad Regional Paraná of the Universidad Tecnológica Nacional.

[c]https://la.mathworks.com/support/requirements/matlab-system-requirements.html
[d]https://www.gnu.org/software/octave/about.html

The initial objective of the project was to develop a tool that would allow users with basic knowledge of GNU Octave to develop applications that incorporate GUI.

The first works consisted of the implementation of a library developed in C++ and with a counterpart in scripts for GNU Octave. Both allowed communication between an application developed using the Qt development environment that constituted the graphic interface and GNU Octave. In this model, the application captured data entered by the users, sent the operations to be carried out to Octave and subsequently presented the results of them again in the application. Using this methodology, the first version of a toolbox for the Process Control teaching[20,21] was developed. With this methodology, it was observed that the consumption of hardware resources when running graphical applications developed with Qt libraries was significantly lower than equivalent interfaces developed entirely in Matlab. However, using this methodology, some drawbacks were noted. Because of development was done in C ++ using Qt Creator, the user should have knowledge of all of these technologies. The main disadvantage of this methodology is that the development must be compiled, which makes it difficult to distribute the software in educational environments, since a binary file must be made for each operative system where the application is to be ported.

When GNU Octave 4 incorporated the possibility of representing user graphic controls using the Qt toolkit a new alternative was possible. In this case, functions such as `uicontrol` and `uipanel` allow to make visible controls such as buttons or boxes of edition on dialog windows, thus constituting user interfaces. In this case, the result is much more advantageous than previously achieved since the interfaces so constituted are script files, independent of the operating system where they are executed.

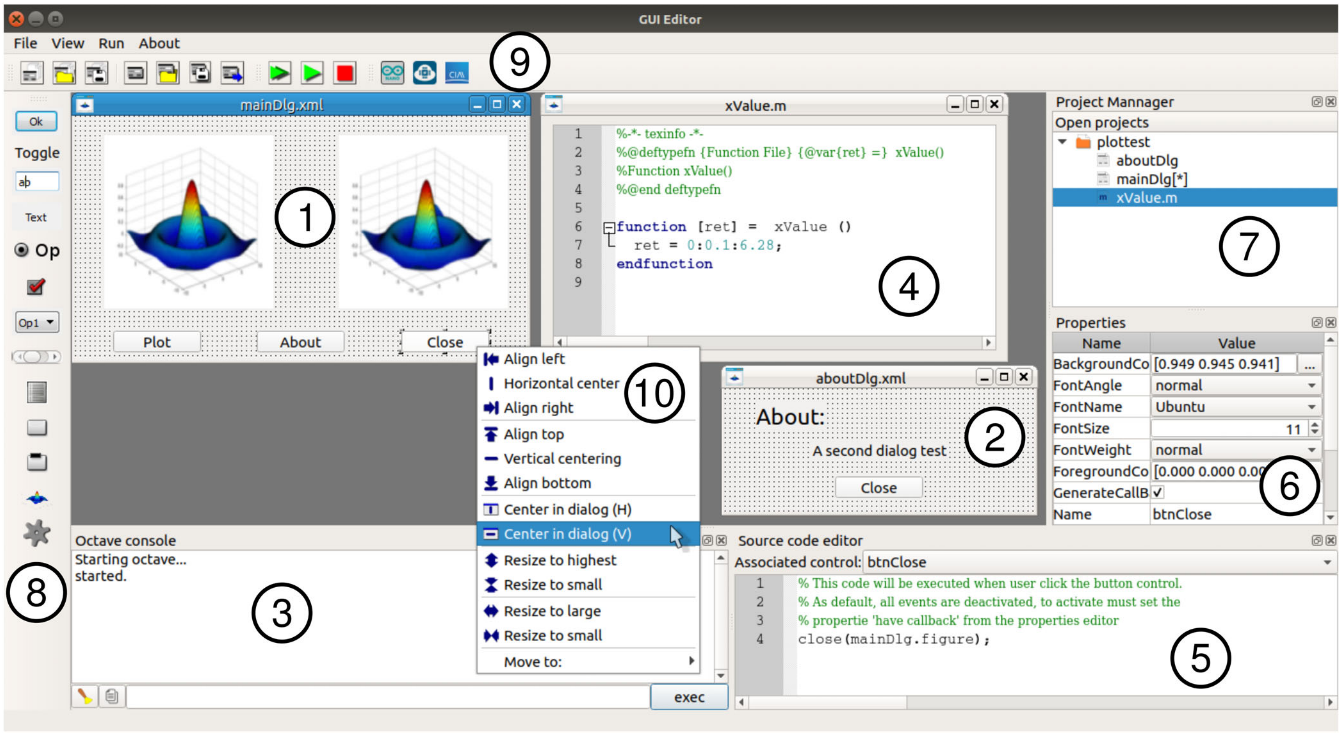
The development environment presented here is the result of the effort of several years; having started as a simple visual editor of graphical interfaces, to allow build dialog windows through a set of scrip files.

In order to build the GUI, guiEditor began to be developed as a visual application editor, which allows the graphical interfaces to be represented as they are constructed and then generate the equivalent script source code that allows to obtain their representation in GNU Octave with applications in education. The development was carried out once more using the Qt framework, the decision is based on two facts, on the one hand the experience in the original development demonstrated low resource consumption hardware and on the other hand, when using the same graphic toolkit as GNU Octave, the representation of the graphical interfaces is similar.

In addition to the development of the application, and its associated model, it was necessary to choose a form (model) of implementation for graphical interfaces such as the GNU Octave script. In this sense, the problem is that the objective was to achieve a transparent environment for the user regarding the use of the controls and their integration with previously developed scripts, but each control is generated through the invocation of a function (`uicontrol`). This implies that once a control has been created and a function associated in response to an event, there is no intuitive representation of controls in the dialog window to permit the access to its properties. To solve this problem, and to have a general method to interact from the script files with the controls that constitute the graphical interface, during the process of generating the scripts the definition of a structure is created. Then, each function associated with an event receives this structure as an argument, with the name assigned to the window that acts as a container. The final result allows working from GNU Octave scripts in a similar way to object-oriented programming languages, where variants of aggregation or composition are used to represent controls in the graphical interface.

In Figure 1 you can see a screenshot showing different elements that constitute the GUI editor:

- 1 and 2: Dialog windows in edition mode.
- 3: GNU Octave terminal to know results of the output variables produced as well as to send commands to the interpreter.
- 4: Function editor.
- 5: Source code editor associated with the default actions of each graphic control.
- 6: Property editor, by selecting each graphic control or a dialog window allowing the user to set the value of their different attributes.
- 7: Project manager. From this window it is possible to add and remove components to the project as well as set its configuration.
- 8 and 9: Visual controls that can be integrated into an application.
- 10: Contextual menu with different ways of resize and align controls.



```
%-*- texinfo -*-
%@deftypefn {Function File} {@var{ret} =} xValue()
%Function xValue()
%@end deftypefn

function [ret] =  xValue ()
  ret = 0:0.1:6.28;
endfunction
```

Name	Value
BackgroundCo	[0.949 0.945 0.941]
FontAngle	normal
FontName	Ubuntu
FontSize	11
FontWeight	normal
ForegroundCo	[0.000 0.000 0.00
GenerateCallB	✓
Name	btnClose

```
Starting octave...
started.
```

Associated control: btnClose

```
% This code will be executed when user click the button control.
% As default, all events are deactivated, to activate must set the
% propertie 'have callback' from the properties editor
close(mainDlg.figure);
```

FIGURE 1 guiEditor running in Ubuntu Linux, editing a graphical script application for GNU Octave

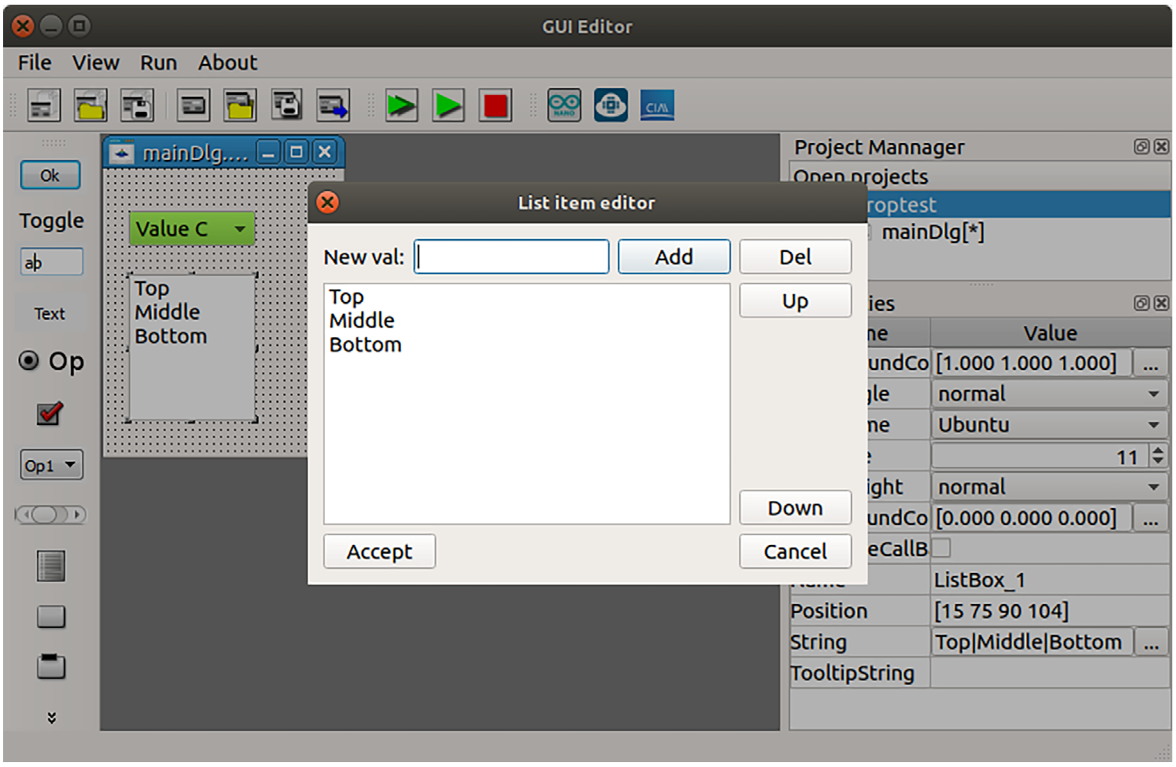
Although it was born as a tool for the construction of graphical interfaces visually, it currently incorporates several functionalities aimed at simplifying the creation of applications by users with little programming knowledge. Among them, it is worth mentioning:

- *Configuration of the control characteristics through a property editor*. Each control, although represented graphically in order to be able to modify its size and position, is represented in the script application as an invocation to the function `uicontrol` or `uipanel`. The values of the arguments of these functions establish their form of representation and behavior. In order to simplify the development process, some of the possible values for them can be selected from a property editor. A property is a particular value of an attribute, in some cases the possible values are restricted to a particular domain, in others, they have special format characteristics. In order to simplify its editing and development, the property editor has particular editors for these cases. An example can be seen in Figure 2, where you can see the use of an item editor in order to configure the options that a list control will present. This option is also applicable to combo box controls.
- *Development of applications as projects*. Each project can contain windows that act as graphical interfaces, files with functions and/or images. Once the development of the application is finished, it is possible to export the project as script files or as GNU Octave packages. If the script application uses multiple windows[e], the window configured as main will become visible when loading and invoking the package from GNU Octave, the others can be made visible by invoking the name of the dialog window. Both, the windows incorporated in an application, as well as the functions used, will be incorporated into the search paths automatically when loading the package or running the application from the development environment.

  Figure 3 shows the structure of a typical project through the folders that constitute it and they are created automatically. Here, the *pkg* folder will contain the file resulting from the construction of a package and *export* the script files associated with the project executable from GNU Octave.

  The folders named as *gui*, *img*, and *fcn* will contain the representations of the dialog windows as files *xml*, the images that are part of the project and M files that implement functions. These will be processed every time a project is opened in the editor, so to reuse existing dialog windows or functions, just copy the files to the corresponding folder. The *libs* folder contains additional script files that can be added in order to be part of the project, and its editing is not possible from the development environment.

[e]Burgos ES. Ej03: Using multiple dialog windows in applications. https://youtu.be/IiTSBQRh5pQ. Accessed February 2020.



	Value
undCo	[1.000 1.000 1.000]
gle	normal
ne	Ubuntu
	11
ight	normal
undCo	[0.000 0.000 0.000]
eCallB	
	ListBox_1
Position	[15 75 90 104]
String	Top|Middle|Bottom
TooltipString	

FIGURE 2 Visual edition of the elements associated to a list control through properties editor

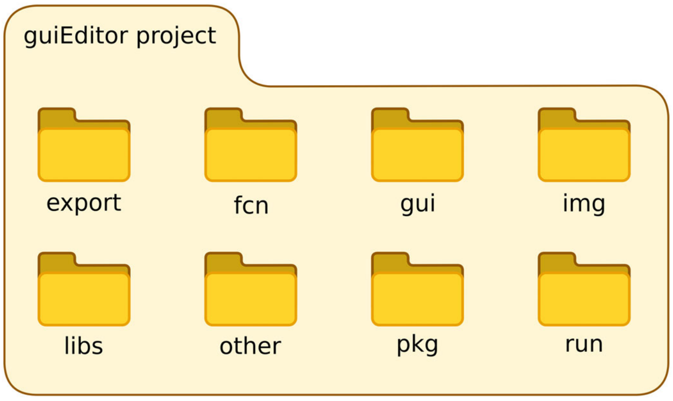


FIGURE 3 Directory structure used to represent a project

Text files containing definitions necessary for the construction of packages are created automatically and added to the *other* folder. It can be modified from the environment in order to adjust specific parameters of the project.

Finally, the *run* folder contains a replica of the project structure as script files, so that it can be executed in GNU Octave from the editor.

- *Integration with GNU Octave*. Although it is not essential to have the interpreter in order to develop GUI, if available, the editor can invoke it in order to test the application as it develops. This also simplifies the debugging of the application since both messages generated using the *disp* function as well as warnings or error messages will be visible from the console built into the editor.
- *Integrated source code editor* In order to allow the development of the different aspects of a project, the possibility of editing the script source code of the applications is incorporated. Either associated to functions linked automatically to events such as additional scripts that are part of the application.

Finally, we consider it is important to remark that, the guiEditor interface has been developing considering the possibility of including different languages. In the current version it is possible to select between English and Spanish, but it is expected in future versions to incorporate different languages since its design allows it.

4 | APPLICATION EXAMPLES

The editor presented in the previous section has been the result of the last 4 years of work, along which different applications used in teaching Process Control in undergraduate and postgraduate courses have been built.

By way of examples and in order to demonstrate the results obtained, two applications developed with guiEditor are presented below. The first one seeks to show a use case where a graphic interface for a function included in GNU Octave is implemented. Here, the resulting script application is introductory to GNU Octave and mode of use. The second example presents an elaborated toolbox, where different tools are incorporated to identify the parameters of process transfer functions based on data monitored in the plant.

4.1 | An interface for `polyfit` function

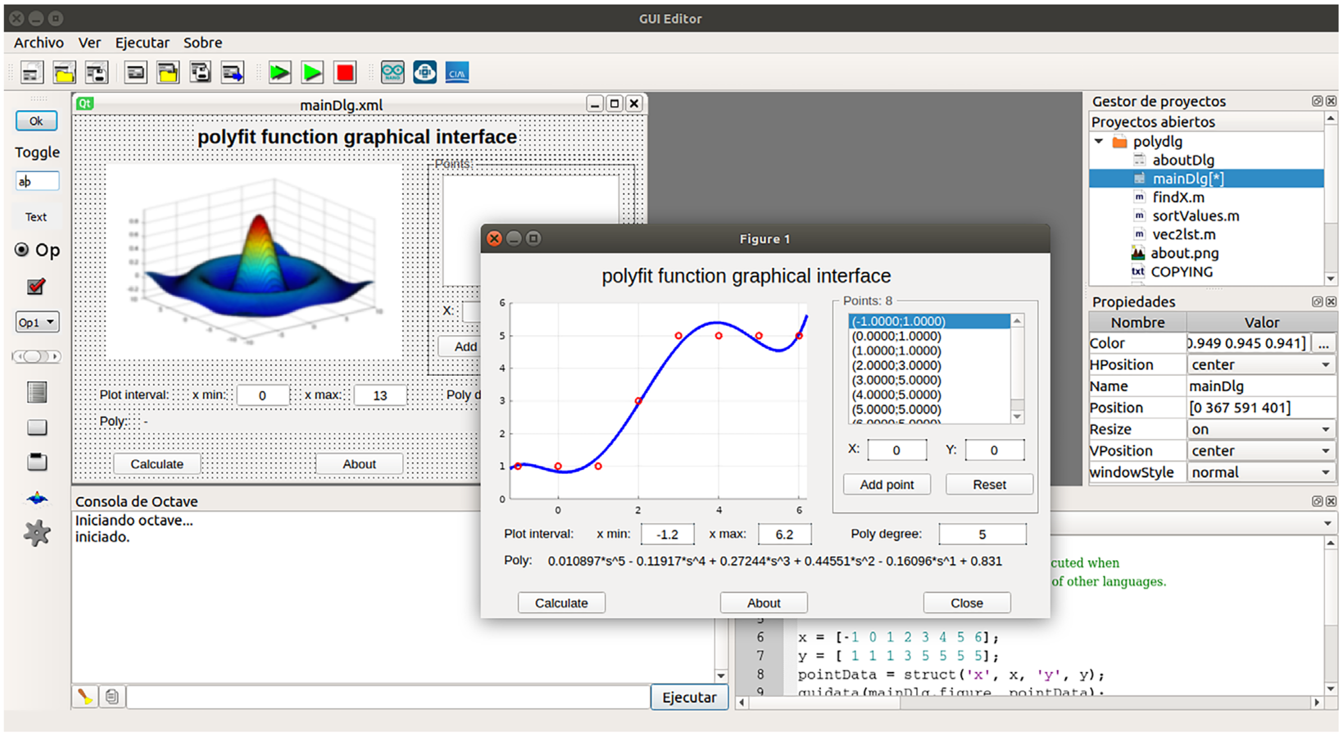
The `polyfit` function, included in GNU Octave core, allows a set of points to be approximated by a polynomial. For this function, a graphical interface is built that allows the user a manual entry of the interest values along with the polynomial degree, and a graph of the resulting function with the points entered is also incorporated.

Figure 4 shows the design process of this application, which it is executed from the development environment. Notice that, a part of the main window is used for representation through the use of an axis control. This allows to establish a section where the graphic representation will be made.

Notice also that,

1. The project includes both a dialogue window and functions developed with the objective of maintaining the point sequence in order, by converting a data vector into a string of characters with a suitable format to be represented in the list control.
2. Furthermore, it is possible to see the additional files included in the project manager that they are associated with the generating process of packages.

Figure 5 shows the same application installed as a package and running with GNU Octave 5.1.0.



mainDlg.xml: Plot interval: x min: 0 x max: 13 Poly: -

Figure 1: Plot interval: x min: -1.2 x max: 6.2 Poly degree: 5 X: 0 Y: 0

Poly: 0.010897*s^5 - 0.11917*s^4 + 0.27244*s^3 + 0.44551*s^2 - 0.16096*s^1 + 0.831

```
(-1.0000;1.0000)
(0.0000;1.0000)
(1.0000;1.0000)
(2.0000;3.0000)
(3.0000;5.0000)
(4.0000;5.0000)
(5.0000;5.0000)
```

Proyectos abiertos:

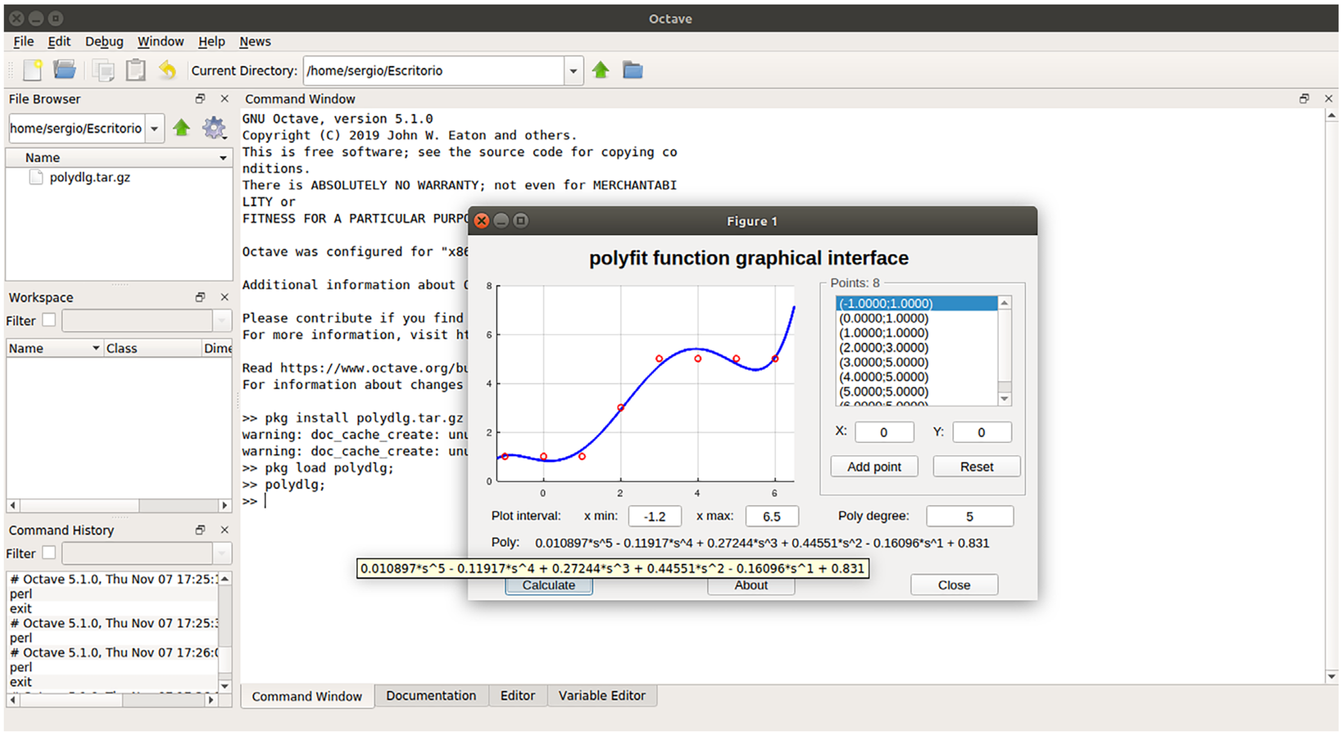
- polydlg
  - aboutDlg
  - mainDlg[*]
  - findX.m
  - sortValues.m
  - vec2lst.m
  - about.png
  - COPYING

Nombre	Valor
Color	0.949 0.945 0.941]
HPosition	center
Name	mainDlg
Position	[0 367 591 401]
Resize	on
VPosition	center
windowStyle	normal

```
Iniciando octave...
iniciado.
```

```
x = [-1 0 1 2 3 4 5 6];
y = [ 1 1 1 3 5 5 5 5];
pointData = struct('x', x, 'y', y);
```

FIGURE 4 Graphical interface built for the `polyfit` function



```
GNU Octave, version 5.1.0
Copyright (C) 2019 John W. Eaton and others.
This is free software; see the source code for copying co
nditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABI
LITY or
FITNESS FOR A PARTICULAR PURPO

Octave was configured for "x86

Additional information about O

Please contribute if you find
For more information, visit ht

Read https://www.octave.org/bu
For information about changes

>> pkg install polydlg.tar.gz
warning: doc_cache_create: unu
warning: doc_cache_create: unu
>> pkg load polydlg;
>> polydlg;
>> |
```

```
# Octave 5.1.0, Thu Nov 07 17:25:1
perl
exit
# Octave 5.1.0, Thu Nov 07 17:25:3
perl
# Octave 5.1.0, Thu Nov 07 17:26:0
perl
exit
```

polyfit function graphical interface

Points: 8

```
(-1.0000;1.0000)
(0.0000;1.0000)
(1.0000;1.0000)
(2.0000;3.0000)
(3.0000;5.0000)
(4.0000;5.0000)
(5.0000;5.0000)
```

Plot interval: x min: -1.2 x max: 6.5 Poly degree: 5

Poly: 0.010897*s^5 - 0.11917*s^4 + 0.27244*s^3 + 0.44551*s^2 - 0.16096*s^1 + 0.831

0.010897*s^5 - 0.11917*s^4 + 0.27244*s^3 + 0.44551*s^2 - 0.16096*s^1 + 0.831

FIGURE 5 Installation of the obtained GUI as a package and its subsequent loading and execution. GUI, graphical user interfaces

As an example, in listings 1 and 2 the source code linked to the "Calculate" and "Add point" buttons are presented (see Appendix A). In both cases, the integration between graphic interface and application code can be observed. It is important to remark that in listings 1 and 2 the entity `mainDlg` identifies the main window that through itself by invoking the get method it is possible to access the different attributes of the controls used. In listing A.1, the `findX` and `sortValues` functions are invoked, which have been defined and implemented for this particular application. The invocation to them is possible because the application launcher adds the routes associated to them so that their management is transparent.

4.2 | A simple linear system identification toolbox

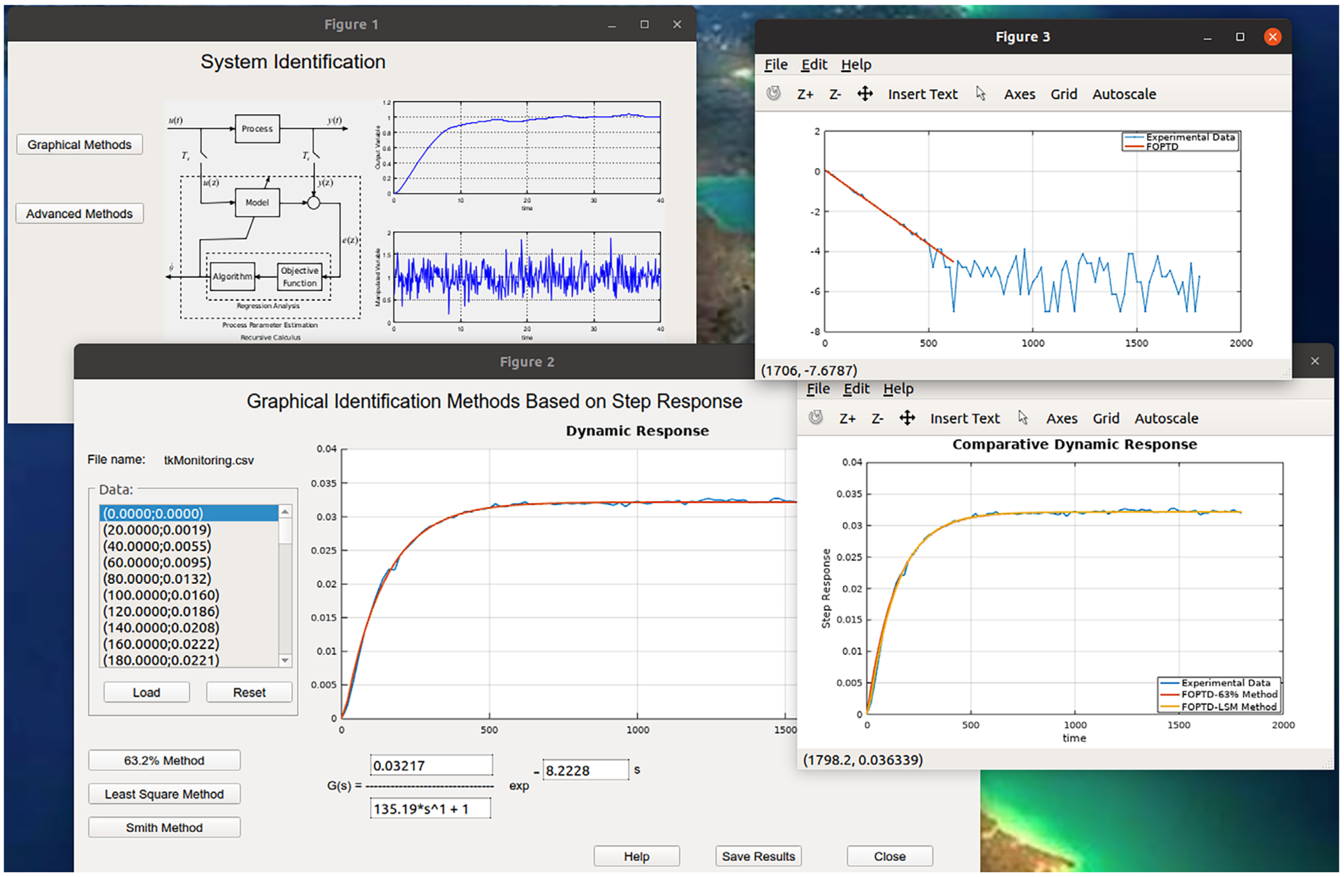
This section presents an application of an identification procedure, which implements simple calculations and by using guiEditor an attractive graphical presentation is achieved.

Due to some models used in Process Control topics has a nonlinear behavior and in some cases, such nonlinearities have considerable magnitude; for this reason, the approximation of these systems with linear models around an operation point and within an operation window is commonly used.[22-24] Thus, there are several methods for identifying parameters so that a reduced-order linear system fit well at a nonlinear one.

As an example, it is presented a GUI developed to read monitored data in plant and then to carry out a passive data analysis of them, in order to identify a parameter set of a linear plant such that it fit the nonlinear dynamic of the plant. Experimental data was collected at the Process Control Laboratory and they correspond to a liquid storage tank system.[22]

Figure 6 shows the windows developed with guiEditor to identify parameter set of a first-order plus time delay (FOPTD) system that they correspond to physical process mentioned above. For this identification, the chosen method was the minimum square one applied to a semilogarithmic curve according to classical textbook.[22] This curve that allows the calculation of parameters for a continuous transfer function, and the time step response of the experimental system and for the FOPTD approximation proposed can also be seen in such figure.

Furthermore, a FOPTD dynamic step response with parameters calculated by 63 % method ([Reference 24, chp. 7, p119], 22, among others) can be incorporated for comparative purposes in an additional window. This option could be chosen, if the user wanted it.



(1706, -7.6787)

(0.0000;0.0000)
(20.0000;0.0019)
(40.0000;0.0055)
(60.0000;0.0095)
(80.0000;0.0132)
(100.0000;0.0160)
(120.0000;0.0186)
(140.0000;0.0208)
(160.0000;0.0222)
(180.0000;0.0221)

G(s) = 0.03217 / 135.19*s^1 + 1 exp -8.2228 s

(1798.2, 0.036339)

FIGURE 6 Identification of FOPTD system parameters and comparative dynamic step responses. FOPTD, first-order plus time delay

5 | EXPERIENCES AND RESULTS

In the Faculty of Chemical Engineering of the Universidad Nacional del Litoral, the Instrumentation and Process Control degree course is taught. This course covers introductory topics to linear control systems, including system modeling, identification and design of controllers, among many other topics. Over the past 15 years, and for the reasons stated above, it has been decided to use open and freely distributed software tools. Specifically, it works with GNU Octave and a set of scripts that simplify the development of the contents, facilitating the tasks of the students.

However, in recent years, some difficulty has been observed in students for the use of simulators and the various tools proposed, which led to the development of a toolbox designated as *LTI System Control Toolbox*[f,g,h,i](LTITOOL). The observed difficulties made it possible to add new features to guiEditor, improving the development of both tools.

At the end of the course, students were request to complete a survey on the experience that they had with this new version of the LTITOOL toolbox. The students were instructed to answer on a 5-level scale and, they were given ample time (10 minutes) to complete their answers.

The questions that were asked are presented bellow and the results obtained summarized in Table 1.

1. It was easy to understand the software management (Yes/No/Not much)
2. Rate the graphical interface developed (Poor 1 - Very Good 5)
3. The software was useful for understanding the topics taught in class (Not much 1 - Much 5)
4. Which of the following topics the software helps you understand them for? (Not much 1 - Much 5)

[f]Adam EJ. Installing Ltitool version 1.0.0. https://youtu.be/T-kPkdVgSZs. Accessed November 2019.
[g]Adam EJ. Analysis and simulation of linear systems to open loop. Part II. https://youtu.be/DnRsB5ohVqk. Accessed November 2019.
[h]Adam EJ. Closed loop LTI system with unity feedback using LTITOOL. Part I. https://youtu.be/J7ufQ0vS53o. Accessed January 2020.
[i]Adam EJ. Closed loop LTI system with unity feedback using LTITOOL. Part II. https://youtu.be/zHzpQVgcQTo. Accessed January 2020.

Questions	Answers					Min	Max	Avg
	Yes		No	Not much				
1	19		1	1				
	Min				Max			
	1	2	3	4	5			
2	0	0	2	9	10	3	5	4.38
3	0	0	0	8	13	4	5	4.62
4 (a)	0	0	3	7	11	3	5	4.38
4 (b)	0	0	1	8	12	3	5	4.52
4 (c)	0	2	2	7	10	2	5	4.19
4 (d)	0	0	2	8	11	3	5	4.43
4 (e)	0	1	6	9	5	2	5	3.86
4 (f)	0	0	2	10	9	3	5	4.33
4 (g)	1	5	3	8	4	1	5	3.43
4 (h)	2	5	2	9	3	1	5	3.29
Mean value:								4.14

TABLE 1 Results of the processing of the answers given by 21 students

(a) Open loop dynamics
(b) Root locus
(c) Frequency domain
(d) Stability
(e) Tuning PID controllers
(f) Closed loop dynamic behavior
(g) Feedforward control
(h) Cascade control

According to the results of Table 1, it is observed that for the first question, most students find the developed software very easy to manipulate, being a merit of the developed graphical environment. While for the remaining questions, the results obtained in all questions are very good, exceeding the average value of 4 over a maximum value of 5, with an average value of all questions of 4.1429. Also in Figures 7 and 8 the results are shown in graphical form.

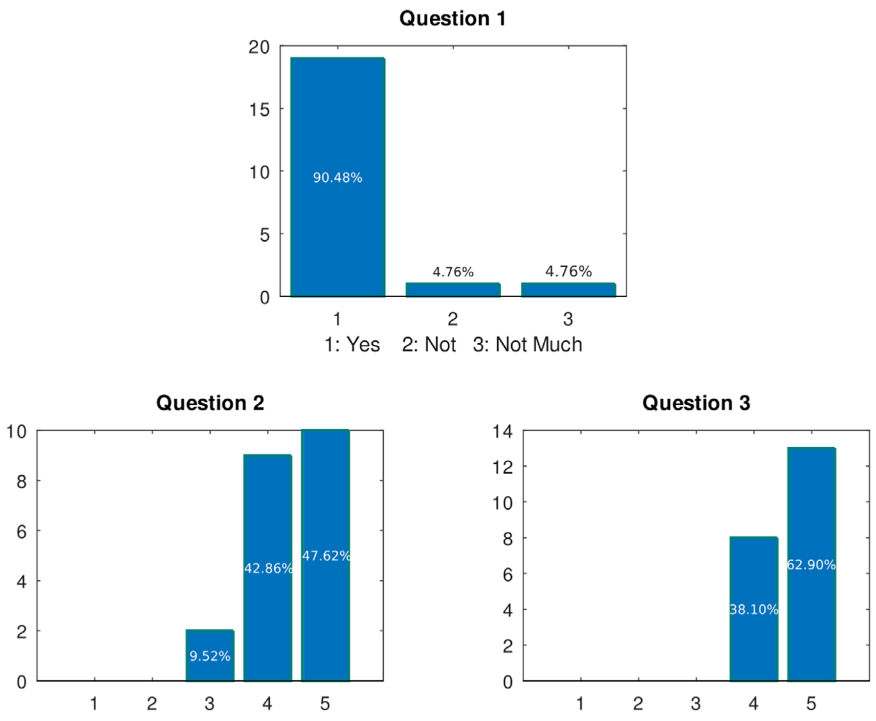


FIGURE 7 Bar graphs for questions 1, 2, and 3

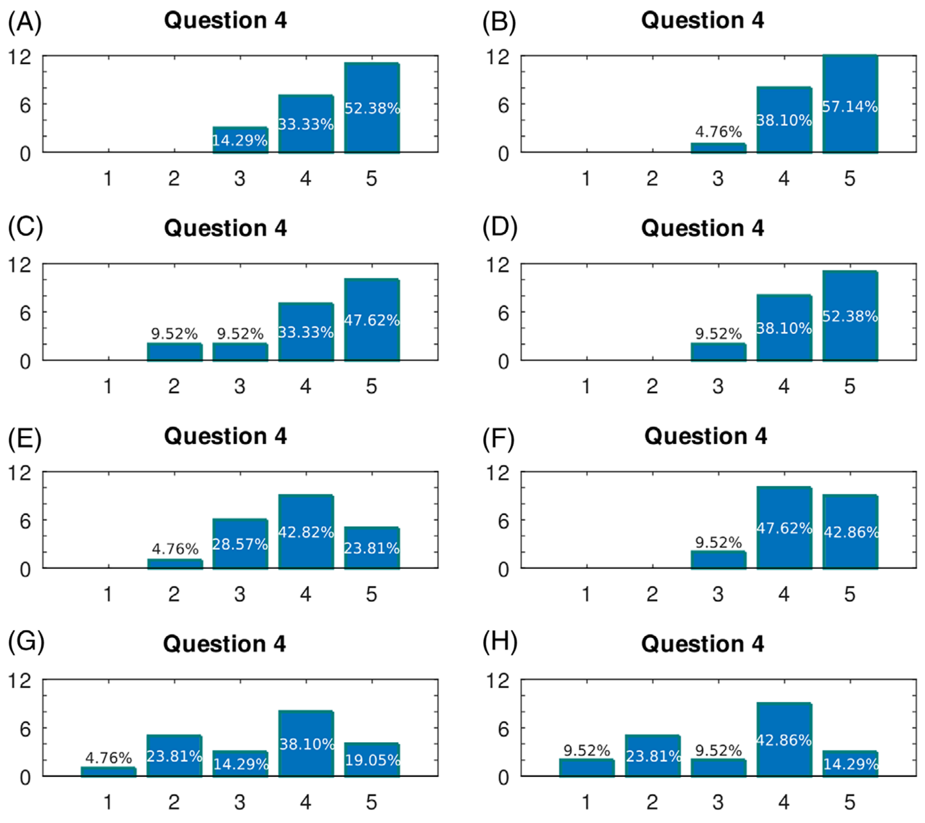


FIGURE 8 Bar graphs for questions 4

TABLE 2 Results of the *t*-test obtained with paired data reported in Table 1 based on the survey questions

Paired questions	Mean difference related	*t*-Value	2 *P*-Value
3-4 (a)	0.238	2.500	.021
3-4 (b)	0.095	1.451	.163
3-4 (c)	0.429	2.905	.009
3-4 (d)	0.190	2.169	.042
3-4 (e)	0.762	6.478	≈0
3-4 (f)	0.286	2.828	.010
3-4 (g)	1.190	7.278	≈0
3-4 (h)	1.333	7.135	≈0

It is thought that the high score obtained with questions 1, 2, and 3 is the result of a simple graphical interface and at the same time easy to understand by inexperienced students and users. While for question 4, this includes topics that are very complex to understand by students of chemical and food engineering and, therefore having such good results suggests that the development with the GUI editor has achieved its main objective.

This last conclusion is supported by a statistical analysis known as a unilateral *t*-test for dependent samples. For this, it was adopted as null hypothesis $H_0 : \mu_1 = \mu_2$ and as true hypothesis $H_1 : \mu_1 \neq \mu_2$, where μ_1 and μ_2 are the true means corresponding to questions 3 and 4, respectively.

The results of the *t*-test were obtained using the GNU Octave statistics toolbox, particularly with the `ttest` function. Table 2 summarized these results.

According to the results of Table 2, it is observed that there is statistical evidence to conclude that,

1. the null hypothesis is rejected in all cases except for pair 3-4 (b) and therefore it is concluded that for most cases the software developed was recognized as useful for learning topics of the subject.
2. except in the comparison of pair 3-4 (b) where the rating is similar on average for this sample.

Finally, another advantage observed in this toolbox version with respect to previous versions is its easy distribution and the low hardware requirement. That greatly benefits the students and is another reason why it is attributed a high score.

6 | CONCLUSIONS

The developed environment presented here opens new opportunities, not only for the migration of existing applications but also for the construction of new applications based on GUI for teaching and research. The main contributions of this work are summarized below.

This development environment allows the construction of graphical interfaces visually from functions incorporated in GNU Octave, in addition, new elements have been added to simplify the creation of objects for didactic and/or research purposes. In this sense, the interaction with Arduino[j] has been significantly improved, allowing the incorporation of communication with these devices.[10,20,27]

Although the most significant contribution of this work are the interfaces with embedded systems, with the same methodology it is possible to add visual controls with their counterpart in scripts[k], making it possible to extend the use of the editor to other areas of knowledge.

Based on analysis of student opinions, these advantages were verified through the LTITOOL toolbox, concluding that the experience of using a teaching tool such as LTITOOL has shown a strong reception by students, because it facilitates the understanding of advanced course topics, as shown by the statistical survey done so far. Thus, from the student-centered learning perspective, this development and similar ones give teachers the flexibility to adapt the teaching objects to the needs that they are considered necessary to emphasize.

ACKNOWLEDGEMENTS
The authors would like to thank support from the Facultad Regional Paraná at Universidad Tecnológica Nacional and the Facultad de Ingeniería Química at Universidad Nacional del Litoral of Argentina.

PEER REVIEW INFORMATION
Engineering Reports thanks Erik de Graaff, Juvenal Rodríguez Reséndiz, and other anonymous reviewers for their contribution to the peer review of this work.

CONFLICT OF INTEREST
Authors have no conflict of interest relevant to this article.

ETHICS STATEMENT
The authors declare that informed consent has been obtained from the participants of the questionnaire.

ORCID
Enrique S. Burgos https://orcid.org/0000-0002-8919-0159
Eduardo J. Adam https://orcid.org/0000-0003-0156-9832

REFERENCES
1. Milano F. An open source power system analysis toolbox. *IEEE Trans Pow Syst*. 2005;20(3):1199-1206.
2. Vanfretti L, Milano F. Experience with psat (power system analysis toolbox) as free and open-source software for power system education and research. *Int J Electr Eng Educ*. 2010;47(1):47-62.
3. Ferreira S, Faezipour M. Advancing the development of systems engineers using process simulators. *Proc Comput Sci*. 2012;8:81-86.
4. Malec J, Toškan D, Snoj L. Pc-based jsi research reactor simulator. *Ann Nucl Energy*. 2020;146:107630.
5. Abdulameer A, Sulaiman M, Aras M, Saleem D. Gui based control system analysis using pid controller for education. *Indones J Electr Eng Comput Sci*. 2016;3(1):91-101.
6. Silva PHO, Nardo LG, Martins SAM, Nepomuceno EG, Perc M. Graphical interface as a teaching aid for nonlinear dynamical systems. *Europ J Phys*. 2018;39(6):065105.

[j]Burgos ES. Ej05: Communication with Arduino development boards. https://youtu.be/ZmANGssRCDw. Accessed February 2020.
[k]Burgos ES. Ej04: Callbacks, events and the callback control. https://youtu.be/xaWTpLStAxY. Accessed February 2020.

7. Mitchell R. Improved matlab guis for learning frequency response methods. *IFAC Proc Vol*. 2013;46(17):303-308.
8. Lanagran-Soler F, Vazquez R, Arahal MR. A matlab educational gui for analysis of gnss coverage and precision. *IFAC-PapersOnLine*. 2015;48(29):93-98.
9. Oravec J, Bakošová M. Piddesign–software for pid control education. *IFAC Proc Vol*. 2012;45(3):691-696.
10. Coll H, Bri D, Garcia M, Lloret J. Free software and open source applications in higher education. Paper presented at: Proceedings of the Mathematics and Computers in Science and Engineering WSEAS International Conference, WSEAS, Heraklion, Greece; 2008:5.
11. Eaton JW. Gnu octave and reproducible research. *J Process Control*. 2012;22(8):1433-1438.
12. Rawlings JB, Mayne DQ, Diehl M. *Model predictive control: Theory, computation, and design*. Vol 2. Madison, WI: Nob Hill Publishing; 2017.
13. Risbeck, M. and J. Rawlings, 2016: *MPCTools: Nonlinear model predictive control tools for CasADi (Octave interface)*. https://bitbucket.org/rawlings-group/octave-mpctools.
14. Richter H. Mote3d: an open-source toolbox for modelling periodic random particulate microstructures. *Modell Simulat Mater Sci Eng*. 2017;25(3):035011.
15. Čisar P, Odry P, Maravić Čisar S, Stankov G. Teaching spread spectrum in the course telecommunication systems using octave. *Comput Appl Eng Educ*. 2020;28(2):367-383.
16. McAndrew A. Open source computer systems in mathematics education. Paper presented at: Proceedings of the 20th Asian Technology Conference in Mathematics; 2015:162-177; Leshan, China: Mathematics and Technology LLC.
17. Igual R, Marcuello JJ, Plaza I, García-Magariño I, Arcega FJ. Experiences using free software simulation tools in engineering higher education. The EDULEARN16 conference was 4th-6th July 2016, Barcelona, Spain; 2016;8653-8662.
18. Burgos ES, Adam EJ. The guiEditor proyect site; 2020. https://gitlab.com/labinformatica/guieditor/.
19. Shaukat K, Tahir F, Iqbal U, Amjad S. A Comparative Study of Numerical Analysis Packages. *International Journal of Computer Theory and Engineering*. 2018;10(3):67–72. http://dx.doi.org/10.7763/ijcte.2018.v10.1201.
20. Adam EJ, Burgos ES. Aplicación de las librerías qt en el desarrollo de un toolbox de sistemas de control para enseñanza en ingeniería. Paper presented at: Proceedings of the XXIII Congreso Argentino de Control Automático (AADECA 2012), Buenos Aires; 2012; Buenos Aires, Argentina, (CD Version).
21. Adam EJ, ES Burgos. Desarrollo de interfases gráficas y simuladores dinámicos para enseñaza de control de procesos. Paper presented at: Proceedings of the CAIQ2013 - VII Congreso Argentino de Ingeniería Química; 2013; Rosario, Santa Fe, Argentina, (CD Version).
22. Adam, E. J., 2020. *Instrumentación y Control de Procesos. Notas de Clase*. 3rd ed. Santa Fe, Argentina: Ediciones UNL. https://bibliotecavirtual.unl.edu.ar:8443/bitstream/handle/11185/5542/instrumentacionprocesos.pdf?sequence=1&AMP;isAllowed=y.
23. Åström KJ, Murray RM. *Feedback Systems. An Introduction for Scientists and Engineers*. Princeton, NJ: Princeton University Press; 2008.
24. Seborg DE, Edgar TF, Mellichamp DA, Doyle FJ III. *Process Dynamic and Control*. 4nd ed. Hoboken, NJ: John Wiley & Sons; 2016.

How to cite this article: Burgos ES, Adam EJ. Graphical user interface editor for Octave applications. *Engineering Reports*. 2020;2:e12269. https://doi.org/10.1002/eng2.12269

APPENDIX A. SOURCE CODES

A.1 polyfit application

Selected fragments of scripting code from example applications:

Listing 1: “Calculate” button Octave code

```
pointData = getappdata(mainDlg.figure, 'pointData');

d = str2num(get(mainDlg.edDeg, 'String'));
p = polyfit(pointData.x, pointData.y, d);

xMin = str2num(get(mainDlg.edXN, 'String'));
xMax = str2num(get(mainDlg.edXM, 'String'));

step = (xMax - xMin)/100;
xRange = [xMin:step:xMax];
plotImg = mainDlg.Image_1;
newplot(plotImg);
```

```
pVal = polyval(p, xRange);
hold(plotImg, 'on');
plot(plotImg, xRange, pVal, 'b', 'linewidth', 3);
grid(plotImg, 'on');
strPoly = polyout(p);
plot(plotImg, pointData.x, pointData.y, 'or', 'linewidth', 2);
hold(plotImg, 'off');
axis(plotImg, [xMin xMax]);
set(mainDlg.labPoly, 'String', strPoly);
set(mainDlg.labPoly, 'TooltipString', strPoly);
```

Listing 2: "Add point" button Octave code

```
pointData = getappdata(mainDlg.figure, 'pointData');

newX = str2num(get(mainDlg.edX, 'String'));
if(findX(pointData.x, newX))
  errordlg (['There is a point with the same value at x (' num2str(newX) ')'],
  'Invalid value');
else
  newY = str2num(get(mainDlg.edY, 'String'));
  pointData.x(length(pointData.x) + 1) = newX;
  pointData.y(length(pointData.y) + 1) = newY;
  [pointData.x, pointData.y] = sortValues(pointData.x, pointData.y);
  set(mainDlg.pointGroup, ...
    'Title', ['Points: ' num2str(length(pointData.x))]);
  set(mainDlg.pointLst, 'String', vec2lst(pointData.x, pointData.y));
  setappdata(mainDlg.figure, 'pointData', pointData);
endif
```

A.2 Source code of an identification toolbox button

Listing 1: Button to Apply Least Square Method

```
% global variable
global _sysidentBasePath;

% s-variable is defined
s = tf('s');

% Loaded data capture
pointData = guidata(graphMethWnd.figure);

% Apply Least Square Method (LSM)
t = pointData.x;
y = pointData.y;

% Estimate K, T and theta by mean of LSM
[K, T, theta] = LSM(t,y);
Gs = K/(T*s+1); % undelayed Gs

% Step response with FOPTD for LSM
for n=1:length(t),
  if t(n) < theta
```

```
    yaproxSenilog(n)=0;
  else
    yaproxSenilog(n)=K*(1-exp(-(t(n)-theta)/T));
  end
end
% Plot dynamic response and monitored data
plotImg = graphMethWnd.Image_1;
newplot(plotImg);
hold(plotImg, 'on');
grid(plotImg, 'on');
plot(plotImg, pointData.x, pointData.y,'linewidth',1.5, ...
t,yaproxSenilog,'linewidth',2);
legend('Experimental Data','FOPTD-LS Method','location','southeast')
title('Dynamic Response','FontSize',14)
hold(plotImg, 'off');
% ------------------------------------------------------------------------------
% Information is loaded on the simple method window
% ------------------------------------------------------------------------------
set(graphMethWnd.ngs,'string',polyout(Gs.num{1,1}));
set(graphMethWnd.dgs,'string',polyout(Gs.den{1,1}));
set(graphMethWnd.delaygs,'string',num2str(theta));
```