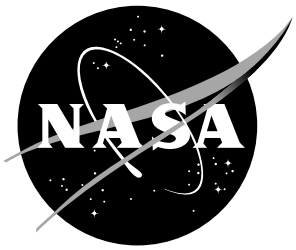
NASA/TM-2020-5005521



LaRC SmartLab Apps For Instrument Control And Data Processing: Optical Micrometer Data Visualizer

Godfrey Sauti
NASA Langley Research Center, Hampton, Virginia

Benjamin D. Jensen
NASA Langley Research Center, Hampton, Virginia

Mathew L. Bartgis
Christopher Newport University, Newport News, Virginia

James D. Tobin
Christopher Newport University, Newport News, Virginia

Emilie J. Siochi
NASA Langley Research Center, Hampton, Virginia

August 2020

NASA STI Program. . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collection of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

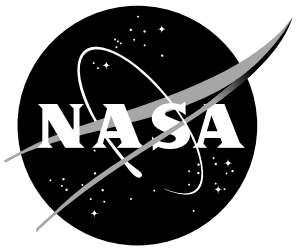
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.
- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI Program, see the following:

- Access the NASA STI program home page at `http://www.sti.nasa.gov`
- E-mail your question to `help@sti.nasa.gov`
- Phone the NASA STI Information Desk at 757-864-9658
- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM-2020-5005521



LaRC SmartLab Apps For Instrument Control And Data Processing: Optical Micrometer Data Visualizer

Godfrey Sauti
NASA Langley Research Center, Hampton, Virginia

Benjamin D. Jensen
NASA Langley Research Center, Hampton, Virginia

Mathew L. Bartgis
Christopher Newport University, Newport News, Virginia

James D. Tobin
Christopher Newport University, Newport News, Virginia

Emilie J. Siochi
NASA Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

August 2020

Acknowledgments

Jae-Woo Kim for providing test data.

The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

The **LaRC SmartLab** applications are a series of software tools to greatly enhance researcher efficiency by streamlining and automating workflows. Python scripts and applications are increasingly being used in scientific workflows, including for instrument control and data processing. Interactive Python scripting environments such as JupyterLab provide powerful tools for using Python. In some use cases, the development of standalone applications with dedicated graphical user interfaces (GUIs) can enhance the utility of the code and open it up to more users, including non-programmers. Here, we describe a GUI based optical micrometer data visualization application developed as part of the **LaRC SmartLab** project. We highlight its use in visualizing experimental data and briefly discuss its implementation to give pointers to programmers who wish develop work based on this application's or similar code.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

1 Introduction **5**

2 Graphical User Interface **6**
2.1 The controls 7
2.2 About 11

3 Getting Started **12**
3.1 Guiding the user 12
3.2 Loading data 14
3.3 Checking data 18
3.4 Plot options 19
3.5 Surface colormaps 20

4 Data Visualization Examples **21**
4.1 Visualizing experimental data 21
4.2 Visualizing synthetic data 24
4.2.1 Ellipses with varying widths & heights 24
4.2.2 Ellipses with constant widths & heights 27
4.2.3 Offsets 28

5 Anatomy of the Plot **31**
5.1 Ellipses of varying diameters 31
5.2 Circles and ellipses of constant diameter 34
5.3 Offsets 36
5.4 Experimental data 38

6 Implementation **39**
6.1 Top level 39
6.2 Application specific supporting modules 40
6.2.1 lslmdvbase 40
6.2.2 lslmdvcontrols 41
6.2.3 lslmdvplots 42
6.3 Generating Microsoft Windows Installers 44

References **45**

A Microsoft Windows Version of the Application **46**

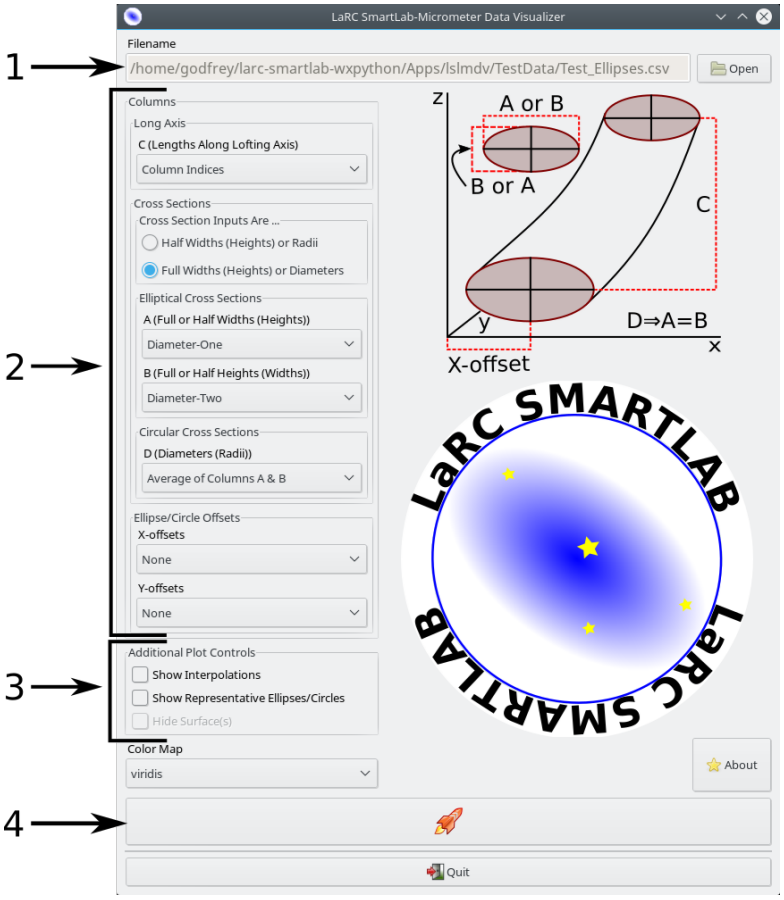
B Examples Using Different Colormaps **50**

C Video Examples **52**

lslmdv Quick Start Guide

LaRC SmartLab Team*

August 4, 2020



1. Select file with the data to be visualized (data must be comma separated variables e.g. select a "*.csv" file).
2. Select data columns
3. Configure the plots to be shown
4. Visualize the data in the selected file and columns.

*POC: godfrey.sauti-1@nasa.gov

1 Introduction

The **LaRC SmartLab**  project develops tools to assist the researcher by eliminating time intensive data manipulation processes as well as providing new ways to interact with and gain insight from the data. Data visualization is an important step in the research workflow. The **Micrometer Data Visualizer** application provides the user with visualization of optical micrometer data in three dimensions (3D). For a tubular object, this data will consist of a distance corresponding to the micrometer's location along the tube's long axis and the diameter (more generally ellipse width and height) along each of the (usually orthogonal) axes that the micrometer measures. Using this data, it is possible to construct (approximate) representations of the 3D structure of the object that is being analyzed. These representations are a quick aid to the researcher in the interpretation of the data.

We describe the application user interface in Section 2 and set up of the data visualization in Section 3. In Section 4 we provide some examples of using the application to visualize experimental data as well as synthetic data. The later helps to further familiarize the user with the features and capabilities of the application. It should be noted here, and will be discussed in detail later, that the generation of the 3D surfaces to represent data necessarily requires making some interpolations. While this will work in most cases, there are situations where the interpolations will introduce artifacts. To help the user distinguish these cases, the application provides options (Section 3.4) that show how the 3D renderings are being generated, in addition to the 3D surface plots. The generation of the surfaces will be discussed in detail in Section 5.

For the interested developer, we briefly outline the implementation of the application in Section 6. This outline is not meant to be a full listing of the program code, but is rather aimed at providing pointers to the tools that are being used.

In addition to the multi-platform Python source code that is used for the examples in most of this document, an installer for the Microsoft (MS) Windows operating system is also available. In Appendix A we provide some screenshots from an installation of the application running on MS Windows 10. We show some examples of some of the many colormaps that are available for the surfaces that are generated in Appendix B. In Appendix C we provide information on videos with examples of the application in action that are available from the authors. The videos include additional features that are not described in this document, such as the access to more plot options including more colormaps.

2 Graphical User Interface

Here, we introduce features of the **Micrometer Data Visualizer** application's graphical user interface and describe the functions of various controls and indicators. Figure 1 shows the main window of the application as it appears on startup. To guide the user with setting up of their visualization, some controls are disabled until suitable data have been loaded (details in Section 3).

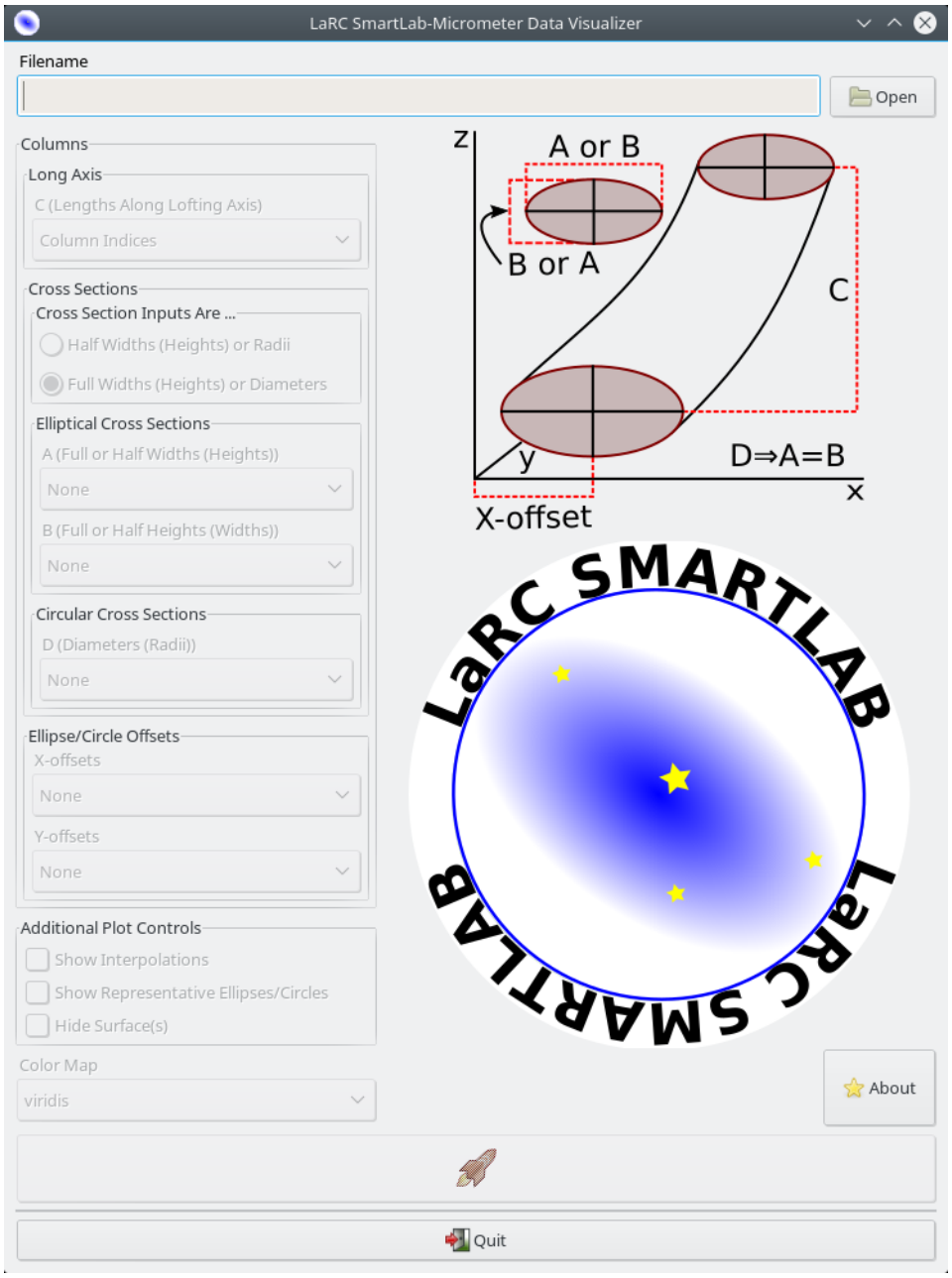


Figure 1: The main window of the application as it appears on startup.

2.1 The controls

Figure 2 shows the main window of the **Micrometer Data Visualizer** application with key control groups labeled.

The functions of the controls shown in Figure 2 are briefly described below.

1. **Filename:** Input for choosing the file with data to be visualized
2. **Long axis column selector:** Selector for the data column to be used as the long (lofting) axis
3. **Cross section type:** Whether the cross section data in the file are to be read as full ellipse widths/heights (diameters for circles) or as half the ellipse widths/heights (radii for circles)
4. **Ellipse widths/heights:** Selectors for the columns to use for the ellipse widths and heights
5. **Diameter column:** Selector for the column to use for circular cross section diameters (radii)
6. **Offset columns selectors:** Selector for the columns to use as offsets to the ellipses/circles
7. **Additional plots:** Selectors of additional plots showing the generation of the 3D visualizations
8. **Colormap:** Selector of the colormap to be used for the rendered surfaces
9. **About:** Button to obtain information about the application
10. **Execute:** Button to render the visualization
11. **Quit:** Button to exit the application

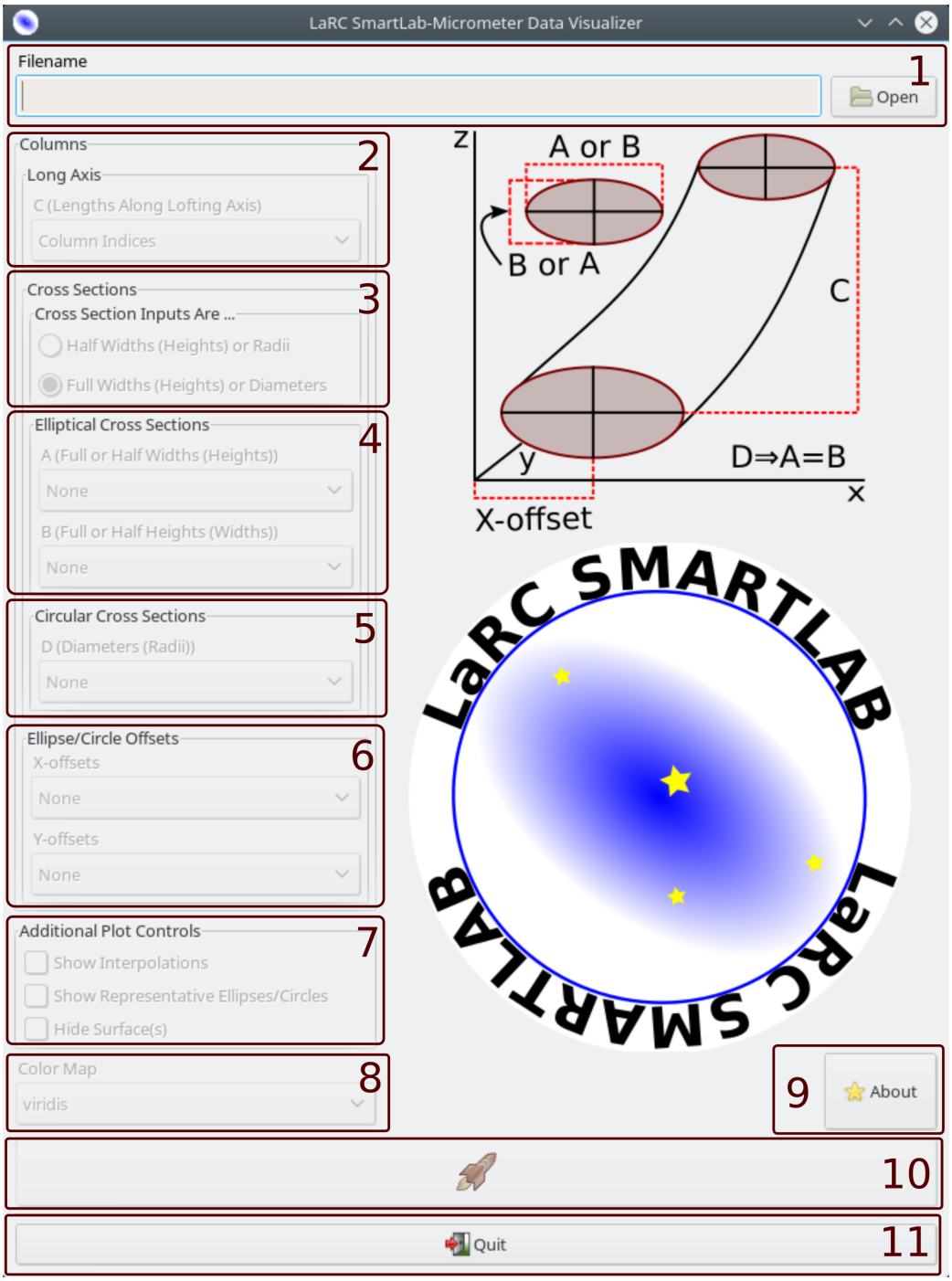


Figure 2: The main window of the **Micrometer Data Visualizer** application with the controls labelled for ease of description.

When visualizing data, the main application will spawn (multiple) child windows to display the plots as shown in Figure 3. How many and which windows are spawned depends on the plot settings (Section 3.4). These child windows can be resized (Figure 4) to better display the data. The child windows have controls for plot manipulation including saving the graphs as image files (Figure 5).

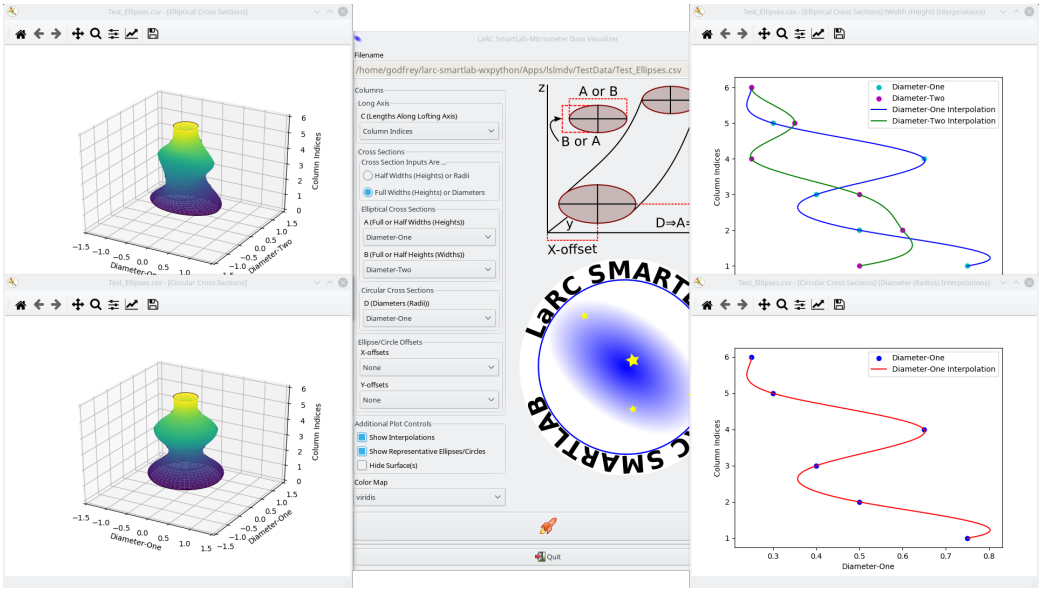


Figure 3: The main window with child display windows.

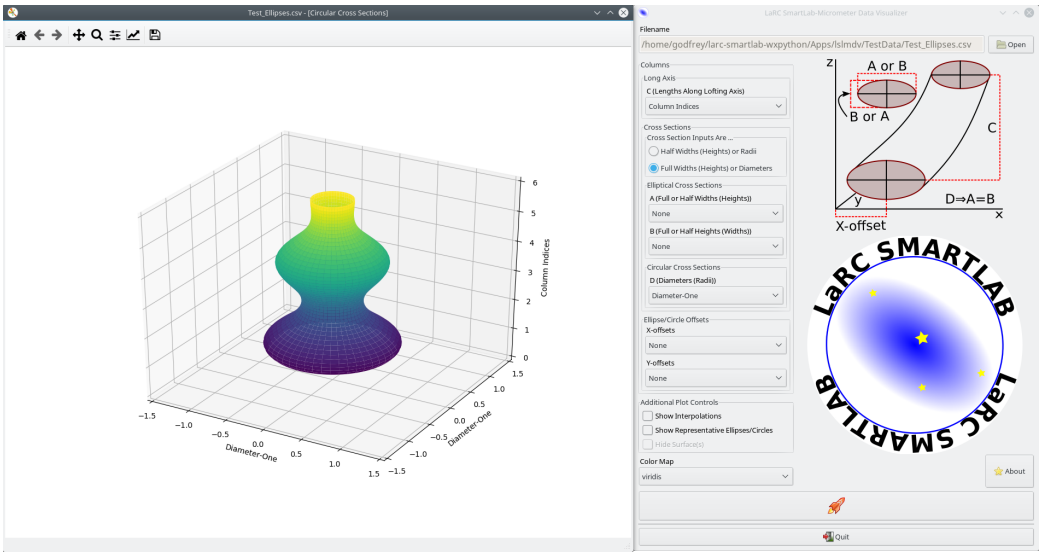


Figure 4: A child window can be resized to better display the data.

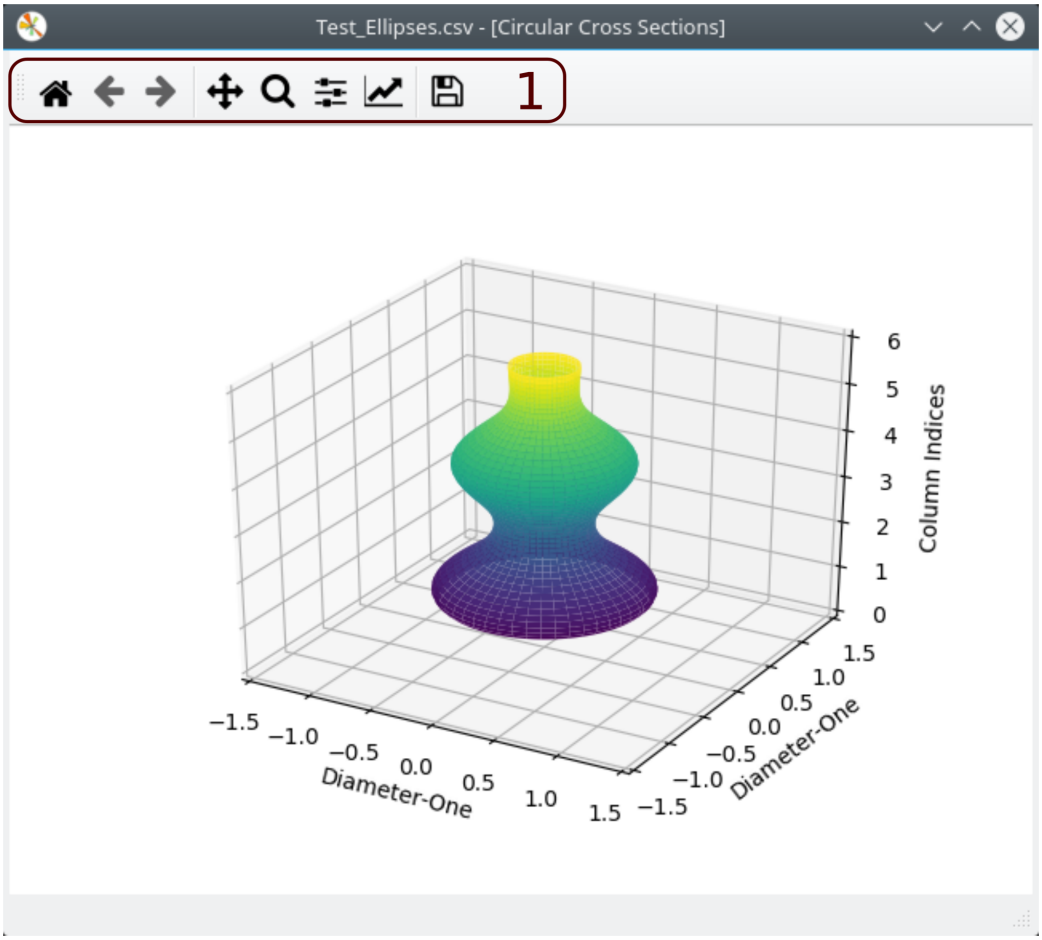


Figure 5: The child windows include controls (1) for manipulating and saving the plots.

2.2 About

Figure 6 shows dialog boxes with information about the application. The information displayed is read from a configuration file in the application directory. The generation and rendering of these dialog boxes is enabled by library modules called up by the main application (Section 6).

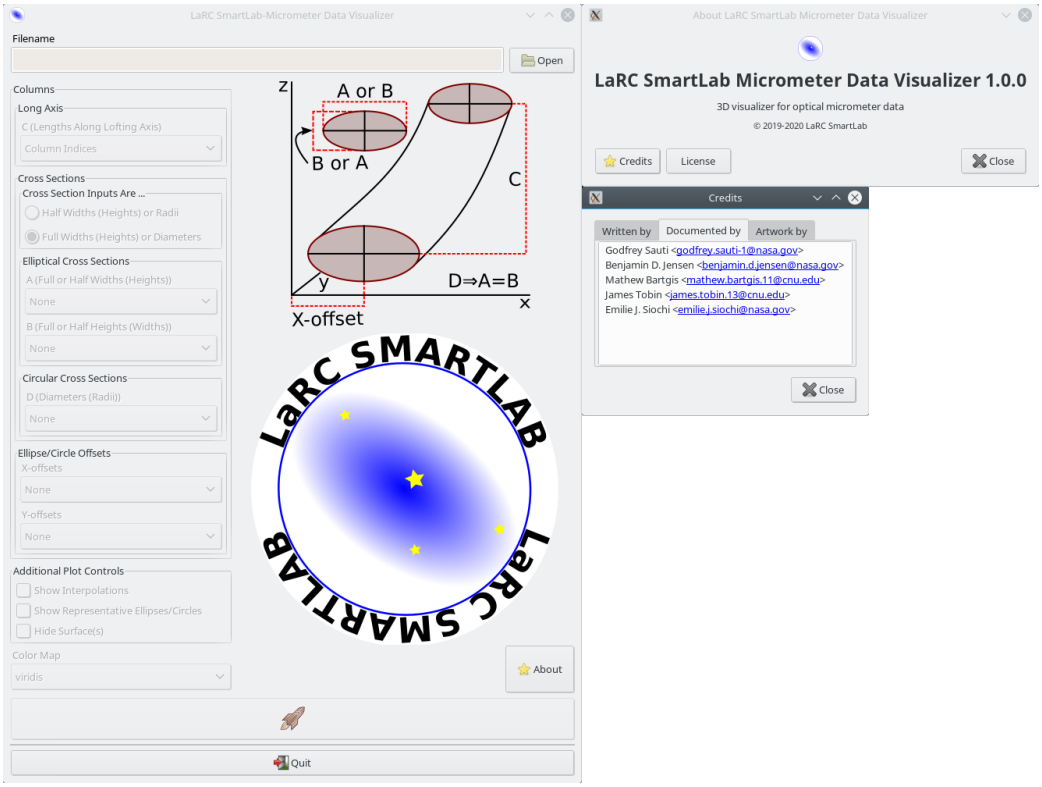


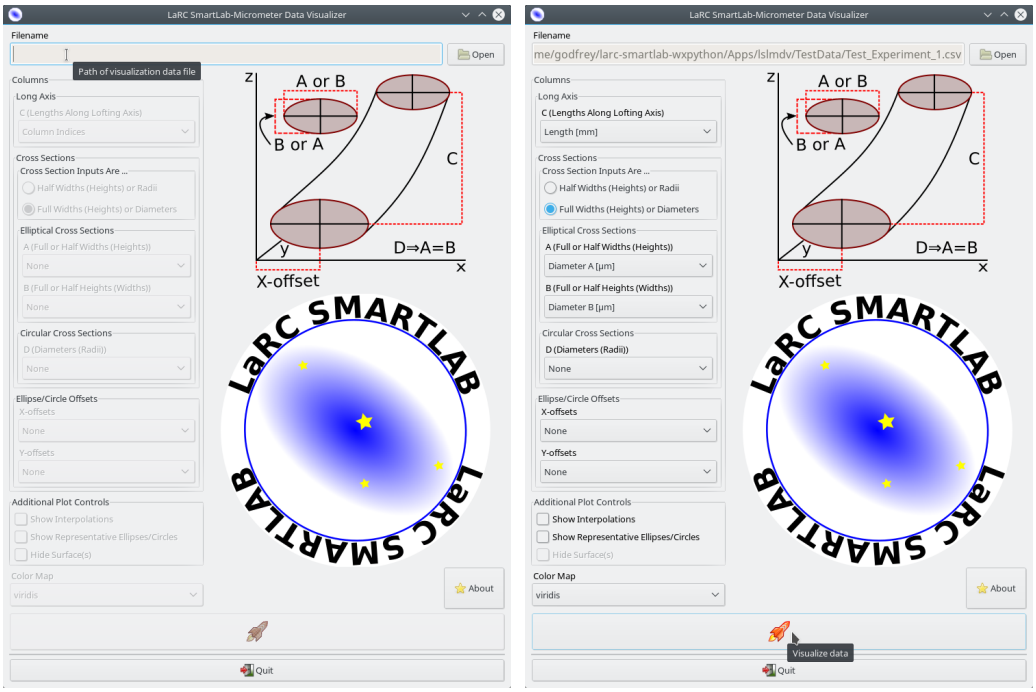
Figure 6: Dialog boxes showing some information about the application.

3 Getting Started

In this section, we discuss some features of the application as well as the process of setting up data visualizations.

3.1 Guiding the user

The application uses three main features to guide the user through setting up and displaying the visualization. The first are tooltips that appear when the user hovers over a control. Examples of these tooltips, which are very brief descriptions of the functions of the controls, are shown in Figure 7.

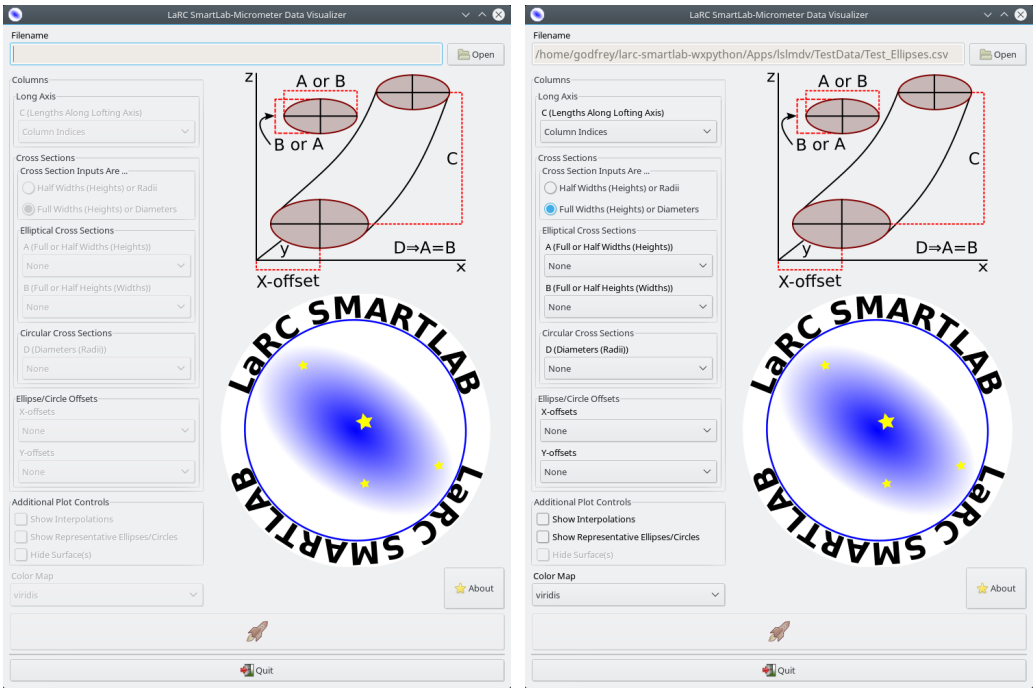


(a) File (b) Visualize data

Figure 7: Tooltips provide some information on the application controls.

The second method, for guiding the user, is the enabling and disabling of various controls depending on what data has been loaded as well as plot selections that the user may have made. Examples of controls in disabled and enabled states are shown in Figure 8. The enabling and disabling of the controls, by the application, gives the user visual cues as to what they may need to do next as they are setting up the data display. Disabling of some controls also assists in making sure that the user does not attempt to render visualizations for which there is insufficient data or which are not compatible with the settings they have selected.

The third method for assisting the user is dialog boxes that will be displayed when specific messages need to be communicated to them by the application. An example of the dialog boxes can be seen in Section 3.3 where the source of the message is also discussed in more detail.



(a) No file selected (b) No columns selected

Figure 8: Controls for setting up the visualization are enabled and disabled depending on the data loaded as well as the currently selected settings.

3.2 Loading data

The **Micrometer Data Visualizer** will load and visualize data from a single file at a time. Figure 9 shows the file selection dialog. The application looks for comma separated variables (**.csv*) files.

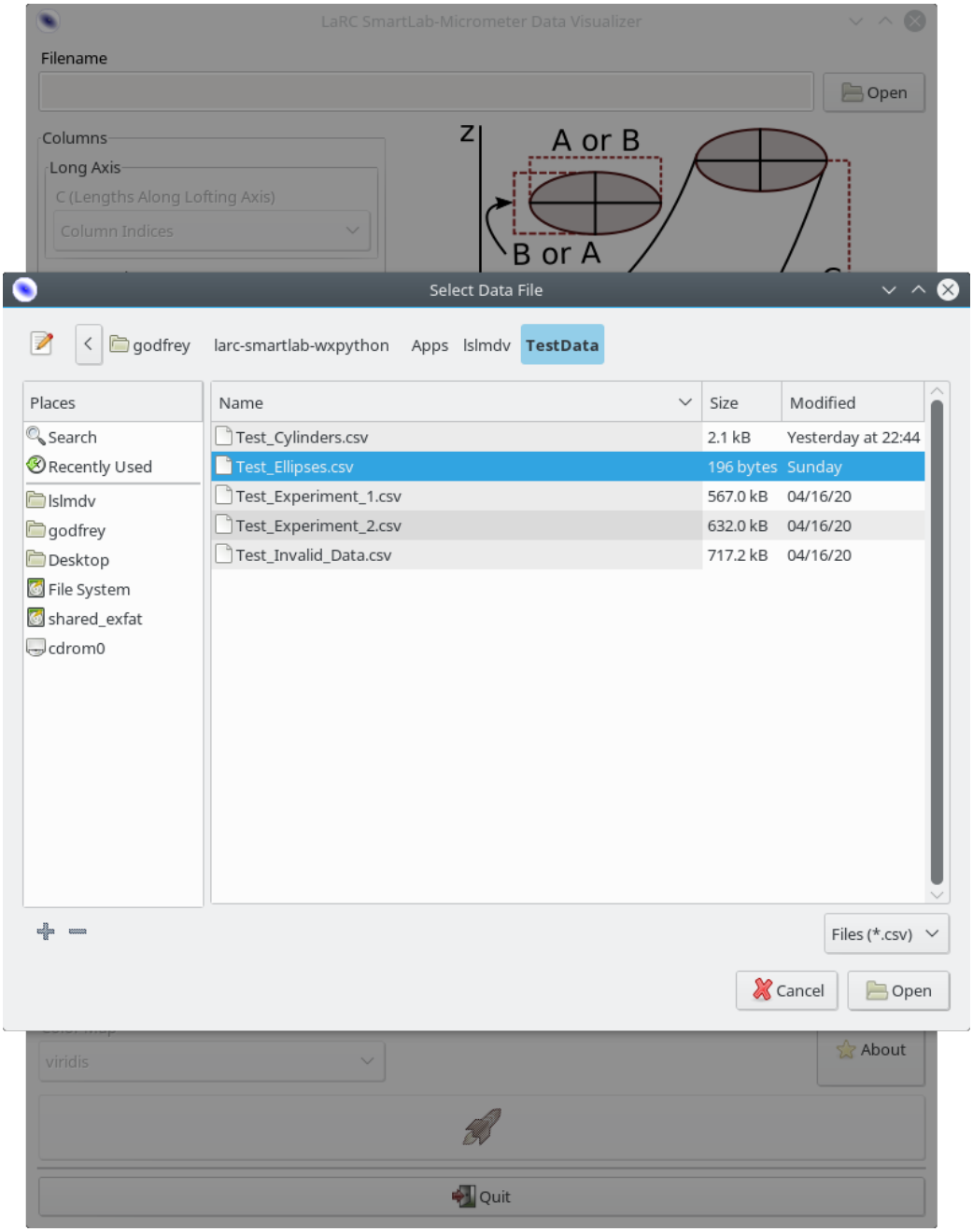


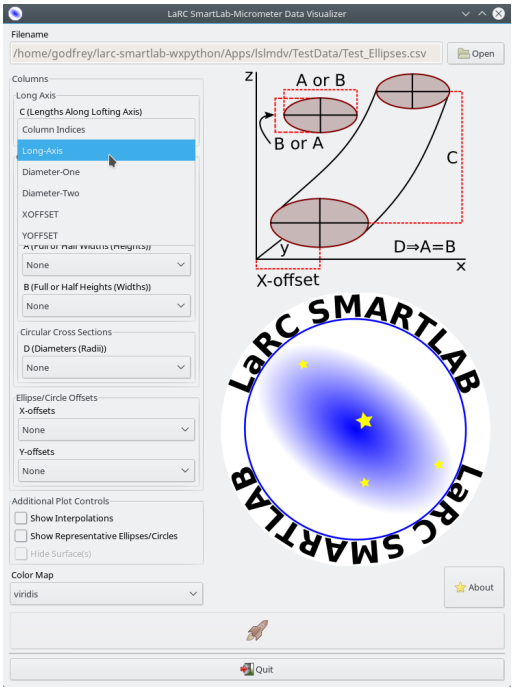
Figure 9: Selecting a **.csv* data file for visualization.

Once a file has been selected, the application will read the headers to find the column names. These names are used to populate the column selectors on the front panel. It is only after the column selectors have been populated that those controls are enabled. The user can then step through selecting the columns of data to display. Other controls which depend on valid selections for the columns (i.e. not "None") will remain disabled. For any visualization to be displayed, the selection for the "Long/Lofting Axis" (Column ***C***) must be a value that is not "None." If data with elliptical cross sections is to be displayed then the selections for both axes ***A*** and ***B*** must also be values that are also not "None." If both ***A*** and ***B*** have valid values and the circular cross section ***D*** also has a valid selection then two 3D visualizations are generated. One will have elliptical cross sections based on ***A*** and ***B*** while the other has circular cross sections based on ***D***. If ***D*** has a valid selection and both ***A*** and ***B*** are "None" then a single visualization with circular cross sections based on ***D*** is generated. The requirement that the selections for both ***A*** and ***B*** must be strictly "None" when only requesting visualizations with circular cross sections based on ***D*** helps to prevent confusion as to what is being rendered. Table 1 summarizes the plots obtained from different column selections. Figure 10 shows examples of options for the column ***C***, ***A***, ***B*** and ***D*** selectors.

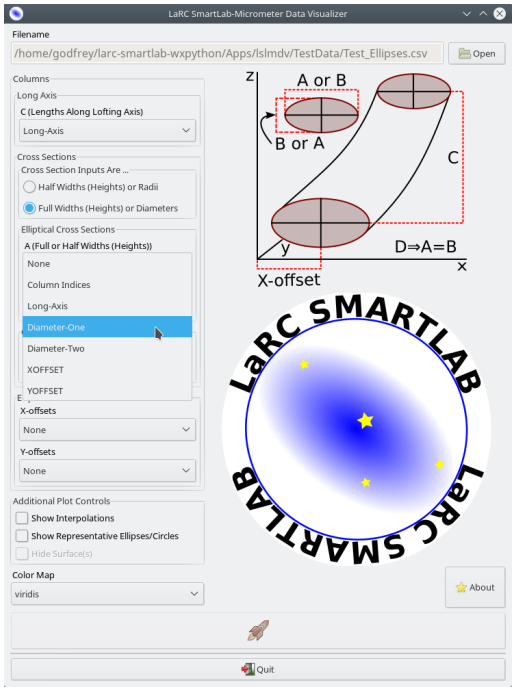
A note on the notation: While it is possible to generate visualizations with circular cross sections by selecting the same value for **A** *and* **B**, *the application shall denote the plot that is generated as being one with an elliptical cross section irrespective of the specific selections for the column values.*

Table 1: Visualizations obtained from various column selections. (X="Don't Care", ~~None~~=not "None").

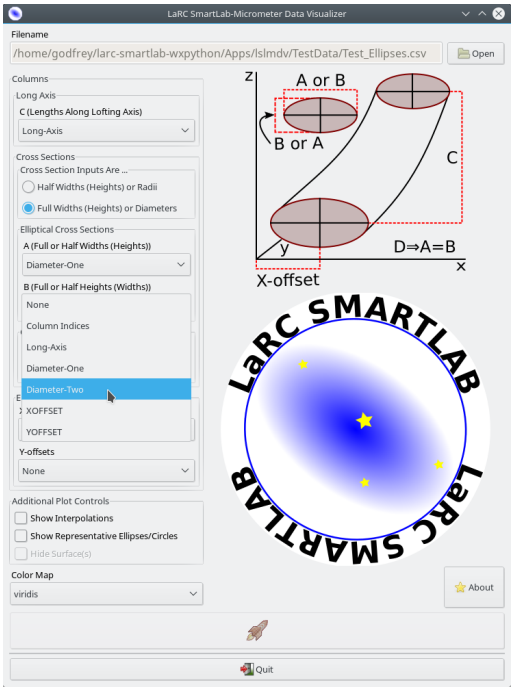
"Long/Lofting Axis" (***C***)	***A***	***B***	***D***	3D Plots	Cross Sections
None	X	X	X	0	N/A
~~None~~	~~None~~	None	X	0	N/A
~~None~~	None	~~None~~	X	0	N/A
~~None~~	~~None~~	~~None~~	None	1	Elliptical
~~None~~	None	None	~~None~~	1	Circular
~~None~~	~~None~~	~~None~~	~~None~~	2	Elliptical & Circular



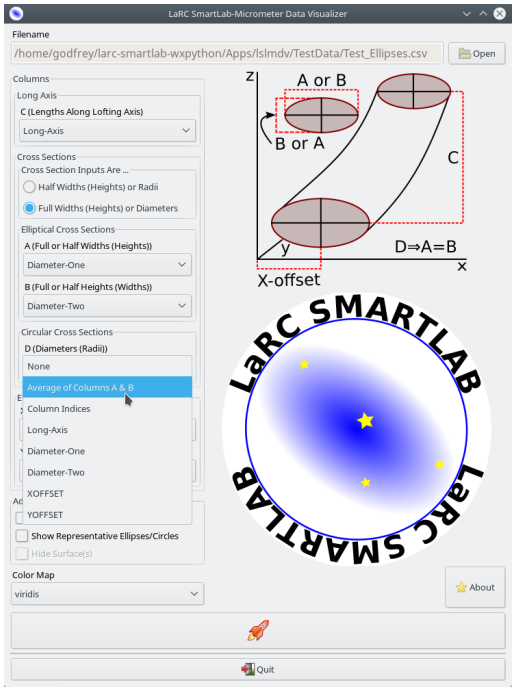
(a) "Long/Lofting Axis" ($\boldsymbol{C}$)



(b) Ellipse width or height ($\boldsymbol{A}$)



(c) Ellipse height or width ($\boldsymbol{B}$)



(d) Circle diameter ($\boldsymbol{D}$)

Figure 10: Options for selecting columns.

When all the required settings have been entered, the data visualization button is enabled (Figure 11).

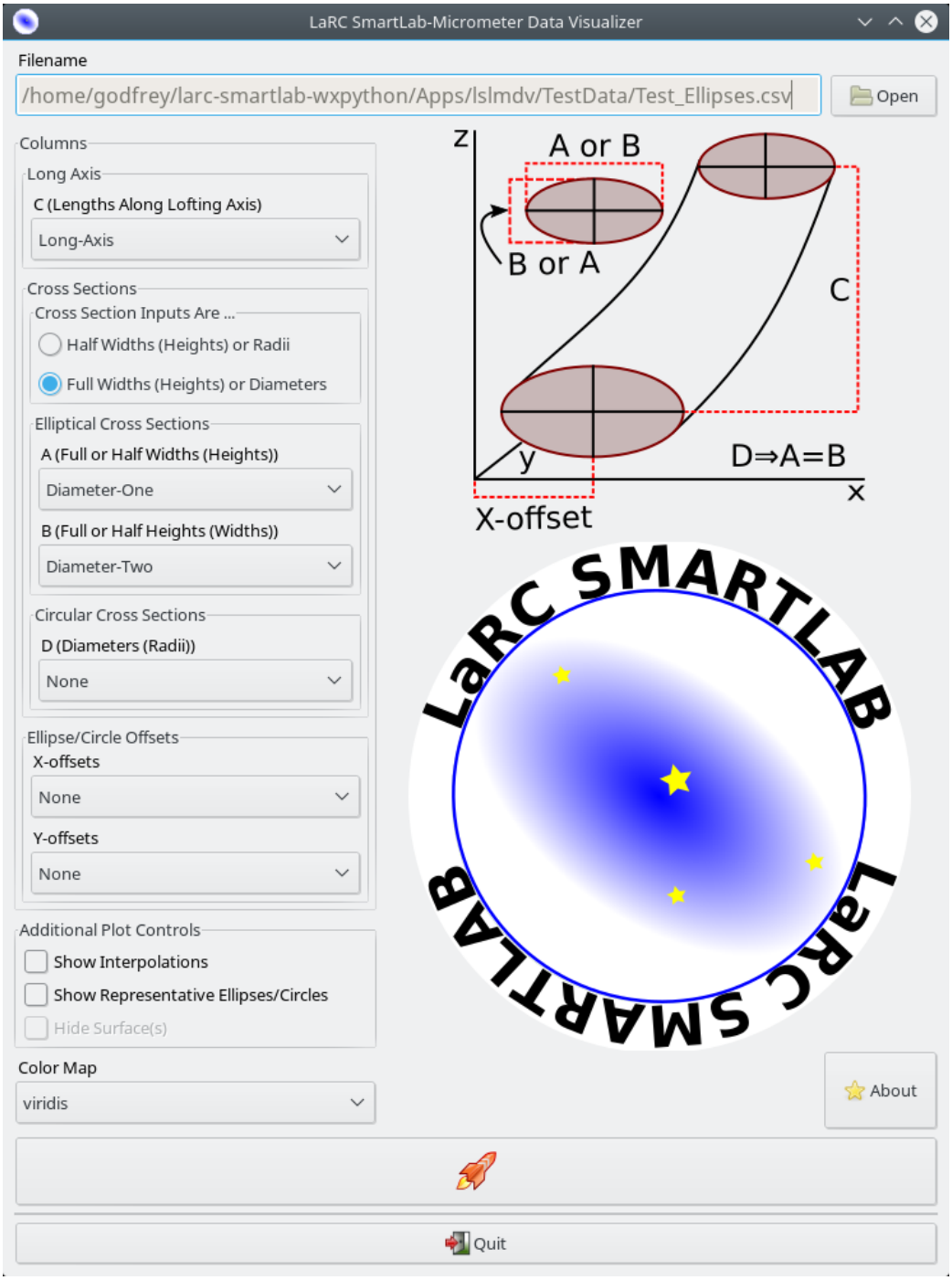


Figure 11: View of the user interface when the application is ready to display data.

3.3 Checking data

In addition to key column selections being required to have values that are not "None," as shown in Table 1, the data along the "Long/Lofting Axis" ($\boldsymbol{C}$) must be ***strictly*** monotonically increasing. The application carries out a check of the monotonicity when the user actually requests for data to be visualized using their column selections. The check is not carried out at the point of column selection to prevent unnecessary access to the loaded data table, and the user being inundated with message dialogs, while they are still in the process of setting up. If the data is not monotonic, the dialog message shown in Figure 12 is displayed letting the user know this.

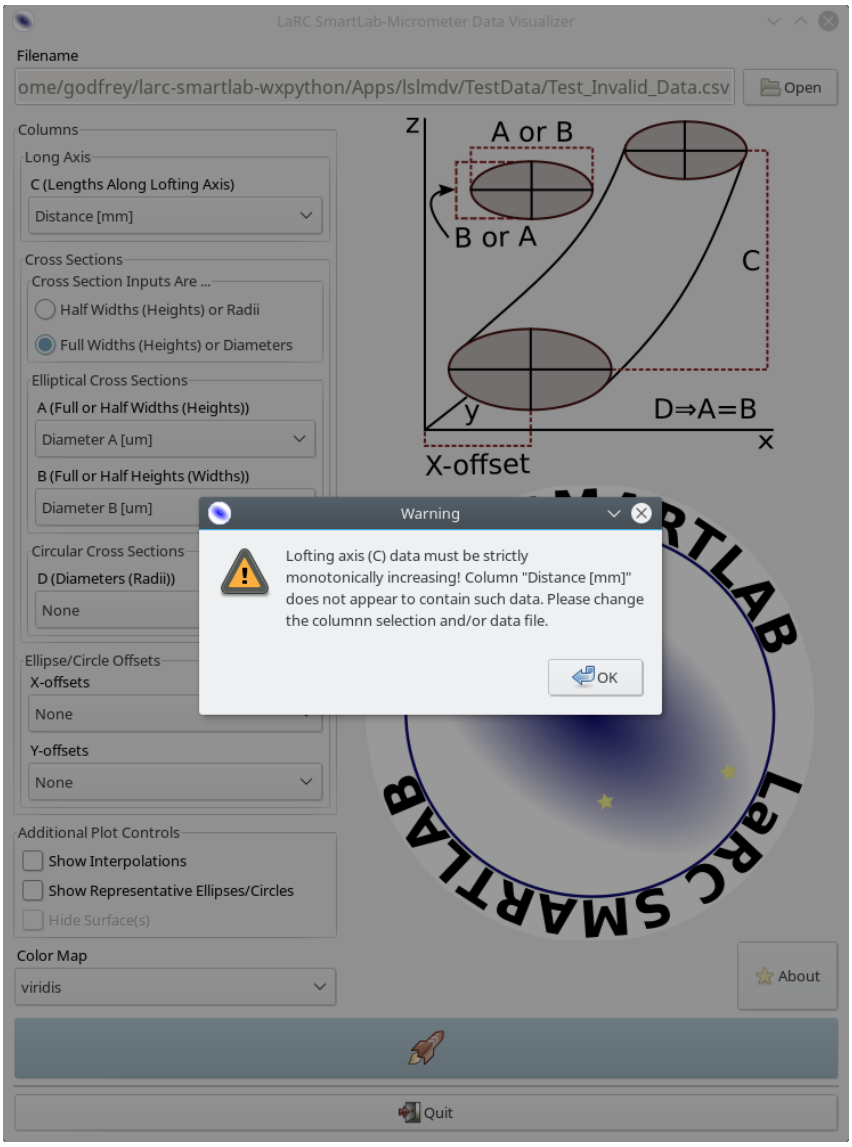


Figure 12: Dialog indicating an invalid selection of data for the "Long/Lofting Axis."

3.4 Plot options

There are a number of other options that the user can select for the visualization. These include whether or not to plot some additional graphs showing how the 3D visualizations are being generated (Section 5) and a selector of the colormap used for the surface (more details and examples in Section 3.5). The additional plots available are:

(a) A plot showing the interpolations being used to create a continuous connection of the ellipses/circles along the "Long/Lofting Axis." For elliptical cross sections there are two interpolation functions generated while circular cross sections will have only one. Note that when considering visualizations which include offsets (Sections 4.2.3 and 5.3), the interpolation functions are calculated before the offsets are applied, i.e. the interpolation functions will look the same whether or not offsets will be added.

(b) A representative sampling of the ellipses/circles that are being generated from the data points in the file can be shown in the 3D plot.

(c) The surface can be hidden so that only the ellipses/circles are shown.

3.5 Surface colormaps

The user can select one of several colormaps to be used for rendering the surfaces that are generated from their data (Figure 13). Results of plotting with some of the colormaps that are available can be seen from the examples that are presented throughout this document and some additional ones (specifically on colormaps) that are discussed in Appendices B and C.

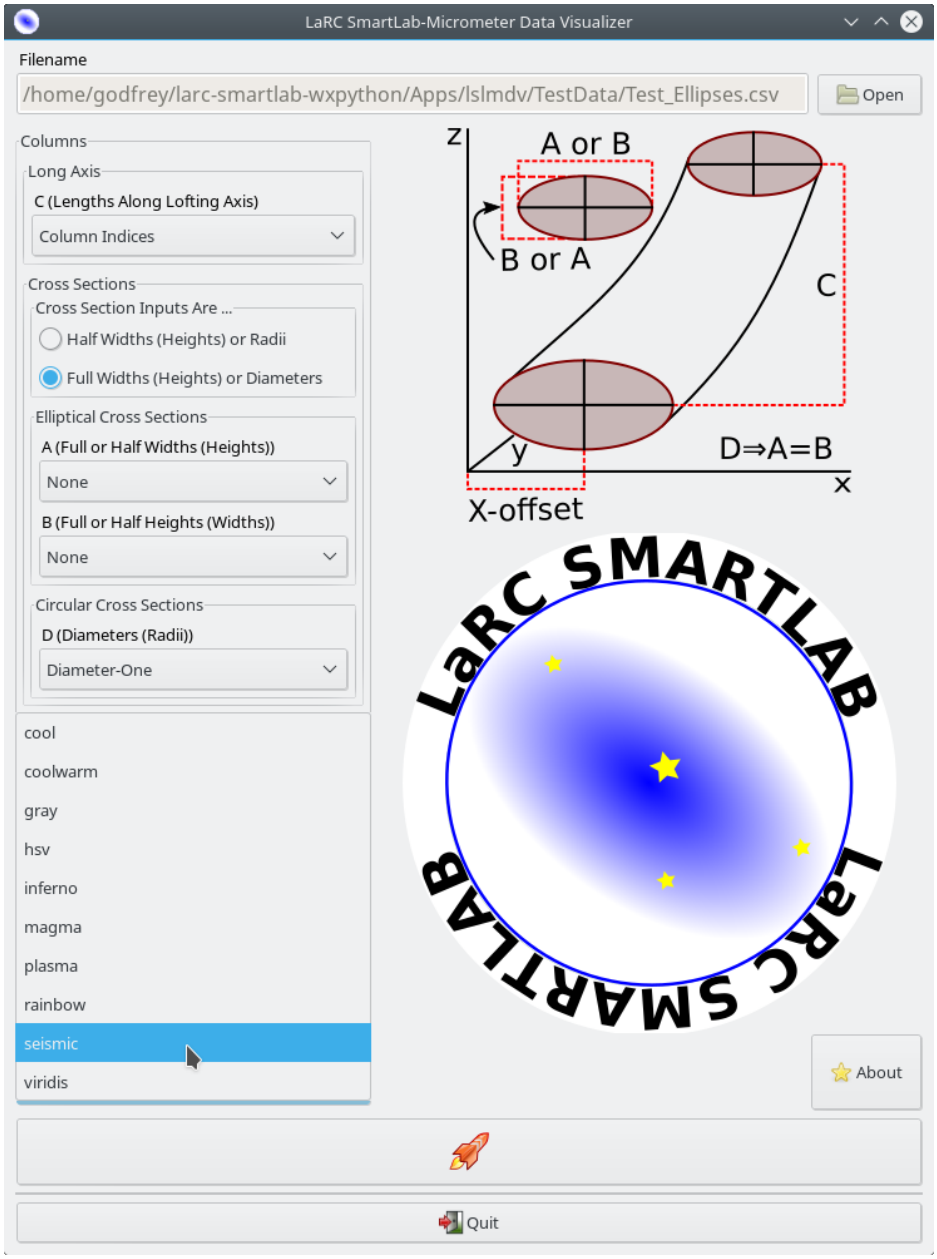


Figure 13: Options for the colormap used in painting the surface(s) that are rendered.

4 Data Visualization Examples

We now look at some examples of using the application to visualize experimental data as well as synthetic data. The later enables further exploration of the capabilities of the application.

4.1 Visualizing experimental data

The main use case for the **Micrometer Data Visualizer** application is visualizing experimental data. We show here some examples of this with visualization of optical micrometer data. The data are taken from two-axis diameter measurements taken along the length of a sewing thread-like carbon nanotube (CNT) yarn. Figure 14 shows the 3D rendering generated from measurements of the yarn.

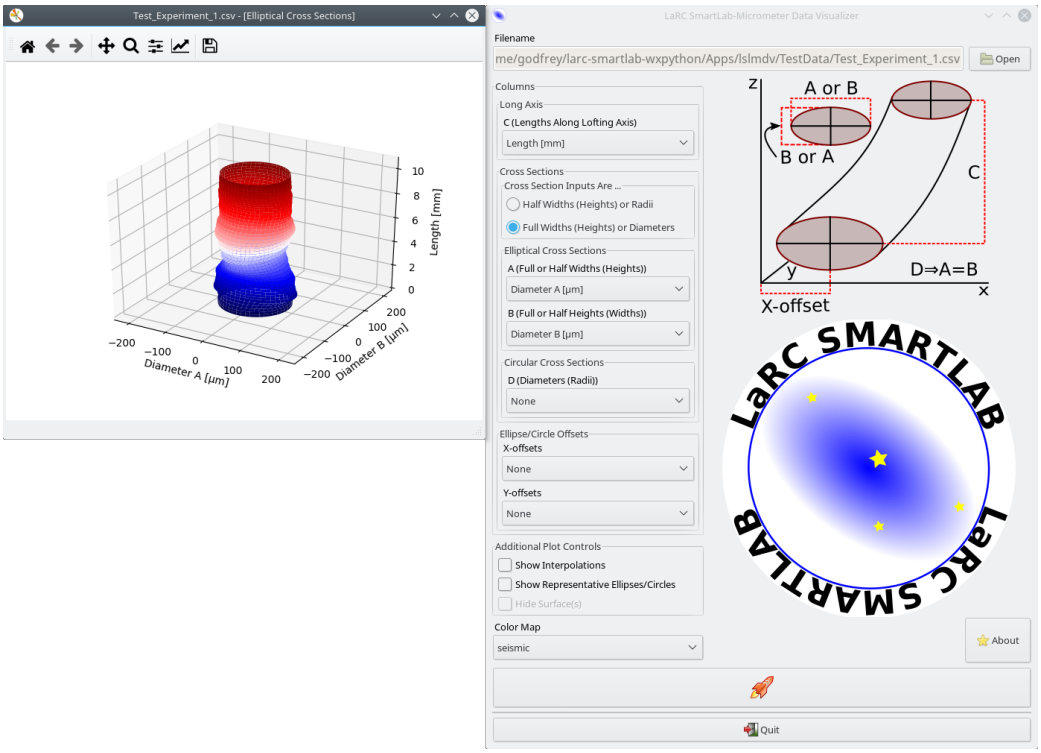


Figure 14: The settings and 3D rendering of the data from experimental two-axis optical micrometer diameter measurements of a CNT yarn.

Figures 15 and 16 show close up plots of the experimental data. The graphing window of the **Micrometer Data Visualizer** allows for the rendered data to be rotated to multiple orientations as can be seen in Figure 16.

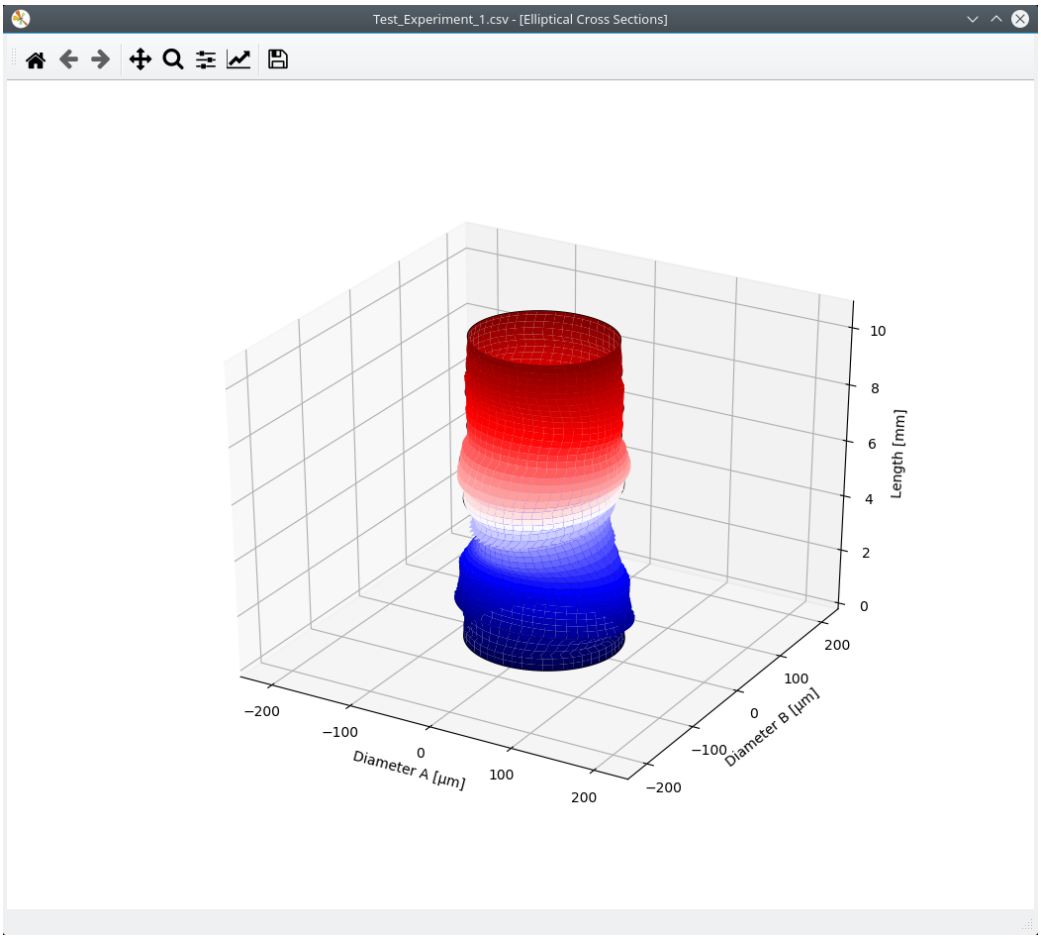
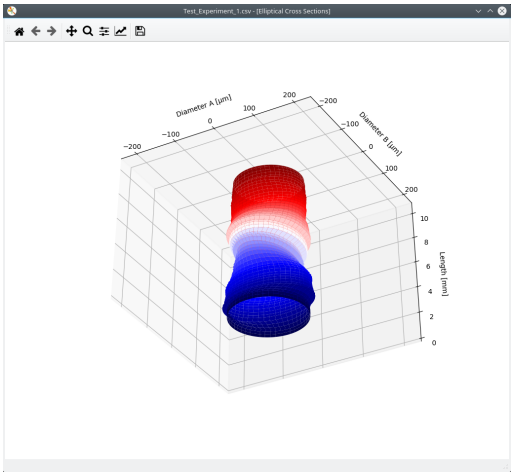
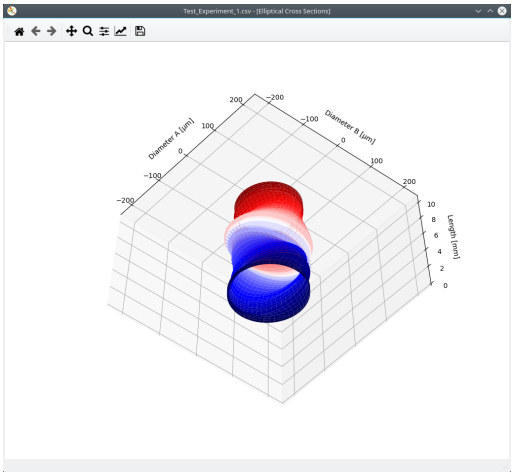


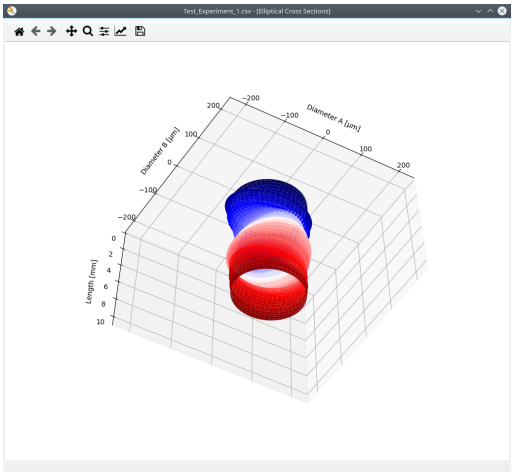
Figure 15: A closeup of the rendering of the experimental data. The viewing orientation (see also Figure 16) is [Azimuth=-60°, Elevation=30°].



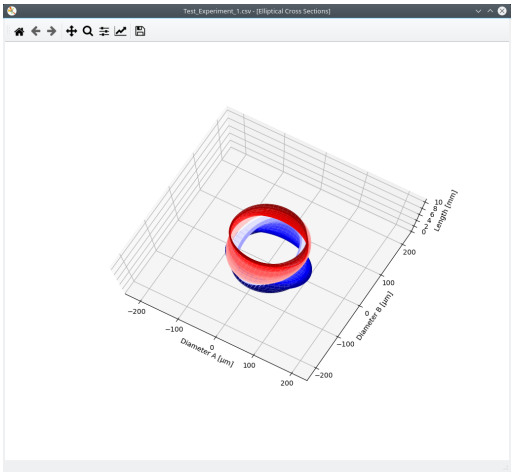
(a) Azimuth=-20°, Elevation=-40°



(b) Azimuth=-40°, Elevation=-60°



(c) Azimuth=-60°, Elevation=120°



(d) Azimuth=-60°, Elevation=80°

Figure 16: Experimental data can be viewed rotate to different orientations.

4.2 Visualizing synthetic data

To explore the range of data that can be rendered we look at the visualization of some synthetic data.

4.2.1 Ellipses with varying widths & heights

We start by showing the general case for ellipses and circles where at least some of the ellipses along the "Long/Lofting Axis" have unequal diameters. A reminder of the notation of the application, and this document, that is introduced in Section 3.2: The plots generated using column selections ***A*** and ***B*** will be referred to as the plots "with elliptical cross sections" even though the plotted cross sections will be circular in the special case where the user selects ***A*** to be the same as ***B***. The term plots "with circular cross sections" will be used to specifically refer to renderings made using the column selection ***D***.

Figure 17 shows the 3D rendering of the surface generated by lofting ellipses where the data in the widths and heights columns (***A*** and ***B***) are not all equal along the "Long/Lofting Axis."

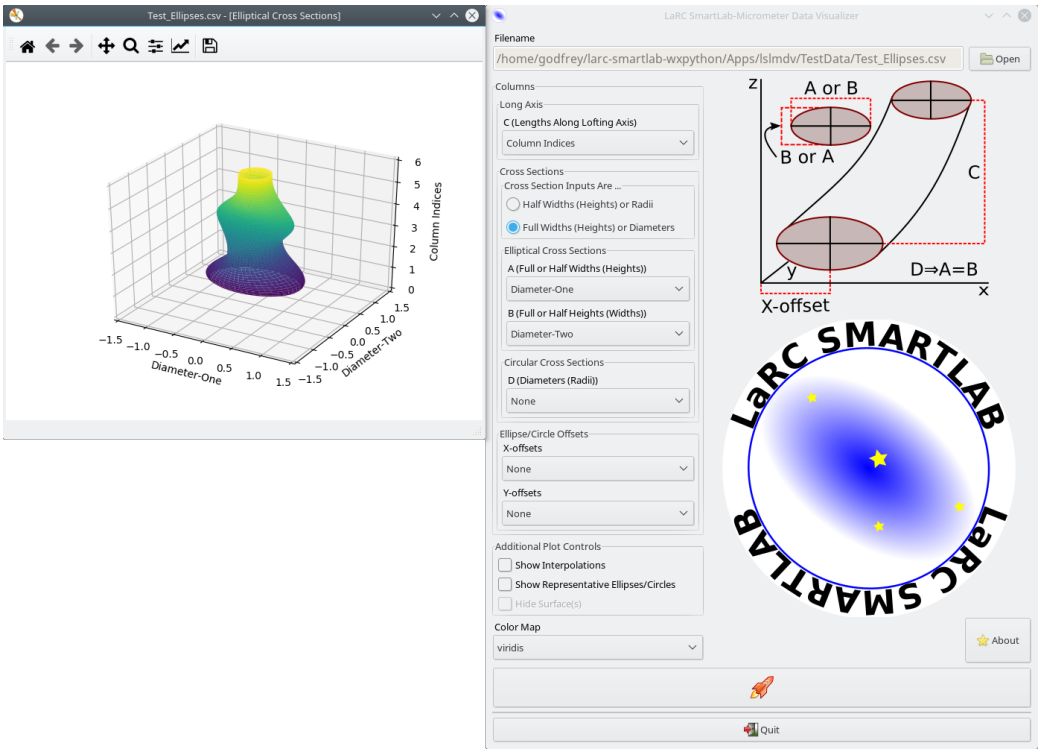


Figure 17: 3D visualization generated from ellipses which have unequal widths and heights at different locations along the "Long/Lofting Axis."

Figure 18 shows the 3D rendering of the surface generated by lofting circles of unequal diameters. The data for the diameters are read from column selection **D**.

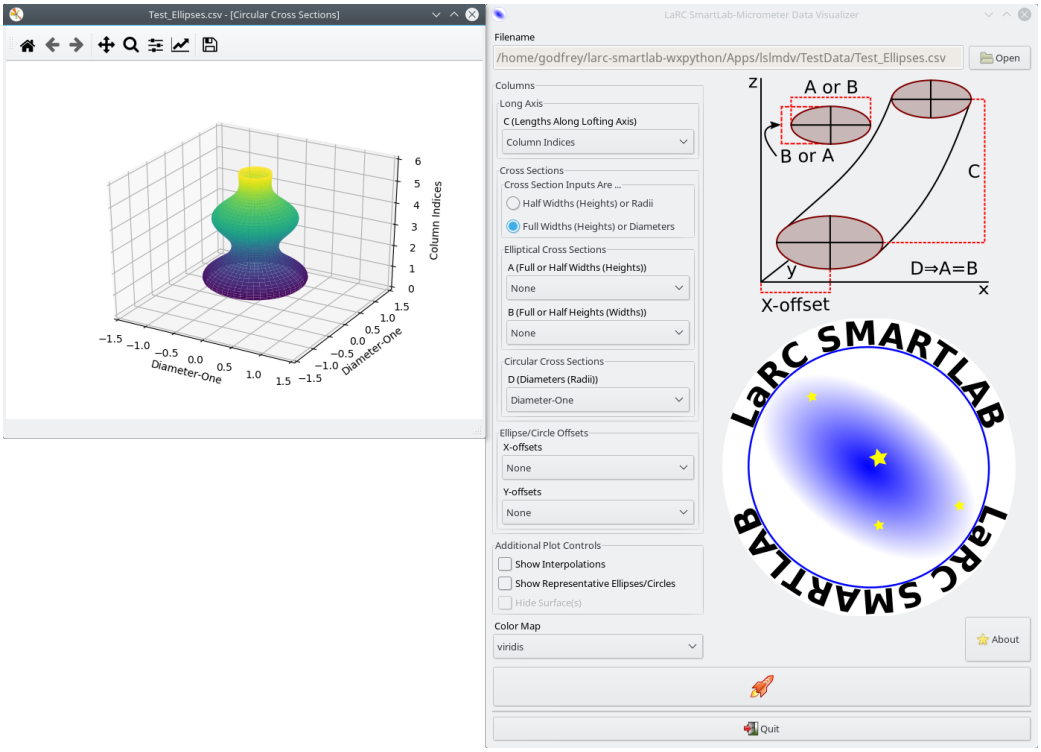


Figure 18: 3D surface generated from circles with varying diameters.

The application allows for the generation of both the 3D rendering with the elliptical cross sections (data from $\boldsymbol{A}$ and $\boldsymbol{B}$) and the one with a circular cross sections (data from $\boldsymbol{D}$) at the same time as shown in Figure 19.

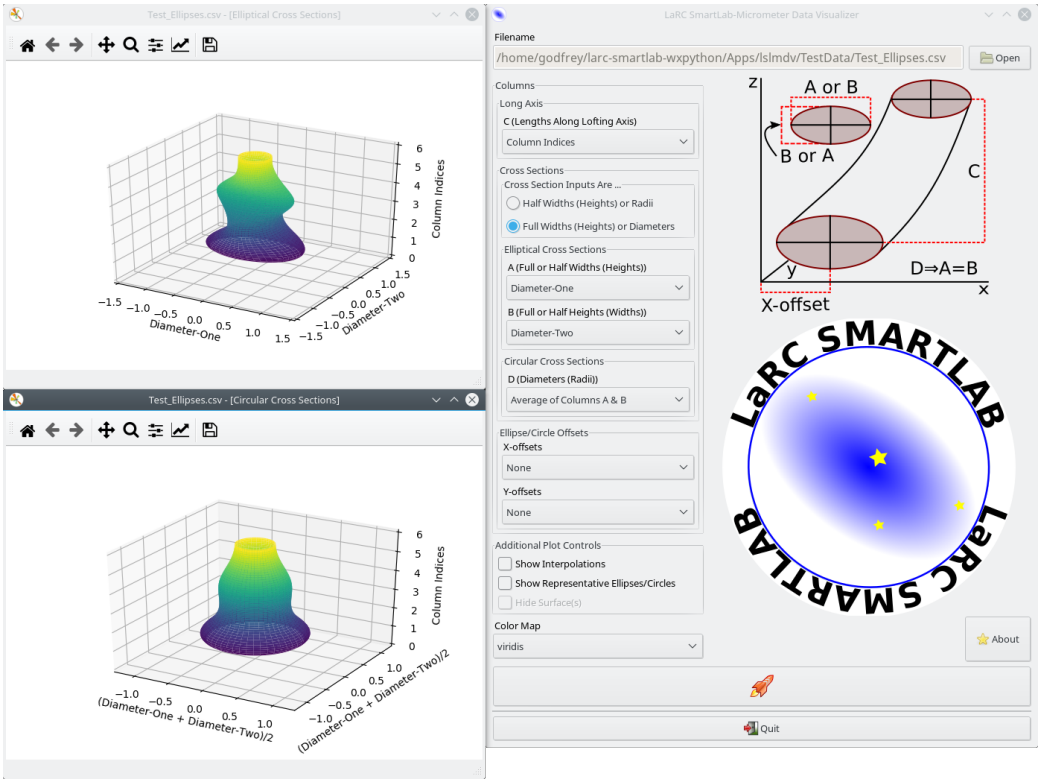


Figure 19: Lofted ellipses and circles can be generated with a single call to the rendering function.

4.2.2 Ellipses with constant widths & heights

A special case for the rendering is where the circles/ellipses all have the same diameters. Figure 20 shows the cylinder generated when each of the cross sections that are being lofted along the "Long/Lofting Axis" is a circle.

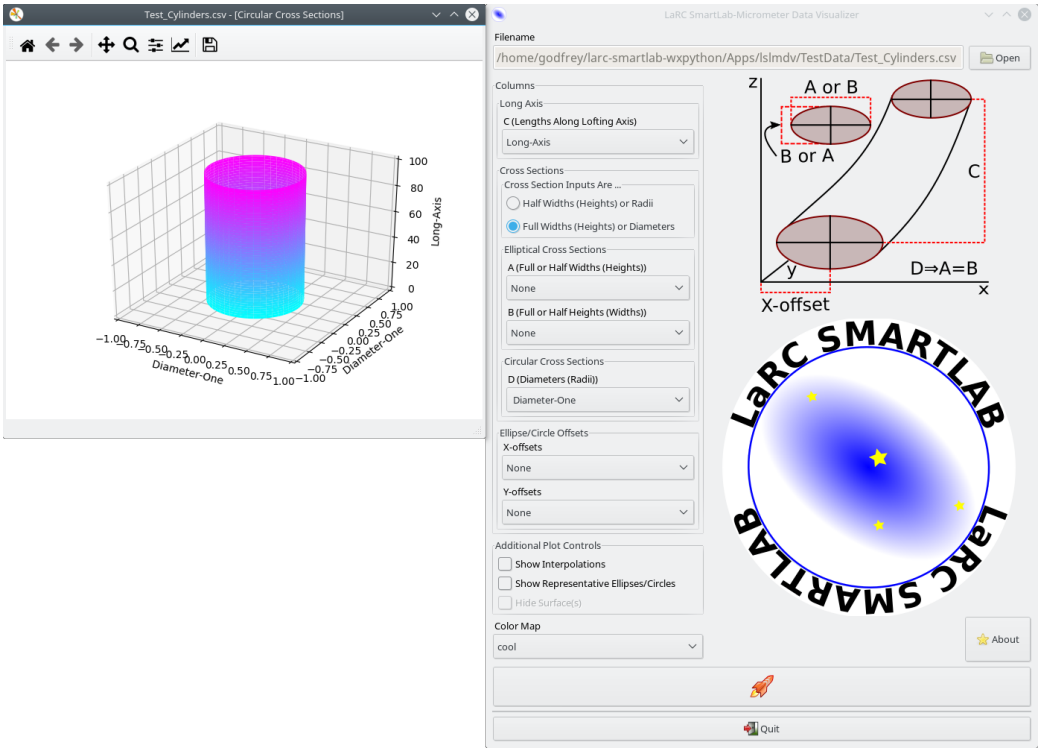


Figure 20: Cylinder generated from lofting equal diameter circles.

Figure 21 shows the elliptical cylinder generated when ellipses with equal widths and heights are lofted.

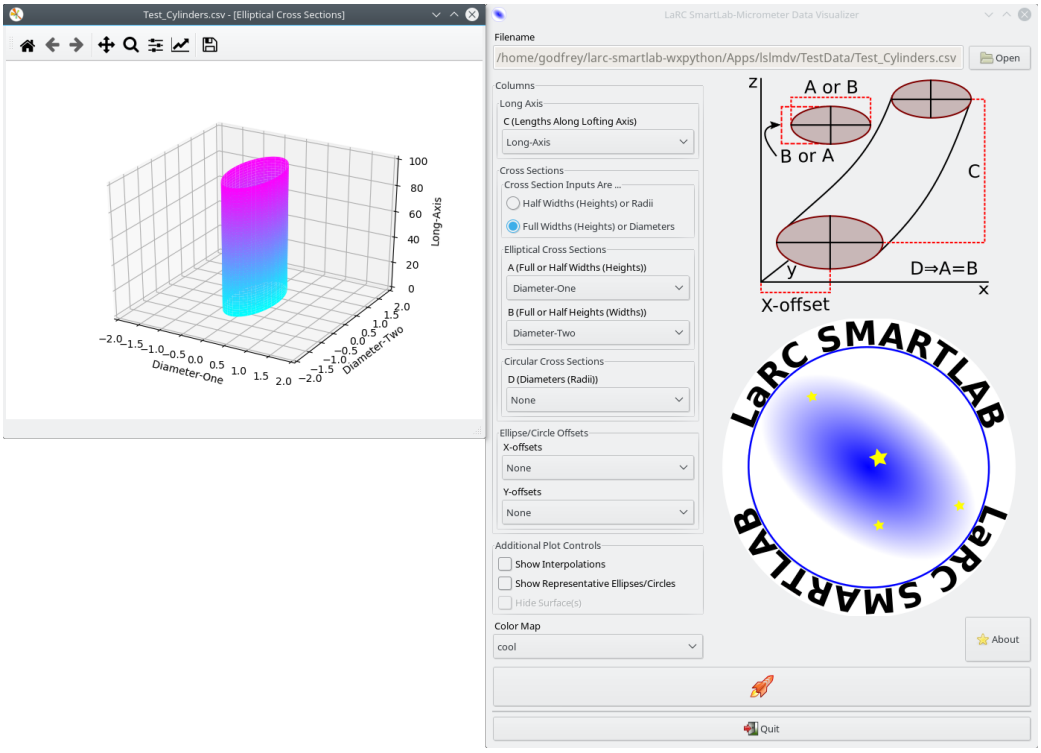
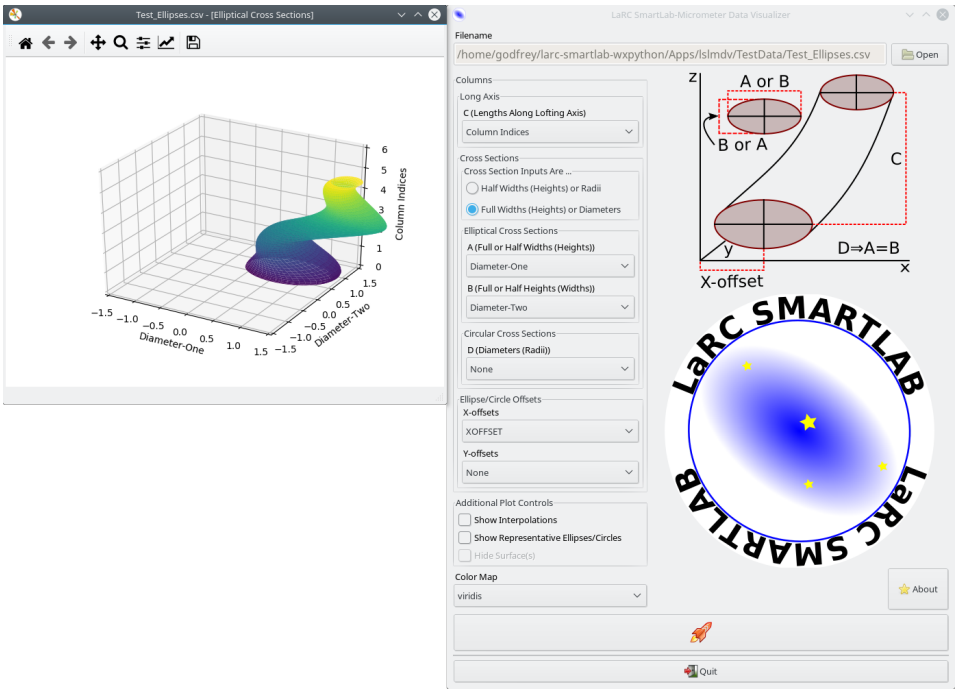


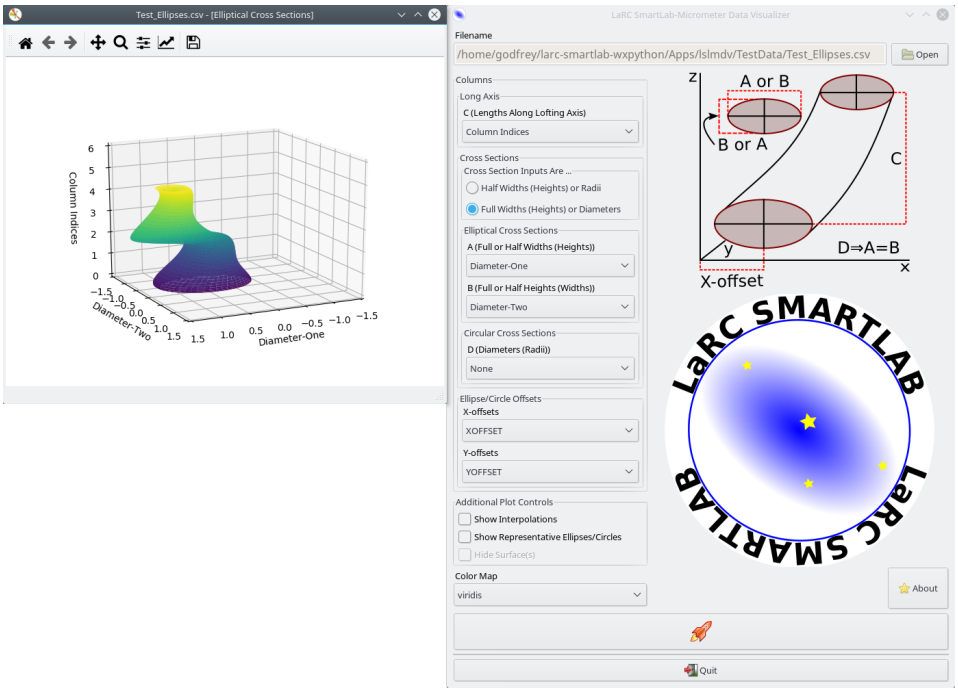
Figure 21: Elliptical cylinder generated from lofting ellipses of equal widths and heights.

4.2.3 Offsets

The application will accept data for offsetting the ellipses or circles along the "X" and "Y" axes (X and Y axes from here on). These data must be provided for each ellipse or circle. In the capture of experimental data, offsets could be introduced if for example there is a need to move the measuring device, perpendicular to the long axis, to follow the contour of a non straight tubular test sample. Figure 22(a) shows ellipses lofted with offsets along the X axis while Figure 22(b) shows ellipses lofted with offsets along both the X and Y axes. Figure 23(a) shows a cylinder with offsets along the X axis and an elliptical cylinder with offsets on the X and Y axes. Figure 23(b) shows elliptical cylinders with offsets to the ellipses on the X and Y axes.

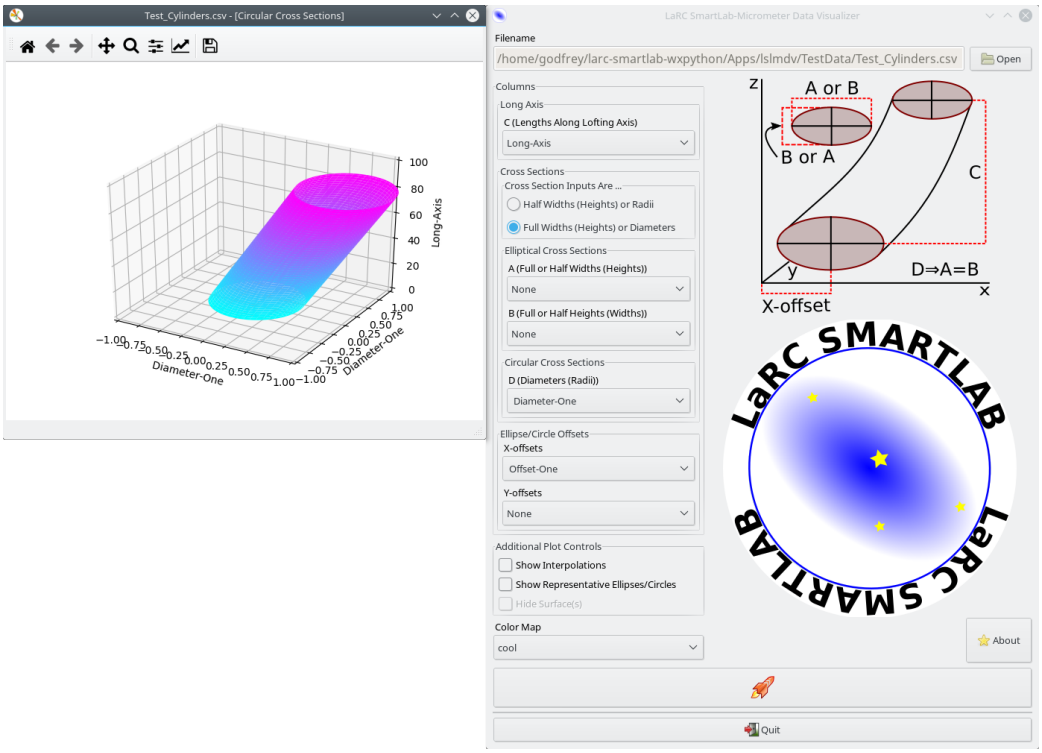


(a) Offsets along the X axis

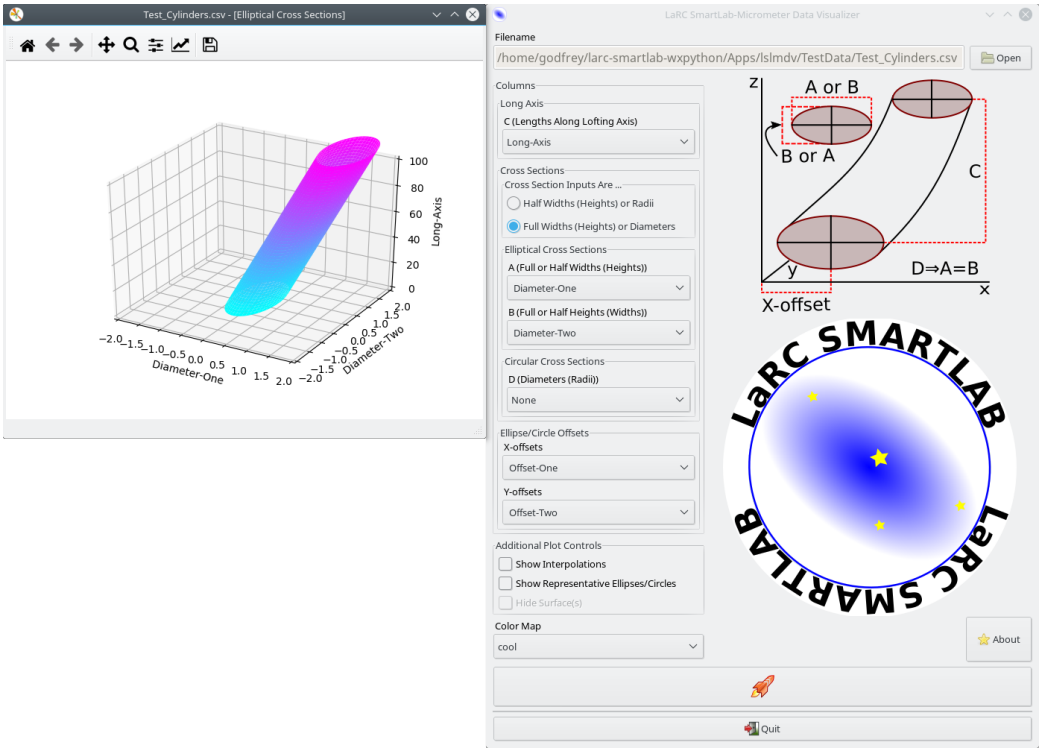


(b) Offsets along both the X & Y axes

Figure 22: Ellipses lofted with offsets along the X and Y axes.



(a) Offsets along the X axis



(b) Offsets along both the X & Y axes

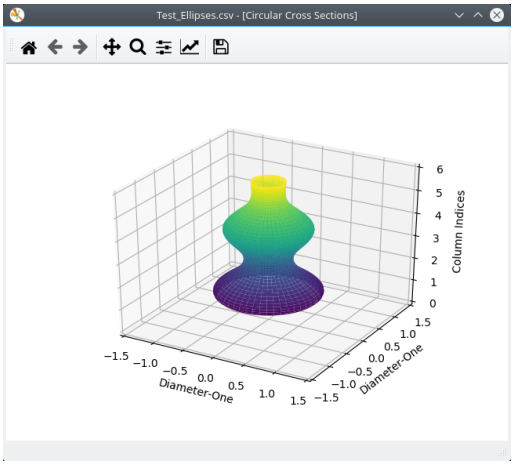
Figure 23: Constant diameter circles as well as constant width and height ellipses lofted with offsets along the X and Y axes.

5 Anatomy of the Plot

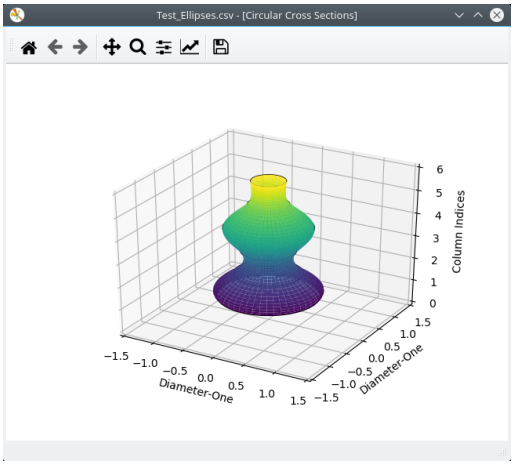
To help understand how the data is being rendered, we now look at some additional plots that can be rendered by the application and which show stages or components used to generate the 3D surfaces from the input data. To illustrate how the surfaces are generated, we shall superimpose some of the ellipses (circles) on the 3D surface plots, hide the surfaces to show only the ellipses (circles) and show the interpolations that are used to generate the "Long/Lofting Axis" profiles.

5.1 Ellipses of varying diameters

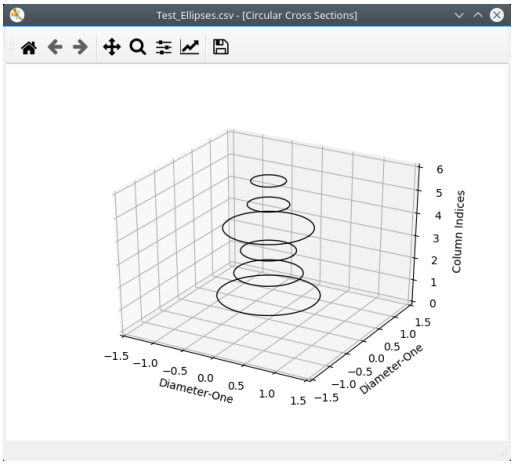
We start by looking at the generation of the 3D surfaces in the case of the circles with varying diameters. Figure 24 shows the elements that make up the surfaces. An important note is that a cubic spline function is being used to interpolate the "Long/Lofting Axis" data (Figure 24(d)). In some cases, generating a smooth cubic interpolation will lead to the formation of prominent ridges along the profile. These undulations become less pronounced when there are more data along that axis. This will be seen more clearly in the rendering of the cylinders (Section 5.2). In the 3D rendering from circular cross sections, there is only one "Long/Lofting Axis" outline that requires an interpolation. Except for very small data sets, the circles (ellipses) such as shown in Figures 24(b) and 24(c) are generally only a small sampling of the data that are used to generate the surfaces. This helps to prevent cluttering of the plot.



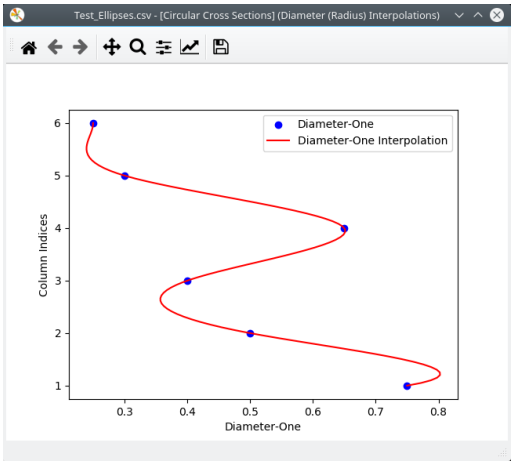
(a) 3D surface



(b) 3D surface & representative circles



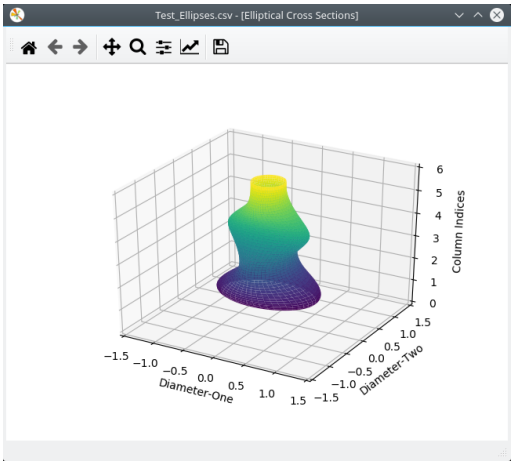
(c) Surface hidden



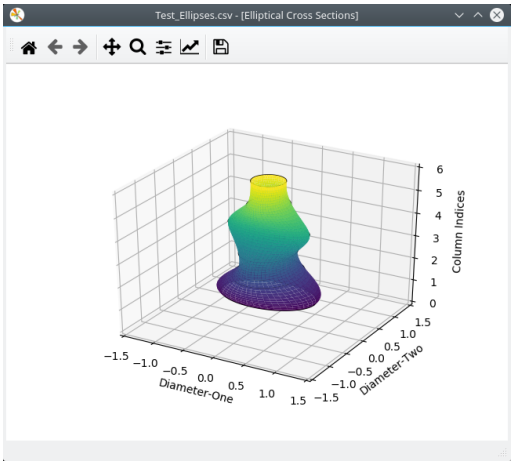
(d) Interpolations

Figure 24: Generating surfaces from lofting circular cross sections.

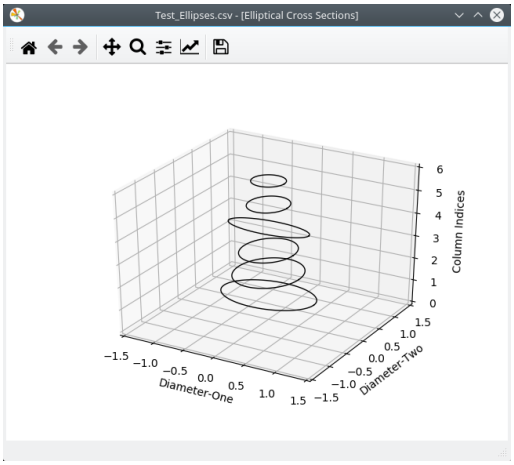
Figure 25 shows the lofting of ellipses with varying widths and heights. When generating surfaces from elliptical cross sections, two interpolations are required (Figure 25(d)). When the interpolations are calculated from datasets with only a few points there may be exaggerated undulations of the surface and therefore care must be taken in interpreting the 3D surfaces generated.



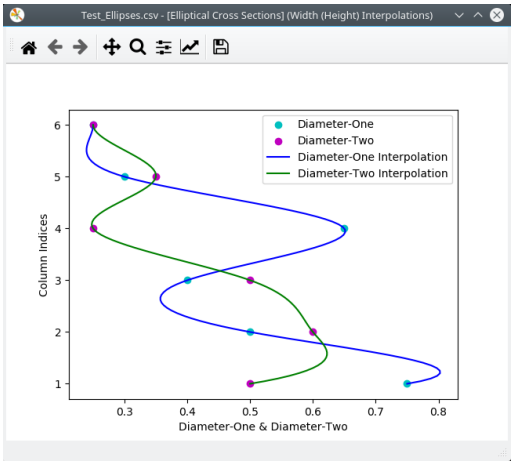
(a) 3D surface



(b) 3D surface & representative ellipses



(c) Surface hidden

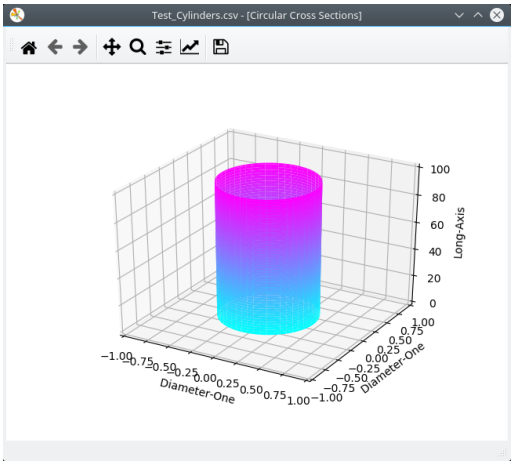


(d) Interpolations

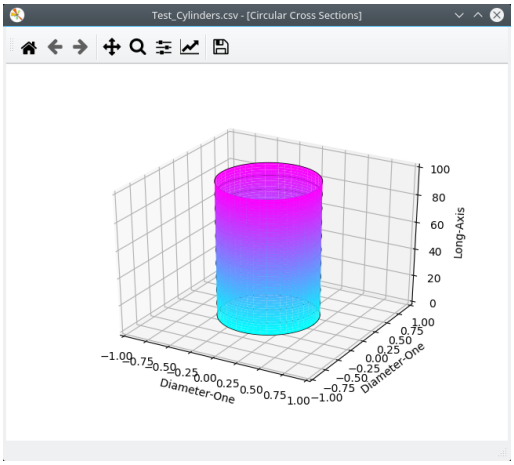
Figure 25: Creating a surface from ellipses with varying widths and heights.

5.2 Circles and ellipses of constant diameter

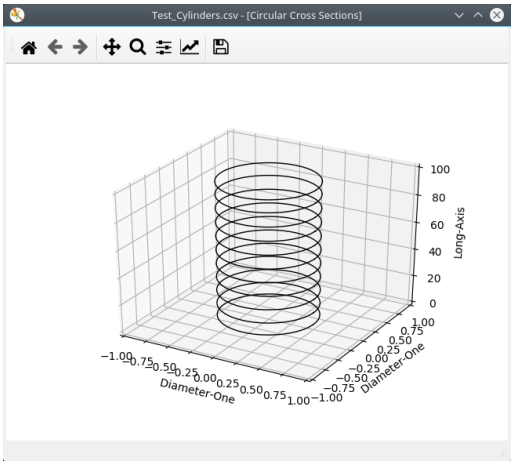
Figure 26 now shows the case where a cylinder is generated from circles of constant diameter. The oscillations from the interpolation function can be seen at the bottom of the Figure 26(d). It can be seen though that the function quickly stabilizes and a usable rendering of the cylinder is generated. For the case with the circles, we see (as expected) the single interpolation.



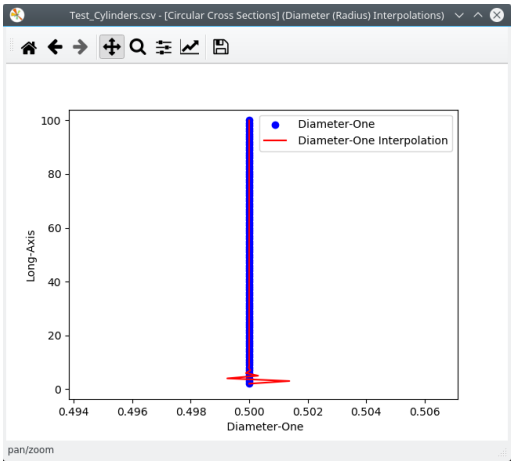
(a) 3D surface



(b) 3D surface & representative circles



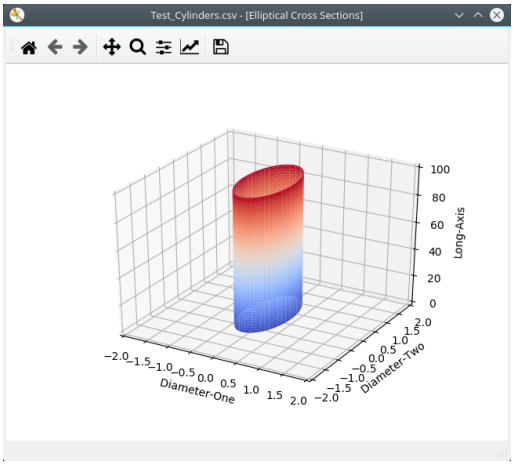
(c) Surface hidden



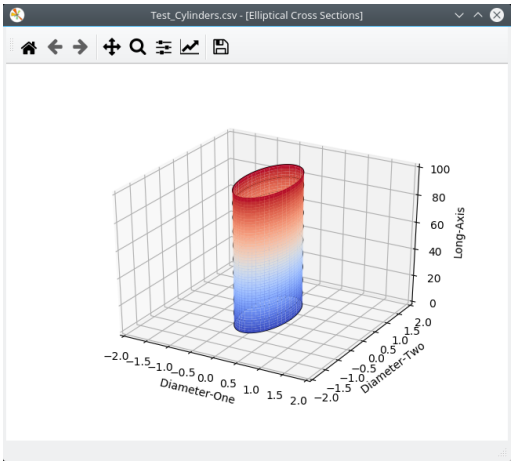
(d) Interpolations

Figure 26: Creating a surface from lofting constant diameter circles.

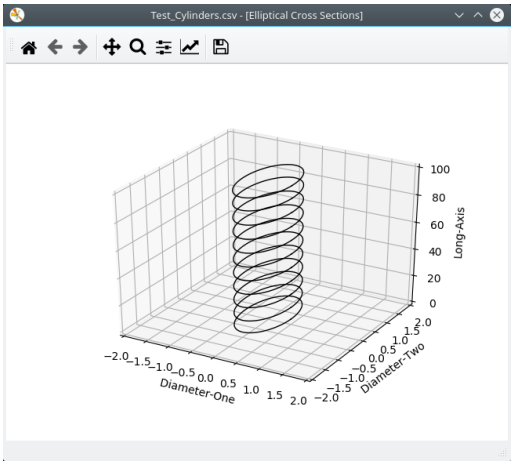
Figure 27 shows the plots when generating an elliptical cylinder. In this case we have two interpolations, one for each of the axes of the ellipses as seen in Figure 27(d). Here again it is clear to see how the interpolation function starts with some initial oscillation that quickly settles and we are able to see the walls of the elliptical cylinder.



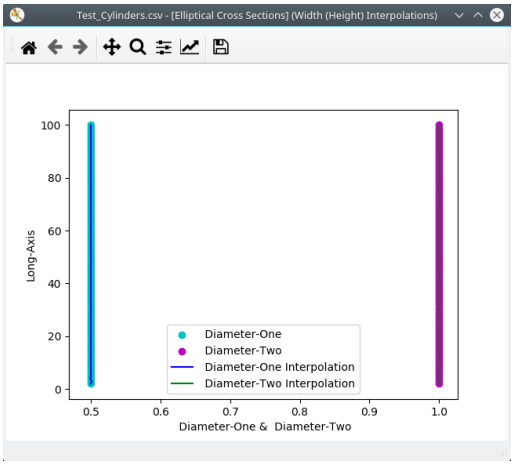
(a) 3D surface



(b) 3D surface & representative ellipses



(c) Surface hidden

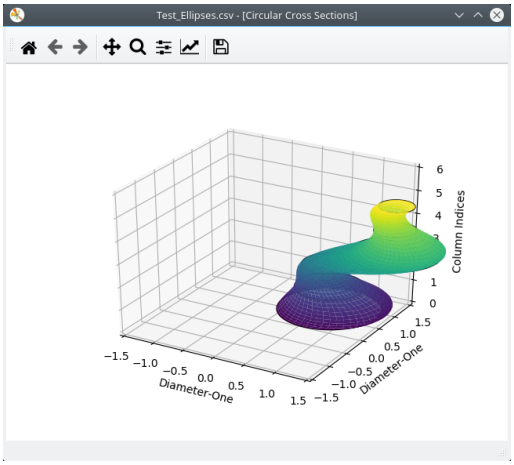


(d) Interpolations

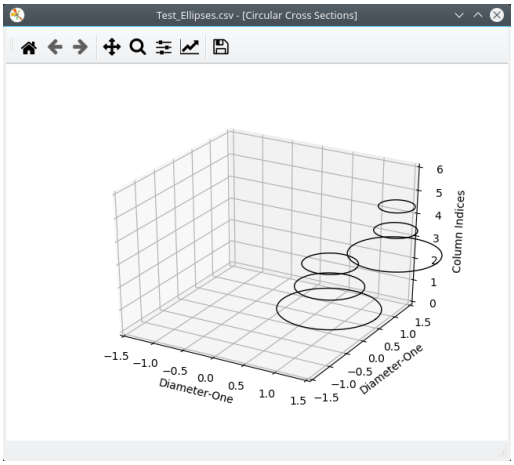
Figure 27: The generation of an elliptical cylinder.

5.3 Offsets

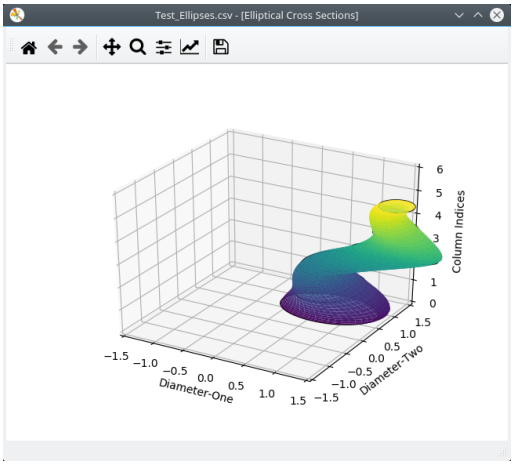
The previous renderings all have no offsets on either axis. We now take a look at the structure when we do have some offsets. Figure 28 shows plots generated from lofted circles and ellipses when there are offsets along the X axis. Apart from the offsets, the data and settings for these renderings are similar to those for the plots in Figures 24 and 25. In the plots with the surfaces hidden (Figures 28(b) and 28(d)), it is clear to see the locations of the circles and ellipses that are being joined together to create the surface. Since, as previously mentioned, the interpolations are carried out before the offsets are added to the circle and ellipse positions, the plots of the interpolations for these surfaces look the same as those in Figures 24(d) and 25(d).



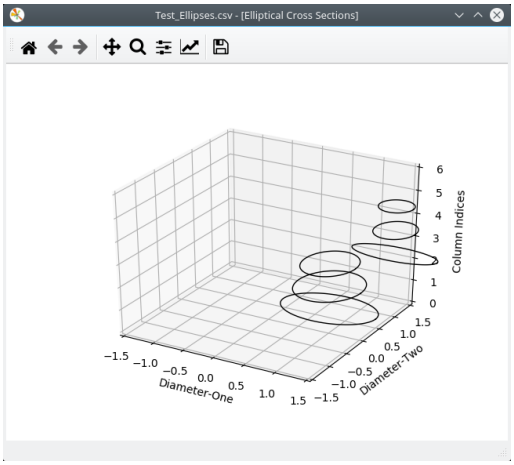
(a) 3D surface & circles



(b) Surface hidden



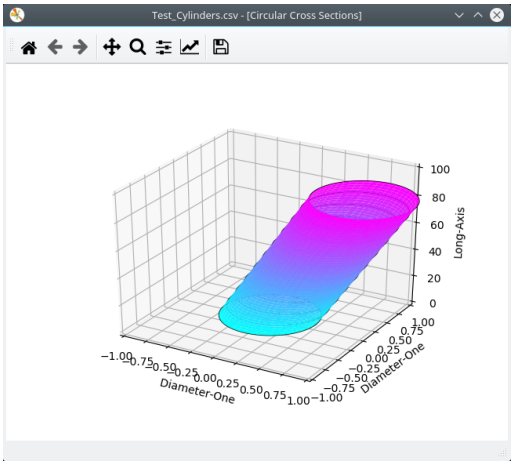
(c) 3D surface & ellipses



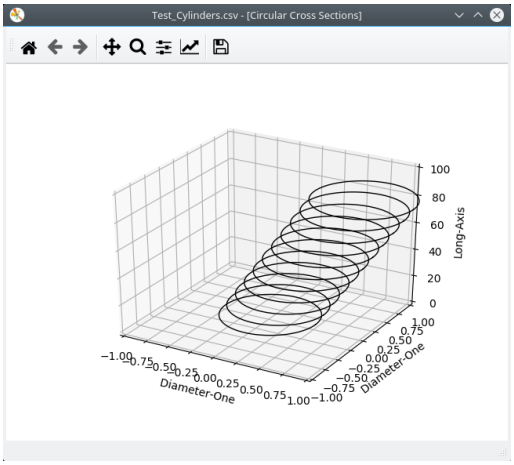
(d) Surface hidden

Figure 28: Lofting of circles and ellipses with offsets along the X axis.

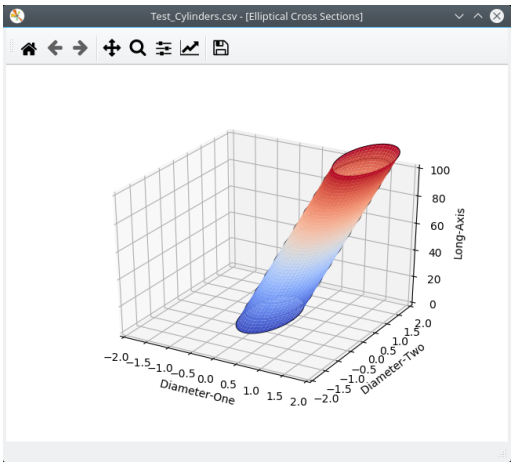
To further illustrate how rendering with the offsets works, we apply them to some regular and elliptical cylinders. The resulting *leaning* surfaces and representative circles and ellipses are shown in Figure 29.



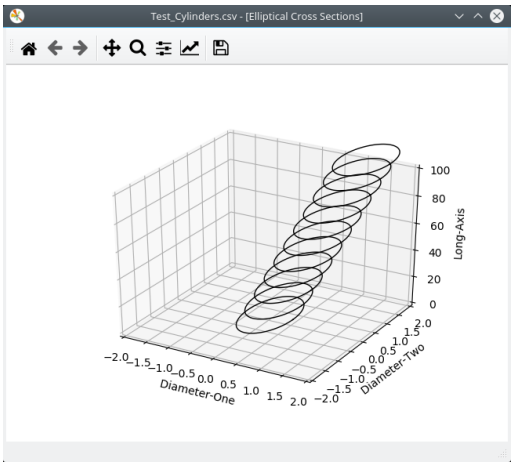
(a) 3D surface & circles



(b) Surface hidden



(c) 3D surface & ellipses

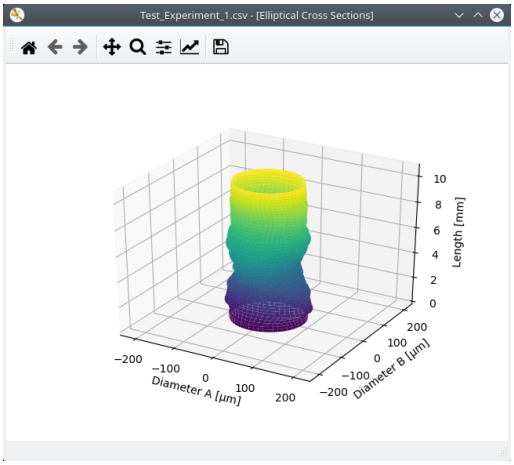


(d) Surface hidden

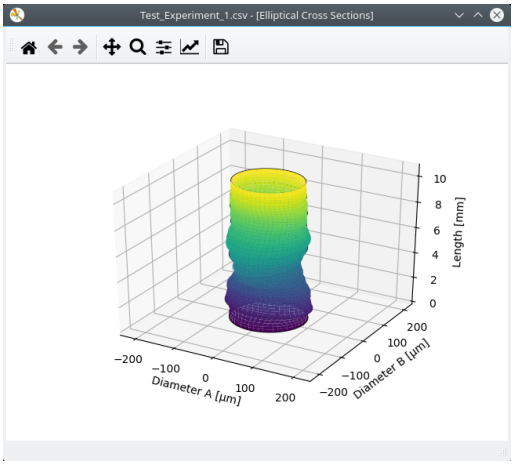
Figure 29: Surfaces and the representative circles (ellipses) generated when lofting cylinders and elliptical cylinders with offsets.

5.4 Experimental data

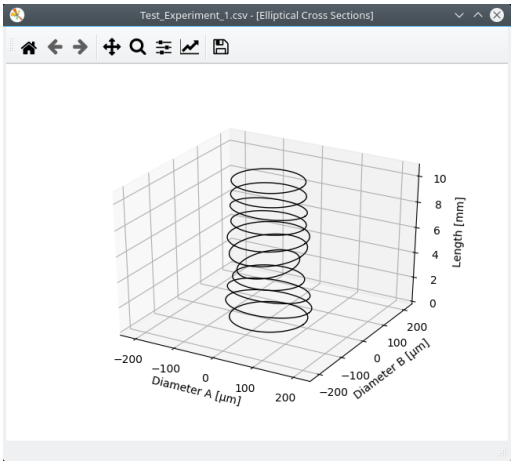
We conclude the look at the generation of the 3D surfaces by looking at an example using experimental data. This data are shown in Figure 30. It can clearly be seen from the high density of points in Figure 30(d) that a lot of data are being used for the interpolation and only a tiny fraction is shown as the representative ellipses in Figure 30(c). The large quantity of data that are generated by the experiment minimizes artifacts from the interpolation functions, rendering these representations of the data valuable.



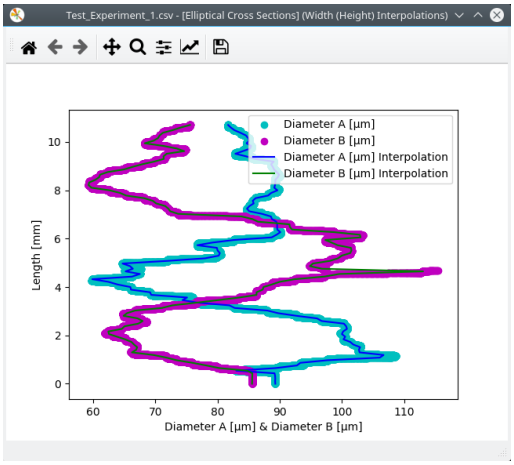
(a) 3D surface



(b) Surface & ellipses



(c) Surface hidden



(d) Interpolations

Figure 30: Creating the visualization from experimental optical micrometer data.

6 Implementation

We now briefly discuss the implementation of the application. It is not intended to provide the complete code here (this can be separately requested from the authors) but rather, to provide interested developers with pointers to the tools that are being used. The **Micrometer Data Visualizer** application is implemented as a series of modules to aid code maintenance and reuse. We briefly describe some of the key modules below.

6.1 Top level

Listing 1: Import of required modules.

```
8  from pathlib import PurePath
9  import numpy as np
10 import pandas as pd
11 import wx
12 from matplotlib import pyplot
13 import libgsalt.gui.miscdialogs as miscdialogs
14 import libgsalt.gui.icons as icons
15 import lslmdvbase
16 import lslmdvplots
17 import lslmdvcontrols
```

The top level application calls in a number of standard Python modules as well as application specific modules. We start by listing the standard Python modules that are imported, the functionality they provide, and we cite relevant resources.

1. **pathlib:** provides classes for managing files[1]
2. **numpy:** a library for processing multi-dimensional arrays[2,3]
3. **pandas:** data analysis and manipulation tool[4,5]
4. **wx:** GUI toolkit for python[6]
5. **matplotlib:** a library for creating visualizations[2,7,8]

Next are the **libgsalt.*** custom library modules, available from the authors, which provide general GUI and data processing functionality (Listing 1). The functionality provided by the **libgsalt.*** modules is described below.

1. **libgsalt.gui.miscdialogs:** provides standard dialog boxes such as the "About" dialog box (Section 2.2, Figure 6)
2. **libgsalt.gui.icons:** provides the icons used in the application as a Python module to make them more portable.

The top level application contains one class (Listing 2) and some supporting functions. The application widgets are described in the **MainWindow** class and its parent class **lslmdvcontrols.MainWindowControls**.

Listing 2: Class.

```
20 class MainWindow(lslmdvcontrols.MainWindowControls):
21     """
22     The class provides the main interface of the application.
23 
24     Extends lslmdvcontrols.MainWindowControls
25     """
26 
27     def __init__(self, parent, _id, title):
```

An initialization function defined in Listing 3 is called up at the start up of the application (Listing 4). The application launches as a **wx.App**[6].

Listing 3: Initialization function.

```
363 def initialize_the_application_main_window():
364     """
365     Create the application's main window.
366 
367     Returns
368     -------
369     None.
370 
371     """
372     frame = MainWindow(
373         None, wx.ID_ANY, 'LaRC SmartLab-Micrometer Data Visualizer')
```

Listing 4: Application start up.

```
378 app = wx.App(False)
379 initialize_the_application_main_window()
380 app.MainLoop()
```

6.2 Application specific supporting modules

There are additional modules (**lslmdv...**) that are specifically tailored for this application that are imported as shown in Listing 1. These modules are briefly described below.

6.2.1 lslmdvbase

The base module provides a number of constants used by the application, including paths created using functions from the path manipulation module **pathlib**, which

is imported as shown in Listing 5. The constants that are defined are shown in Listing 6.

Listing 5: **lslmdvbase** imports.

```
30 from pathlib import Path, PurePath
```

Listing 6: **lslmdvbase** constants.

```
32 PATH_TO_PLOT_SCHEMATIC = str(PurePath.joinpath(
33     Path(__file__).resolve().parent, 'lslmdv_plot_schematic.png'))
34
35 COLUMN_SELECTION_INDEX = 'Column␣Indices'
36 COLUMN_SELECTION_NONE = 'None'
37 AVERAGE_OF_COLUMNS_A_AND_B = 'Average␣of␣Columns␣A␣&␣B'
38 BASE_CHOICES_FOR_COLUMN_C = [COLUMN_SELECTION_INDEX]
39 CHOICES_FOR_WIDTH_OR_HEIGHT_COLUMNS_A_AND_B = [COLUMN_SELECTION_NONE]
40 CHOICES_FOR_DIAMETER_COLUMN_D = [
41     COLUMN_SELECTION_NONE, AVERAGE_OF_COLUMNS_A_AND_B]
42 CHOICES_FOR_XOFFSET_AND_YOFFSET_COLUMNS = [COLUMN_SELECTION_NONE]
43
44 COLORMAPS = [
45     'cool', 'coolwarm', 'gray', 'hsv', 'inferno', 'magma', 'plasma',
46     'rainbow', 'seismic', 'viridis']
```

6.2.2 lslmdvcontrols

The controls displayed in the application GUI are defined in this module. Generation of these elements is enabled by some standard Python, **libgsalt.*** and **lslmdv...** modules that are imported into **lslmdvcontrols** as shown in Listing 7. A notable import is the module **libgsalt.gui.labelledcontrols** which combines wxPython widgets such as a *textbox* (wx.TextCtrl) or a *numeric control* (wx.SpinCtrl), with a *label* and a *caption* as text (wx.StaticText) laid out in a sizer (e.g. wx.FlexGridSizer). **labelledcontrols** allows these elements to be handled as a single entity in building the application and includes tools for validation and acceptance of the user input data entered into the controls.

Listing 7: **lslmdvcontrols** imports.

```
10 import wx
11 import libgsalt.gui.labelledcontrols as lc
12 import libgsalt.gui.icons as icons
13 import lslmdvbase
```

One class is defined in **lslmdvcontrols**. This class, whose parent is the wx.Frame class, is shown in Listing 8.

Listing 8: **lslmdvcontrols** class.

```
16 class MainWindowControls(wx.Frame):
17     """
18     The class provides the main interface for the application.
19 
20     Extends wx.Frame.
21 
22     """
23 
24     def __init__(self, parent, _id, title):
```

6.2.3 lslmdvplots

The plotting tools for the **Micrometer Data Visualizer** application are provided by this module. **lslmdvplots** calls in a number of modules to support its data handling and rendering. These packages are shown in Listing 9. One constant is defined in the module and shown in Listing 10. The module defines two "public" functions as shown in Listing 11.

Note that the Python language does not have strictly private functions.

Listing 9: **lslmdvplots** imports.

```
15 import numpy as np
16 from scipy.interpolate import CubicSpline
17 from matplotlib import pyplot as plt
18 from mpl_toolkits.mplot3d import Axes3D
19 import lslmdvbase
```

Listing 10: **lslmdvplots** constants.

```
22 DEFAULT_PLOT_OPTIONS = {
23     'offsets_along_x_axis': None,
24     'offsets_along_y_axis': None,
25     'colormap': lslmdvbase.COLORMAPS[0],
26     'plot_title': '',
27     'lofting_length_c_label': 'C',
28     'ellipse_width_or_height_a_label': 'A',
29     'ellipse_height_or_width_b_label': 'B',
30     'diameter_or_radius_d_label': 'D',
31     'show_representative_ellipses': False,
32     'show_profile_interpolations': False,
33     'hide_surfaces': False,
34     'number_of_points': 100,
35     'maximum_number_of_representative_ellipses': 10
36     }
```

The lofting functions in this package use solutions suggested on site Stack Overflow by Ted-Ort and Sabhuku (GS) for lofting ellipses in Matlab and Python respectively[9,10].

Listing 11: **lslmdvplots** functions.

```
def loft_ellipses(lofting_lengths_c, ellipse_widths_or_heights_a,
                  ellipse_heights_or_widths_b, plot_options=None):
    """
    Generate surface from ellipses.

    Parameters
    ----------
    lofting_lengths_c : pandas series
        The long axes for lofting the ellipses.
    ellipse_widths_or_heights_a : pandas series
        Widths or heights along one axis of the ellipses.
    ellipse_heights_or_widths_b : pandas series
        Widths or heights along another axis of the ellipses.
    plot_options : dict, optional
        Additional parameters to control the surface.
        The default is None.

    Returns
    -------
    None.

    """
    ellipse_loft_options = dict(DEFAULT_PLOT_OPTIONS)
```

```

def loft_circles(lofting_lengths_c, diameters, plot_options=None):
    """
    Generate surface from circles.

    Parameters
    ----------
    lofting_lengths_c : pandas series
        The long axes lofting the circles.
    diameters : pandas series
        Diameters of the circles.
    plot_options : TYPE, optional
        Additional parameters to control the surface plot.
        The default is None.

    Returns
    -------
    None.

    """
    circle_loft_options = dict(DEFAULT_PLOT_OPTIONS)
```

6.3 Generating Microsoft Windows Installers

In addition to the Python code described above and which can be run in a suitable environment, preferably a virtual Python environment such as those that can be set up with **venv** or **Anaconda**, the application has been successfully packaged as a Microsoft Windows installer. Thus it can be installed and run in Windows 10 (tested) or possibly other versions (≥Windows 7, but not tested) without the need to access any of the underlying code. Compilation of the Windows installer was done using PyInstaller[11]. To build the windows installer, on a Debian Linux platform (https://www.debian.org/), Python was installed on top of a Wine compatibility layer (https://www.winehq.org/) capable of running Windows applications on several POSIX-compliant operating systems. Some screenshots obtained when running the Windows version of the application are shown in Appendix A.

References

1. *Python 3.8.2 documentation*. https://docs.python.org/3/index.html. URL: https://docs.python.org/3/index.html (visited on 05/17/2020).
2. Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: https://doi.org/10.1038/s41592-019-0686-2.
3. Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.
4. Wes McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2017. ISBN: 978-1-4919-5766-0.
5. Wes McKinney and the Pandas Development Team. *pandas: powerful Python data analysis toolkit Release 1.0.3*. 2020. URL: https://pandas.pydata.org/pandas-docs/stable/pandas.pdf (visited on 05/17/2020).
6. N. Rappin and R. Dunn. *WxPython in Action*. Manning Publications, 2006. ISBN: 9781932394627. URL: https://books.google.com/books?id=RgvFQgAACAAJ (visited on 05/17/2020).
7. J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science Engineering* 9.3 (2007), pp. 90–95.
8. John Hunter et al. *Matplotlib: Release 3.2.1*. 2020. URL: https://matplotlib.org/Matplotlib.pdf (visited on 05/17/2020).
9. Teddy Ort (https://stackoverflow.com/users/1139661/teddy-ort). *How to loft with ellipses to create a 3d hollow pipe in MATLAB or Python? (Matlab Solution)*. stack overflow. eprint: https://stackoverflow.com/questions/42824105/how-to-loft-with-ellipses-to-create-a-3d-hollow-pipe-in-matlab-or-python. URL: https://stackoverflow.com/questions/42824105/how-to-loft-with-ellipses-to-create%20-a-3d-hollow-pipe-in-matlab-or-python%20(version:%2004-27-2020) (visited on 06/25/2020).
10. Sabhuku (https://stackoverflow.com/users/10913741/sabhuku). *How to loft with ellipses to create a 3d hollow pipe in MATLAB or Python? (Python Solution)*. stack overflow. eprint: https://stackoverflow.com/questions/42824105/how-to-loft-with-ellipses-to-create-a-3d-hollow-pipe-in-matlab-or-python. URL: https://stackoverflow.com/questions/42824105/how-to-loft-with-ellipses-to-create%20-a-3d-hollow-pipe-in-matlab-or-python%20(version:%2004-27-2020) (visited on 06/25/2020).
11. David Cortesi. *PyInstaller Documentation: Release 3.6*. 2020. URL: https://readthedocs.org/projects/pyinstaller/downloads/pdf/stable/ (visited on 05/17/2020).

Appendix A

Microsoft Windows Version of the Application

The **Micrometer Data Visualizer** application comes as multi-platform Python source code or as a Microsoft (MS) Windows installer. Here we show some screenshots from the MS Windows version running on Windows 10. Figure A1 shows the graphical user interface of the application on startup. Figures A2 and A3 show file selection and data visualization respectively. Figure A4 shows examples of some of the application's modal dialogs as they appear on Windows.

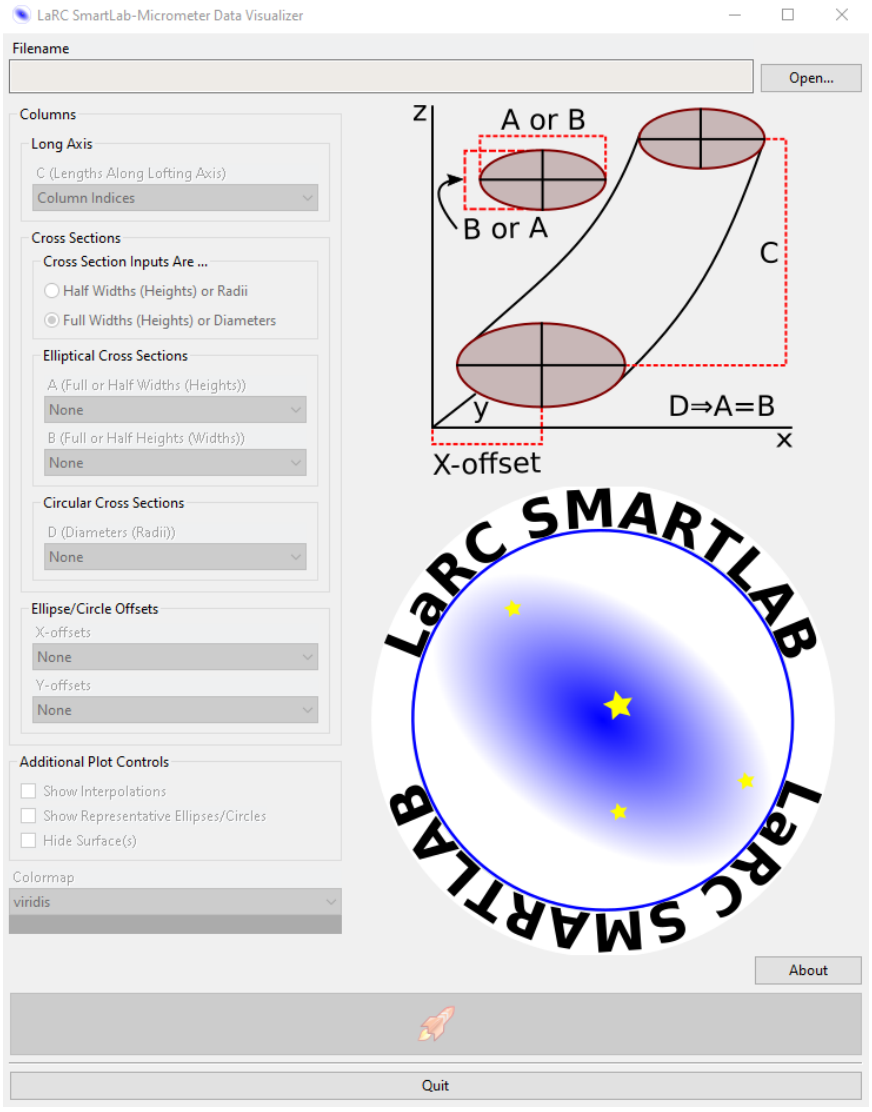


Figure A1: The GUI on application startup when running on MS Windows 10.

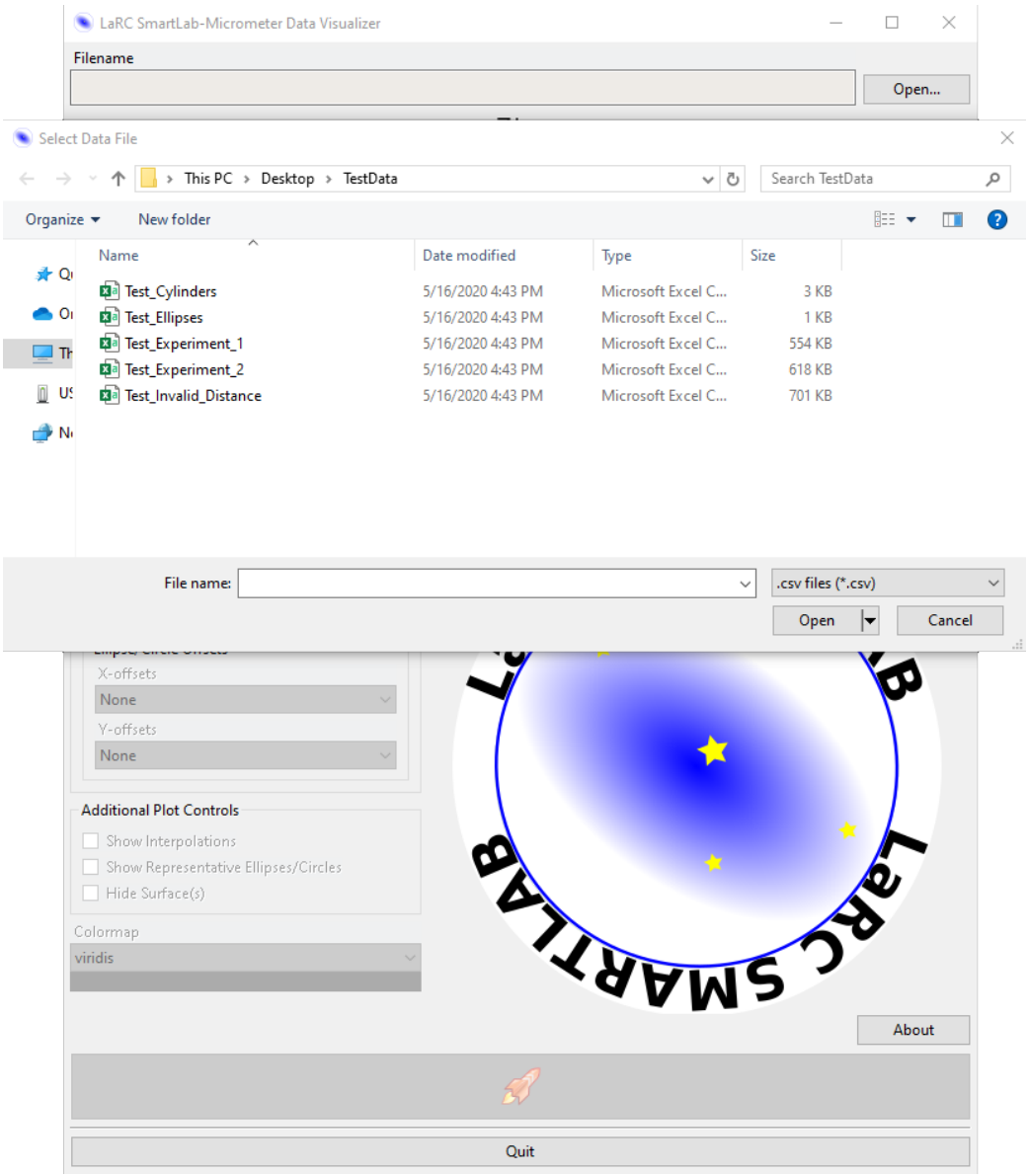
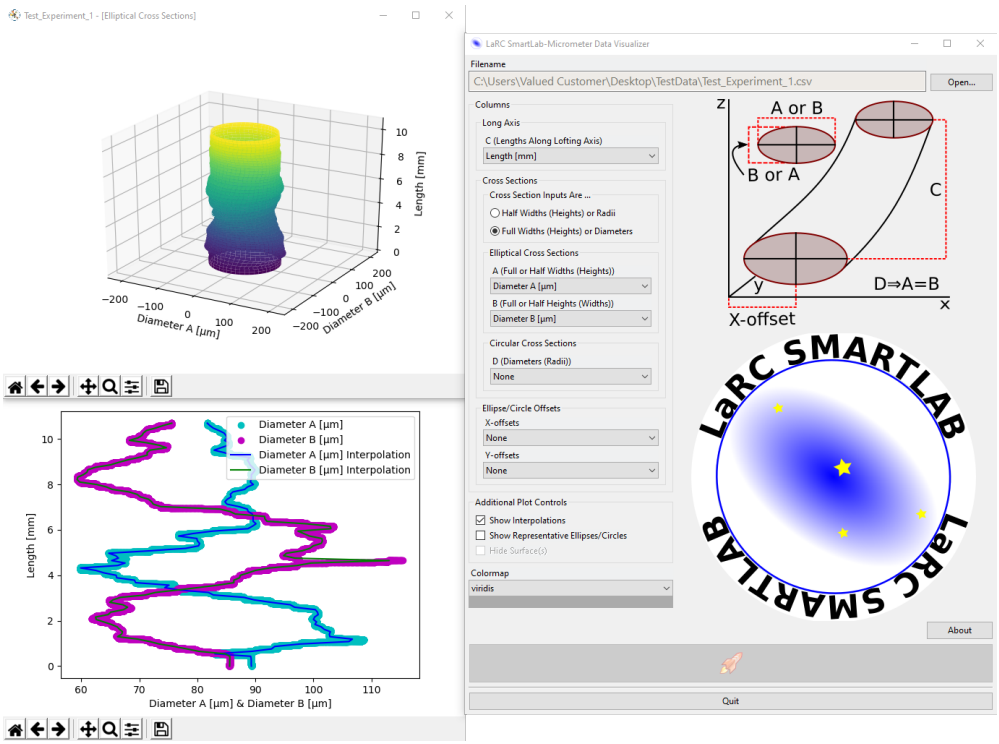
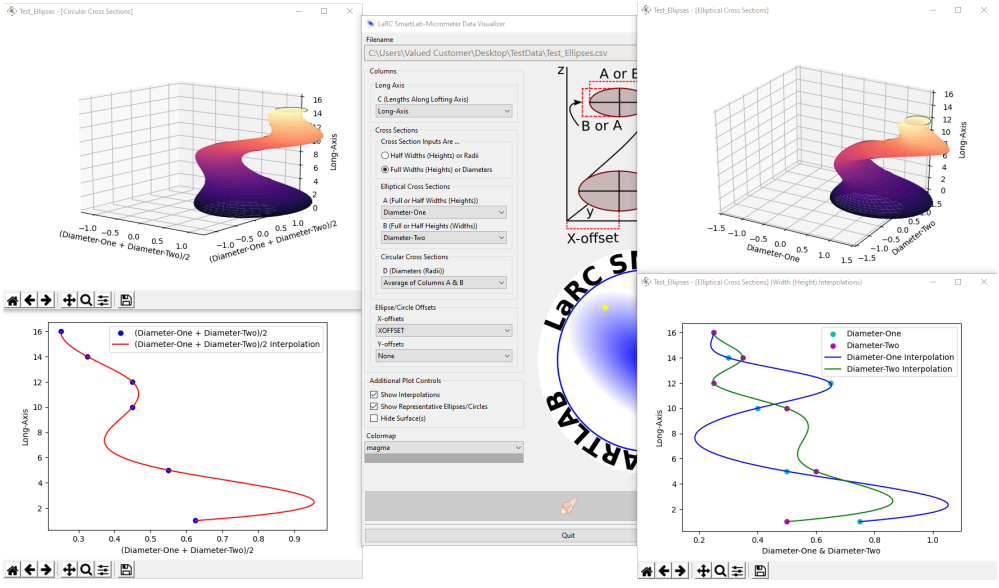


Figure A2: File selection dialog box while running on MS Windows 10.

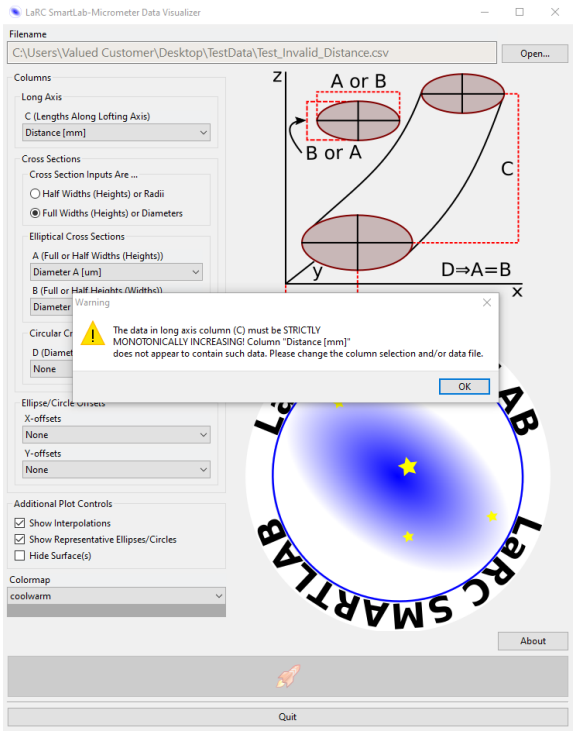


(a) Experimental data

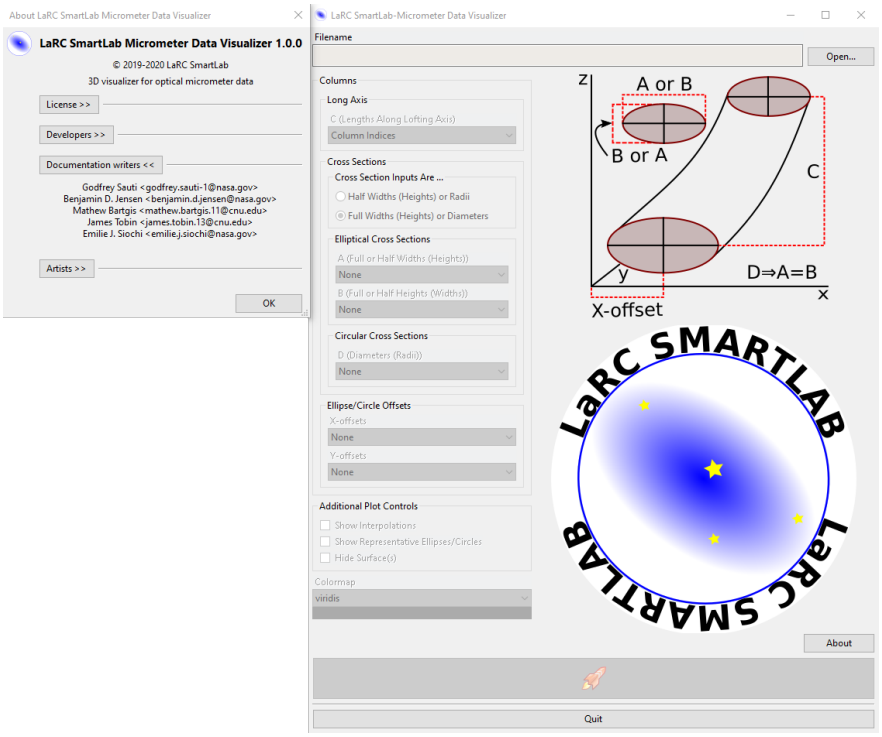


(b) Synthetic data

Figure A3: Visualizing experimental and synthetic data on MS Windows 10.



(a) "Long/Lofting Axis" data monotonicity warning



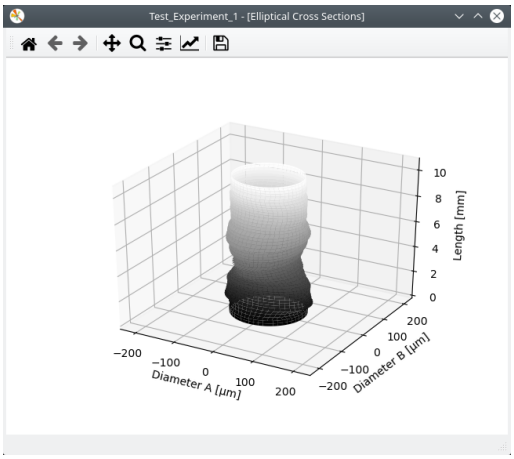
(b) About

Figure A4: Examples of the modal dialogs displayed on MS Windows 10.

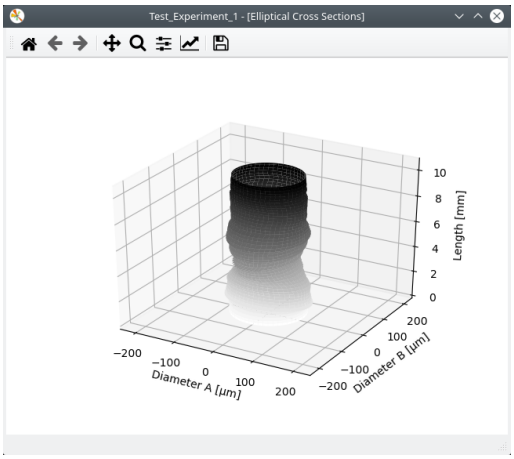
Appendix B

Examples Using Different Colormaps

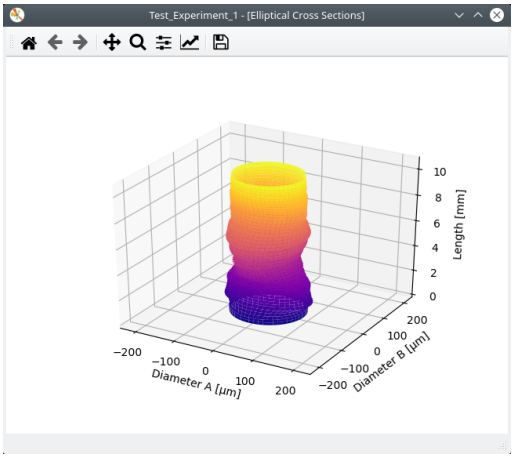
Figures B1 and B2 show examples of the colormaps that are available for the surfaces. When the application is run natively in a Python environment, even more options for colormaps (and other plot options) may be available to the user. Information on these is available in the videos that are described in Appendix C.



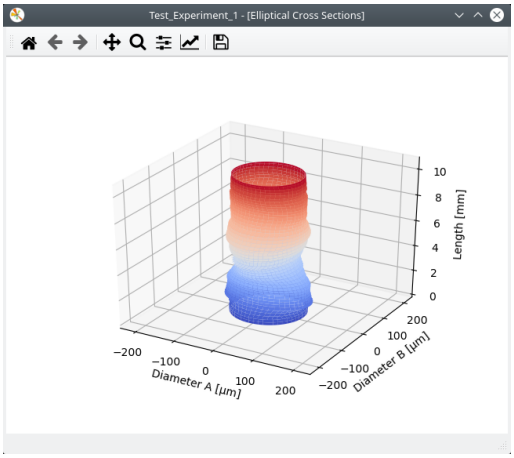
(a) "gray"



(b) "gray_r"

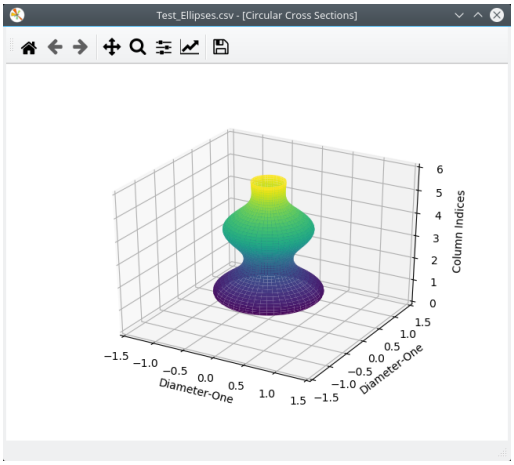


(c) "plasma"

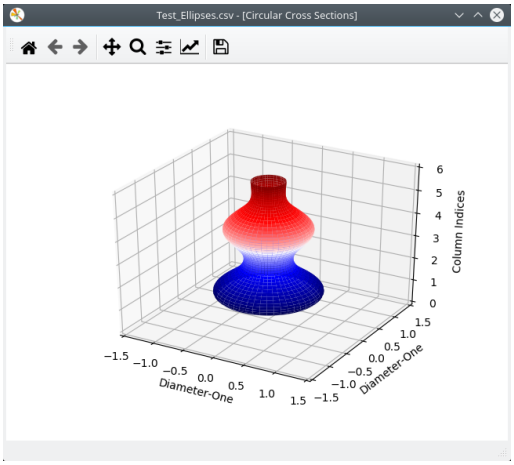


(d) "coolwarm"

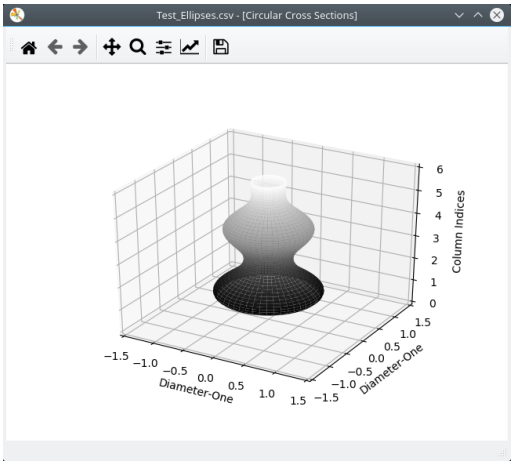
Figure B1: Examples of rendering experimental data using different colormaps.



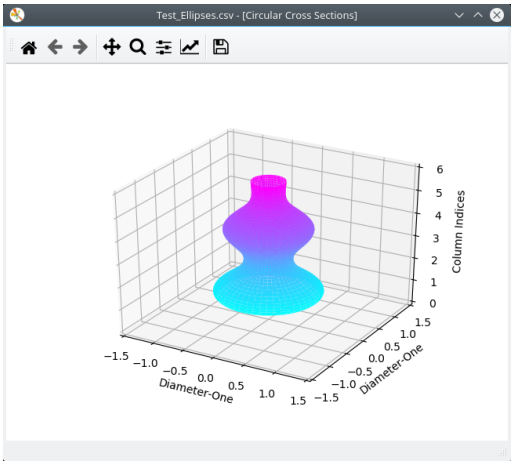
(a) “viridis”



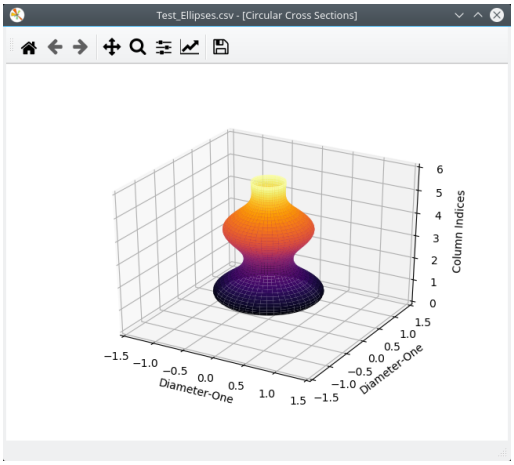
(b) “seismic”



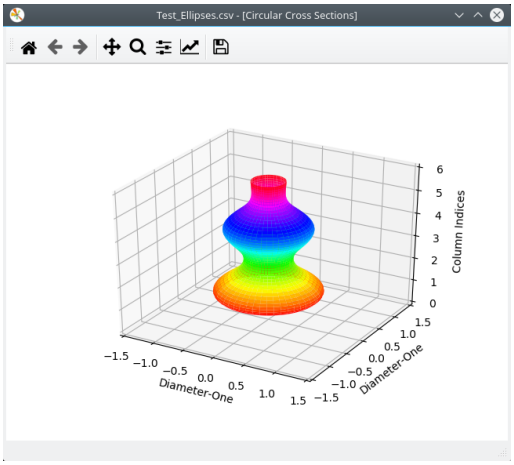
(c) “gray”



(d) “cool”



(e) “inferno”



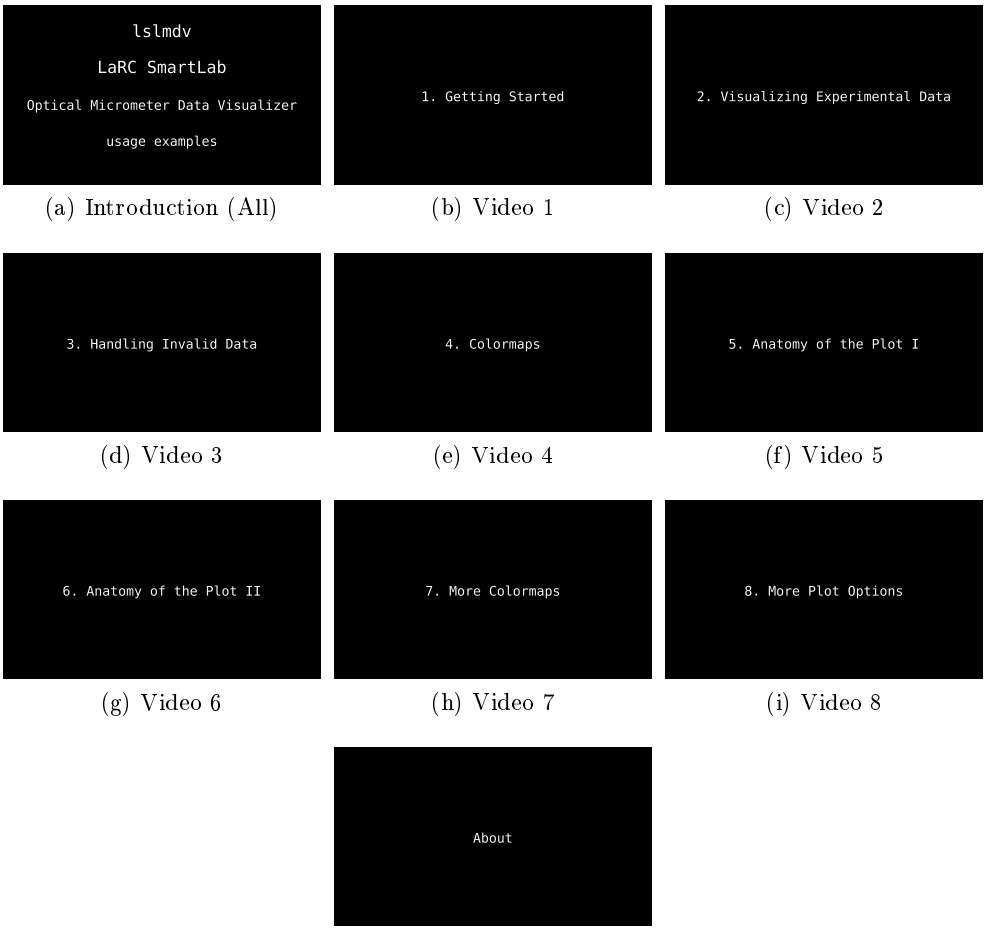
(f) “hsv”

Figure B2: More examples showing rendering of synthetic data using different colormaps.

Appendix C

Video Examples

The following videos showing use of the application are available as supplementary information:



(a) Introduction (All) (b) Video 1 (c) Video 2

(d) Video 3 (e) Video 4 (f) Video 5

(g) Video 6 (h) Video 7 (i) Video 8

(j) About (All)

Figure C1: Section headers from the videos with examples of the usage of the application.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-08-2020	Technical Memorandum	

4. TITLE AND SUBTITLE
LaRC SmartLab Apps For Instrument Control And Data Processing: Optical Micrometer Data Visualizer

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Godfrey Sauti, Benjamin D. Jensen, Mathew L. Bartgis, James D. Tobin, Emilie J. Siochi

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
736466.01.08.07.20.59.02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, Virginia 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S
NASA/TM-2020-5005521

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 61
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES
An electronic version can be found at http://ntrs.nasa.gov.

14. ABSTRACT

The **LaRC SmartLab** applications are a series of software tools to greatly enhance researcher efficiency by streamlining and automating workflows. Python scripts and applications are increasingly being used in scientific workflows, including for instrument control and data processing. Interactive Python scripting environments such as JupyterLab provide powerful tools for using Python. In some use cases, the development of standalone applications with dedicated graphical user interfaces (GUIs) can enhance the utility of the code and open it up to more users, including non-programmers. Here, we describe a GUI based optical micrometer data visualization application developed as part of the **LaRC SmartLab** project. We highlight its use in visualizing experimental data and briefly discuss its implementation to give pointers to programmers who wish develop work based on this application's or similar code.

15. SUBJECT TERMS
LaRC SmartLab, Python Programming, Graphical User Interfaces (GUIs), wxPython GUI Programming, Data Visualization, Optical Micrometer, Laser Micrometer

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Information Desk (help@sti.nasa.gov)
U	U	U	UU	58	19b. TELEPHONE NUMBER *(Include area code)* (757) 864-9658