Nathaniel Rose

August 30th, 2018

Gaussian Optics Tool and its Applications to the Wendelstein 7-X Stellarator Phase Contrast Imaging Diagnostic

I - Intro

Over the summer of 2018 software was developed, primarily to be used for simple guassian optics calculations. Inspiration for this tool came from the complex design of a Phase Contrast Imaging (PCI) diagnostic system. Due to the large number of lenses used in the system, it was recognized that a tool which could quickly model lens arrays and return information regarding radius of curvature, and beam waist would be useful. The PCI diagnostic was developed for use at the Max Planck Institut für Plasmaphysik. This institute houses a stellarator type fusion device known as the Wendelstein 7-X (W7-X). The goal of W7-X is to analyze how well a large scale stellarator "(can achieve) … high heating-power and high confinement in steady-state operation" [1]. The PCI diagnostic has been developed to analyze turbulence and waves generated within the plasmas produced by the fusion device. Understanding turbulence "... is important for finding operational scenarios that can lead to improved performance at fusion-relevant temperatures and densities." [3]. This diagnostic uses a 100 Watt CO_2 laser to make measurements in the plasma, as well as a Helium Neon laser which is used in the alignment process. Both beams travel through a series of lenses and mirrors which direct them through a large duct that passes the beams through the stellarator device and out to a receiving table, where the resulting beams are passed through another series of lenses and mirrors to a detector. The primary goal for developing the software was to create a tool with an intuitive graphical user interface, with the intention of simplifying the process of designing such lens and mirror arrays.

II - Goals

It was determined early in the project that the top priorities would be to develop a back-end that can calculate beam waist and radius of curvature of an ideal scenario (i.e. no aberrations from the spherical optics) and to create a graphical user interface (GUI) that is intuitive for the user to both input data and display results. It was also recognized that it would be beneficial for the user to have the option of working with both thin and thick lens approximations. Another design criterion was to develop a piece of software that could handle at least two light sources at a time. This constraint was inspired by the PCI diagnostic, the intention being that the software would be able to model the arrays that guide the HeNe and the CO_2 beams. It is often the case that the two sources will share optical elements, so it was recognized that a desirable feature might be to have software that allows the information about shared optical elements be duplicated automatically to make the process of inputting data more streamlined. One final criterion was that the software be designed in such a way that it is possible to conduct simple sensitivity studies.

III - Development

This software was developed using the Python scripting language in combination with the KIVY GUI library. Python is an object-oriented and procedural programming language that comes with several libraries that are useful for scientific applications, such as Numpy, Pandas, and Matplotlib. These libraries give the developer access to tools that can simplify code relating to calculations, data handling and organization, and plotting.

The KIVY library was designed specifically for creating GUI applications. There are several other choices of GUI libraries for python, such as the tkinter and qt libraries, however, KIVY was chosen for its cross-platform compatibility. KIVY is designed to work with most popular PC operating systems such as Windows, mac OS, Linux, etc. as well as with several mobile platforms such as iOS, and android. Similar to the cascading style sheets (CSS) used by the qt library, KIVY allows all styling of the software to be handled in separate (.kv) files, while the logic is handled primarily within the python (.py) files. This separation of logic and style helps to improve the readability of the code, it also serves to make it simpler for changes within the user interface to be made without disturbing the functionality of the program.

IV - User Interface

The user interface is currently comprised of three elements; a data table, a one-dimensional representation of the lens array, and a window containing two plots. The data table has been designed as a separate pop-up window which is composed of a tabbed panel (figure 1). The first tab, labeled 'Src 1' holds all data regarding optical elements which effect the beam coming from the first source, and the tab labeled 'Src 2' holds data regarding the elements that effect the beam from source two. Presently, the software can handle only two light sources at a time. Within each tab the user can add, remove, or modify optical elements in the lens array associated with that source. The user can also pull data from the source 1 data table into the source 2 data table by adding a new element and selecting 'Combine' as the surface type. This will then allow the user to assign a range of elements to be shared (example: 103 – 107 would grab the third through the seventh elements from data table 1). Once the user has finished inputting data they can close the data table pop-up and use the 'Update' button on the main window to update both the one-dimensional lens array and the two plots. The one dimensional lens array is a visual representation of the position of the lenses, as well as the beam waist as it passes through the optical elements and free-space (figure 2). The two plots above the lens array diagram correspond to the radius of curvature and the beam waist as they travel through space (figure 3). These plots can be viewed together or individually. The user may also select which beam (source 1 or source 2) they wish to plot, or plot both simultaneously. These plots are generated using the Matplotlib library, however additional features have been developed to allow the user to 'pan' using a simple mouse click and to zoom vertically using the '+/-' icons on the side.

V - Back-end

The back-end of the software has been developed to take the user defined information, store this information into a dataframe using the Pandas library. Once the user clicks the 'Update' button on the user interface, this dataframe is then passed to a script which will check to see if there user is asking to copy any information from data table 1 to data table 2. If such a request is detected, the script will automatically populate the second dataframe with the requested information from the first. This is done by first splitting each column of the dataframe into separate arrays. These arrays can then be modified using either an append or insert function built into the default python library. Once the modifications have been made, the new arrays are used to replace the columns in the second dataframe. After this script has run, the software passes the updated information to a script that calculates the beam waist and radius of curvature at every point in space between the first element and the last, using at a step size of approximately 10µm. These values are stored into arrays, along with an array holding position information, which are then returned at the end of the script. These arrays are then used when generating the radius and width plots, and the lens-diagram.

VI - Optics Calculations

The calculations done by the software consist of a few simple linear matrix equations [1]. Primarily the software is utilizing the complex beam parameter, expressed as $\frac{1}{q} = \frac{1}{R} - \frac{i\lambda}{\pi n w^2}$, where *q* is the complex beam parameter, *R* is the radius of curvature of the beam, *w* is the beam waist. To analyze what is happening at each surface, a linear transformation in the form of $\begin{pmatrix} q \\ 1 \end{pmatrix} = k \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} q_i \\ 1 \end{pmatrix}$ is used, where k is a normalization constant and the ABCD matrix is dependent on the surface that the beam encounters.

1. All subsequent equations and matrices were found in the Optics textbook. [2]

A. Free space propagation matrix. $\begin{pmatrix} 1 & d \\ 0 & 1 \end{pmatrix}$

The free space matrix depends only on the distance travelled through the medium. The medium is described by the index of refraction (n) which is specified in the complex beam parameter equation.

B. Thin lens and mirror matrix. $\begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix}$

The thin lens and mirror matrix assumes the lens to be infinitely thin and so treats the lens as a single surface and is dependent only on the focal length.

C. Thick lens matrix. $\begin{pmatrix} 1 & 0 \\ \frac{n_2-n_1}{R_2 n_1} & \frac{n_2}{n_1} \end{pmatrix}\begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \frac{n_1-n_2}{R_1 n_2} & \frac{n_1}{n_2} \end{pmatrix}$

Finally, the thick lens matrix is a product of two surface matrices and a free space propagation matrix (where *t* is the lens thickness and can be treated the same as *d*). The thick lens ABCD matrix considers the space between the two surfaces and so is dependent on both the index of refraction inside and outside of the lens. It is also dependent on the radius of curvature of both surfaces.

VII - Test Cases

The following is a description of the results of two separate test cases. In the first test case, two identical lens arrays were constructed, the only difference being that in one array a thin lens was used and in the other, a thick lens was used (figure 3). It is clear from the resulting plot (figure 4) that there is a deviation in the beam waist between the two lenses. This result is expected since in the thick lens approximation, the beam is propagating for a short period of time through the lens, at an index of refraction larger than air. The second test case involves checking for sensitivity in a three-lens telescope. In this case two identical arrays, both using thin lens approximations, were constructed (figure 5). Then, in one array the displacement of the central lens was deviated a small amount (approximately 5 cm).

The resulting plots (figure 6) show that varying the lens position by this amount does cause a possibly significant deviation in the beam waist.

VIII - Conclusion

This project not only demonstrates the practicality and usefulness of a piece of software such as this, but also showcases the power of the KIVY library. In a span of approximately 6 weeks it was possible to both learn the .kv language and develop a functioning piece of software. Currently, the biggest strength of this software is simplicity, both in the way that data is input and the way that the results are displayed and can be navigated. Currently, the largest weakness is that no error handling exists in the software.

Going forward there are plans to improve the overall aesthetic and functionality of the user interface as well as add some additional features, including error handling. For future versions there are plans to give the user the ability to get more specific information from the one-dimensional lens diagram by simply moving the mouse over a specific part of the diagram, and having the position, beam waist and radius of curvature at that point in the array displayed on the screen. There are also plans to implement dimensionally realistic lenses into the diagram, that will allow the user to see if the beam waist has become too large to fit through a lens. There are also plans to add the capability of handling non-linear solutions as well as give the user the ability to model lens arrays in two-dimensions.

The progress made thus far has clearly demonstrated the feasibility of the ambitions held for this project and gives a solid foundation from which the software can continue to grow and develop. There are plans to make the software available and open source soon, which will almost certainly foster continued growth and improvements.

Citations:

1) H.-S. Bosch *et al.*, Nucl. Fus. **57**, 116015 (2017).

2) Hecht, E. (2017) *Optics.* Garden City, New York: Pearson.

3) E.M. Edlund *et al*., Rev. Sci. Inst. **89**, 10E105 (2018).