Graphical User Interfaces for Teaching and Research in Optical Communications

Telmo Almeida[ab], Rogério Nogueira[ab], Paulo André[ab]
[a]Instituto de Telecomunicações, Campus Universitário de Santiago, 3810-193, Aveiro, Portugal.
[b]Departamento de Física, Universidade de Aveiro, Campus Universitário de Santiago, 3810-193, Aveiro, Portugal.

ABSTRACT

This paper highlights the use of graphical user interfaces (GUIs) developed with the guide tool from Matlab® for university level optical communications courses and research activities. Graphical user interfaces programmed with Matlab® would not only improve the learning experience, making models easier to understand, but also could be tweaked and improved by students themselves. As Matlab® is already taught in many universities, this would ease the process. An example of a model for a stationary EDFA is given to demonstrate the ease of use and understanding of the role of all the different parameters of the model, so students can get a real interactive experience. Another considered potential application is in research. With GUIs, researchers can make real-time parameter optimization, quick assessments and calculations, or simply showcase their work to broader audiences who may not be so familiar with the topic. A practical example of a research application is given for a parameter optimization of a model for non-linear phenomena in uncompensated long-haul transmission links is given.

Besides all the emphasis given to practical applications and potential situations for its use, the paper also covers the basic notions of the critical steps in making a successful Matlab® GUI. Ease of use, visual appearance and computation time are the key features of a successfully implemented GUI.

Keywords: Graphical User Interface, Teaching app, Research Tool

1. INTRODUCTION

Graphical User Interfaces (GUIs) have been proposed over the years as innovative tools to improve teaching at several levels, from primary school to university, in a plethora of subjects and areas of knowledge[1].

In the particular case of optics and photonics, many approaches have been proposed, such as toolbox-based interfaces[2], custom designed interfaces[3] and web based simulators[4]. Although they have been valid and often inspiring contributions, GUI's are far from being everyday tools in optics and photonics classrooms. Why? We believe that this is due to four main reasons:

-i) The choice of programming language for the interface: Choosing a programming language is inevitably choosing a set of pros and cons: C++ provides the best computational times but no real plotting and consequent real-time result assessment ability. Visual based languages such as Visual Basic and Visual C++ provide good computational times and real-time assessment, but at an increased complexity that cannot be afforded in the classroom. MATLAB® is the best choice because it is a very popular language in engineering and science courses, it provides real-time assessment with graphical data and there is no requirement of a compiler or a specific operating system[3]. MATLAB® is a tool that may have been avoided in the past for its low computational yield, but with the improvements in hardware that the Moore's law has continuously provided, this is a minor issue.

-ii) The lack of open-source: Why are Legos® so popular? One of the undisputable reasons of why Legos are so popular is that these toys can be transformed according to the users needs, tastes and sensibilities. This is exactly the same powerful feature open-source software brings to the table in terms of educational applications. On top of the right choice of programming language, GUI based software absolutely needs to be open-source so that improvements can be made. These can include modifications, small tweaks, and even visual changes. Most of the GUIs presented so far are just

12th Education and Training in Optics and Photonics Conference, edited by
Manuel F. P. C. Martins Costa, Mourad Zghal, Proc. of SPIE Vol. 9289, 92891U
© 2014 SPIE, OSA, IEEE, ICO · doi: 10.1117/12.2070562

showcasing a certain model, which can have flaws and intricacies that may not appeal to every audience. The most flagrant case is the one of the language. If an educator can't even translate the software to the mother tongue of the students, how can this application be expected to disseminate among the academic world? Open-source freeware developed in a familiar programming environment to the class is an absolute must.

-iii) The application shelf life: Are these GUIs something that the students will be able to use in the future? Can they be valuable tools that they will use further ahead in their professional careers? Extending the shelf life of any kind of knowledge or tool improves its chances of success. Students are keen to invest the majority of their interest in what is foreseeably useful in the future. Open-Source GUIs in MATLAB® have an extended shelf life because they go beyond the mathematical model or physical phenomena that they are showcasing. Mastering GUI programming can be useful for researchers who want to do quick assessments or showcase they're work in a more engaging way or simply reach a broader audience. Graphical user interfaces are wonderful tools in multidisciplinary workgroups where results need to be presented in the clearest way possible or phenomena needs to be explained without getting into too much technicality. A well-programmed GUI will help the student to learn the classroom material but also it will present itself as something useful and appealing for the future.

-iv) Visual Impact, User experience: GUIs are all about creating a user-friendly, engaging experience. From the moment a user opens a GUI, its design has to appeal immediately and invite for experimentation. First looks are always important, as important as providing a stable, bug free environment, with very quick computation times. Bugs, slowdowns or lack of polish are not welcome in any type of software, and this is especially true for educational software. The programmer has to think as a student. What will be the most likely inputs? What error messages to display when a prohibitive range of values is inserted into the program? Where the results should be displayed and the inputs inserted? What cues should be given and how should they be triggered? These are just some examples of issues that the programmer must consider when developing GUIs, and they are often disregarded.

This paper intends to highlight the critical steps in GUI creation with the GUIDE tool from MATLAB®, but also demonstrate practical examples of it can be used as both a teaching and a research tool.

2. IMPORTANT ASPECTS OF BUILDING A GUI

There are many tutorials on how to create a GUI on Matlab®. This section of the paper will not focus in the practical aspect of constructing a GUI step by step but rather address the key aspects mentioned earlier with some useful hints and tips.

It is important to first code a single m.file® a base code where all the calculations are made with a fixed set of input variables. The GUI will only add another dimension on top of it, where it is the user that chooses and experiments with different inputs. From the base code, the programmer decides what inputs will be asked to the user and which ones will remain constant. The programmer can also define a range of values that can be inserted. For example, the step size should be a ranged input. Its limits should be ceiled to a number where the computation time is still reasonable and floored to the minimum number where the presented results are consistent.

To sum up, it is recommended that the GUI creation is a two-step process: first a single m.file with the "core" code is created and tested, to assure that the model works perfectly. The GUI is programmed on top of the base code, where all the visual elements are added. This assures that at leas two distinct debugging sessions are made, one for the code and the "mechanics" of the model itself and other for the visual elements of the application.

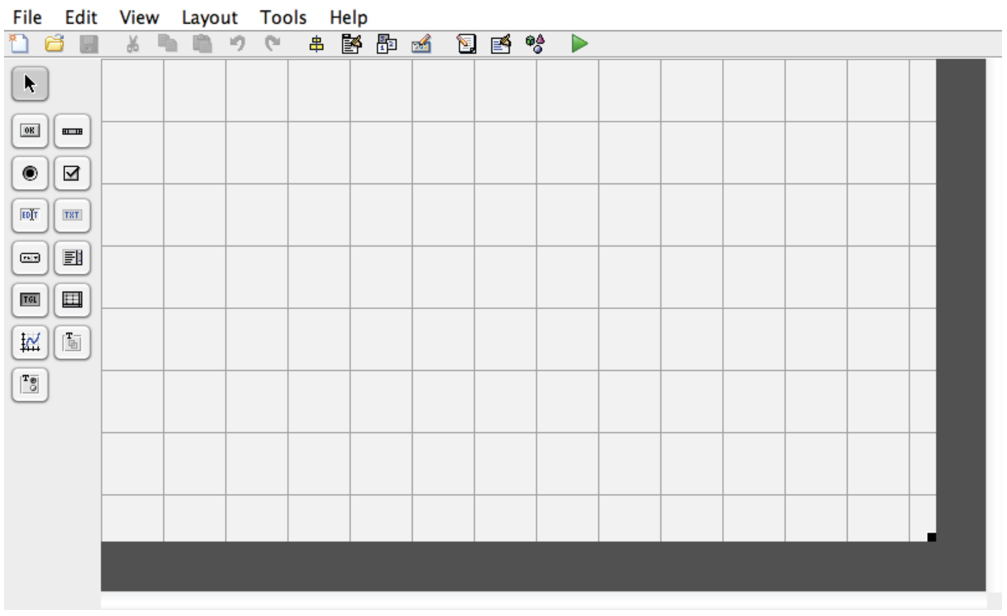


Figure 1: The GUIDE main window.

Figure 1 shows the main window of the GUIDE tool. On the left the user has a set of interface elements to drag and drop to the drawing area on the right. Data can be presented in three different forms: in text boxes, plots or tables. We strongly recommend plots over tables. Text boxes should no be used to present results but rather the values off constants or single parameters that are not to be editable.

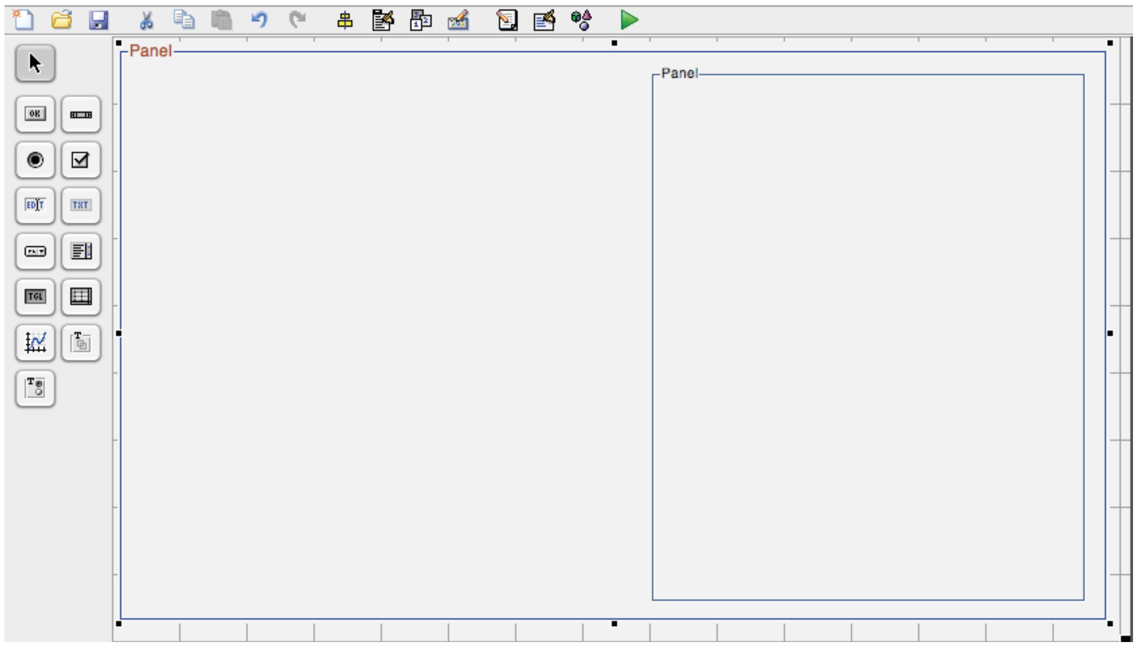
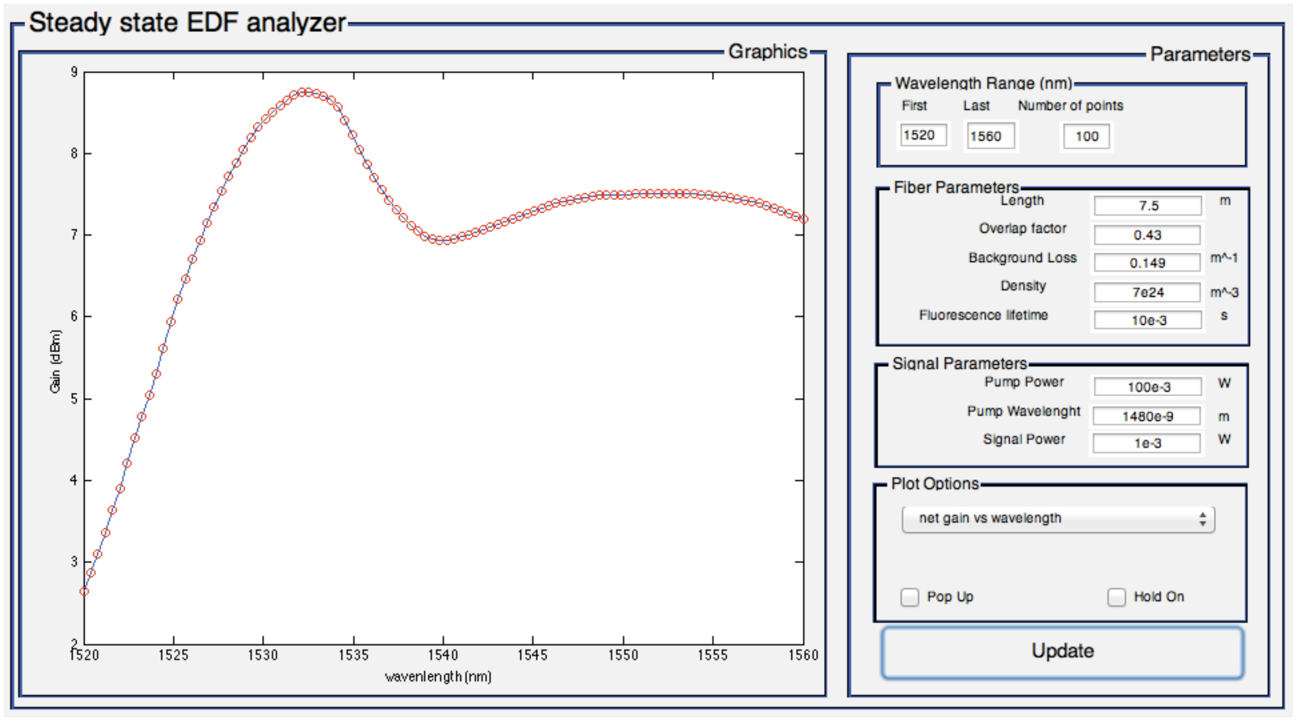


Figure 2: Two panels in an example basic layout.

Panels are very important design elements that are often ignored or put aside. They improve the aspect of the GUI and have the important task of grouping large number of interface elements: it is easier for the programmer to move or resize the entire panel instead of doing it for every single button, and it is easier on the eye of the user to have categorized sets of buttons. An example of an already built interface is shown in the next figure.



Wavelength Range (nm)
First 1520
Last 1560
Number of points 100

Fiber Parameters
Length 7.5 m
Overlap factor 0.43
Background Loss 0.149 m^-1
Density 7e24 m^-3
Fluorescence lifetime 10e-3 s

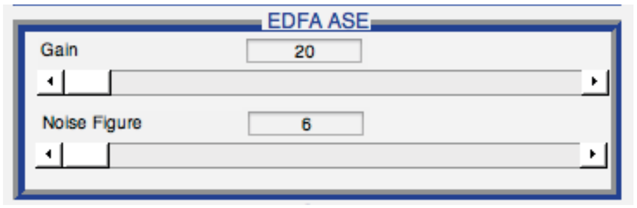
Signal Parameters
Pump Power 100e-3 W
Pump Wavelenght 1480e-9 m
Signal Power 1e-3 W

Plot Options
net gain vs wavelength

Figure 3: A steady-state erbium-doped fiber analyzer.

Having a single window with categorized button panels provides ease of interaction and a large area dedicated to plotting is crucial for quick assessments. If the results often require larger plot areas a button to pop up the graphical area onto a separate window should be made available.

Alternative forms of input such as sliders are welcome, because the user can only use his mouse to control all the GUI, bringing not only commodity, but also making it easier to use for incapacitated students.



Gain 20

Noise Figure 6

Figure 4: Sliders placed beneath the text input boxes.

Having a single window with categorized button panels provides ease of interaction and a large area dedicated to plotting is crucial for quick assessments. If the results often require larger plot areas a button to pop up the graphical area onto a separate window should be made available. Plot options should be organized in a pop-up menu so they are concealed yet promptly available on the menu is activated.

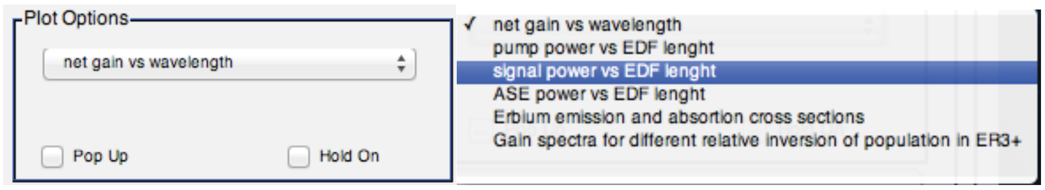

Plot Options
net gain vs wavelength
Pop Up
Hold On
✓ net gain vs wavelength
pump power vs EDF lenght
signal power vs EDF lenght
ASE power vs EDF lenght
Erbium emission and absortion cross sections
Gain spectra for different relative inversion of population in ER3+

Figure 5: On the menu, pop-up button and "hold on" button, on the right the several plot types that can be chosen when the menu is selected.

Having a single window with categorized button panels provides ease of interaction and a large area dedicated to plotting is crucial for quick assessments. If the results often require larger plot areas a button to pop up the graphical area onto a separate window should be made available. Plot options should be organized in a pop-up menu so they are concealed yet promptly available on the menu is activated.

3. EXAMPLES OF GUIS

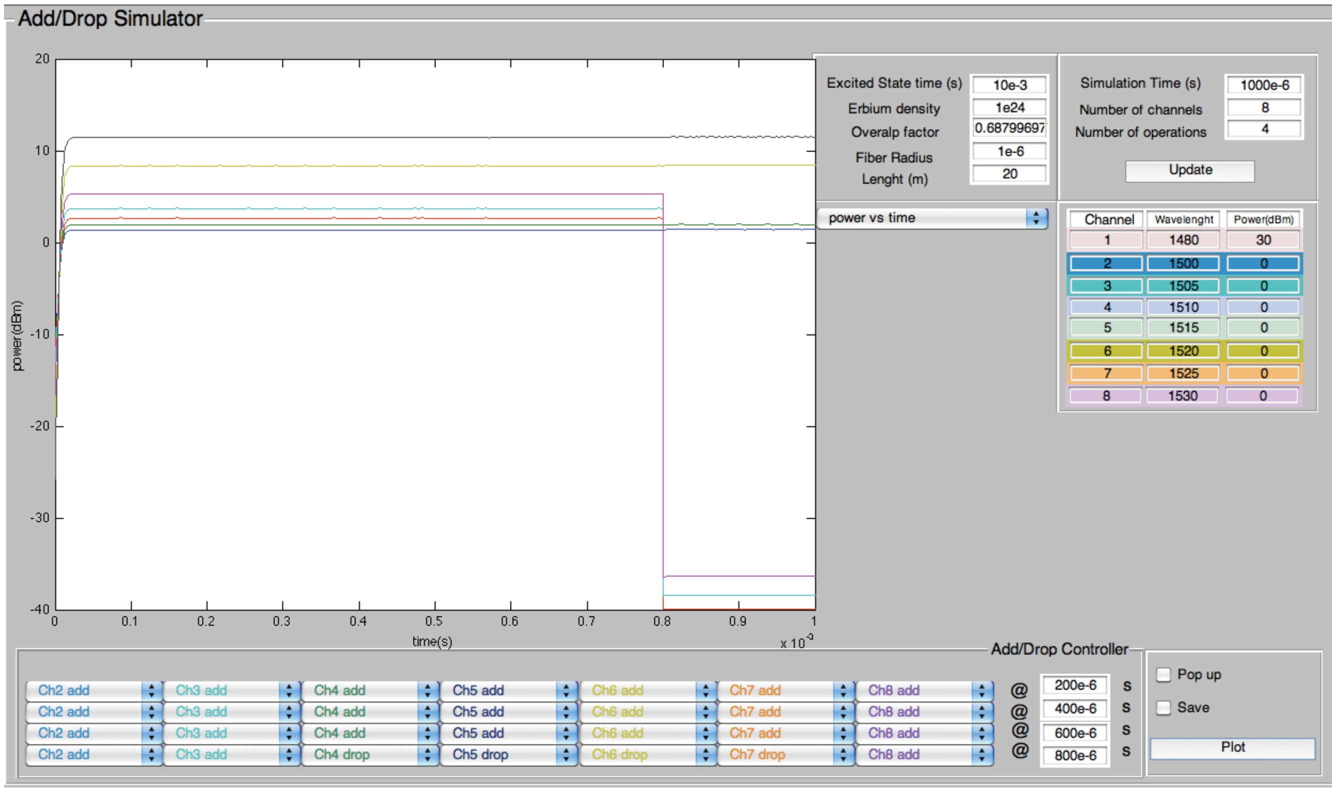
The first example of GUI is a steady-state EDFA analyzer, presented above in Figure 3. It was developed to be both a research and an educational tool. The aim of this particular work was to create a simulation platform using MATLAB®, a tool that is both available to educators and researchers: an accessible tool, easy to use, that allows users to change all the desired parameters and functions in a quick fashion. Steady-State models are useful when wants to study the influence of each individual parameter in the overall performance of the amplifier. In order to do so a GUI has to provide quick computation times and a good visualization so the impact brought by parameter variations can be immediately assessed. As it can be seen in figure 3, this GUI consists in a single control window. All the important parameters (every non-fundamental constant) are editable and shown in the panels right to the graphical area where the results will be plotted. There is a number of plot options from which the user can choose. There is also the option to hold on the current plot so different results can be compared in the same window, or, as mentioned before, the current graphic can be shown in a new window outside the GUI by ticking the "pop-up" box. If neither the "pop-up" nor "hold on" boxes have a tick, the GUI will simply replace the old plot with a new one as soon as the update button is pressed. There is also an important pop-up menu with all the important options.

It is almost excused to mention that the GUI is completely open-source, every exampled showcase in this paper is. New users can add their code lines to the already existing ones, creating more plot options for the rollover menu, or simply change the visual aspect of the GUI with simple drag and drop operations with the mouse.

This GUI was originally presented at EUROCON 2011- International Conference on Computer as a Tool as a research and education tool[5]. It is a valuable GUI not only to be presented in a classroom to illustrate the basic principles of operation of an Erbium Doped Fiber Amplifier (EDFA) but also a handy tool for the laboratory, where quick calculations can be made before or during experiments. This GUI also proved to be useful to explain concepts and discuss results in a multidisciplinary team with physicists and electronic engineers, where the latter were no familiar with the physical concepts behind an EDFA.

In steady-steady models the input power is assumed to be constant, making them fast computational models ideal to be presented on a GUI. Although time dependence is a very important aspect of modeling EDFAs, it is not necessary if the user wants to assess and study the influence of the physical parameters of the amplifier, making the GUI a very suitable application for that same purpose.

But since there was an interest and a need for a time-dependent model, a GUI was created in order to do so. This presented a challenge, because the main interest of creating a dynamic model is to study add/drop scenarios. The algorithm as to adaptive in order to solve the necessary number of equations which varies accordingly with the number of add or drop operations and the GUI as somehow to allow it without losing its key features. Ideally, the user would have two separate windows in the GUI, one for defining the add/drop operations and other for visualizing the results but that means complex programming in the second stage of the application development, which is highly undesirable and against the principles of a good interface. GUIs have to be maintained has simple as possible, not only for the sake of an user-friendly interface but also for the aspect of computation time. The computation time should be associated to the core of the code and not to the graphical element and kept as low as possible. Despite all these constraints, compromises can be made and a valuable tool for education can still be produced, one that would illustrate with rigor the concept of adding or dropping a channel and its associated phenomena. In order to do so, the maximum number of add/drop operations in the GUI was limited to 4 while the number of channels limited to 8. The core algorithm that the interface is running is more than capable of calculating a limitless number of add/drop operations, but turning it in a successful GUI is a difficult task. That is not a limiting factor if proper limitations are performed.



Excited State time (s)	10e-3
Erbium density	1e24
Overalp factor	0.68799697
Fiber Radius	1e-6
Lenght (m)	20

Simulation Time (s)	1000e-6
Number of channels	8
Number of operations	4

Channel	Wavelenght	Power(dBm)
1	1480	30
2	1500	0
3	1505	0
4	1510	0
5	1515	0
6	1520	0
7	1525	0
8	1530	0

Ch2 add	Ch3 add	Ch4 add	Ch5 add	Ch6 add	Ch7 add	Ch8 add	@	200e-6	s
Ch2 add	Ch3 add	Ch4 add	Ch5 add	Ch6 add	Ch7 add	Ch8 add	@	400e-6	s
Ch2 add	Ch3 add	Ch4 add	Ch5 add	Ch6 add	Ch7 add	Ch8 add	@	600e-6	s
Ch2 add	Ch3 add	Ch4 drop	Ch5 drop	Ch6 drop	Ch7 drop	Ch8 add	@	800e-6	s

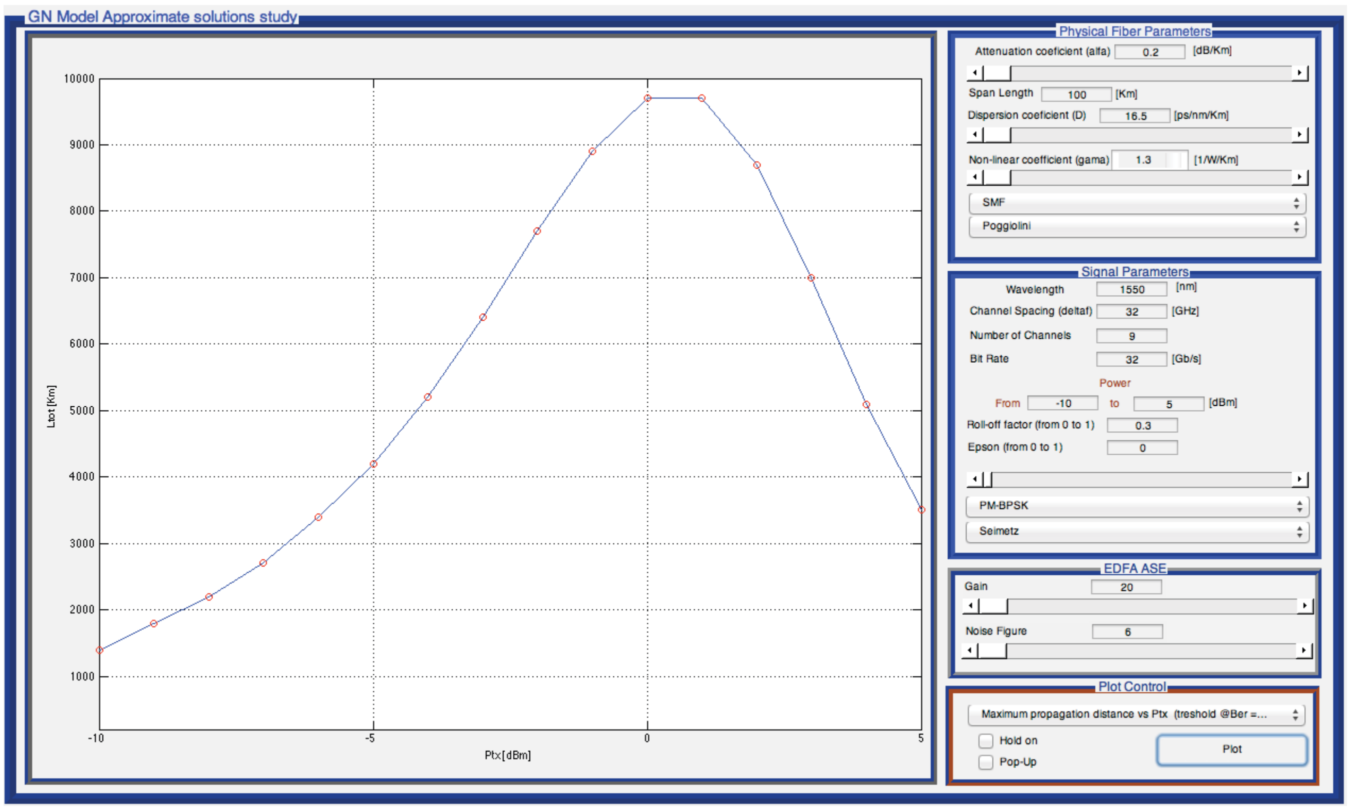
Figure 6: Dynamic EDFA GUI.

The developed GUI can be seen in figure 6 below. When the GUI is started, all the necessary input boxes and pop-up menus for 8 channels and 4 operations are displayed. But, as soon as the user chooses an inferior number of operations and channels these boxes disappear, only to show up again if corresponding number of operations or channels is inputted again. These dynamic features are very easy to program with *IF* cycles that scan the properties of every visual element (button, box, pop-up menu) and are of great visual aid to the user.

As for the rest of the GUI, it follows the guidelines of its antecessor: A good graphical area, with the possibility to pop-up and hold on. It is also useful to remind that this GUI is running on MATLAB®, and that means that every usual plot tool (zoom, legends, save) is available from the start.

The last GUI example (figure 7) is a recently developed GUI for studding closed form expressions of an analytical model for nonlinear propagation in uncompensated long haul transmission links[6].

The idea once again was to study the model for a very large number of scenarios, modulation formats and parameter values. The ideal tool for a researcher to showcase results and fine-tune every parameter individually is a GUI. Since this model is able to predict quiet accurately the reach in transmission links, a very handy tool even for commercial purposes was developed.



Attenuation coeficient (alfa) 0.2 [dB/Km]

Span Length 100 [Km]

Dispersion coeficient (D) 16.5 [ps/nm/Km]

Non-linear coefficient (gama) 1.3 [1/W/Km]

SMF

Poggiolini

Wavelength 1550 [nm]

Channel Spacing (deltaf) 32 [GHz]

Number of Channels 9

Bit Rate 32 [Gb/s]

Power

From -10 to 5 [dBm]

Roll-off factor (from 0 to 1) 0.3

Epson (from 0 to 1) 0

PM-BPSK

Seimetz

Gain 20

Noise Figure 6

Maximum propagation distance vs Ptx (treshold @Ber =...

Hold on

Pop-Up

Figure 7: GUI for parameter optimization of a model for non-linear phenomena in uncompensated long-haul transmission.

4. CONCLUSION

The main factors for successful GUI were addressed, and important tips and hints for building a GUI using the GUIDE tool in MATLAB® were addressed. A GUI needs to be visually appealing and easy to use, open source and have a broad range of applicability, rather than just being focused in education. Examples of GUIs were demonstrated, as well as possible challenges and the means to overcome them. The programmer must learn to develop always with the users in mind and compromise and adapt when necessary. The example GUIS were always developed in a research context but its adaptation to the classroom is very easy and straightforward. Depending on the model or subject in question, GUIs may even have commercial value.

REFERENCES

[1] Metros, S. E., Hedberg and J. G., Harvey, "More than just a pretty (inter) face, The Role of the Graphical User Interface in Engaging eLearners," The Quarterly Review of Distance Education, Volume 3(2), 191-205 (2002).

[2] Melo JR, C. F., N., Lima, C.A., de Alcantara, L.D.S, dos Santos, R.O and Costa, J.C.W.A, "A Simulink™ toolbox for simulation and analysis of optical fiber links," Sixth International Conference on Education and Training in Optics and Photonics, Proc. SPIE 3831, (2001).

[3] Depick, C. and Assanis, D.N., "Graphical User Interfaces in an Engineering Education Environment," Computer Applications in Engineering Education, Volume 13, 48-59 (2005).

[4] Campbell, S., Madden, K., Strickland, A., Williams, R., Jovanovic, N., Johnston, B.F. and Dawes, J.M., "Web-based photonics simulator for secondary school students", Eleventh International Conference on Education and Training in Optics and Photonics, Proc. SPIE, (2009).

[5] Almeida, T., Girão, J., André, P.S. and Teixeira, A., " GUI model for simulation of steady state Erbium dopped fiber amplifiers", Eurocon - International Conference on Computer as a Tool, Proc. IEEE, (2011).

[6] Poggiolini, P. " The GN Model of Non-Linear Propagation in Uncompensated Coherent Optical Systems", Journal of Lightwave Technology, Volume 30, 3857-3879, (2012).