Role of Simulations in Optics Education

V. Lakshminarayanan*[a], H Ghalila[b,c], A Ammar[b,c] and S Varadharajan[d]

[a]School of Optometry and Vision Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada,[b]Laboratoire de Spectroscopie Atomique, Moléculaire et Applications, Faculté des Sciences de Tunis – Université de Tunis El Manar, Tunis, Tunisia, [c]Société Tunisienne d'Optique, Tunis, Tunisia,[d]Brien Holden Institute of Optometry and Vision Science and Prof. Brien Holden Eye Research Center, L V Prasad Eye Institute, Hyderabad, India

ABSTRACT

Simulations can play an important role in science education. Simulations (enabled by powerful numerical and visualization methods) are excellent tools for teaching optical phenomena. The advantages of using simulations as a tool for teaching optics include, amongst others, (1) giving students an engaging, hands-on active learning experience, (2) helping in understanding equations as physical relationships among experimental measurements and (3) allowing students to investigate phenomena that would not be possible to experiment on in a laboratory or classroom setting. We illustrate the utility of simulations in optics by describing some examples from geometric and physical optics using the open source programming language Python.

Keywords: Active learning, Simulations, Python programming language, numerical methods, numerical experiments

1 INTRODUCTION

In general, teaching has not changed over the centuries. Figure 1 shows a scene from a classroom from about 1350 CE; the image is a reproduction of a parchment painting rendered by the artist Laurentius de Voltolina. It depicts a university lecture during the fourteenth century at the University of Bologna in Italy.[1] It shows students talking to each other or falling asleep while the teacher lectures. This scene depicted will be very familiar to those of us who teach/lecture. John Amos Comenius (1592-1670) who was a Czech teacher, writes in the frontispiece to the Didactica Magna (1628) “Let the long and the short of our didactic be to investigate and discover the means for teachers to teach less, and learners to learn more.”[2] According to Albert Camus, “Some people talk in their sleep, lecturers talk while other people sleep.”[3] So, the teacher acts as a presenter of facts and the students are passive absorbers of this knowledge.

There is considerable evidence that lectures are ineffective in teaching STEM (Science, Technology, Engineering, Mathematics) subjects. Numerous studies have examined the effectiveness of active learning.[5] In a recent meta-analysis of undergraduate STEM education, Freeman et al. [4] found that the failure rate was 55% higher in traditional lecturing method than in the active learning modes. One of the methods to incorporate active learning in the classical lecture format is the interactive lecture demonstration.[6] The other ways of active learning include visual learning, problem solving or problem based learning, cooperative learning, role playing, peer education, games, computer based education, and simulations.[7] A more recent development is the flipped classroom paradigm [8] wherein students read or study about an assigned topic by themselves and classroom contacts are utilized to solve problems or to do what would be part of traditional homeworks. In this paper we deal with another important component of active learning, namely, computer simulations.

* vengu@uwaterloo.ca; Phone: +1-519-888-4567 ext.38167

Optics Education and Outreach IV, edited by G. Groot Gregory, Proc. of SPIE Vol. 9946, 99460N
 · doi: 10.1117/12.2236159

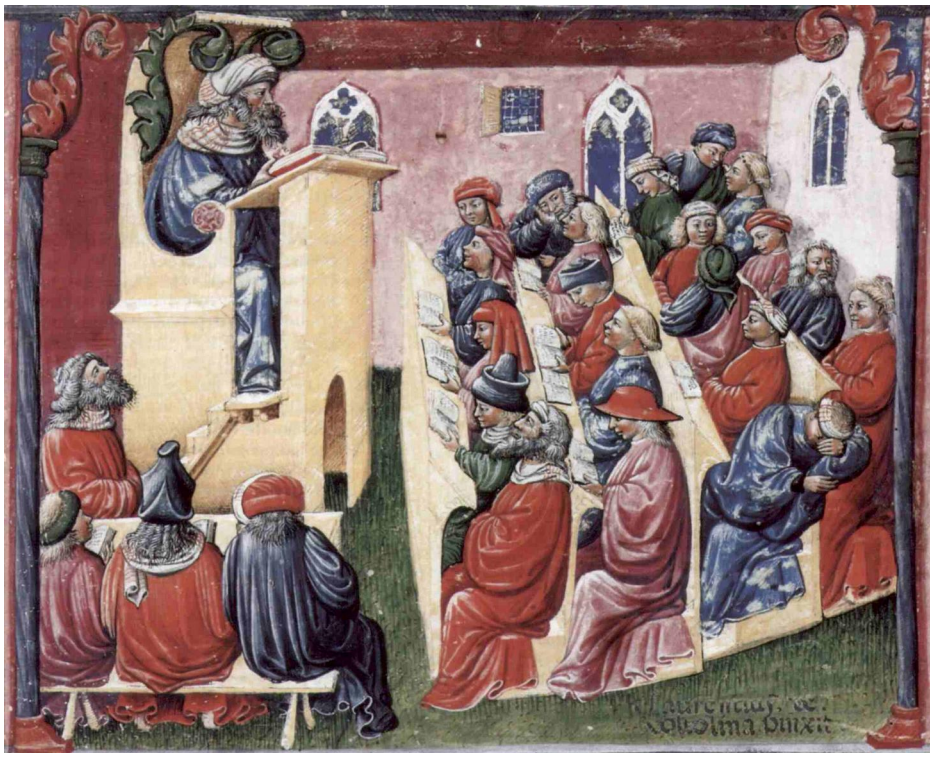

Figure 1: Henry of Germany lecturing at the University of Bologna. Painting by Laurentius de Voltolina (1350).

2 COMPUTER SIMULATIONS

Over the past couple of decades, there has been a great increase in the use of technology in STEM education. More specifically information technology has enabled great advances in teaching.[9] The mass availability of computers, laptops, and mobile devices such as iPads, tablets and the like has led to the implementation of computer simulations in a variety of science subjects. Simulations have been around ever since the advent of computers, and computer simulations have become available for a wide range of science subjects.[10]. According to de Jong and van Joolingen [10], a computer simulation is "a program that contains a model of a system (natural or artificial; e.g., equipment) or a process." Computer simulations offer many advantages including the fact they are often less expensive to create and construct than their real life counterparts. They can remove danger or uncertainty from the situation and explore hypothetical scenarios and hypotheses. They can be paused so that students can assess the results. In recent times, experimental situations have become complex; in addition, in traditional situations only few students can be trained at a time. In these situations, computer simulations offer a great alternative.

What are some of the criteria for a good computer simulation for classroom use? A number of guidelines for "best practice" can be found in the literature.[11] Some of the criteria that are relevant for learning via simulation include:

- The simulations should be true to life. That is, they should simulate the activity so well that there is little difference between the simulated environment and the real one, and the same kind of learning experience can take place.

- The simulations should be "hand-on," involving the students so that the students become participants in the simulation activity. This implies that students interact with the simulation, for example, by changing parameters,

changing lines in the code, discussing the simulated results, etc.

- Simulations should motivate learning. Student involvement in the activity should be such they are motivated to learn more about the activity or the subject matter.

- Simulations should be customizable to the students' needs. Students' developmental requirements should be taken in consideration while designing simulations specifically for them.

- Simulations should be inspirational. Student input should be welcomed and activities designed to encourage students to enhance the activity by contributing their own ideas.

- Simulations are meant to supplement, not replace other teaching modes. Integrating simulations into the curriculum also ensures that connections to domain knowledge and real-world applications are made explicit.

- As with any instructional technology, computer simulations should be chosen to meet the teaching objectives and teach the content.[12]

In the use of computer simulations, there are two important questions, namely: (1) How can they enhance traditional education, and (2) how can they be optimally used to improve learning outcomes. Rutten et al. [13] have analyzed results from education research over a period of a decade (2001-2010) to answer these questions. They also find that computer simulations can enhance traditional instruction, especially with laboratory activities. However, they note that the use of computer simulations has been approached without consideration of the possible impact of teacher support, the lesson scenario, and the computer simulation's place within the curriculum. Well-designed computer simulations provide a model of those elements most relevant to the problem at hand and the learning objective [14]. In addition, "they inform the instructor and the learner of aspects of the real-life system or process that have been simplified" or eliminated.[15]

How can simulations be applied to optics education? The earliest published literature on the application of computer simulation to optics was published by Eylon et al. In 1996.[16] These authors used a simulation called RAY and studied the potential of simulations to improve optics learning in a high school classroom. RAY was a learning environment that included a flexible ray tracing simulation, graphic tools, and task authoring facilities. They showed the importance of three factors to students' understanding of concepts and their ability to use the ray model: the computerized environment (versus written instruction of similar kind); a task design that addresses directly conceptual difficulties in ray tracing; and the explicit reformulation of ideas.

Carnicer et al. [17] at the University of Barcelona developed an educational optical software package for undergraduate students. It consisted of a web based textbook with several applets for illustrating the theory to simplify the teaching tasks in the classroom. The applets were written in the Java language using the Java Network Launching Protocol. These programs could also used as a method for self-learning in an on-line environment.[17] These applets can be downloaded from: http://www.ub.edu/javaoptics/index-en.html. Foley et al. at Mississippi State University created WebTOP, a three-dimensional, web-based, interactive computer graphics system that covered waves and optics. The subject areas include waves, geometrical optics, reflection and refraction, polarization, interference, diffraction, lasers, and scattering. WebTOP was developed with VRML, JAVA, JavaScript and VRML's JavaEAL.[18]. The website for this is: http://webtop.msstate.edu/. There are other optics simulation programs available on the web. Notable amongst them is PHET (http://phet.colorado.edu). Founded in 2002 by Nobel Laureate Carl Wieman, the PhET Interactive Simulations project at the University of Colorado, Boulder makes available for free interactive math and science simulations. The Light and Radiation page (http://phet.colorado.edu/en/simulations/category/physics/light-and-radiation) has over 20 different simulations covering a wide variety of optical phenomena. The programs are well documented and the source code is available. The Wolfram demonstration project, (http://demonstrations.wolfram.com/) boasts of having over 10,000 interactive demonstrations covering a whole gamut of areas in engineering, math, physics, business and social systems, etc. There are over 200 optics demonstrations available as of May 2016. This is a crowd sourced project – any user of Mathematica software can write programs and submit for publication on the website. These demonstrations run from simple to very complex.

More recently, our group, presented a paper on simulations of diffraction using the Python language.[19] The precursor to this was the development of "Python Optics" [20]. Python has numerous advantages, especially for a novice programmer (*vide infra* Section 3).

3 OPTICS SIMULATION WITH PYTHON

3.1 What is Python?

Python (https://www.python.org/) is an interpreted, interactive, object-oriented programming language that is open-source and easy to learn. It provides high level data structures such as list and associative arrays (called dictionaries), dynamic typing and dynamic binding, modules, classes, exceptions, automatic memory management, etc. It has a remarkably simple and elegant syntax that allows experienced programmers in any other language to pick up Python very quickly. Beginners would find the clean syntax and indentation structure easy to learn. It was designed in 1990 by Guido van Rossum. Like many other scripting languages it is free, even for commercial purposes, and it can be run on practically any modern computer. A python program is compiled automatically by the interpreter into platform independent byte code which is then interpreted.

A large number of specialized modules or applications are written in Python. Modules in Python are simply Python files with the ".py" extension, which implement a set of functions or instructions. Modules can be "import"-ed from or in to other modules thereby facilitating the development of complex software. The full list of built-in modules in the Python standard library can be found at https://docs.python.org/3/library/. In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the Python Package Index (https://pypi.python.org/pypi).

3.2 An easy way to install Python and its packages

Anaconda (https://www.continuum.io/downloads) is an easy-to-install free package and environment manager. Python distribution (Python 2.7, 3.4., and/or 3.5), and collection of over 150 open source pre-built and tested scientific and analytic Python packages that include NumPy, Pandas, SciPy, Matplotlib, and IPython, with over 250 more packages can be installed through Anaconda. In addition, Anaconda includes several open source integrated development environments (IDE) such as Jupyter/IPython and Spyder. Anaconda is available for Linux, OS X and Windows.

3.3 Powerful Python packages for teaching and research

Below is a list of modules/packages that are of utmost importance for using Python as a platform for optics simulations:

IPython (https://ipython.org/): It is an advanced Python shell that offers an end-user environment for interactive work. Ipython is a set of tools originally developed to make it easier for scientists to work with Python and data. It allows you to combine interactive Python exploration with prewritten programs and even text and equations for documentation.

Jupyter/IPython Notebook (https://jupyter.org/): It is a web-based graphical notebook interface that lets users execute, store, load, re-execute a sequence of Python commands, and to include explanatory text, images and other media in between. It can be used for data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more. All of these features are available in a single environment. This is a recent and exciting development that has the potential to develop into a tool of great significance, for example, for documenting calculations.

NumPy (http://www.numpy.org/): All Python numerical codes today are based on the NumPy library. The NumPy array is the basic data structure that virtually all libraries understand. NumPy arrays have a rich Python interface, but they also can be readily accessed from C, C++, and Fortran, making it easy to optimize performance bottlenecks or reuse existing libraries that have no knowledge of Python.

SciPy (https://www.scipy.org/): Scipy can be thought of as an extension of NumPy with a large number modules that are optimized for specific scientific calculations. The SciPy community is Python's most compelling assets as a platform for scientific computing. The SciPy community is a well-established and growing group of scientists, engineers, and researchers using, extending, and promoting Python's use for scientific computing, research and education.

Matplotlib (http://matplotlib.org/): Matplotlib is the most widely used library for high-quality plotting, with support for a wide array of 2D and 3D plot types, precise layout control, a built-in LaTeX typesetting engine for label equations, and publication-quality output in all major image formats. Matplotlib tries to make easy things easy and hard things possible. One can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code.

3.4 Examples

In this article, we illustrate the great advantages offered by Python and all the new tools accompanying it, using examples from traditional topics in geometric and physical optics. We start with a simple demonstration of determining the positions of the cardinal planes for a lens system with many refracting surfaces. Then, a simple demonstration of how we can simulate light diffraction produced by slits and circular diaphragms in the approximation of the far field – the so-called Fraunhofer diffraction is provided. For diffraction by rectangular aperture, we choose the simplest case of the diffraction by a single slit because it is a well known problem and our goal here is not to describe the complexity of the physical phenomena. In the case of diffraction by circular apertures, we consider the Rayleigh criteria because it concerns a wide range of applications. Simultaneously, we demonstrate how this numerical environment (Jupyter/IPython) provide teachers with a set of rich tools that help to generate documents for education and outreach programs.

3.4.1 Cardinal planes

We begin with a demonstration of determining the positions of the cardinal planes for an optical system with an arbitrary number of refracting surfaces. In general, one first needs to determine the front vertex power (FVP), back vertex power (BVP) and the nominal equivalent power (NEP) of the system. This could be done by defining functions for do this as shown in Figure 2.

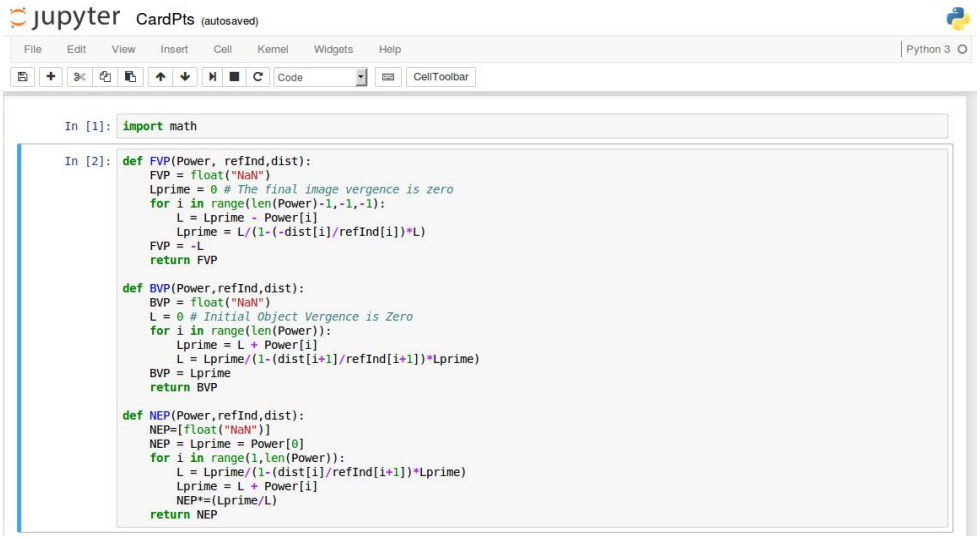

Figure 2: Sample code demonstrating functions definitions in Python. Note the usage of Jupyter. The various boxes with names line 'In [1]:' denote the various 'cells' in the Jupyter Python Notebook. A cell is nothing but a block of code that is executed at one go.

We could use the IPython widgets to get input from the users as shown in Figure 3. With these input values, we can then use the functions as defined in Figure 2 to calculate the various quantities that are of interest to us. This is demonstrated in Figure 4. With the calculated values, one can then create the canvas as demonstrated in Figure 5 that results in a display as shown in Fig 6. As a demonstration we have used the Gullstrand Schematic Eye Model that models the optical portion of the human eye as a system of four refracting surfaces. In this particular demonstration, the Principal Planes are denoted as P1 and P2, the Focal Planes as F1 and F2 and the nodal planes as N1 and N2. Users can change the input values in Figure 3 and rerun the code to see how the positions of the cardinal planes change.

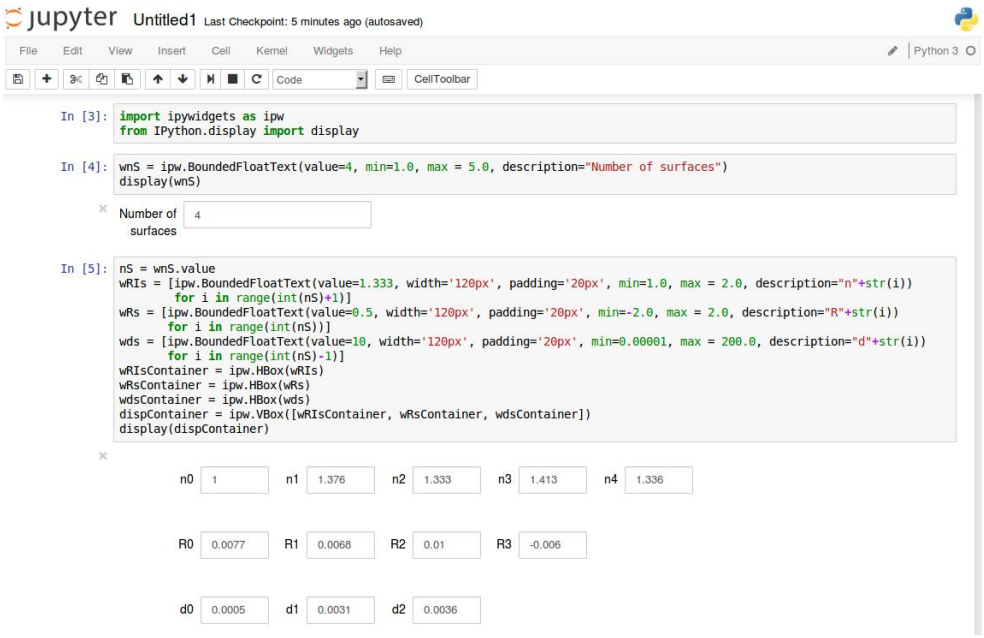

Figure 3: It is easy to get user interaction using the ipywidgets module. Note that we begin with importing the required modules (ipywidgets and Ipython.display in this case). Note that creating the widgets, arranging them and displaying are all done using simple one line commands.

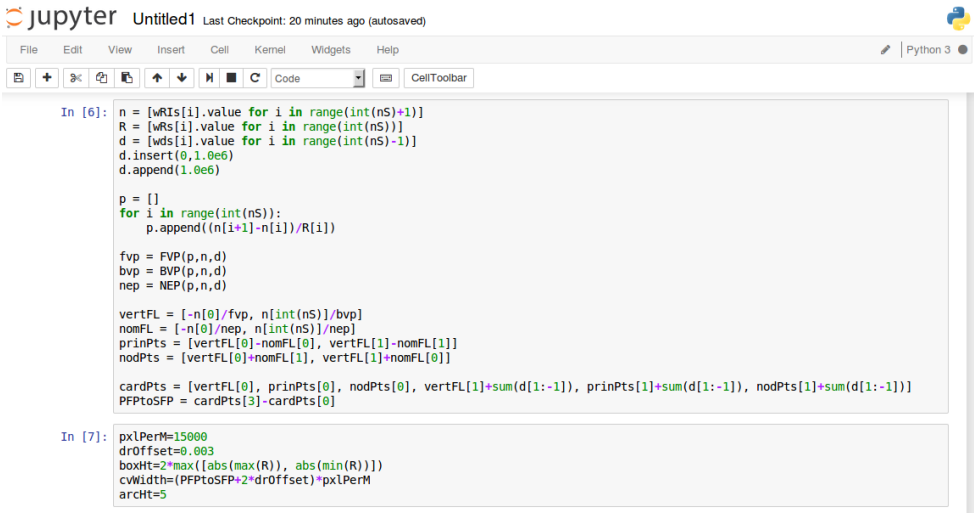

Figure 4: Demonstration of simple commands and function calls in Python in the Jupyter notebook.

jupyter Untitled1 Last Checkpoint: 27 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

In [8]:
```
def arcBoxExtnt(rad,arcHt,pos,pxlPerX):
    out=[pos[0], pos[1]-abs(rad), pos[0]+2*rad, pos[1]+abs(rad)]
    extent = 2.0*abs(math.atan(arcHt/rad)*180.0/math.pi)
    angle=180*(1+int(rad<0))-extent/2.0

    out=[pxlPerX*out[n] for n in range(len(out))]
    out.append(angle)
    out.append(extent)
    return out
```

In [9]:
```
import tkinter as tk
```

In [10]:
```
mW = tk.Tk()
cv=tk.Canvas(mW, height=boxHt*pxlPerM, width=cvWidth, background='white', relief='sunken')
cv.create_line(drOffset/2*pxlPerM,boxHt/2*pxlPerM,cvWidth-drOffset/2*pxlPerM, boxHt/2*pxlPerM)
dDraw = [0.0]
for i in range(1,len(d)-1):
    dDraw.append(d[i])

for i in range(len(R)):
    arcBEx=arcBoxExtnt(R[i],0.005,[drOffset-cardPts[0]+sum(dDraw[:i+1]),boxHt/2],pxlPerM)
    cv.create_arc(arcBEx[0], arcBEx[1], arcBEx[2], arcBEx[3], style='arc', outline='blue',
                  start=arcBEx[4], extent=arcBEx[5])

arcBEx=arcBoxExtnt(-cardPts[3]/2.0,0.01,[drOffset-cardPts[0]+sum(dDraw[:3+1])+vertFL[1],boxHt/2],pxlPerM)
cv.create_arc(arcBEx[0], arcBEx[1], arcBEx[2], arcBEx[3], style='arc', outline='red', start=arcBEx[4],
              extent=arcBEx[5])
namesCards=['F1','P1','N1','F2','P2','N2']
for i in range(len(cardPts)):
    cv.create_line(((drOffset-cardPts[0])+cardPts[i])*pxlPerM, 1*boxHt/5*pxlPerM,
                   ((drOffset-cardPts[0])+cardPts[i])*pxlPerM, 4*boxHt/5*pxlPerM)
    cv.create_text(((drOffset-cardPts[0])+cardPts[i]-0.0005+0.001*(int(i>2)))*pxlPerM,
                   (1.15+3.7*int(i>2))*boxHt/6*pxlPerM, text=namesCards[i], font=('times', 10, 'bold'))

cv.grid(row=0,column=0)
mW.mainloop()
```

Figure 5: Demonstration of code to draw the various surfaces and the cardinal planes.

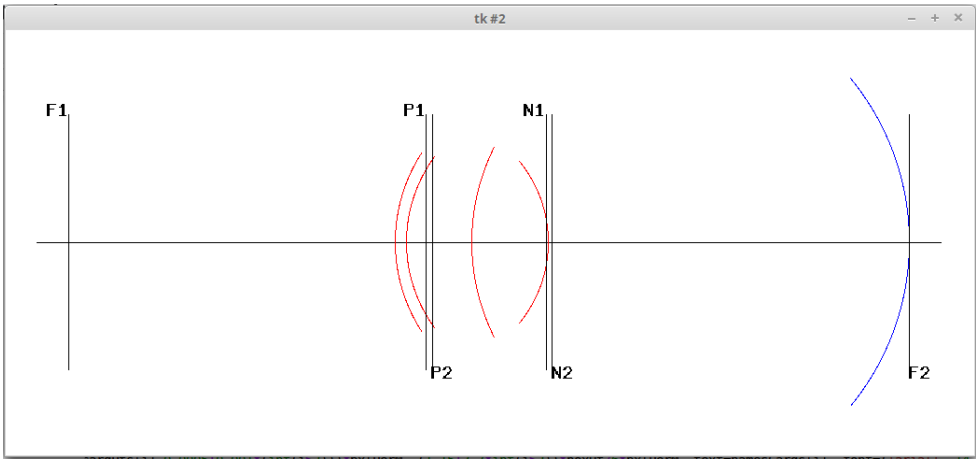

Figure 6: The output of the codes using user input shown in Figures 2-5.

3.4.2 Diffraction by a single slit

The choice of the diffraction by a single slit is very useful because it allows us to simply show how easy it becomes to interactively simulate a well-known function – in this case the sinc function involving a few physical parameters. That is to say that we can reproduce observable phenomena by varying dimensions or distances. Figure 7 below shows a schematic of diffraction of light by single slit indicating the main parameters that influence the intensity distribution observed on a screen.

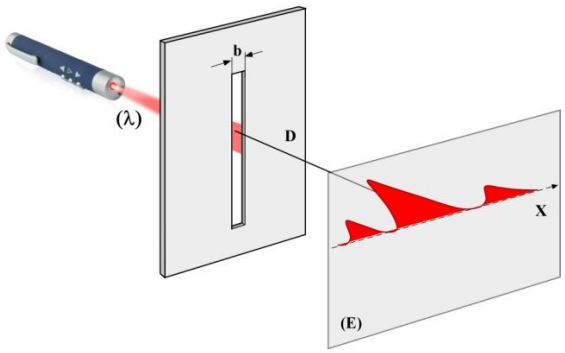

Figure 7: Diffraction obtained by a single slit with a width (d) observed on screen at a distance (D) from the slit.

The expression for the intensity of light observed on the screen (E) along the X direction for a monochromatic light with wavelength (λ) is given by:

$$I(X) = I_0 \mathrm{sinc}\left(\frac{\pi b}{\lambda D} X\right) \quad (1)$$

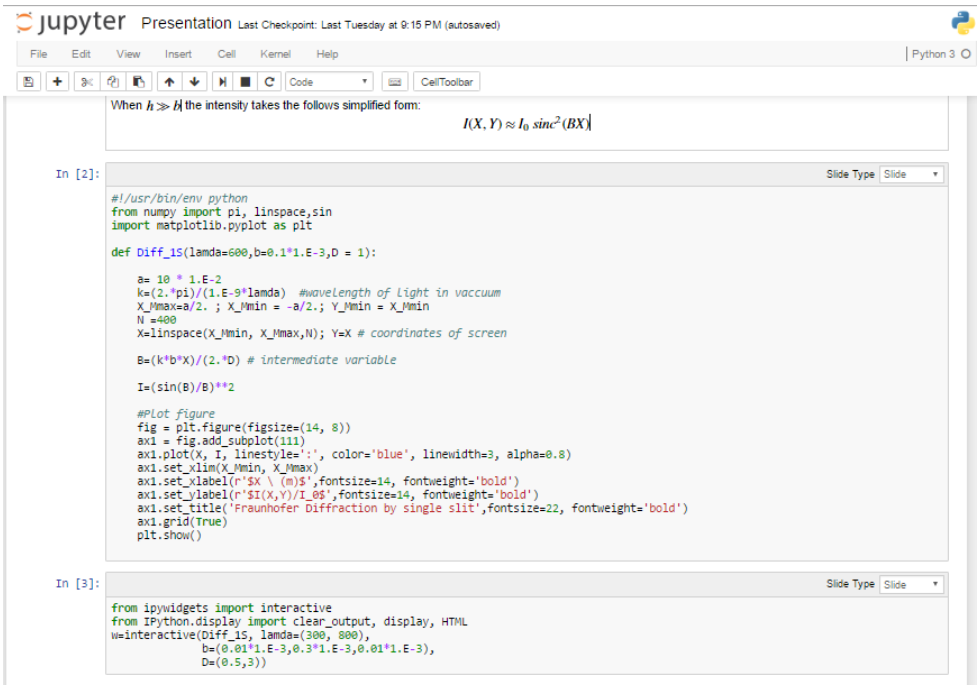

Figure 8: Jupyter/IPython notebook cells showing codes that we need to write and execute in order to have figure 9 as output.

To simulate equation (1), we write a function “Diff_1S(lamda=600,b=0.1*1.E-3,D = 1)”. The default parameters of this

function are: λ = 600 nm, b = 10^{-03} m and D = 1 m. This function generates a plot of the intensity of the diffracted monochromatic light beam as function of 'X' coordinate of the screen (Figure 9). The plot changes if we change the values of the defaults parameters. We can generate a Jupyter/IPython notebook and load this program and run it to generate the plot. We can also use sliders to vary the three parameters (λ, b and D) in order to simulate various configurations we want to test. The program and the output are shown in Figures 8 and 9, respectively. Sliders are imported from the ipywidgets package. IPython widgets are interactive HTML widgets for Jupyter notebooks and the IPython kernel.

3.4.3 Rayleigh criterion

For the second case, we consider the diffraction by a circular diaphragm; we focus on the resolution issue. As for the previous case all the work can be done inside Jupyter/IPython notebook. The three figures 10-12 demonstrate this.

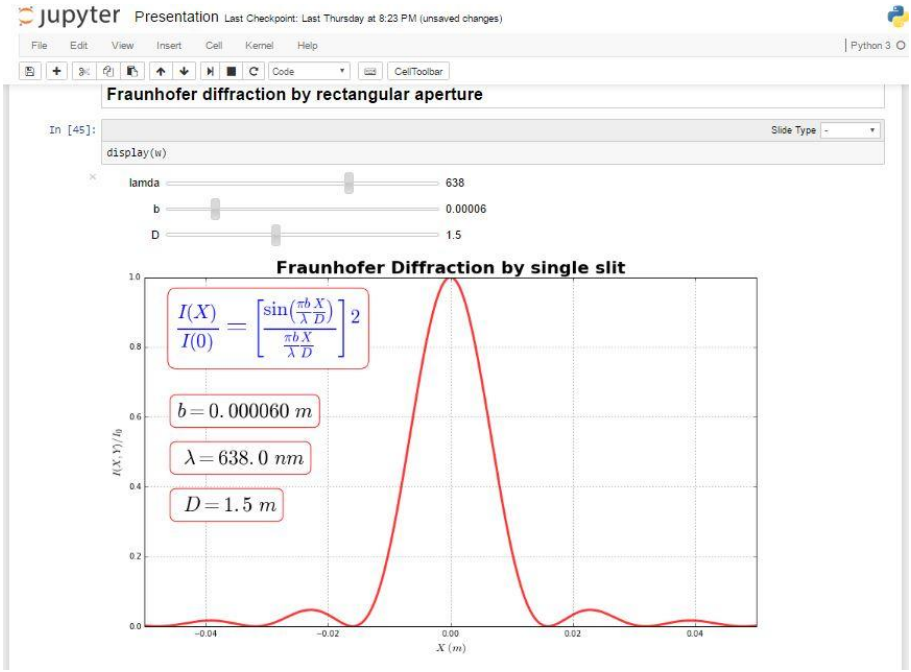

Figure 9: Diffraction pattern we obtained with the sliders associated with the three parameters (λ,b, and D).

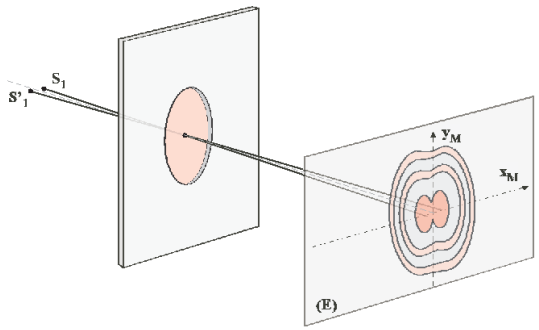

Figure 10. Diffraction by circular diaphragm of radius (r) for two sources separated by a distance d.

The Python program is as as shown in Figure 11 and the the output in Figure 12.

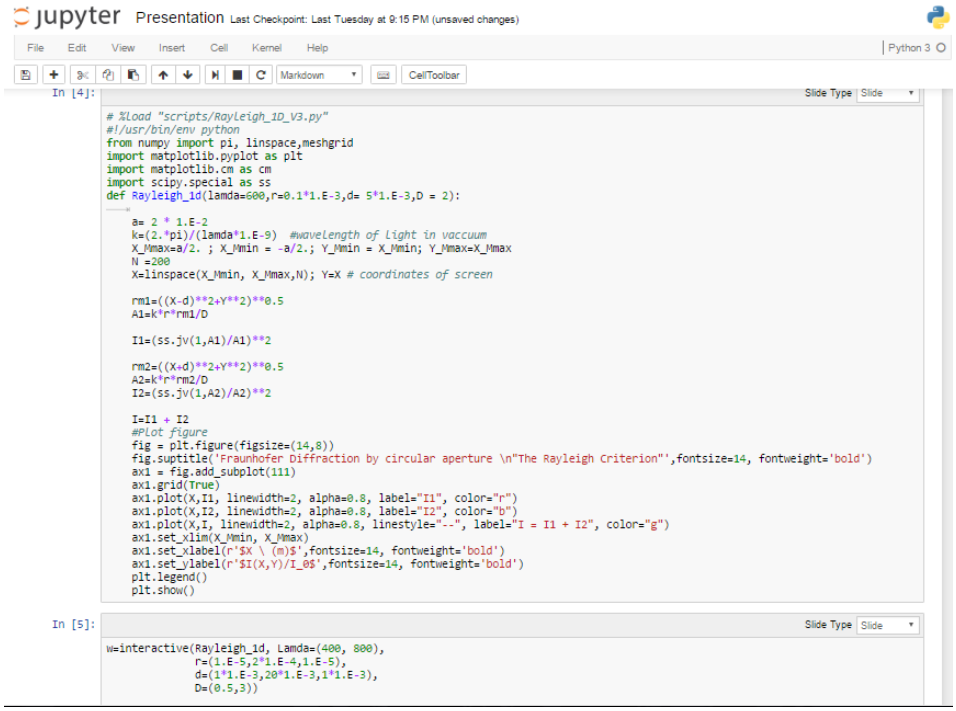

Figure 11: Jupyter/IPython notebook cells showing codes that we need to write and execute in order to have Figure 12 as output.

We can check easily the Rayleigh criterion by dragging the slider associated with (d) or (λ) and noting the values corresponding to the situation where the minimum of the red curve is aligned with the maximum of the blue curve.

4 DISCUSSION

Simulations have a major impact on learning and understanding science concepts. We have described in this paper some uses of the Python programming language to teach optics via simulations. As noted previously, Python and associated programs are open source and free, thus making them very cost effective for use in developing nations or in situations where cost of software is a consideration. In addition, Python is an object-oriented programming language. Known for its easily readable syntax, uncluttered visual layout and extensibility to other programming languages, Python can be used to develop for a variety of uses including the Web, GUI and other applications that need a programmable interface. Python can be introduced in workshops to both teachers and students to learn physics at high school and lower grades. [21] We are currently working on a suite of Python programs to teach introductory optics and these will be published elsewhere.[22]

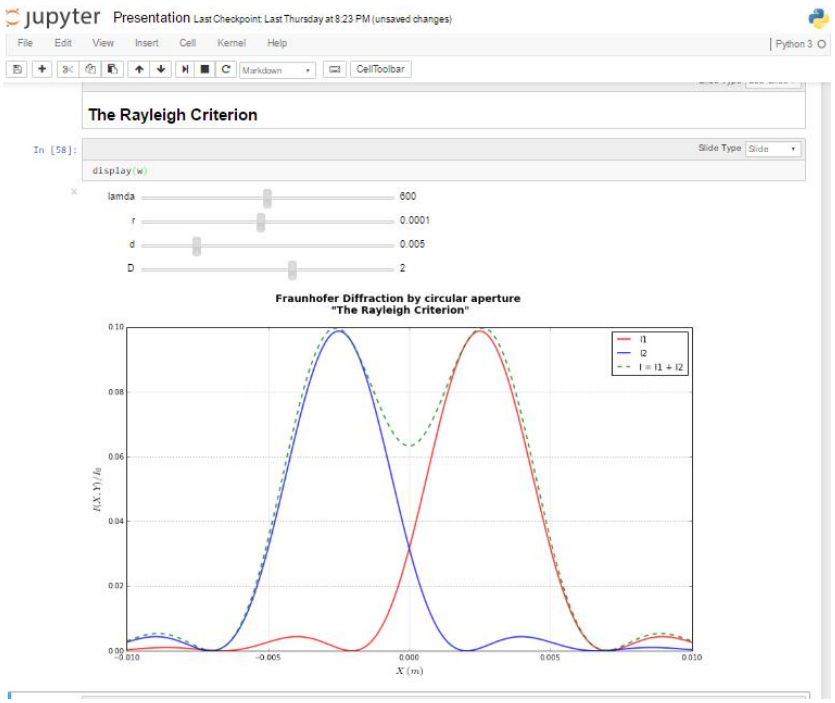

Figure 12. Intensity distribution obtained with the parameter's values indicated on the sliders.

REFERENCES

[1] Laurentius de Voltolina, Liber ethicorum des Henricus de Alemannia; Kupferstichkabinett SMPK, Berlin/Staatliche Museen Preussiischer Kulturbesitz, Min. 1233

[2] Comenius, J.A., "The Great Didactic", https://archive.org/stream/cu31924031053709/cu31924031053709_djvu.txt, (23 May, 2016).

[3] Camus, A., http://www.brainyquote.com/quotes/quotes/a/albertcamu384330.html, (accessed 05/23/2016).

[4] Freeman, S., Eddy, S.L., McDonough, M., Smith, M.K.,Okoroafor, N., Jordt, H. and Wenderoth, M.P., "Active learning increasers student performance in science, engineering and mathematics", Proc. Nat. Acad. Sci. (USA), 111(23):8410-8415, 2014.

[5] Prince, M., "Does active learning work? A review of the research", J. Eng. Education, 93:223-231, 2004.

[6] Lakshminarayanan, V., "Interactive lecture demonstrations, active learning and the ALOP project", SPIE Eco-Photonics: Sustainable design, manufacturing, and engineering work force education for a green future, Proc. SPIE, 8065, DOI: 10.1117/12.589508, 2011.

[7] Bonwell, C.C., and Elson, J.A., "Active learning: creative excitement in the classroom", ASHE-ERIC Higher Education Report No. 1. School of Education and Human development, The George Washington University, WASHINGTON DC, 1991. This report has an extensive bibiliography.

[8] Bishop, J.L., and Verleger, M.A., "The flipped classroom: a survey of the research", Paper ID#6219, 120th ASEE Annual Conference and Exposition, Atlanta GA., https://www.asee.org/public/conferences/20/papers/6219/view, (21 July, 2016).

[9] Lakshminarayanan V. and McBride, A.C., “The use of high technology in STEM education”, Education and Training in Optics and Photonics, ETOP 2015, Proc SPIE, 9793, DOI: 10.1117/12.2223062, 2015.

[10] de Jong, T., and van Joolingen, W. R., Scientific discovery learning with computer simulations of conceptual domains. Review of Educational Research, 68(2), 179–201, 1998.

[11] Bell, R., and Smetana, L., Using Computer Simulations to Enhance Science Teaching and Learning, 2011, static.nsta.org/files/pb217x-3.pdf, (21 July, 2016).

[12] Flick, L., and Bell, R., Preparing tomorrow's science teachers to use technology: Guidelines for science educators. Contemporary Issues in Technology and Teacher Education [Online serial], 1(1). Retrieved from http://www.citejournal.org/volume-1/issue-1-00/science/preparing-tomorrows-science-teachers-to-use-technology-guidelines-for-science-educators/, (21 July, 2016).

[13] Ruten, N., Van Joolingen, R.W., and van der Ween, J.T., “The learning effects of computer simulations in science education” Computers & Education 58:136–153, 2012.

[14] D’Angelo, C., Rutstein, D., Harris, C., Bernard, R., Borokhovski, E., and Haertel, G., Simulations for STEM Learning: Systematic Review and Meta-Analysis, SRI International, Menlo Park, CA. 2014, https://www.sri.com/work/publications/simulations-stem-learning-systematic-review-and-meta-analysis-full-report, (21 July, 2016).

[15] Sternberg, R. J. Cognitive Psychology. (2nd ed.)., Harcourt Brace Johanovich, Fort Worth, Texas, page 17, 1999.

[16] 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.

[17] Carnicer, A., Andilla, J., Ferre, J., Ferre, J., Juvells, I, Martin-Badosa, E., de F. Moneo, J.R., Pleguezuelos, E., Tudela, R., and Vallmitjana, S., 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: 10.1117/12.22077251.

[18] Foley, J.T., Zoughi, M., Herring, S.D., Morris, M., Gilbert, P.J., and 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.

[19] Ammar, A., Burman, R., Ghalila, H., Ben Lakhdar, Z., Varadharajan, 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.

[20] Lakshminarayanan, V. and Burman, R. “Optics Tutorials with Python”, Tech. Report, Univ. of Waterloo, 2015, doi:10.13140/RG.2.1.2940.2325, 2015.

[21] Allein, R., “How to use Python to teach high school physics”, Wired, 12 July, 2016, http://www.wired.com/2016/07/use-python-teach-high-school-physics/ (21 July, 2016).

[22] Lakshminarayanan ,V., Ghallila, H., Varadharajan, L.S. Ammar, A. [Understanding Optics With Python], CRC Press/Taylor and Francis, Boca Raton, FL., in preparation, 2015.