PAPPy - Help File

Addison Ballif

June 2021

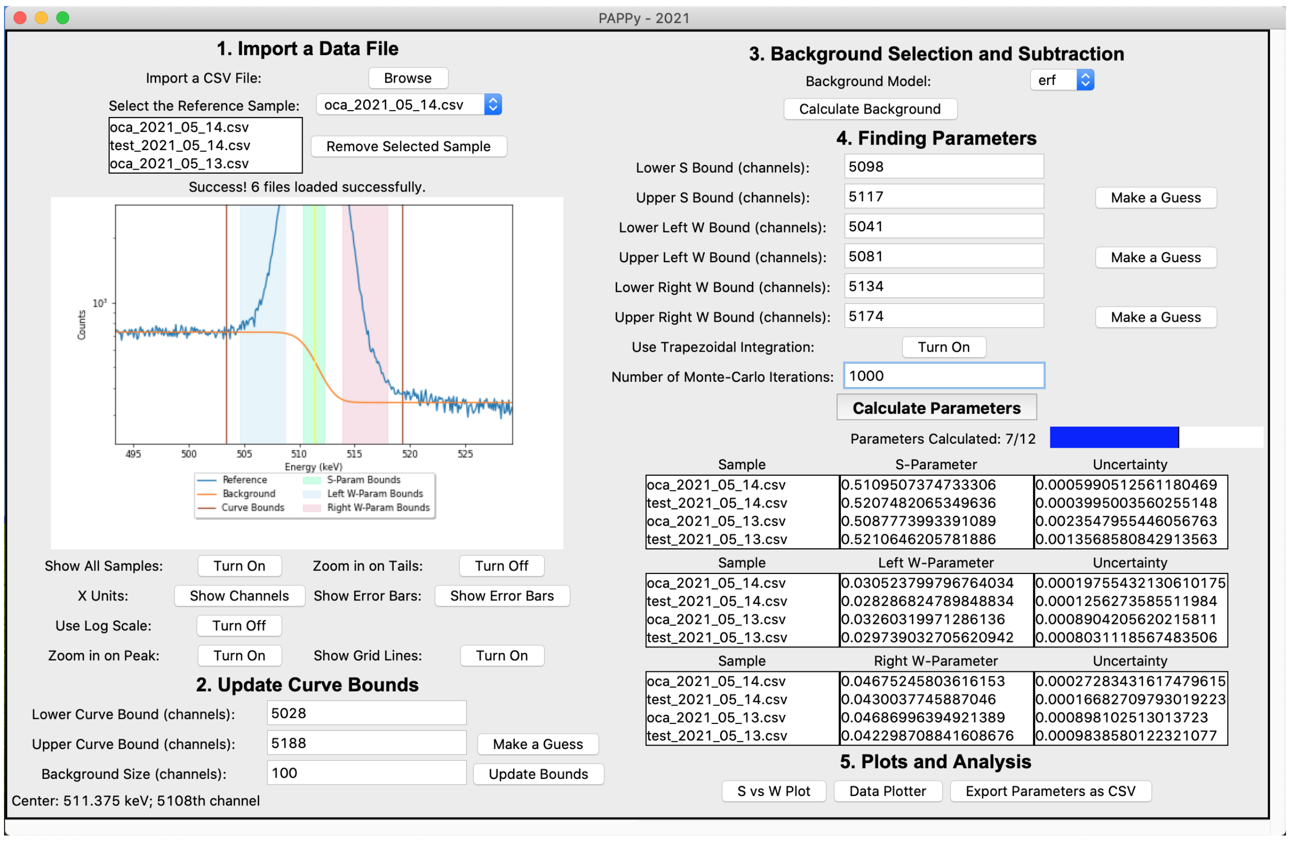
Contents

1 Introduction

In the Spring 2021 Semester, I was involved in the Positron Annihilation Spectroscopy team. My group's goal was to change the model of the subtracted background. Because we were changing part of the way the data is analyzed, a change to the PAS analysis program was necessary.

The program that was being used when I joined the group was PASDA. It was written in Matlab about 10 years ago by Timothy Harbison. I was able to get the code from Timothy Harbison, but because of it's age, I was not able to get it to run and was unable to add a new background model to the analysis program.

I decided to rewrite the program in Python. PAPPy is a program written in Python which was based of Harbison's PASDA program. Hopefully the rewrite into a new language will make modifying and taking control of the analysis program more accessible for future students.



oca_2021_05_14.csv
test_2021_05_14.csv
oca_2021_05_13.csv

Success! 6 files loaded successfully.

Lower Curve Bound (channels): 5028
Upper Curve Bound (channels): 5188
Background Size (channels): 100
Center: 511.375 keV; 5108th channel

Background Model: erf

Lower S Bound (channels): 5098
Upper S Bound (channels): 5117
Lower Left W Bound (channels): 5041
Upper Left W Bound (channels): 5081
Lower Right W Bound (channels): 5134
Upper Right W Bound (channels): 5174
Use Trapezoidal Integration: Turn On
Number of Monte-Carlo Iterations: 1000

Parameters Calculated: 7/12

Sample	S-Parameter	Uncertainty
oca_2021_05_14.csv	0.5109507374733306	0.0005990512561180469
test_2021_05_14.csv	0.5207482065349636	0.0003995003560255148
oca_2021_05_13.csv	0.5087773993391089	0.0023547955446056763
test_2021_05_13.csv	0.5210646205781886	0.0013568580842913563

Sample	Left W-Parameter	Uncertainty
oca_2021_05_14.csv	0.030523799796764034	0.00019755432130610175
test_2021_05_14.csv	0.028286824789848834	0.0001256273585511984
oca_2021_05_13.csv	0.03260319971286136	0.0008904205620215811
test_2021_05_13.csv	0.029739032705620942	0.0008031118567483506

Sample	Right W-Parameter	Uncertainty
oca_2021_05_14.csv	0.04675245803616153	0.00027283431617479615
test_2021_05_14.csv	0.0430037745887046	0.00016682709793019223
oca_2021_05_13.csv	0.04686996394921389	0.000898102513013723
test_2021_05_13.csv	0.042298708841608676	0.0009838580122321077

2 How To Use PAPPy

The purpose of PAPPy is to calculate S and W parameters from PAS data. S and W parameters can be used to examine vacancies in a material.

To run PAPPy, run the `PAPPy.py` file. Make sure that the other three code files are in the same folder. After running the python file, the window should appear.

> **Note:**
> On the BYU-I lab room computer, I have set up a bash script that will run PAPPy. The bash script should have a star shaped icon.

2.1 Requirements for PAPPy

PAPPy is a program written in Python. Python 3 is required to run the program, as well as the following libraries.

- Tkinter (comes default with python) - Code written using version 8.6
- Matplotlib - Code written using version 3.3.1
- Numpy - Code written using version 1.20.2
- Scipy - Code written using version 1.6.2

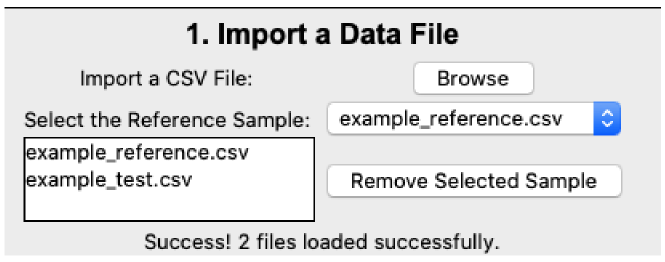
The code was written using python 3.8.3. I don't believe it uses any functionality that a slightly older version wouldn't have.

2.2 Importing the Data Files

The Software that we have been using to collect data at BYU-Idaho this semester is called Prospect. It reads energies from the connected detector, and stores the data as a histogram of photon energies. The data is exported as a csv file.

PAPPy can read these csv files. The top of the file contains some miscellaneous information, and the rest of the file contains a counts column, giving counts for each channel. Channels are listed in the bins column, and corresponding energy for each channel (based of the detector's calibration) is given in the energy column.

> **Note:**
> The Energy Data is used for plotting the data in a more reasonable way, but all calculations in the PAPPy program are based off of the channels data. Notice while using that program that all bounds entries are in channels and are integer values.



example_reference.csv
example_test.csv

Success! 2 files loaded successfully.

Figure 1: The file import section of PAPPy.

To import a CSV file, press the "Browse" button. This will import and load the data into the program.

When the program is first opened, two sample data files are loaded. To delete these files, or any other files you want to remove, select the file in the file list. You can then press the "Remove Selected Sample" button to remove the sample file. [1]

It is important to select the reference sample. This is the sample to which you want to compare all other files. You can select the reference sample using the drop-down menu. By default, the reference should be set to the first file loaded.

> **Note:**
> PAPPy assumes that the x-scale of each data file will be the same; it assumes that the width of each channel is the same for each file. If you change the number of channels between collection of the different samples, calculations in PAPPy will not be correct.

2.3 The Data Plot

After the data files are loaded, they will be shown in the data plot on the left side of the screen. You can adjust settings for the plot using the buttons below.

[1]To prevent the default files from being loaded, you can comment the two lines at the very bottom of file where it says "load some sample data. "

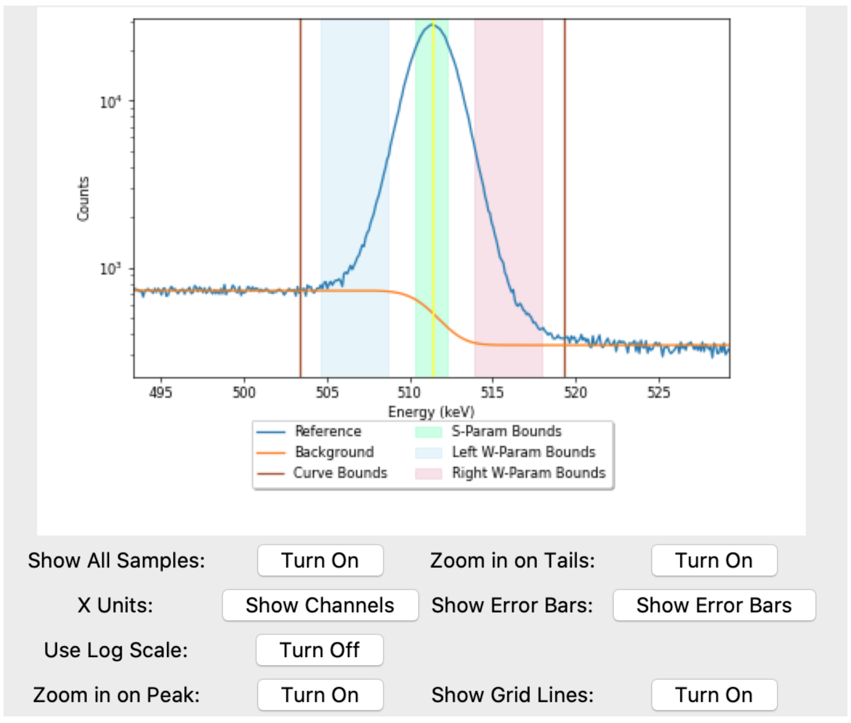


Figure 2: The data plot section of PAPPy allows the user to view the annihilation peak.

"Show All Samples" determines whether the plot should show all files, or just the reference sample. When only the reference sample is being shown, the background signal can also be plotted. Background will not be shown when all the samples are plotted.

"Zoom in on Tails" tries to set the y-axis limits of the plot so that the user can more easily see the shape of the background.

The x-axis units can be changed from energy (keV) to channels by pressing the button next to "X Units."

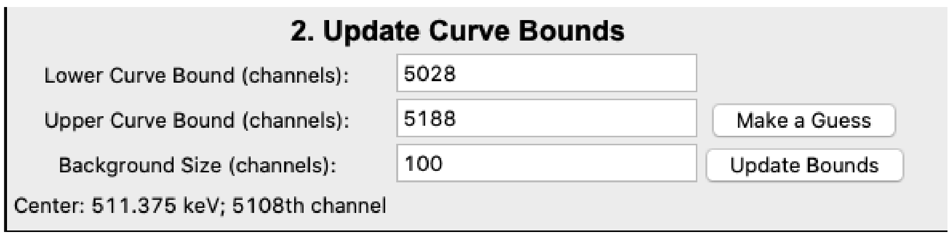
The error bars for the signal can be shown by pressing "Show Error Bars." The size of the error bar for each channel is equal to the square root of the counts in that channel. When a background model contains uncertainty calculations, the error bars for the background will also be plotted.

You can change the scale of the y-axis to a logarithmic scale by pressing the button next to "Use Log Scale."

Being able to see the parameter and curve bound regions is important for selection of good parameter and curve bounds. The "Zoom in on Peak" button will hide the background region, so you can more easily see the parameter regions. The "Show Grid Lines" function may also be used for better visualization.

For more control on the looks of the plot, see section 2.7.1

2.4 Selecting Curve Bounds


2. Update Curve Bounds

Lower Curve Bound (channels): 5028

Upper Curve Bound (channels): 5188

Background Size (channels): 100

Center: 511.375 keV; 5108th channel

Figure 3: The curve bounds selection window allows the user to set the bounds for the annihilation peak.

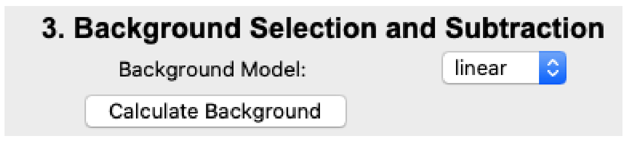
The next step is to select the region that captures the annihilation peak. When the data files are loaded, the program tries to guess a region for the curve bounds. The guess will be loaded in by default, but can be adjusted by the user.

You can also adjust the size of the background region on either side. This background region is used to calculate the background to be subtracted from the signal before parameter calculation. The default value is 100 channels, and this was the recommended amount in the PASDA program.

After changing the curve bounds or the background region size, you can press enter or press the "Update Bounds" button. To reset the bounds back to the software guess, press "Make Guess." (Pressing "Make Guess" will also update the bounds.)

2.5 Subtracting the Background Signal

It is our current understanding that the detector will read some photons incorrectly, and this will distort the measured signal. There will also be other photons throughout the entire spectrum from other sources that we want to ignore.


3. Background Selection and Subtraction

Background Model: linear

Figure 4: The background subtraction window allows the user to subtract a background signal from the data by choosing one of the background models.

PAPPy will subtract a background signal based off of a model of that signal. The original PASDA used a linear model, assuming the shape of the background beneath the correctly measured signal was a constant slope. Part of the reason for rewriting the program was to be able to add our new s-curve background model.

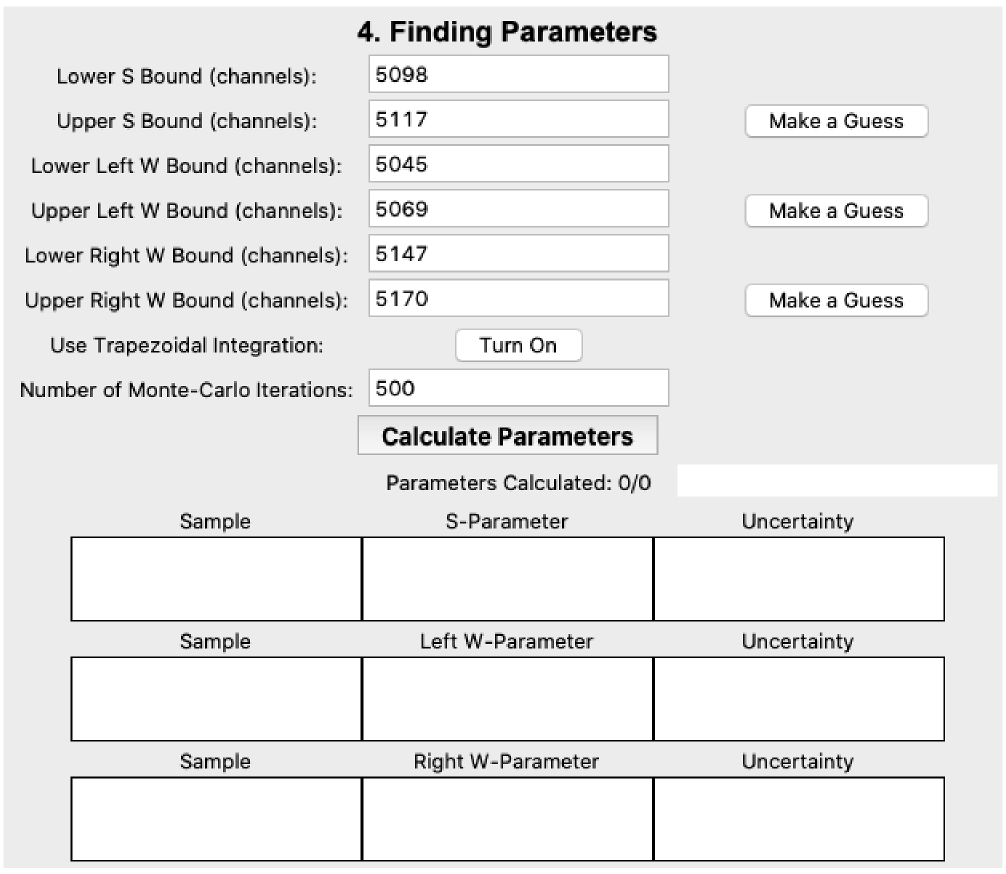
To subtract a background model, first select the model using the drop-down menu. You can press the "Calculate Background" button to see it on the graph. If you have "Use Log Scale" or "Zoom in on Tails" turned on, it will be easier to see the shape of the background model.[2] Background is re-calculated during parameter calculation, but the "Calculate Background" button will show the background shape on the graph.

> **Note:**
> Depending on the background model, the channels may not be independent after selection. Some background models use the Monte-Carlo method for uncertainty and parameter calculation. This requires recalculation of the background signal for each iteration. Be aware that the curve shown on the plot may not be the exact curve being used for calculation. For more info, see section 3.3.3

If you do not want a background signal to be subtracted, you can select the "none" option. All later calculations will work the same without background being subtracted. By default, PAPPy uses the linear model of background subtraction.

[2]Be aware that with the log scale on, the linear background will not look as linear.

2.6 Finding the Parameters



Lower S Bound (channels):	5098
Upper S Bound (channels):	5117
Lower Left W Bound (channels):	5045
Upper Left W Bound (channels):	5069
Lower Right W Bound (channels):	5147
Upper Right W Bound (channels):	5170
Use Trapezoidal Integration:	Turn On
Number of Monte-Carlo Iterations:	500

Parameters Calculated: 0/0

Sample	S-Parameter	Uncertainty

Sample	Left W-Parameter	Uncertainty

Sample	Right W-Parameter	Uncertainty

Figure 5: The primary purpose of PAPPy is to calculate S and W parameters. This is the window were the calculation takes place.

PAPPy can find S and W parameters. To calculate the parameters, enter the bounds for each parameter region. To allow the software to make a guess, press the "Make a Guess" button. The S parameter guess tries to pick a region that captures 50 percent of the center data on the reference. The W parameter guesses try to keep the W parameter bounds far away from the s-parameter region.

Note:
It is not required to put bounds in for each parameter. You only need to input the bounds for the parameter you want to calculated. Parameters with invalid or empty bounds will be ignored.

To use trapezoidal integration when calculating parameters, press the button next to "Use Trapezoidal Integration." When it is off, the program uses a Riemann sum. Uncertainty using trapezoidal integration is lower.[3]

When using the error function background model, the uncertainty is calculated using the Monte-Carlo method. The progress is shown using the progress bar when Monte-Carlo is being used. There is also a Monte-Carlo version of the linear model that also uses the error bar. The error bar is not used for calculations not involving Monte-Carlo method.

2.7 Plots and Analysis

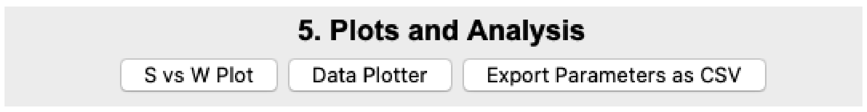


Figure 6: The plots and analysis section allows further analysis windows to be opened.

The option is there to add additional analysis to the program. Currently the program includes a simple data plotter, which can be used to produce more polished plots of the data, and an S vs W Plot. The S vs W plot can be used to compare how the two parameters are

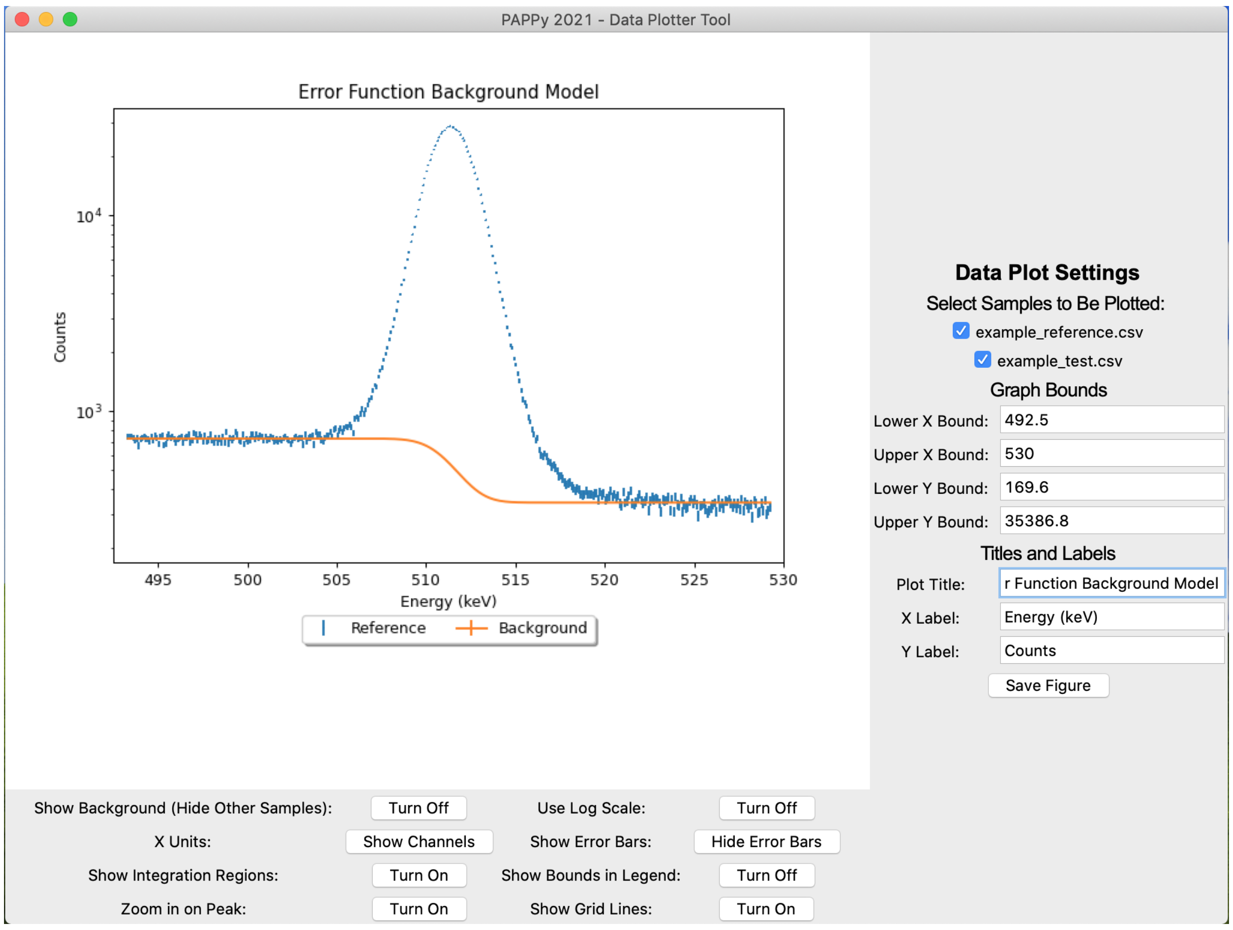
[3]The uncertainty in the trapezoidal integration is $\frac{1}{\sqrt{2}}$ of the uncertainty using a Riemann sum. See section 3.3.2

related for different samples. There is also a button that exports the calculated parameters in CSV format.

The code is setup in a way to make it easy to add additional functions to the "Plots and Analysis" section. Functions in this section open a new window, making it easier for others to add new GUI functions without having to touch the original code much.

2.7.1 The Data Plotter Tool

The Data Plotter Tool allows the user to make a more polished version of the data plot. The plot can be exported for use in reports or presentations.



Data Plot Settings

Select Samples to Be Plotted:

✓ example_reference.csv

✓ example_test.csv

Graph Bounds

Lower X Bound:	492.5
Upper X Bound:	530
Lower Y Bound:	169.6
Upper Y Bound:	35386.8

Titles and Labels

Plot Title:	r Function Background Model
X Label:	Energy (keV)
Y Label:	Counts

Save Figure

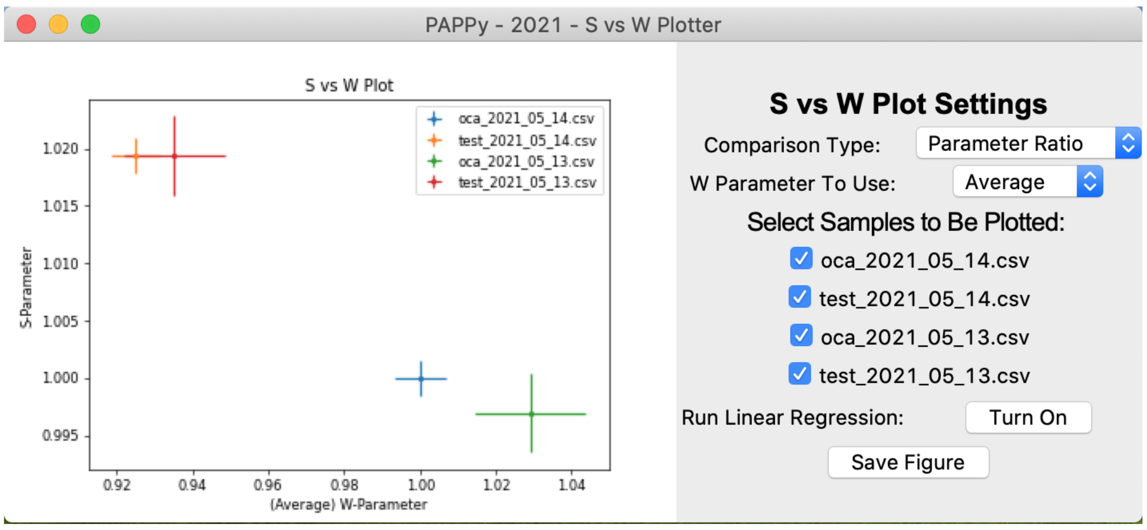
Show Background (Hide Other Samples):	Turn Off	Use Log Scale:	Turn Off
X Units:	Show Channels	Show Error Bars:	Hide Error Bars
Show Integration Regions:	Turn On	Show Bounds in Legend:	Turn Off
Zoom in on Peak:	Turn On	Show Grid Lines:	Turn On

Figure 7: The data plotter tool gives more control for the appearance of the data plot.

The user has control over the x and y limits of the graph, as well as the title, x-label, and y-label. Different samples can also be disabled from the plot. Many of the regular plot features are also still present.

A high quality image of the plot can be saved by pressing the "Save Figure" button.

2.7.2 The S vs W Plot



S vs W Plot Settings

Comparison Type:	Parameter Ratio
W Parameter To Use:	Average

Select Samples to Be Plotted:

✓ oca_2021_05_14.csv

✓ test_2021_05_14.csv

✓ oca_2021_05_13.csv

✓ test_2021_05_13.csv

Run Linear Regression: Turn On

Save Figure

PAPPy can create S vs W plots. To create an S vs W plot, first select how you want the parameters to be compared. “Absolute Parameter” will plot using the absolute values of each parameter. “Parameter Difference” will plot the difference between a given parameter and the parameter of the reference sample. “Parameter Ratio” will plot the ratio between a given parameter and the parameter of the reference sample.

The user can also select which W parameter to use for the x-axis. Select “Right,” “Left,” “Average,” or “Sum.” Selecting average will use the average of the two wing parameters, and sum will just use the sum of the two.

The user can also choose to run a linear regression on S vs W plot. The linear regression will be shown. By default, the S vs W plotter will plot all samples, but samples can also be disabled from the plot. This will affect the linear regression results.

3 PAPPy Parameter Calculations

The main purpose of PAPPy is to calculate S and W parameters. After importing and cropping the data to a reasonable range, the background is subtracted. The parameter can then be calculated with the subtracted signal.

The cropped data includes the curve bounds plus a certain amount of background on each side. This background region is determined by the background size input on the bounds selection window. The region on each side is used for calculating background.

3.1 Calculation of the Linear Background

Linear background is determined by taking the mean height of the background region on each side and interpolating a line through the peak region.

The height of the background in each side is the mean number of counts for each background region. The background regions do not overlap with the curve region. If the height of the left background region is B_l, the height of the right background B_r, and the lower and upper curve bounds are P_0 and P_f, then the slope of the line will be:

$$m = \frac{B_r - B_l}{P_f - P_0}$$

The slope will usually be negative. The intercept of this line will be:

$$b = B_l - mP_0$$

This gives us a line with the equation form:

$$y_b = B_l + \frac{B_r - B_l}{P_f - P_0}(x - P_0) \tag{1}$$

where x is the channel, and y_b is the background to be subtracted.

It is important to note that to find the background heights B_l and B_r, the background regions and the curve region must be independent and must not overlap. This is especially useful when finding the uncertainty.

To find the height of the left background region, we use the mean:

$$B_l = \frac{\sum y_i}{l_f - l_0 + 1}$$

where y_i is the number of counts in each channel, l_f is the lower bound for the left background region, and l_f is the upper bound in the background region. (In PAPPy, $l_f = P_0 - 1$ so the regions do not overlap. This is the case for the right background region as well.)

The partial derivative with respect to a single channel is:

$$\frac{\partial B_l}{\partial y_i} = \frac{1}{l_f - l_0 + 1}$$

Because we use counting statistics and assume that the uncertainties are Poisson distributed, we know that $\delta y = \sqrt{y}$. If we assume that each channel is independent, then we can propagate the uncertainty independently. Propagating the uncertainty using this partial derivative gives us:

$$\delta B_l = \frac{\sqrt{\sum y_i}}{l_f - l_0 + 1} \tag{2}$$

$$\delta B_r = \frac{\sqrt{\sum y_i}}{r_f - r_0 + 1} \tag{3}$$

The uncertainty propagation is the same with the right background region. With the uncertainty in the background height on each side, the uncertainty in the line can now be found.

$$\delta y_b = \sqrt{\left(1 + \frac{P_0 - x}{P_f - P_0}\right)^2 \delta B_l^2 + \left(\frac{x - P_0}{P_f - P_0}\right)^2 \delta B_r^2} \tag{4}$$

This is the equation PAPPy uses to find uncertainty in the linear background. Because the background signal and it's uncertainty were calculated based off of a region of the data which is separate from the curve bound region, we can say that the background and the original observed data are independent. The subtracted signal will be:

$$y_s = y - y_b \tag{5}$$

and the uncertainty will be:

$$\delta y_s = \sqrt{\delta y^2 + \delta y_b^2} \tag{6}$$

where $\delta y^2 = y$ and δy_b is given in equation 4.

3.2 Calculation of the Error Function Background

In the spring semester of 2021, my group worked on implementing a Gaussian error function model of the background signal. In order to find the s-curve that fits the data, PAPPy uses a two step fit.

The first curve fit is to determine the mean and standard deviation parameters of the Gaussian error function. Assuming the annihilation peak is close to the shape of a Gaussian[4], PAPPy fits a Gaussian to the data. The parameters which are being fit are the mean and standard deviation, as well as a scaling constant and a vertical shift constant. The first curve fit is given all of the data in the cropped region. It includes the background regions and the peak region. The mean and standard deviation of the error function correspond to the mean and standard deviation of the Gaussian, so we use the same parameters.

With the mean and standard deviation locked, PAPPy then cuts out the middle peak of the data, and fits an error function to the tails of the data. Data passed into the second curve fit is only the data in the background regions. The only two parameters being fit in this second stage are the scaling and vertical shift constants. The mean and standard deviation of the error function are locked to the values given from the first curve fit.

Because a two step curve fit approach is used, it is more difficult to propagate the uncertainty through the procedure. We also do not believe that the background is independent from the data because the first curve fit uses the data inside the peak region. To find uncertainties on s-parameter calculations, PAPPy uses the Monte-Carlo method and recalculates the background for each iteration. This means that this curve fit approach is done many times. With the Monte-Carlo method for propagating uncertainty, we can ignore how the uncertainty would be propagated analytically and we can just say:

$$y_s = y - y_b \tag{5 revisited}$$

3.3 Calculation of the Parameters

An S or W parameter is a measure of how much data is within an S or W region, compared to the amount of data in the whole curve. To find this ratio for a continuous function, we would use:

$$\text{S Param} = \frac{\int_{S_0}^{S_f} y_\mathrm{s}\,\mathrm{d}x}{\int_{P_0}^{P_f} y_\mathrm{s}\,\mathrm{d}x} \tag{7}$$

[4]We know that the true shape of the annihilation peak is not Gaussian. We are just making this assumption in order to find a reasonable standard deviation parameter. In reality, because the true curve is not Gaussian, the background subtracted should not be exactly a Gaussian error function.

where S_0 and S_f are the lower and upper parameter bounds, P_0 and P_f are the lower and upper curve bounds, and y_s is the data with background subtracted.

Because we don't have a continuous function, we will have to find some way to integrate numerically, and the method of integration will affect the uncertainty in the result.

3.3.1 Riemann Sum Integration

The first approach to integrating a region is to do a Riemann sum. This is the default method of integration in PAPPy. When integrating this way, the S parameter is found using:

$$\text{S Param} = \frac{\sum_{S_0}^{S_f} y_s}{\sum_{P_0}^{P_f} y_s} \tag{8}$$

Because we are just taking the sum of the counts in each channel, and each channel is assumed to be independent[5], then we can just add the variances for each channel.

$$\delta A_s = \sqrt{\sum_{S_0}^{S_f} \delta y_s^2}$$

$$\delta A_t = \sqrt{\sum_{P_0}^{P_f} \delta y_s^2}$$

where A_s is the sum of the data in the S parameter region, and A_t is the sum of the data for the whole curve. PAPPy uses equation 9 to calculate the uncertainty in each parameter.

$$\delta\text{S Param} = \sqrt{\left(\frac{\delta A_s}{A_t}\right)^2 + \left(\frac{A_s\, \delta A_t}{A_t^2}\right)^2} \tag{9}$$

3.3.2 Trapezoidal Integration

In an effort to improve the integration for the discrete function, PAPPy has an option to use trapezoidal integration.[6]

To find the area under a curve using trapezoidal integration, we use:

$$A = \sum_{x_0}^{x_f} \frac{y_i + y_{i-1}}{2} \Delta x$$

For the data being used, y_i is the number of counts in a given channel, y_{i-1} is the number of counts in the previous channel, x_0 and x_f are the lower and upper bounds for a given region, and Δx is the width of each channel. (The width of the channels won't matter for the ratio of the areas.) This area calculation can be used to find the area for the parameter or the total curve region.

$$\text{S Param} = \frac{A_s}{A_t} \tag{10}$$

The partial derivative with respect to each channel (or the previous channel) is one half. The uncertainty in the area of the S region (or the total area) can be found by propagating independent uncertainty.

[5]When it isn't independent, the error calculation will have to be different. This error propagation is mostly only used for the linear model. When the channels aren't independent, PAPPy uses Monte-Carlo method.

[6]We noticed that this is the way Joey Watkins calculates his S and W parameters in his PAS GUI.

$$\begin{aligned}
\delta A &= \sqrt{\sum\left(\left(\frac{\delta y_i}{2}\right)^2+\left(\frac{\delta y_{i-1}}{2}\right)^2\right)} \\
&= \sqrt{\sum\left(\frac{\delta y_i^2}{4}+\frac{\delta y_{i-1}^2}{4}\right)} \\
&= \sqrt{\frac{\delta y_1^2}{4}+\frac{\delta y_2^2}{4}+\frac{\delta y_2^2}{4}+\frac{\delta y_3^2}{4}+...+\frac{\delta y_n^2}{4}+\frac{\delta y_{n-1}^2}{4}} \\
&= \sqrt{\frac{\delta y_1^2}{4}+\frac{\delta y_2^2}{2}+\frac{\delta y_3^2}{2}+...+\frac{\delta y_{n-1}^2}{2}+\frac{\delta y_n^2}{4}} \\
\delta A &\approx \sqrt{\frac{\delta y_1^2}{2}+\frac{\delta y_2^2}{2}+\frac{\delta y_3^2}{2}+...+\frac{\delta y_n^2}{2}} \\
&\approx \sqrt{\sum\frac{\delta y_i^2}{2}} \\
&\approx \sqrt{\frac{1}{2}}\sqrt{\sum\delta y_i^2}
\end{aligned}$$

Propagating uncertainty this way gives:

$$\delta A_{\text{trapezoidal}} = \frac{\delta A_{\text{Riemann}}}{\sqrt{2}} \tag{11}$$

After the uncertainty in the S area and the total area is known, the uncertainty in the S parameter can be found using equation 9. The uncertainty is lower when using trapezoidal integration. Trapezoidal integration seems to be good for integration on a function that is assumed to be continuous in reality.

$$\delta\text{S Param} = \sqrt{\left(\frac{\delta A_s}{A_t}\right)^2+\left(\frac{A_s\,\delta A_t}{A_t^2}\right)^2} \tag{9 revisited}$$

Note:
The math shown in the previous sections was in terms of the S parameter, but calculations for each W parameter are the same but with different bounds.

3.3.3 Monte-Carlo Parameter Calculation

The previous uncertainty propagation techniques are only valid when the uncertainty in each channel is independent, and as long as the background and original data are independent within the curve region. For the error function background model, the background and original data are not independent, because the curve data is used to determine the shape of the background. For calculation of the uncertainty in the S parameter when the error function is used, the Monte-Carlo method of the background subtraction is used.

For each iteration, every channel is wiggled independently according to a Poisson distribution where λ is the number of counts in each channel. The error function background is then fit for each iteration. This unique fit is subtracted from the wiggled data, and the S parameter is calculated.

The mean and standard deviation of all the S parameters calculated are the value and uncertainty reported by PAPPy.

3.4 Calculations for Bound Guesses

PAPPy has functions that can make initial guesses for curve bound and parameter bound values. These guesses are based on a Gaussian fit of the curve. A Gaussian curve is fit to the data, and the mean and standard deviation of that fit are used to make guesses for upper and lower bounds.

The curve bounds are calculated as being six standard deviations away from the mean. S parameter bounds are calculated as being 0.725 standard deviations from the mean.[7] The

[7] If the data was a perfect Gaussian distribution this should be 0.6745 standard deviations away from the mean. The value was adjusted because the peak is not a perfect Gaussian, and I felt that a slightly larger z-score gave better S parameter results.

W parameter guesses are found by picking the region between 2 and 5 standard deviations away from the mean on either side of the curve.

4 How the Program is Set Up

The code for PAPPy is separated among four python files. The main file is called `PAPPy.py`. It contains all the GUI code for the main program. The second file is called `pasdatalib.py`, and it contains functions use for importing and exporting and transforming the data.

> **Note:**
> The `pasdatalib.py` file is also contained in a few places in my PAS folder. However, the version with PAPPy is an altered newer version. In order for PAPPy to work, the `pasdatalib.py` file must be the same version written for PAPPy. There should be a comment at the top identifying it as the correct version.

The third file is the `pasphysics.py` file. It contains all the physics for the main part of the program. It includes functions used for background subtraction and parameter calculation. It also contains the functions that make guesses for various bounds in the program.

The fourth file is called `analysisFunctions.py`. This file only contains additional analysis function such as the S vs W plotter or the data plotter. The file contains GUI and physics for these functions. For the analysis functions I included, the physics is included inside the GUI code. The main purpose of the analysis functions file was to make it easier for others to add new operations to the data. Analysis functions open a new GUI window, rather than operating inside the main program.

```
│ Main Application (class inheriting tk.Frame)
│ ├──Left Side (tk.Frame)
│ │  ├──File Picker (class inheriting tk.Frame)
│ │  ├──Graph Window (class inheriting tk.Frame)
│ │  └──Bounds Selection Window (class inheriting tk.Frame)
│ └──Right Side (tk.Frame)
│    ├──Background Selection Window (class inheriting tk.Frame)
│    ├──Parameter Window (class inheriting tk.Frame)
│    │  ├──Progress Bar (class inheriting tk.Canvas)
│    │  ├──S Parameter Table (ParamTable - class inheriting tk.Frame)
│    │  ├──Left W Parameter Table (ParamTable - class inheriting tk.Frame)
│    │  └──Right W Parameter Table (ParamTable - class inheriting tk.Frame)
│    └──Analysis Window (class inheriting tk.Frame)
├──S W Plotter Window (class inheriting tk.TopLevel)
│  ├──S W Graph (class inheriting tk.Frame)
│  └──S W Control (class inheriting tk.Frame)
│     └──Index Selector Window (class inheriting tk.Frame)
├──Data Plotter Window (class inheriting tk.TopLevel)
│  ├──Data Graph Window (class inheriting tk.Frame)
│  └──Data Control Window (class inheriting tk.Frame)
│     └──Index Selector Window (class inheriting tk.Frame)
└──Future Analysis Windows
```

Figure 8: A tree showing the layout and use of each class in the PAPPy code. The S W Plotter Window, Data Plotter Window, and future analysis function windows are independent from the main class structure. The tree includes which tkinter widget each class inherits. Note: The Left Side and Right Side objects inside the Main Application are not custom classes. They are just tk.Frames used to organize the other objects in the Main Application object.

The GUI for PAPPy uses an object oriented approach to Tkinter. Figure 8 shows the layout of the GUI and where each class is used. For the most part, many of the custom classes are only used once.

4.1 PAPPy Main File

The main file contains the GUI organization for the program. The GUI is split into various tkinter frames. The main file contains a class for each part of the main window.

The different classes take in a master argument when being initialized. This refers to the main application class. Through this master, the different windows can find the information they need. For example, when the graph window needs to access the bounds for the S parameter, it can call: `self.master.parameterWindow.lowerSBound`. Using a master rather than referencing parent allows access to anywhere in the program regardless of where the class is located.

Each GUI class will also take in a parent argument. This is mostly used by the tkinter library, but sometimes is used to find certain values. This is used in the analysis function classes. In this case, the master refers to the main window, while parent refers to the parent window of the analysis function.

Below is a summary of each class in the main file of the program. A summary of each class's purpose is also given in the code comments inside the file.

4.1.1 The Main Application Class

The main application class is declared at the bottom of the file. This class contains most of the data and information for the program. The data is stored in a variable which is contained in this class. The calculated background and uncertainties are all stored in this class. The bounds for the curve are stored in this class.[8]

Inside this class are two tkinter frames; one frame for the left side of the window, and one for the right. The purpose of using these two frames was to allow the rows of tkinter widgets on each side to be independent. The only reason to do this is to minimize empty space and unnecessary stretching of tkinter widgets.

Each other class in this file is used in the main application class.

The Crop Data Function

The last important function in the main application class is the Crop Data function. Data in the main application class is stored in several lists. The original data that is imported into the file is stored in the `bins0`, `energy0`, and `data0` lists. After it is cropped, the cropped arrays are stored in the `bins`, `energy`, and `data` lists.

The `data0` and `data` lists are lists of lists. The `data0` (and `data`) list contains each y-data set in list form. It is an NxM matrix, where N is the number of files and M is the number of channels in each file. The assumption is made that each file contains the same number of channels. X data (channels and energy) is only stored once for all files, in the `bins` and `energy` lists.

After data is cropped, a corresponding uncertainty list is created. Corresponding background and background uncertainty lists is also created. The background type is also reset to "None."

When the cropping function is called, the calculated parameters are also deleted.[9]

4.1.2 The File Picker

This is the part of the program where the user can import a file. The file picker doesn't contain any data stored in it except for the file path to each data file. These file paths are not used anywhere else in the program.

4.1.3 The Graph Window

The graph window is the part that graphs the data. It uses Matplotlib to graph the data and display it in the tkinter frame.

The only variables stored in this class are the boolean variables which determine the settings for the graph. Each boolean variable has a corresponding label and button that is

[8] The bounds for the different parameters are stored in the Parameters Window class. This is one of the few exceptions to storing everything in the Main Application.

[9] If the parameters are displayed in the Param Table, they are not removed. However, the S vs W plot and the export button will no longer export the calculated parameters.

used to toggle it on and off. Each boolean also has a setter function that can be used to toggle the boolean, or set it to a specific boolean value using the kwarg "value."

To update the graph window, you can call it's graph function. `GraphWindow.plotData()`. This function will erase the plot and re-plot the data using the current settings and axis limits.

4.1.4 The Bounds Selection Window

The bounds window is the part of the program where you set the bounds for the application. Rather than storing the bounds information in the BoundsWindow class, it directly accesses the bounds in the MainApplication class.

4.1.5 Background Selection Window

The background selection window has an options menu that is used to select the background type, as well as a button that calculates the shape of the background for each file, and stores it in the `self.master.backgrounds` list.

When the background is selected, the settings for the graph as also changed to show the calculated background. This is done using some setting methods in the graph window.

4.1.6 The Param Table

The param table is not a tkinter frame that is used by the main application class. Instead, it is a piece created for displaying the calculated parameters. It is used by the Parameter Window class, and is used to display the results.

4.1.7 The Progress Bar

Like the Param Table, the progress bar is not used by the main application. It is also used by the Parameter Window class to display the progress for Monte-Carlo calculations. While most classes in PAPPy inherit tk.Frame, this class inherits tk.Canvas.

4.1.8 The Parameter Window

The parameter window is the biggest part of the main application. It contains the inputs for the parameter bounds, as well as the outputs for calculated parameters.

The bounds for the different parameter regions are stored in the parameter window class, rather than being stored in the main application. When plotting the parameter bounds, the graph window refers to the parameter window.

This class contains the function that calculates parameters for each sample. However, the math for the calculations is in the `pasphysics.py` file.

4.1.9 The Analysis Window

The analysis window contains buttons that trigger the creation of other GUI windows. When adding a new analysis function, you will want to make sure to add a label, button, and function to the analysis window to trigger the creation of the new analysis function GUI.

It also contains the button that will export the data.

4.2 PAS Data Library File

The `pasdatalib.py` file contains functions that import the data, export the data, and transform the data.

The functions used by the main application class to crop the data are also contained in this file. They are not the same functions as in older versions of this file. They have been updated to use channels as the primary x-axis.

There are also some unit conversion functions to switch between channels and keV.

4.3 PAS Physics File

The Physics File contains the main physics for the application. Hopefully putting the physics in it's own file makes it easier for others to add better physics without having to touch the GUI code for PAPPy.

For more on this file, see section 5.1.

4.4 Analysis Functions

The analysis function file is set up similar to the main `PAPPy.py` file. However, rather than just having a single main application class, there is a main window class for each analysis window.

Currently there are two main analysis windows: the S vs W plotter, and the data plotter. There is also another class and a method that are not specific to these two analysis functions: the Index Selector Window, and the `getSaveFilePath()` method. These can be used by any of the analysis functions.

If it is useful to those who want to add new analysis functions, you could also make a new file for each new analysis function added. There is no reason why it should have to be in this file, as long as it gets imported correctly into the main file.

4.4.1 The Index Selector Window

The index selector is a tkinter frame that shows a checklist with each data file imported. Different files can be selected or deselected, and the class can return a list of indices according to which files are selected. This is useful for plotting specific files rather than all of them. It is used by the S vs W Plotter, and the Data Plotter.

4.4.2 The S vs W Plotter

The S vs W Plotter is built of three main classes. The S vs W Plotter Window is the main window for the S vs W plotter. It is a tkinter Top Level widget. This allows it to be independent from the main window.[10]

The S vs W Plotter Window contains the S vs W Control class and the S vs W Graph class. The graph class is a tkinter frame that uses Matplotlib to plot the S and W parameters. It gets access to the parameters by using the master argument, which is assigned during initialization. The S vs W Control class contains the Index Selector Window as well as a few other buttons to control the looks of the graph and export the results.

The S vs W plotter can run linear regression on the parameters. The `scipy.linregress` library is used to do linear regression. It does not take into account the uncertainty on each parameter. Perhaps a different method could be implemented in the future that uses the uncertainty for linear regression calculation.

4.4.3 The Data Plotter

The data plotter window is set up similar to the S vs W Plotter, but is uses the data plotter shown on the screen of the main window. It is slightly different because it allows the user to adjust the looks of the graph a bit more.

Note:
The Data Graph Window is the class that contains the graph for the Data Plotter. While I originally copied and pasted it from the main program, it is slightly different and should not be used in the main program window.

5 How To Modify the Program

If your goal is to modify something in the program, it will first be important to read section 4. It explains the general setup of the program. Subsections here further explain how to do specific additions or changes to the program. Depending on what you want to change, the section above may be useful to you.

[10]All additional analysis functions should be created using the tkinter Top Level. You can use the two included analysis functions as a reference.

5.1 Adding a Background Model

Adding a background model is one of the easiest additions to make to PAPPy. The first step is to give the name of the background model. It should be put in the `background_models` list as a string.[11]

The next step is to add an `elif` statement inside the `find_background_shape()` function. Here you can calculate the shape and uncertainty of the background signal to be subtracted. The information available to you for background calculation is the channels (bins) and counts data, as well as the lower and upper curve bounds. Make sure that you return the background and uncertainty arrays as a tuple; `return background_array, uncertainty_array`.

If your background model is not independent from the original data, or if subtraction of your background model will result in the channels not being independent each other, then you will have to add an additional `elif` statement in the `calculate_parameter()` function for a more specialized parameter calculation. If you do not add the `elif` statement, the background and channels will be treated as independent.

Through the kwargs, the setting to use trapezoidal integration is also passed in. If you add a new `elif` statement to the `calculate_parameter()` function, make sure you keep this functionality. See the original code for an example.

Using Monte-Carlo Method for Parameter Calculation

For the error function background model, we chose to use Monte-Carlo method for our parameter uncertainty calculation. If you choose to use Monte-Carlo (or another calculation technique that takes a lot of time), you can use the kwargs parameter to attach progress bar functionality. The progress bar class can be connected through the `kwargs.get('progress_bar', None)`. See the two Monte-Carlo examples in the code for reference.

5.2 Adding an Analysis Function

To add an analysis function, you can either add it to the `analysisFunctions.py` file, or you can make a new file. If you make a new file, make sure to import it into the main `PAPPy.py` file.

To make an analysis function, you must start with a class that inherits tk.TopLevel. For the initialization function, make sure that you take in the *args, and **kwargs as you normally would for an object oriented approach to tkinter. The tkinter TopLevel does not require a parent. Also take in a master. This gives you access to all the information from the main program. An example initialization is shown below.

```
self.__init__(master, *args, **kwargs):
    tk.TopLevel.__init__(*args, **kwargs)
    # Do cool stuff here
    self.mainloop() # Call this at the end
```

You can access any part of the program using `self.master.anything_you_are_looking_for`. Most information will be stored directly in master. The only reason to look inside a different class would be to get the parameter bounds, or graph settings. For example, to get the lower S parameter bound, you would call `self.master.parameterWindow.lowerSBound`.

To connect the top level to the main program, you need to add a button and caller function inside the analysis class inside `PAPPy.py`. See the data plotter button and function for reference. The function just has to initialize the new analysis class (that inherits tk.TopLevel), and the button just runs the function.

After you have created the tk.TopLevel class and connected it to the main program, you can set up the analysis function however you want. (Be careful not to use master to change variables. Try to only use master to read information. This is to protect the information and function of the original program.)

An object oriented approach to tkinter is recommended, but anything set up inside the new Top Level class will not affect the original program. Use the new Top Level class as the parent for the main tkinter widgets. For example, if you want to create a label

[11] Don't put the new background model name at the beginning of the list. The first item in this list is used as the default background model. Because linear is the simplest and fastest, it is best to keep the linear model as the default.

in your new analysis window, you would call use: `tk.Label(self, text='This is a new label').grid(row=0, column=0, columnspan=2)`, or something to that extent. The first argument for the tkinter widget is the parent, and would be set to self, because it should be contained in the new Top Level class.

5.3 Adding To or Changing the GUI in the Main Program

Changing the main GUI is not recommended, but if there is a change that needs to be done, you should make sure to study the layout of the program first.

Each section of the program is contained in a separate class (see Section 4). This should make it relatively easy to find the part you are looking for.

If a new functionality is being added that does not affect calculation of the parameters, then it is recommended to create an analysis function window instead. However, if the goal is manipulate the data before calculating the parameters, then it could go in the main GUI.

On smaller screens, the GUI is quite filled. It may be useful to apply a tkinter Top Level technique to a process that can be done before parameter calculation. This would be done similar to the creation of an analysis window, except you would have to use the master for setting values as well.

For example, deconvolution of the data would be an interesting feature to add. Because it would involve some inputs, it would be good to add a button that opens a deconvolution window, rather than trying to fit it all on the main window. This new window would start as a class inheriting tk.TopLevel. It would take in the master. Because it has to manipulate the data, it would deconvolve the data, and save the deconvolved data in the master data lists. It would have to overwrite the original data lists, which is probably okay.

For small changes in the GUI, it should be relatively easy to find the section you want to change, and make a small addition or change. For example, adding a graph option is a simple change that only requires addition of a label, button, and function in the GraphWindow class.

When making changes to the original GUI, be aware of the size of the screen. While the application contains scroll bars in the vertical and horizontal directions, it is not very pleasant to use them. The application will not let the window start bigger than a maximum size. The user can scale the window larger. However, the maximum initial size is one that I feel is reasonable for most computers. Try to not add more than will fit comfortably in the original maximum dimensions.

6 Conclusion

Hopefully the new program is useful to students in the future. I do feel that there are changes to be made. I tried to set it up in a way that was easy for others to change.

For questions on altering the program, you can contact me at `addisonballif@gmail.com`. I may not be available during the 2021-2023 period.

Thanks,

Addison Ballif