

Test Report: SpectrumImageAnalysisPy

Isobel Bicket

December 18, 2017

1 Revision History

Date	Version	Notes
December 18, 2017	1.0	Initial draft

2 Symbols, Abbreviations and Acronyms

See SRS document for further definitions.

symbol	description
RMS	Root Mean Square
SNR	Signal-to-Noise Ratio
T	Test

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Functional Requirements Evaluation	1
3.1	R1: SI Inputs	1
3.2	R2: Spectrum Inputs	1
3.3	R3: Input Verification	1
3.4	R4: Slice (1D) SI and extract Image	1
3.5	R5: Mask (2D) SI and extract Spectrum	1
3.6	R6: Richardson-Lucy Deconvolution	2
3.7	R7: Normalization	5
3.8	R8: Background subtraction	5
3.9	R9: Gain Correction	5
4	Nonfunctional Requirements Evaluation	6
4.1	Usability	6
4.2	Performance	6
4.3	etc.	6
5	Comparison to Existing Implementation	6
6	Unit Testing	6
7	Changes Due to Testing	6
8	Automated Testing	7
9	Trace to Requirements	7
10	Trace to Modules	7
11	Code Coverage Metrics	7
11.1	Statement Coverage	7
11.2	Branch Coverage	7

List of Tables

1	Statement coverage	7
2	Branch coverage	8

List of Figures

1	Original spectrum (0), Point spread function (PSF), the original spectrum convolved with the PSF (conv), and the original spectrum convolved with the PSF with Poisson noise added (noisy). Three images show three different regions of the spectrum.	2
2	Spectra deconvolved using the Richardson-Lucy algorithm. Different numbers of iterations were used, as shown in the legend (RL2000 = 2000 iterations). Four images show different regions of the same plot.	3
3	Root mean square error against number of RL deconvolution iterations performed.	4
4	Signal to noise ratio against number of iterations	5

This document details the results of the testing detailed in the Test Plan, for the code SpectrumImageAnalysisPy, which can be found, along with the other documentation, on https://github.com/icbicket/SpectrumImageAnalysisPy/tree/SpectrumImageAnalysisPy_dev. Data and results from testing can be found under </home/isobel/Documents/McMaster/PythonCodes/DataAnalysis/src/TestCase>.

3 Functional Requirements Evaluation

In this section, the testing for each of the functional requirements is addressed. Some testing is still in progress and will be completed at a later date.

3.1 R1: SI Inputs

Some **unit testing** has been performed for error checking on initializing a spectrum image. More remains to be done.

3.2 R2: Spectrum Inputs

Testing remains to be done.

3.3 R3: Input Verification

Some **unit testing** has been performed for error checking on initializing a spectrum image. More remains to be done.

3.4 R4: Slice (1D) SI and extract Image

Some **unit testing** has been performed on the 1D slice module in Spectrum.

3.5 R5: Mask (2D) SI and extract Spectrum

Testing remains to be done.

3.6 R6: Richardson-Lucy Deconvolution

The test data for testing the RMS error and SNR of the deconvolution algorithm were obtained from Dr. E.P.Bellido and are the same datasets used in [1]. The inputs to the deconvolution function are shown in Figure 1. The original spectrum is simulated, with a narrow Gaussian representing the zero loss peak. An experimental point spread function was convolved with it to represent the blurring of the 'ideal' spectrum with the point spread function of the CCD. Poisson noise was added to the result of this convolution to imitate noise from the detector.

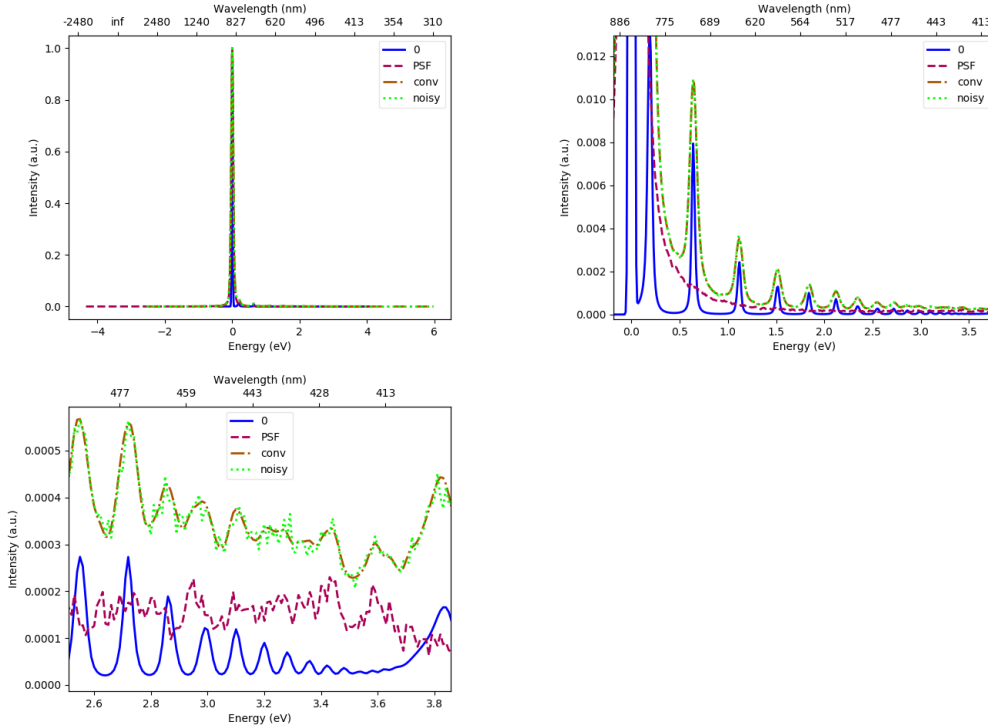


Figure 1: Original spectrum (0), Point spread function (PSF), the original spectrum convolved with the PSF (conv), and the original spectrum convolved with the PSF with Poisson noise added (noisy). Three images show three different regions of the spectrum.

The goal of deconvolution is to retrieve the original spectrum from the spectrum convolved with the PSF. Up to 2000 iterations of the Richardson-Lucy deconvolution algorithm have been performed on this dataset, using

the experimental point spread function, and the noisy convolved spectrum as inputs. The results at several chosen values are shown in Figure 2. As the number of deconvolutions increases, the spectra quickly approach the 'ideal' spectrum (RL1, RL50). Further increments represent small benefits in the approach to the 'ideal' spectrum (RL107-RL350). Using even more iterations results in periodic artifacts starting to appear in the spectrum (RL1000, RL2000), where small peaks appearing in between the main peaks can be seen and grow larger with further iterations.

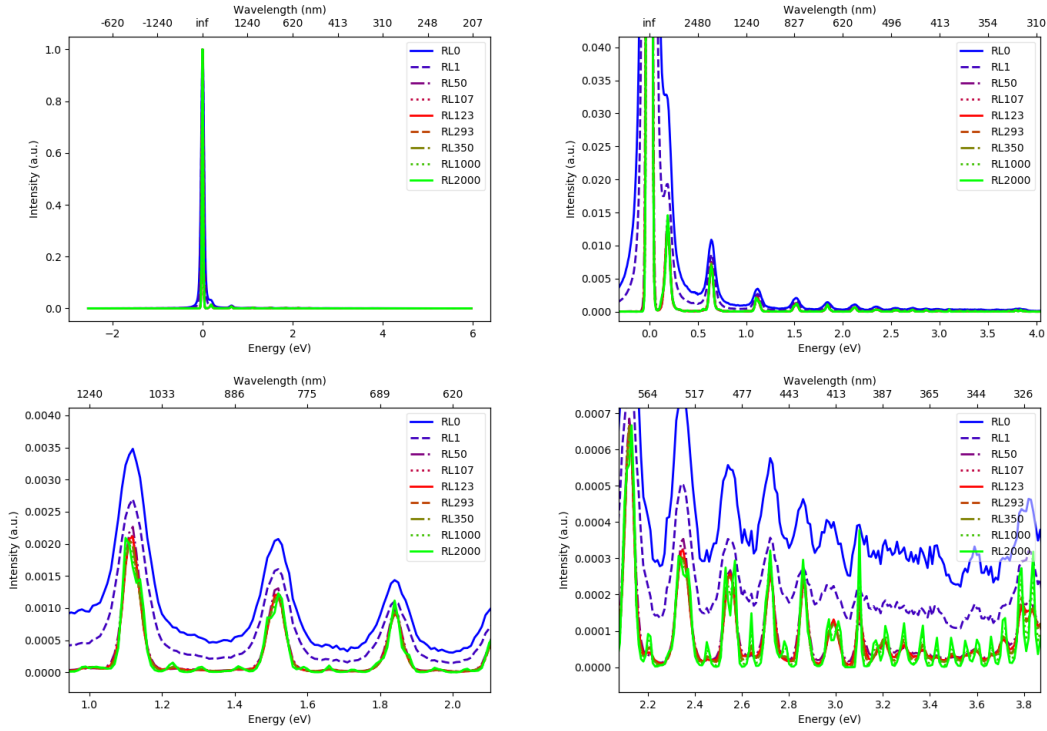


Figure 2: Spectra deconvolved using the Richardson-Lucy algorithm. Different numbers of iterations were used, as shown in the legend (RL2000 = 2000 iterations). Four images show different regions of the same plot.

To quantify the difference, the RMS error is plotted against the number of iterations in 3. There is at first a rapid decrease in the RMS error, ending in a local minimum at around 100-120 iterations (the exact value of the minimum depends on the noise added). Following this, the error increases as the periodic artifacts begin to appear and grow larger. The minimum RMS

error at around 100-120 iterations represents about 0.12% error.

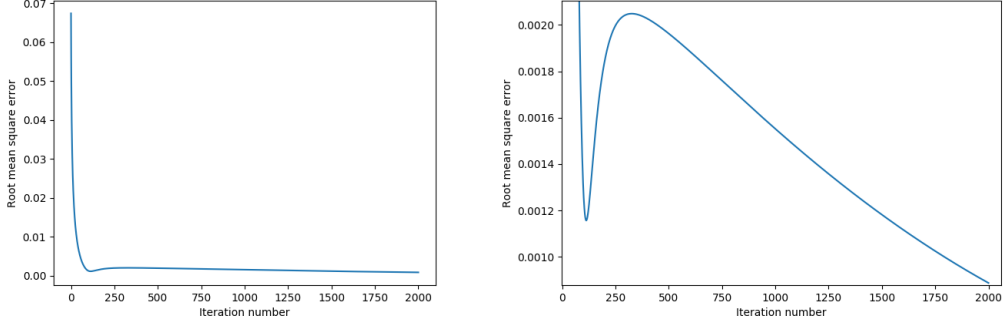


Figure 3: Root mean square error against number of RL deconvolution iterations performed.

To compare more directly with the data from Bellido *et al.* [1], the signal-to-noise ratio is plotted against the number of iterations in Figure 4. As was done by Bellido *et al.*, the calculation for the SNR was split into two sections in the spectrum. The SNR-ZLP section represents the SNR calculated within 0.1 eV of the zero loss peak (at 0 eV), and the SNR-plasmon section represents the spectrum from 0.1 eV to 4.0 eV (the 'plasmon region' of the spectrum). The SNR increases rapidly with the first few iterations, followed by a peak at a similar value to the minimum in the RMS error. Interestingly, the peak in SNR does not coincide exactly with the RMS minimum, but is slightly offset in the number of iterations. There is also a small difference between the SNR-ZLP and the SNR-plasmon peaks. After the peak, the SNR-ZLP begins to increase. The ZLP is easily the strongest feature in the spectrum: adding noise has very little effect on it and the SNR increases with the number of iterations (after a local minimum). The SNR-plasmon, on the other hand, has very little signal and the Poisson noise can easily dominate or replace some of the smaller peaks. After the local maximum, the SNR-plasmon begins to decrease and continues to decrease as the number of iterations increases. Inspecting the results of the deconvolution in Figure 2, it is apparent that this is caused by the periodic artifacts appearing between the peaks and the way that some of the peaks have split into two peaks (*e.g.*, near 2.6 eV).

These results are quite similar to those of Bellido *et al.*, though not exactly the same (likely due to the different Poisson noise used, since the Poisson noise was generated independently for this test). Although it appears that

there is a further decrease in error and an increase in the ZLP-SNR with large numbers of iterations, the results on the SNR-plasmon suggest that there is an optimum number of iterations (dependent on the amount of noise) to balance the RMS error and SNR for the region of interest, the plasmon region of the spectrum.

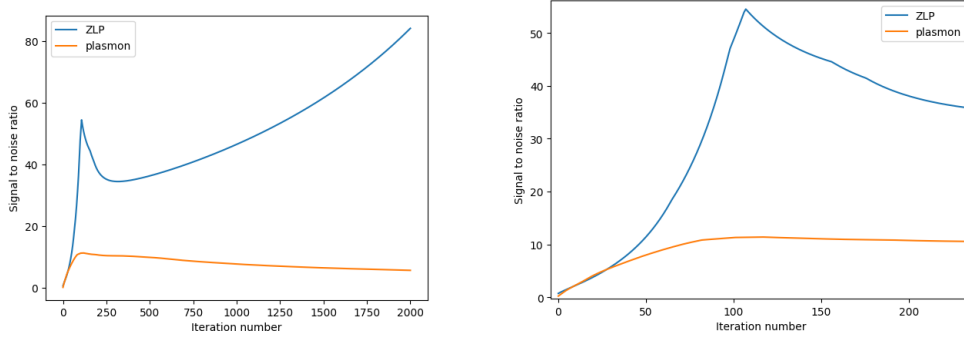


Figure 4: Signal to noise ratio against number of iterations

3.7 R7: Normalization

Testing has not yet been performed for this functional requirement.

3.8 R8: Background subtraction

Testing has not yet been performed for this functional requirement.

3.9 R9: Gain Correction

Testing has not yet been performed for this functional requirement.

4 Nonfunctional Requirements Evaluation

4.1 Usability

4.2 Performance

4.3 etc.

5 Comparison to Existing Implementation

This section will not be appropriate for every project.

6 Unit Testing

To date, partial unit testing has been performed on the following files:

- Filenamer.py, used to name the files to be exported and avoid conflicts with already existing files).
- Spectrum.py, used to implement the Data 1D Spectrum module.
- SpectrumImage.py, used to implement the Data 3D Spectrum module
- ImagePlotter.py, implementing the Display 2D Image module.

Coverage statistics for these files can be found in [Statement Coverage](#) and [Branch Coverage](#). As of the latest update, all unit tests were passing:

```
Ran 50 tests in 0.008s
```

```
OK
```

7 Changes Due to Testing

- A zero in the denominator during RL deconvolution now raises an exception

8 Automated Testing

9 Trace to Requirements

10 Trace to Modules

11 Code Coverage Metrics

Code coverage metrics were obtained using the Python coverage library (coverage.py-4.4.2) and represent the statement and branch coverage of the unit tests, to date.

11.1 Statement Coverage

Statement coverage provides a value for the percentage of lines which were run during the unit tests.

Name	Statements	Missed	Coverage %
FileNamer.py	40	3	92%
ImagePlotter.py	136	97	29%
SpectrumImage.py	247	155	37%
Spectrum.py	136	70	49%

Table 1: Statement coverage

11.2 Branch Coverage

Branch coverage combines the number of statements run during the test with the number of branches covered during the test.

Branch Name	Statements	Missed	Branches	Partial Branches	Coverage %
FileNamer.py	40	3	14	1	93%
ImagePlotter.py	136	97	56	0	23%
SpectrumImage.py	247	155	60	1	37%
Spectrum.py	136	70	32	1	48%

Table 2: Branch coverage

References

- [1] E. P. Bellido, D. Rossouw, and G. A. Botton, “Toward 10 meV Electron Energy-Loss Spectroscopy Resolution for Plasmonics,” *Microscopy and Microanalysis*, vol. 20, pp. 767–778, June 2014.