Optics Simulations: A Python Workshop

H. Ghalila*[a,b], A. Ammar[a,b], S. Varadharajan[d] Y. Majdi[a,b], M. Zghal[b,e], S. Lahmar[a,b], V. Lakshminarayanan*[c]

[a]Laboratoire de Spectroscopie Atomique, Moléculaire et Applications, Faculté des Sciences de Tunis - Université de Tunis El Manar, Tunis, Tunisia, [b]Société Tunisienne d'Optique, Tunis, Tunisia, [c]School of Optometry and Vision Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada, [d]Brien Holden Institute of Optometry and Vision Science and Prof. Brien Holden Eye Research Center, L V Prasad Eye Institute, Hyderabad, India, [e]University of Carthage, Engineering School of Communication of Tunis (Sup'Com), GreS'Com Laboratory, Ghazala Technopark, 2083 Ariana, Tunisia

ABSTRACT

Numerical simulations allow teachers and students to indirectly perform sophisticated experiments that cannot be realizable otherwise due to cost and other constraints. During the past few decades there has been an explosion in the development of numerical tools concurrently with open source environments such as Python software. This availability of open source software offers an incredible opportunity for advancing teaching methodologies as well as in research. More specifically it is possible to correlate theoretical knowledge with experimental measurements using "virtual" experiments. We have been working on the development of numerical simulation tools using the Python program package and we have concentrated on geometric and physical optics simulations. The advantage of doing hands-on numerical experiments is that it allows the student learner to be an active participant in the pedagogical/learning process rather than playing a passive role as in the traditional lecture format. Even in laboratory classes because of constraints of space, lack of equipment and often-large numbers of students, many students play a passive role since they work in groups of 3 or more students. Furthermore these new tools help students get a handle on numerical methods as well simulations and impart a "feel" for the physics under investigation.

Keywords: Active learning, Optics simulations, Python programming language, Numerical experiments

1. INTRODUCTION

"*Active learning*" and "*Optics simulations*" together may be joined as "*Active Learning in Simulating Optics*" (ALSO). The goal of this paper is to encourage teachers and students to take an active part in developing their own codes as they design individual own experiments. There are many useful codes, both free and commercial, software or online codes, with good interfaces which reproducing perfectly analytical solutions allowing an infinite number of configurations numerically imitating experiments. For examples you can see the RAY program, the WebTOP and PHET project and also Wolfram demonstration project[1,2,3,4]. The basic disadvantage with these prepackaged programs is that we (students and teachers) behave as passive users without understanding the contents or the limitations of the material we use (codes or experiments). The idea here is to encourage academics to code and build their own interfaced programs more easily with minimal cost as possible. In this context, the Python environment offers many advantages compared to other object oriented languages. The most important points are the fact that it is open source, it is well documented and it also guarantees cross-platform compatibility. On the other hand optics is very well adapted to realize very simple experiments showing instantaneously the influences of different physical parameters on direct observations.. The combination of these two factors allows one to develop elaborate original lessons for laboratory activities and for numerical modeling of

14th Conference on Education and Training in Optics and Photonics: ETOP 2017, edited by Xu Liu, Xi-Cheng Zhang, Proc. of SPIE Vol. 10452, 1045218 · © 2017 ICO, IEEE, OSA, SPIE
CCC code: 0277-786X/17/$18 · doi: 10.1117/12.2268377

physical phenomena (in the case of optics, both geometrical and physical optics) with very limited means. We have already presented in previous publications the role of simulations in optics educations using the Python language[5,6,7,8]. Here we focus more on the coding side showing explicitly how to manage with the various toolboxes at our disposal.

2. ACTIVE LEARNING IN OPTICS

2.1 ALOP workshops

Since 2004 many workshops were organized throughout the word within the framework of the UNESCO project ''*Active Learning in Optics and Photonics*" (ALOP) with the support of ICTP (Abdus Salam International Centre for Theoretical Physics, Trieste, Italy) and generously supported over the years by SPIE (SPIE – The International Society for Optics and Photonics) Other support has come from as well the Optical Society of America, The National Academies (USA), The European Optical Society, Esssilor Corporation, etc.[9-15]. This project promotes a hands-on method for teachers and students in optics using very simple and inexpensive material (homemade or locally available material when possible) and also avoiding as far as possible reference to mathematical/quantitative expressions. Furthermore, and in order to overcome the passive attitude of students during traditional lecture format, the teaching paradigm is Predictions, Observations, Discussions and Syntheses (PODS) in order to improve the conceptual understanding of optics. For instance, more than fifteen workshops have been organized just in Tunisia since 2005. Lessons focus on several topics usually found in any introductory university/or high school physics programs such as geometrical optics, interference and diffraction, atmospheric optics and optics in communications. Figure.1 illustrates the context in which active learning procedure works.

Figure.1 : From left to right: Laser pointer fixed on a small patch of polystyrene in front of slits made by cutting the backside of a plane mirror - Group of students proposing solutions followed by a discussion - Experience highlighting a virtual image - Demonstration of the refraction of light by means of a laser and a vessel filled with liquid.

2.2 Organization of ALSO workshops

We propose here a new series of workshops (denoted as ALSO workshops) keeping the hands-on approach applied to the same contents e.g. optics but adding mathematical solutions and their numerical modeling. The idea here is to append to those experiments, such as those shown in Figure.1, codes that allow the manipulation of the experimental configurations that can help to generalize and better understand the concepts previously discussed. As for the experiments, we should give a special care to the training during the coding encouraging students to be active during the development of the codes applying the same PODS method. The main drawback here could possibly be the price of computers. These workshops should last almost the same time e.g. two or three days (the full ALOP workshops last for 5 days and cover 5 modules as well as a pre- and post- workshop conceptual evaluation test). We will choose only two or three modules per workshop and for each module we will manage half day for experiment and half day for the program development.

3. BUILDING PROGRAMS

3.1 How to start programming

Starting to program is always something complicated. We never know exactly at which level we should start. Should we explain bit and addressing and occupying memory or should we start with syntax and instructions? Part of the answer to this question resides in what we want to do. As physics educators, in the context of the active learning and hands-on teaching, we suggest starting with subsections of programs carrying out specific tasks. Generally, an entire program can be divided into two parts: one dedicated to analytical solutions and the other for displaying curves and graphs. For example let's focus on the diffraction-interference phenomena and more precisely on the Young double slit experiment. Here, the normalized intensity is expressed as:

$$I = \mathrm{sinc}^2(Bx)\left[1 + \cos(2Ax)\right] \qquad \text{where : } A = \pi a/\lambda D \quad \text{and} \quad B = \pi b/\lambda D,$$

'b' stand for the width of the slits, 'a' stand for the distance between slits, 'D' is the distance of the screen to the plan of the slits and 'λ' is the monochromatic wavelength of the incident light.

This example is very convenient for numerous reasons. It is very easy and inexpensive to realize experimentally and at the same time it highlights phenomena, such as interference and diffraction, which are not easy to understand and teach. The material shown on the left of Figure.1 is sufficient to do that. Slits are made manually on the plate in front of the laser simply using knife-edge. For the Young double slits we try to make two slits as parallel and identical as possible. So it fits well the hands-on approach we whish to promote. Similarly this hands-on approach could be applied on the numerical modeling. The program bellow simulates the Young double slit configuration and as highlighted it can be divided in two parts one dedicated to the mathematical solution and the other for the graphics.

```
from __future__ import division
from numpy import pi, linspace, sin, cos
import matplotlib.pyplot as plt
#---------- Mathematical solution ----------
lamda = 650*1.E-9          # Wavelength (m)
b     = 1.0*1.E-3          # Slits' width (m)
a     = 2.5*1.E-3          # Distance inter-Slits (m)
D     = 2                  # Screen distance (m)
e     = 1*1.E-2            # Size of the screen (m)
X_Mmax=e/2. ; X_Mmin=-e/2.    # Coordinates of screen
N = 400                    # Number of mesh point
X = linspace(X_Mmin, X_Mmax,N) # Space discretization
B = (pi*b*X)/(lamda*D)     # Intermediate variable
A = (pi*a*X)/(lamda*D)     # Intermediate variable
I = 0.5*(sin(B)/B)**2 * (1 + cos(2*A))
Envelop=(sin(B)/B)**2
#---------- Ploting ----------
fig = plt.figure(figsize=(8,5))
fig.suptitle('Young Double Slits',fontsize=14, fontweight='bold')
ax1 = fig.add_subplot(111)
ax1.grid(True)
ax1.plot(X,I,'-k', linewidth=3)
ax1.plot(X,Envelop,'--k', linewidth=2, alpha=0.8)
ax1.set_xlim(X_Mmin, X_Mmax)
ax1.set_xlabel(r'$X \ (m)$',fontsize=14, fontweight='bold')
ax1.set_ylabel(r'$I(X,Y)/I_0$',fontsize=14, fontweight='bold')
ax1.set_title(r"$wavelength \ \lambda = %.1e \ m, \ b = %.1e \ m, \ a = %.1e \ m, \ D = %.0e \ m$"% (lamda,b,a,D))
plt.show()
```

You can copy this script, past it into python editor and run it as it is. This will generate Figure 2

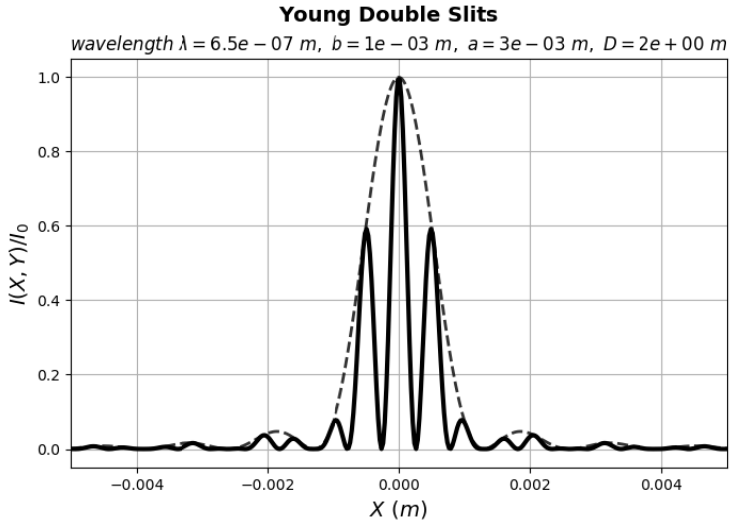

Figure.2 : Distribution of the normalized intensity observed on the screen. The dashed curve represents the sin(Bx)/(Bx).

The hands-on method appears through the ability to easily change the physical parameters (b, a, D, λ) and then determining the different patterns favoring predictions and discussions (OPDS). Of course it is not always easy to experimentally change the wavelength (λ) but thanks to the simulation we can vary it simply through the 'lamda' variable (see script) expanding the possibilities of OPDS. This point shows clearly the complementarity between experiments and programming.

Another interesting aspect here from the point of view of interactive learning in programming is the fact that we can focus on some basic and important notions we systematically encounter when modeling. This has to do with discretization. For instance, let us change in the script above the number of mesh point 'N' and give it the value N=30. The result is displayed in Figure.3.

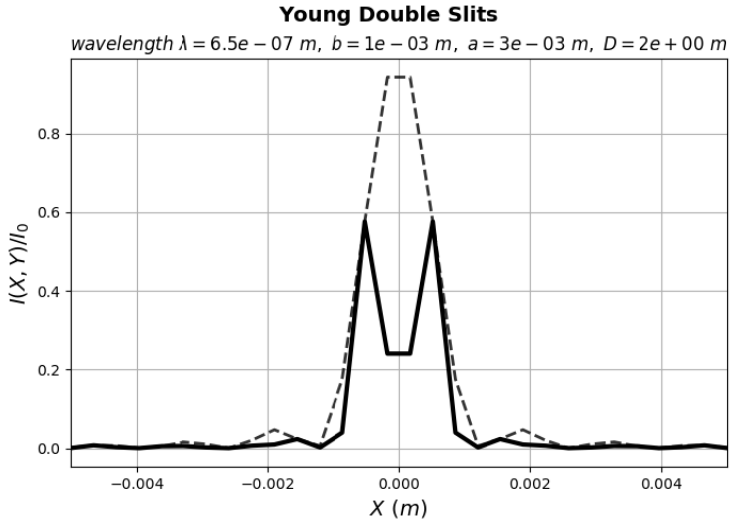

Figure.3 : Same physical configuration as in Figure.2 but with N=30 instead of N=400

As we can rapidly observe on this curve is that the number of peaks inside the central diffraction envelope goes from 5 to only 2 peaks. Actually, when we choose the number of mesh points for any discretization problem we have to deal with resolution problems and here resides many basic aspects in physics, raising many interesting questions.
The second part of the program is fully dedicated to the graphs and here also we can take time to learn how to better manipulate graphics libraries and determine the appropriate physical phenomena representation.

3.2 Interfacing the programs

Interfacing the programs is the next step that we shall realize after having manually tested the correctness of the script. This operation allows us to vary interactively the physical parameters without the need to return to the script. Practically, it involves associating widgets with curves into the graphics, which allow us to modify and extract the important information. Typical widgets come in the form of buttons, menus, box or control on the cursers. All these tools transform the static solutions in quantities similar to the results observed in real experiment, dynamically changing when varying one of the physical parameters. Of course all these tools give us a lot off opportunity in achieving a real hands-on experiences for both the experimental and the numerical point of view. Indeed adding widgets into graphics is equivalent to adding diagnostics in experiences and this is also a good way to push students to become active learners.

Designing a Graphical User Interface (GUI) for an application remains a difficult task and we are not going to develop here the complete steps to achieve this. However there are some procedures that need to be considered which can be roughly summarized in two main steps: the first step is to create a file containing all the elements devoted for interaction with the mathematical function and the second is to convert this file into a language that can be understood by Python. Fortunately the Qt toolkit installed with Python environment provides the necessary material to do that. We have at our disposal the 'Qt Designer' to build interactively our interface and the actual version of the command 'pyuic5'. Figure.4 shows the configuration built for the Young double slit with 'Qt Designer'

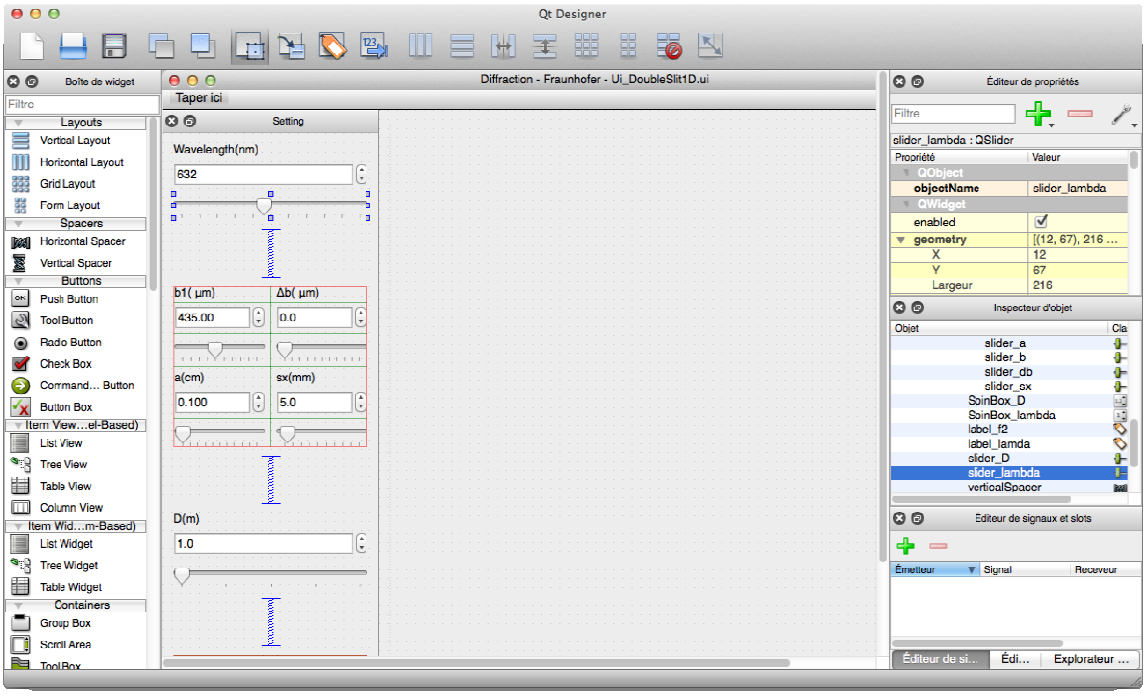

Figure.4 : 'Qt Designer' interface used to build the your own GUI application.

‘Qt Designer’, similarly to other software (Visual Basic, Real Studio, …), allows building GUI by simply dragging widgets listed as icons in the left panel of the windows and control their characteristics (name, size, location, …) with the tools in the right panel. We show here as an example the selection of the slider associated with the wavelength named ‘slider_lambda’ (right panel) accompanied by other information given in the yellow part for the other widgets. After putting in place all the physical parameters (b, a, D, λ) we have to connect them with the intensity ‘I’ implemented in the script given above. How to do this? When saving the configuration shown in Figure.4 we generate a file named by the title on the top of the main window, here ‘Ui_DoubleSlit1D’ with the extension ‘.ui’. This code is translated into Python running on the terminal by the following command ‘pyuic5 -x Ui_DoubleSlit1D.ui -o Ui_DoubleSlit1D.py’. This generates a program in Python language that can be imported as any kind of functions or libraries into our previous script. This is shown in the script bellow where we only exhibit the major additions to the program.

```
from __future__ import division
from numpy import pi, linspace, sin, cos
import matplotlib.pyplot as plt
from Ui_DoubleSlit1D import Ui_MainWindow
#---------------- Mathematical solution ----------------------------
class DoubleSlit1D(QMainWindow, Ui_MainWindow):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.fig1()

    def fig1(self):
            lamda = self.slider_lambda.value()*1.E-9      # Wavelength (m)
            b     = self.SpinBox_b.value()*1.E-6          # Slits' width (m)
            a     = self.SpinBox_a.value()*1.E-2          # Distance inter-Slits (m)
            D     = self.slider_D.value()                 # Screen distance (m)
            e     = self.slider_e.value()*1.E-3           # Size of the screen (m)
…
…

    @pyqtSlot("int")
    def on_slider_lambda_valueChanged(self, value):
        self.SpinBox_lambda.setValue(value)
        self.fig1()
    @pyqtSlot("int")
    def on_slider_b_valueChanged(self, value):
        self.SpinBox_b.setValue(value)
        self.fig1()
…
…

if __name__ == "__main__":
    app = QApplication(sys.argv)
    MyApplication = DoubleSlit1D()
    MyApplication.show()
    sys.exit(app.exec_())
```

Finally the result of this second version is illustrated in Figure 5 showing how the light intensity will vary when manipulating the different widgets displayed on the left panel.

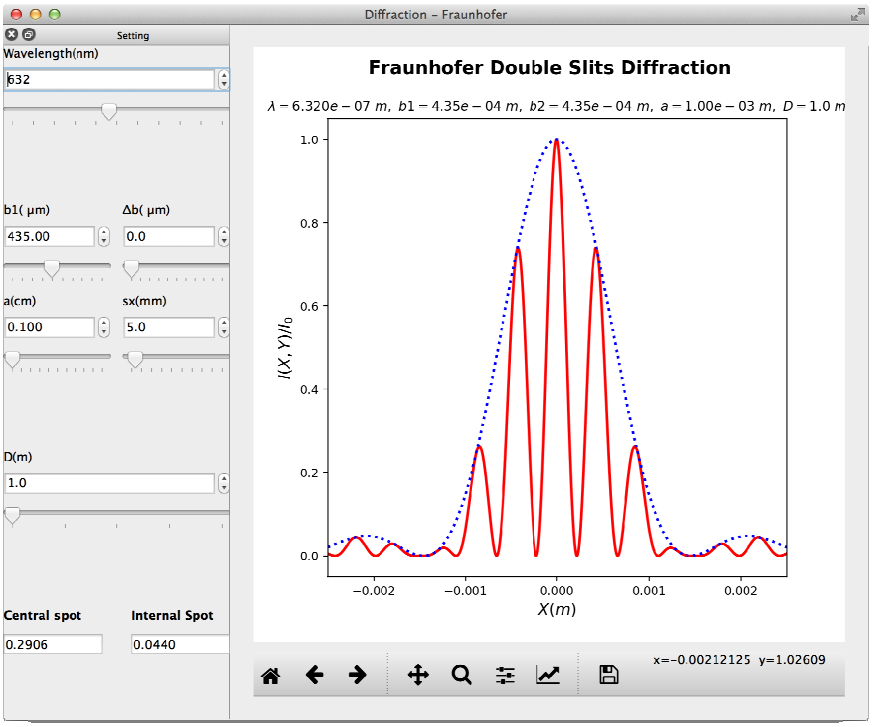

Figure.5 : 'Qt Designer' interface used to build the your own GUI application.

Indeed, the program has increased in complexity but let us focus on the points we are concerned with, i.e. teaching and interactive learning. So we can see here how the values can be varied using the widgets for the physical parameters (a, b, D, λ) instead of the fixed value adopted in the previous static script. We recognize for example, in Figure.4 'slider_lambda' associated with the 'lamda' variable for the wavelength. The lines starting with '@pyqtSlot' in the script insure the new computation of the intensity function 'I' each time one of the widgets is manipulated. So we can see, from this Young double slit example, how we can interactively control any mathematical expression by taking the control of its parameters.

Otherwise and as we noted the implementation of the interface shown in Figure.4 requires numerous operations such as the line 'from Ui_DoubleSlit1D import Ui_mainWindow' inserted in the header but also the need to define: a '__main__' routine, a 'DoubleSlit' class and two functions '__init__' and 'fig1'. This part is perhaps the most subtle and delicate to transmit to teachers and students because we have to insert them manually. The only way out here is to convince participants to firstly accept these notions without understanding all their meanings because our main goal deals with optics and physics and not the subtleties of writing software. The content of the file 'Ui_DoubleSlit1D.py' is also very complicated but we can consider it as a black box. Nevertheless, for all these topics (and more) we can find many documents and guides online[16,17,18].

4. CONCLUSION

Providing opportunities for developing countries to carry out experiments during classroom which they cannot otherwise afford due to lack of means is one of the main goal of this work. The other one is to promote the active learning and hands-on activities during classroom encouraging students to be active learners. Numerical simulation in physics and more specifically in optics is a good way to reach this goal because it is possible to accompany very simple experiments with inexpensive homemade/custom built materials. We have shown in this work that it is possible to elaborate lessons based on the active learning approach for both the experiments in optics and also for the development of codes. This second point is just as important as the first because it also favors questioning and discussions. We can apply similarly to experiments the method based on the Predictions, Observations, Discussions and Syntheses (PODS), which help better understand the basic concepts.

REFERENCE

[1] Eylon, B., Ronen, M., and Uri, G., "Computer Simulations as Tools for Teaching and Learning: Using a Simulation Environment in Optics," J. Sci. Edu. Technology, 5(2), 93-110 (1996).

[2] Carnicer, A., Andilla, J., Ferre, J., et al. "Teaching (and learning) optics using interactive simulations: the JavaOptics course", Proc. SPIE 9664, Ninth International Topical Meeting on Education and Training in Optics and Photonics, 96641G, (2005), doi: 10.1117/12.220772251.

[3] Foley, J.T., Zoughi, M., Herring, S.D., Morris, M., Gilbert, P.J., Moore, D.T., "The Optics Project on the Web: WebTOP ", Proc. SPIE 9663, Eighth International Topical Meeting on Education and Training in Optics and Photonics, 96630K, (2003), doi: 10.1117/12.2207350; http://dx.doi.org/10.1117/12.2207350

[4] Wolfram Demonstrations Project & Contributors, 2017, http://demonstrations.wolfram.com/

[5] Ghalila, H., Ben Lakhdar, Z., Lahmar, S., Dhouaidi, Z., Majdi, Y., "Active Learning in Optics and Photonics: Fraunhofer diffraction", Education and Training in Optics and Photonics: ETOP 2014, Proc. of SPIE Vol. 9289, 92892V, OSA, IEEE, ICO · doi: 10.1117/12.2070776.

[6] Ammar, A., Burman, R., Ghalila, H., Ben Lakhdar, Z., Varadharajan, L.S., Lahmar, S., and Lakshminarayanan. V., "Optics simulations with Python: Diffraction" Education and Training in Optics and Photonics: ETOP 2015, Proc. SPIE Vol. 9793, 97930K, (2015), doi: 10.1117/12.2223072.

[7] Lakshminarayanan, V., and Burman, R.,"Optics Tutorials with Python", Tech. Report, Univ. of Waterloo, 2015, doi: 10.13140/RG.2.1.2940.2325, (2015).

[8] Lakshminarayanan, V., Ghallila, H., Varadharajan, L.S., Ammar, A., Understanding Optics With Python, CRC Press/Taylor and Francis, Boca Raton, FL., (2017).

[9] Alarcon, M., Arthurs, E., Ben Lakhdar, Z., Culaba, I., Lakshminarayanan, V., Maquiling, J., Mazzolini, A., Niemela, J., Sokoloff, D., "Active learning in optics and photonics: experiences in Africa", ETOP 2005, http://spie.org/etop/ETOP2005_040.pdf

[10] Lakshminarayanan, V., "ALOP: Bringing Optics and Photonics to the Developing World", Photonics Spectra, June 2015. https://www.photonics.com/Article.aspx?AID=57453.

[11] Lakshminarayanan, V., "Interactive lecture demonstrations, active learning, and the ALOP project", Proc. SPIE **8065**, 80650S (2011); doi: 10.1117/12.889508

[12] Ben Lakhdar, Z., Derbel, N., Dhaouadi, Z., Ghalila, H., Miled, R., Lahmar, S., Berrada, K., Channa, R., Outzourhit, A.,"Active learning in physics a way for rational thinking - a way for development". http://spie.org/etop/2007/etop07programsI.pdf

[13] Zghal, M., Ghalila, H., Ben Lakhdar, Z., "A simple wavelength division multiplexing system for active learning teaching", ETOP 2009,http://spie.org/etop/2009/etop2009_10.8.60.pdf

[14] Alarcon, M., Ben Lakhdar, Z., Culaba, I., Lahmar, S., Lakshminarayanan, V., Mazzolini, A., Maquiling, J., Niemela, J., " Active learning in optics and photonics (ALOP): a model for teacher training and professional development", ETOP 2010, http://spie.org/x648.html?product_id=860708

[15] Alarcon, M., Arthurs, E., Ben Lakhdar, Z., Culaba, I., Denardo, G., Lakshminarayanan, V., Maquiling, J.,

Mazzolini, A., Niemela, J., Sokoloff, D., “Active learning in optics and photonics: Training Manual” http://unesdoc.unesco.org/images/0021/002171/217100e.pdf
[16] Python Software Foundation, https://www.python.org
[17] Hunter, J. D., “Matplotlib: A 2D graphics environment” Computing In Science and Engineering, Papers 9(3), 90-95, (2007), doi : 10.1109/MCSE.2007.55, http://matplotlib.org.
[18] Riverbank Computing Limited, 2016, http://pyqt.sourceforge.net/Docs/PyQt5/designer.html