Turning Delta Function into Continuous

October 19, 2018

Abstract

This is an attempt to try and describe how modeling the delta function as a continuous function. Right now I'm considering an extremely simple situation, where I just have a flat region with a single delta function. I approximate this delta function as a triangle, and I'm trying to figure out how the width of this triangle affects its accuracy.

1 Run #1

1.1 Generating phonon distributions

I generated the pretend phonon distributions using the "generateInput.py" code, that exists in the "run1" directory. This created a series of 5 fake frequency distributions, shown in Fig. 1. These distributions are generated between 0 and 1, with a spacing of 0.01. The delta function is to be located at 0.5, and the area of the delta function is to be held at 0.2, while the area of the continuous piece (the flat piece) is to be held at 1.0. The thinnest triangle (blue) has a base width of $(2 \times 0.01) = 0.02$, while the largest one (purple) has a base width of $(10 \times 0.01) = 0.1$.

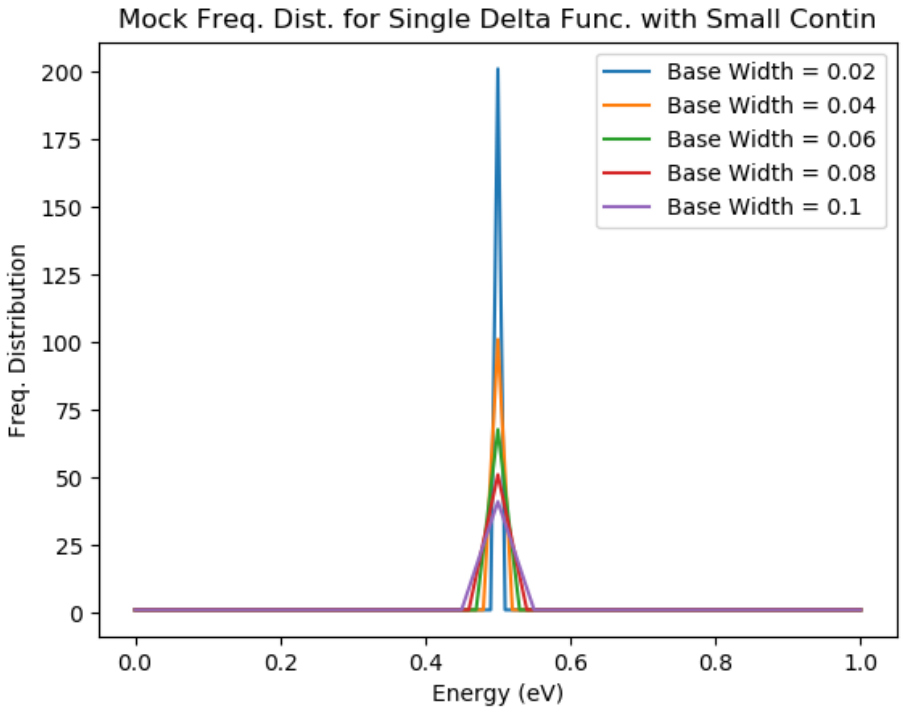

Figure 1: Fake frequency distributions of Run #1 mentioned in Sec.1. For this I consider five different continuous representations of a delta function, each a triangle with varying width. The delta function is normalized to have an area of 0.2, the flat region has an area of 1.0. The delta function is centered at 0.5, and the grid has a spacing of 0.01.

1.2 Running with Delta Funcs. vs. with Triangle-Approximated Delta Funcs.

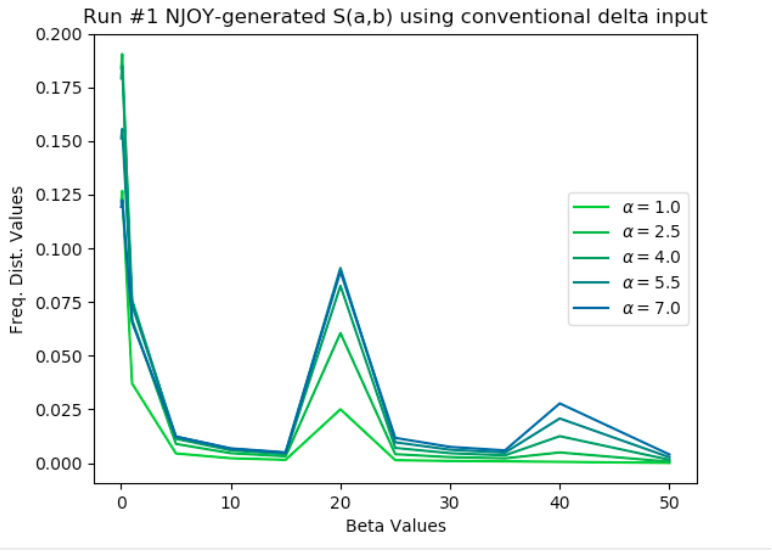

Figure 2: The $S(\alpha, \beta)$ results are shown above, after running the simple, single delta function input, using delta functions instead of triangle-approximated delta functions.

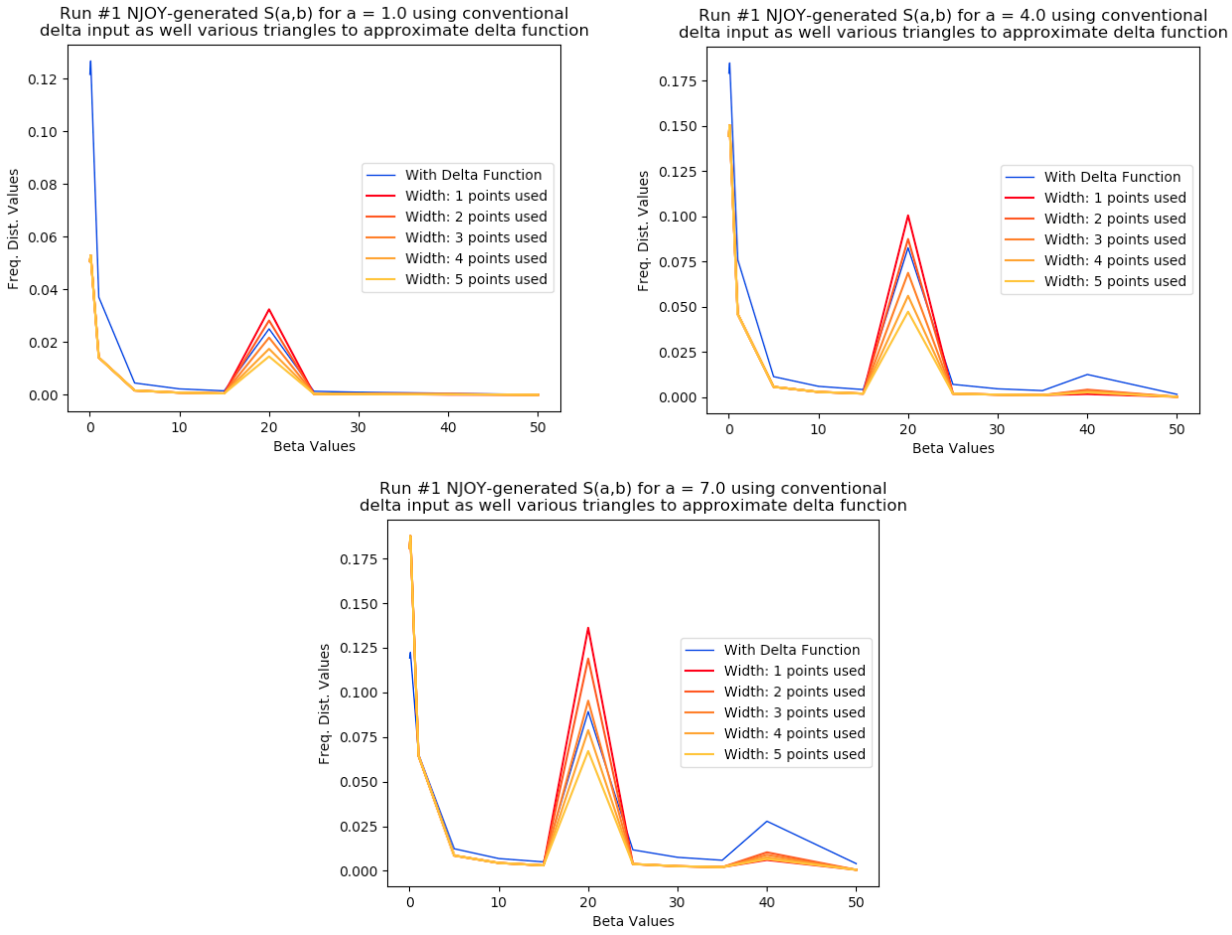

Figure 3: The $S(\alpha, \beta)$ results are shown above, after running the simple single-delta function input. This is run for α =1.0,4.0, and 7.0 separately. Note that the approximated $S(\alpha, \beta)$ does not seem to follow the delta-function $S(\alpha, \beta)$ well at all. The various widths of triangles seem to have a significant effect on the resultant $S(\alpha, \beta)$.

1.3 Percent Errors

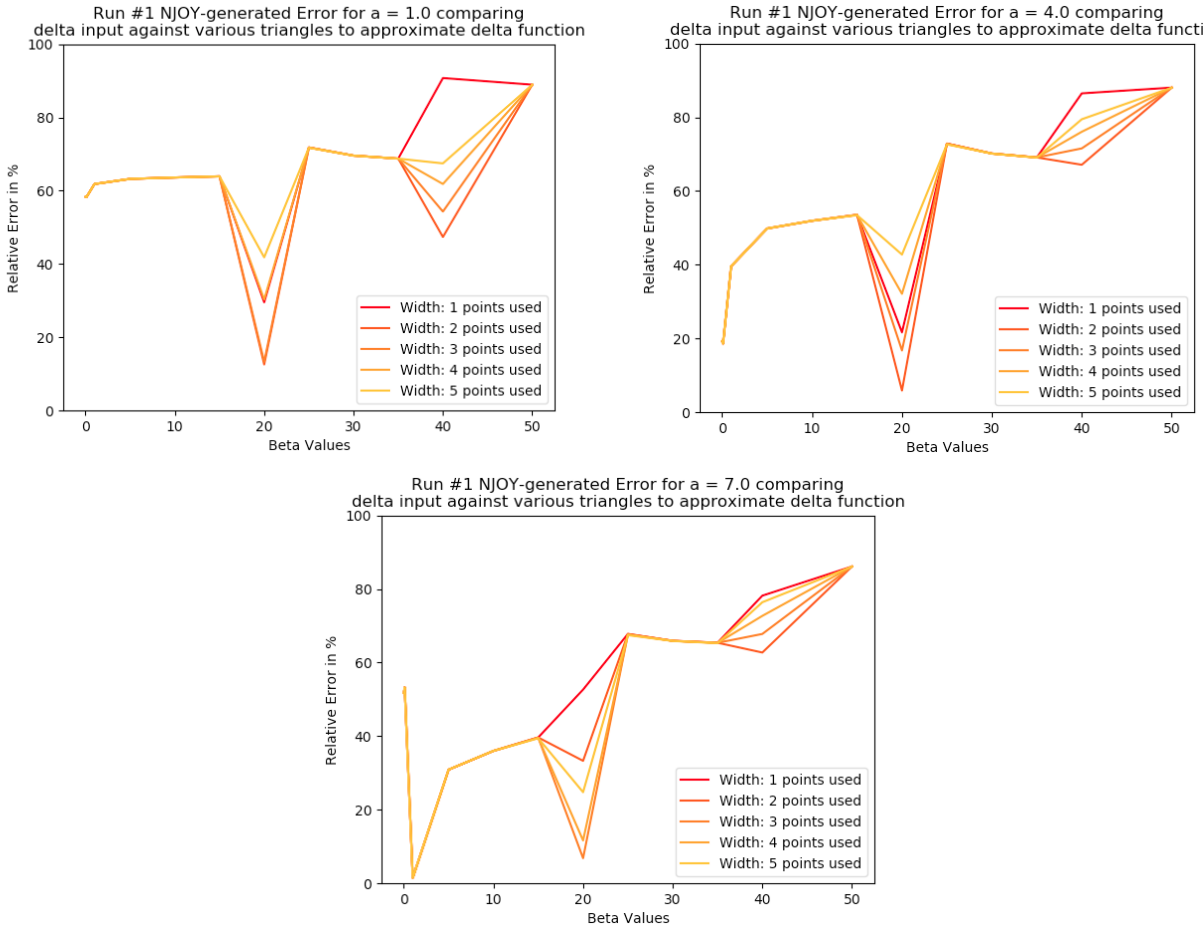

Figure 4: The percent error between the triangle-approximated delta functions and the real delta function is shown above.

1.4 Absolute Errors

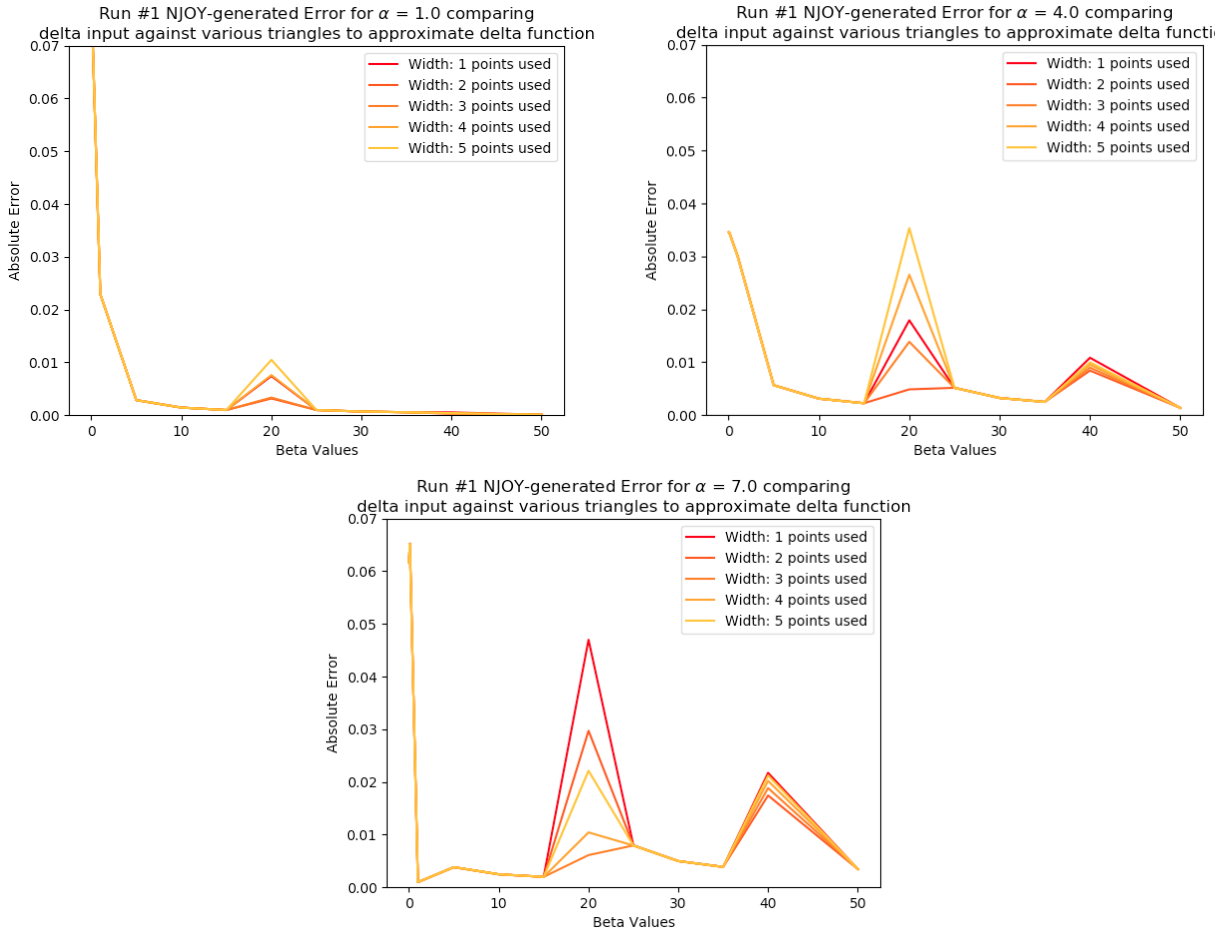

Figure 5: The absolute error between the triangle-approximated delta functions and the real delta function is shown above.

2 Run #2

2.1 Generating phonon distributions

I generated the pretend phonon distributions using the "generateInput.py" code in the "run2" directory. This is meant to create something close to the H in H2O model used in the NJOY test, with approximated delta functions included in the higher energy region.

The low energy distribution was copied from Test 09 of the NJOY 2016 release. This is also where I got the locations and weights for the delta functions. The lower delta function is centered at 0.205 eV, with a weighting of 0.16667, while the higher one is centered at 0.48 eV with a weighting of 0.33333. The continuous piece (which is only nonzero at the low energy region) has a weighting of 0.4444. The spacing used for this run is 0.005, which differs from the 0.00255 used in the Test 09 example. The resultant frequency distribution is shown in Fig. 6.

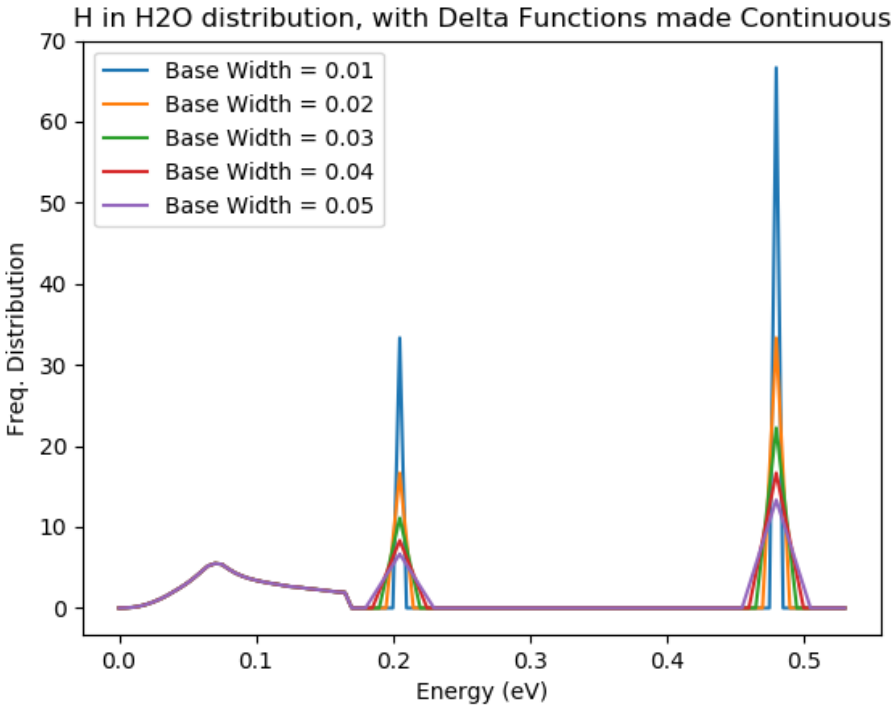

Figure 6: The generated phonon distribution for run #2 is shown here. This Is meant to strongly resemble the H in H_2O input, where the low-energy continuous distribution and its weight is taken from NJOY2016 Test09, along with the positions/weights of the delta functions.

2.2 Running with Delta Funcs. vs. with Triangle-Approximated Delta Funcs.

I used NJOY to run the 6 relevant cases: H in H2O with (1) delta functions approximating the higher energy peaks, and with (2-6) triangles of base width 0.002, 0.004, 0.006, 0.008, and 0.010. These input files are saved in the directory "run2" under the names "yDelta" and "nDelta_triangleWidthX_run2" for $X = 2, 4, 6, 8, 10$.

Fig 7 shows the $S(\alpha, \beta)$ produced by using the conventional δ-function input, plotted against β for various α. Note that there is extremely high variation between the $S(\alpha, \beta)$ values for various α and β.

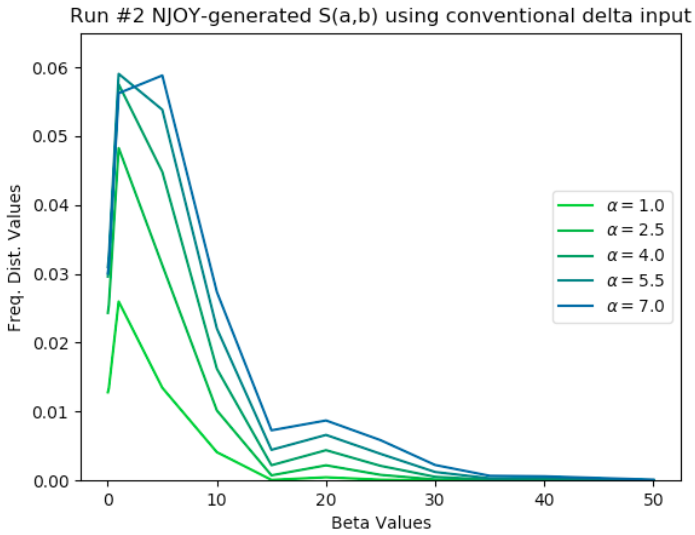

Figure 7: The $S(\alpha, \beta)$ results are shown above, after running the H in H_2O input using delta function inputs. Note that the resultant $S(\alpha, \beta)$ distribution is highly variant on input α, β values. Keep the scale in mind when we look at the absolute difference between with delta functions vs. without delta functions, for the later graphs.

Fig. 8 shows the results of $S(\alpha, \beta)$, comparing the "true" δ function result, vs. the triangle-approximated results, with three separate plots depicting $\alpha = 1.0, 4.0$, and 7.0.

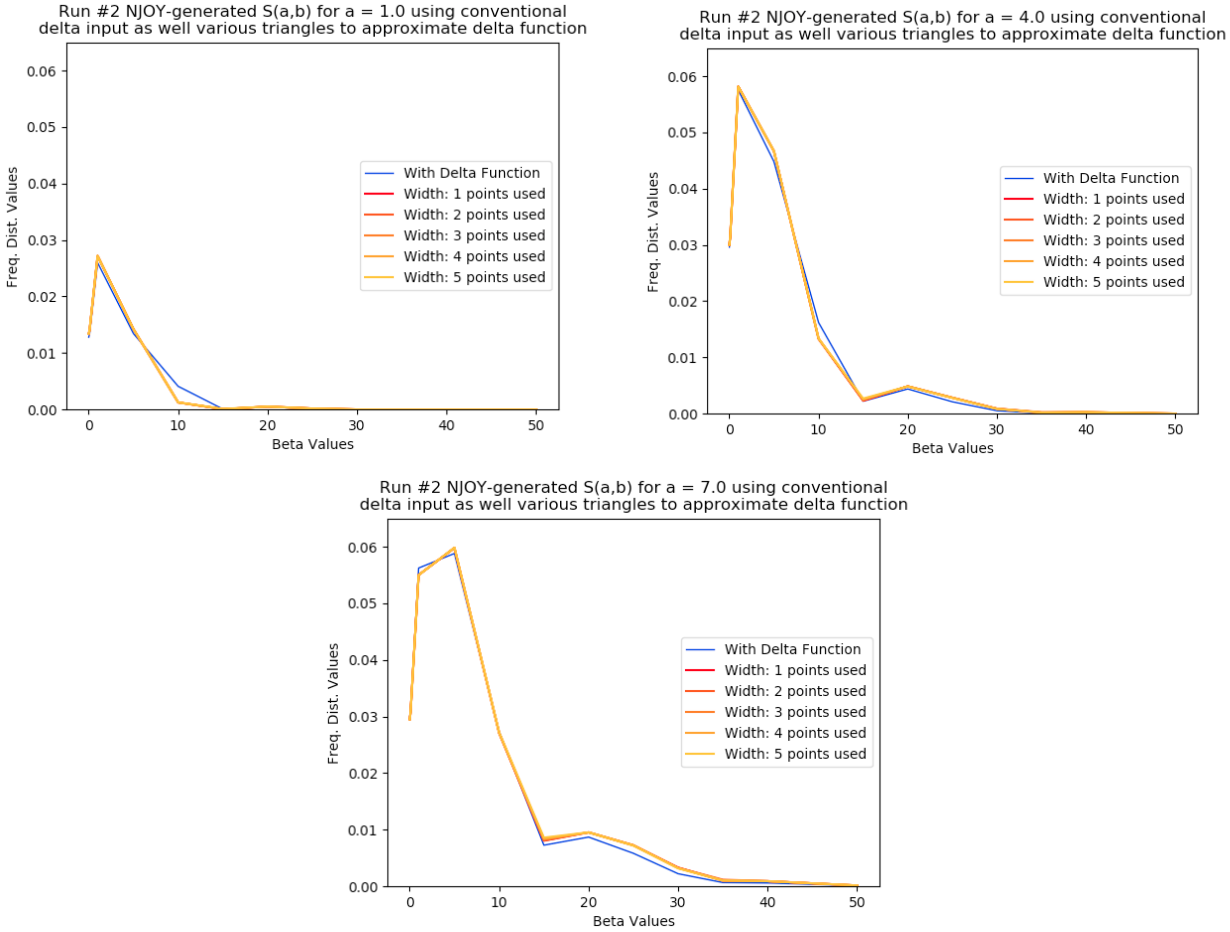

Figure 8: The $S(\alpha, \beta)$ results are shown above, after running the H in H_2O input using both delta function inputs and triangle-approximated inputs. This is run for α =1.0,4.0, and 7.0 separately. Note that the approximated $S(\alpha, \beta)$ follows the delta-function $S(\alpha, \beta)$ very well, and that the various widths of triangles does not seem to have a significant effect on the resultant $S(\alpha, \beta)$.

2.3 Percent Errors

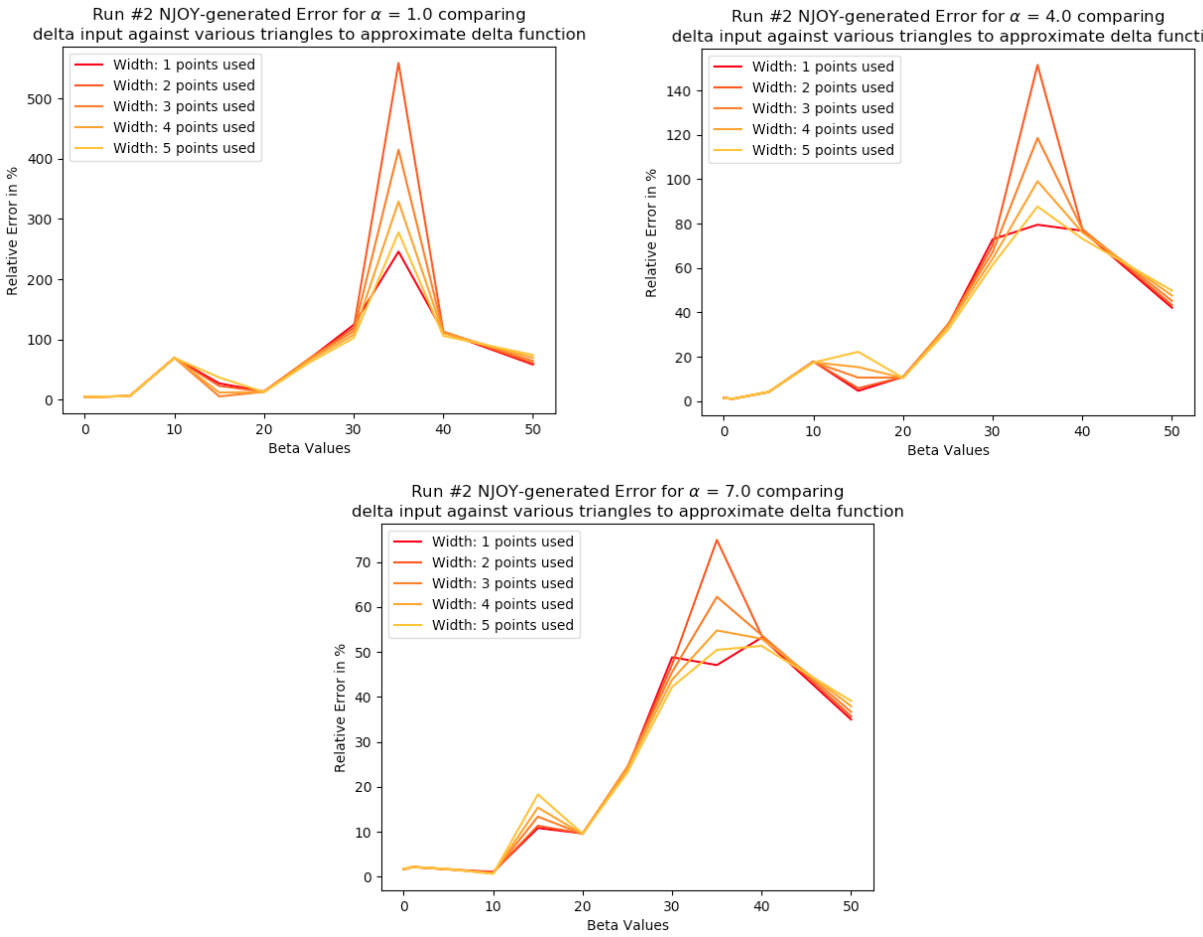

Figure 9: The percent error between the triangle-approximated delta functions and the real delta function is shown above. Note that the different plots have different scales.

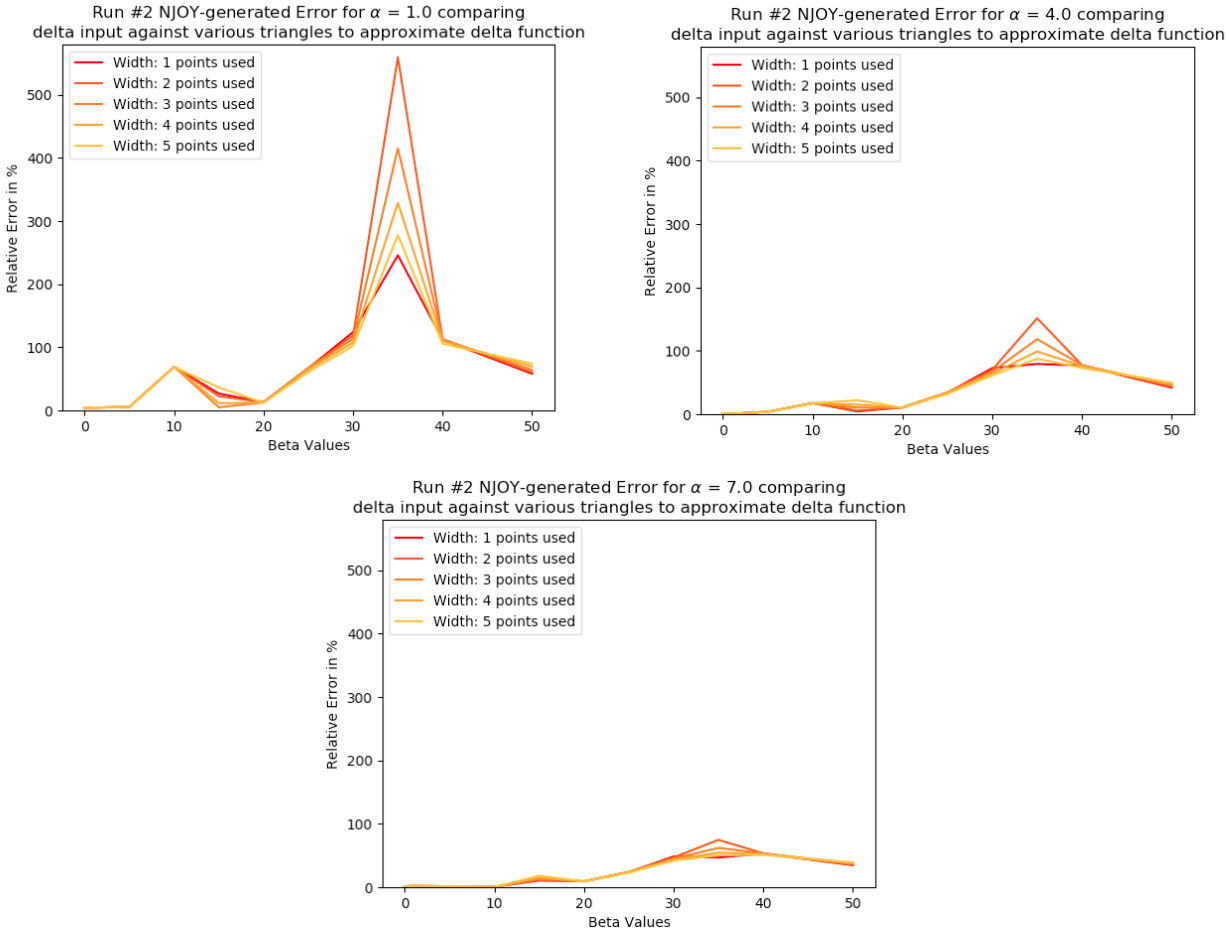

Figure 10: The percent error between the triangle-approximated delta functions and the real delta function is shown above. These plots are scaled so they all use the same axis bounds. The percent error for larger α values are significantly lower than those of lower α values.

2.4 Absolute Errors

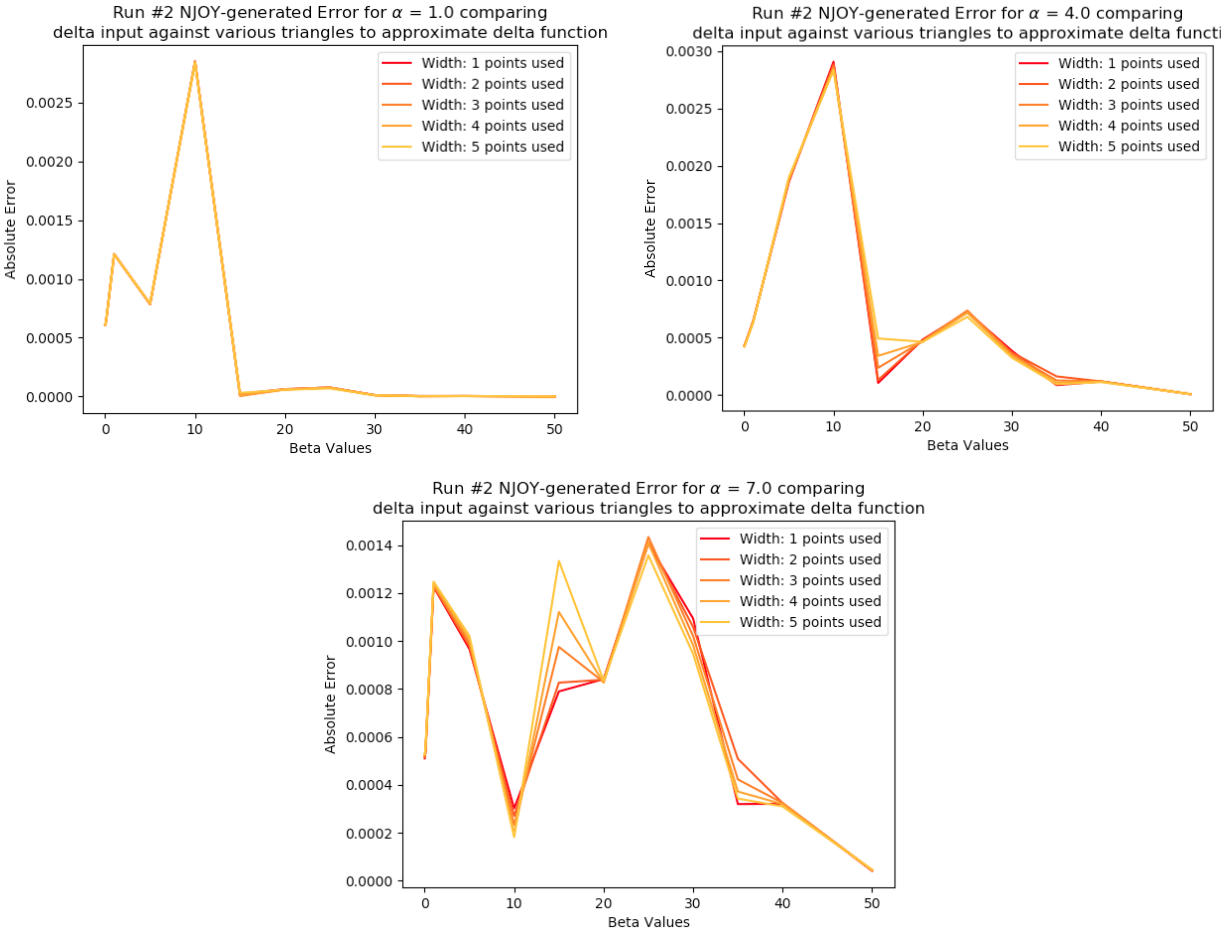

Figure 11: The absolute error between the triangle-approximated delta functions and the real delta function is shown above.