Dissertation

Francesco Tamburi

October 2023

Todo list

expand paragraph 4

Introduction

Results

0.1 Start

Table 1: Instrument resolution function

FWHM (ps)	Shift (ps)	Intensity (%)
213.3	0	80
150	-5	10
267	17	10

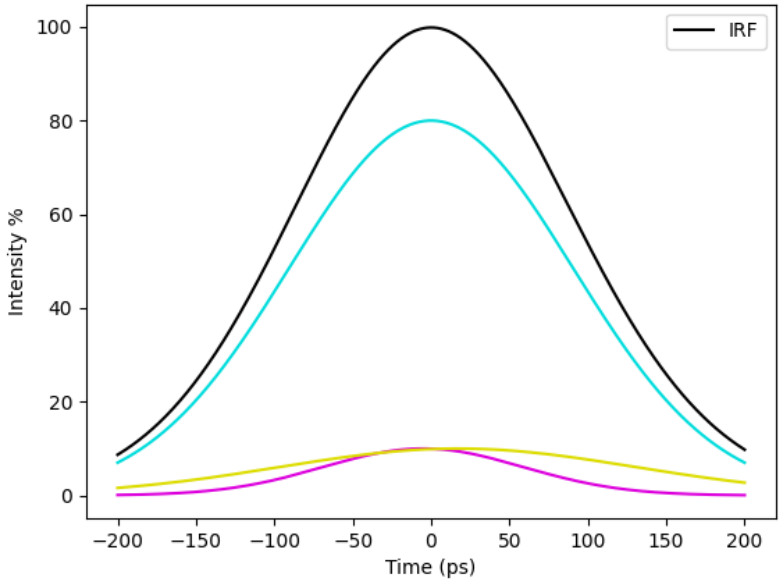


Figure 1: instrument resolution function

Using PALSSIM, a series of spectra containing two positron lifetimes, τ_1 and τ_2, were generated with a three gaussian instrument resolution function (see Table 1 and Figure 1). τ_1 was kept fixed at 180ps and τ_2 varied from 220-280ps, in 10ps intervals. For each value of τ_2, three

spectra were generated with the relative intensities of τ_1 and τ_2 set to 20%-80%, 50%-50% and 80%-20%. All the resulting spectra were then analyzed using PALSFIT, in order to evaluate how well the program could extract the two lifetimes, and their respective intensities, from each simulated spectrum.

In Figure 2a, we can observe how the values of tau_1 outputted by the program change, as the simulated value of tau_2 (on the horizontal axis) increases and the relative intensities (indicated by color and shape of marker) vary. Zooming in to the τ_2 = 250-270ps range, we can see in Figure 2b that, for these values in particular, PALSFIT struggles to determine τ_1 in the 50-50 and 80-20 case.

When evaluating τ_2, unlike with τ_1 where the simulated value of the lifetime was kept fixed, as the simulated value of the second lifetime changes, the difference between the result and the original values of τ_2 is plotted in Figure 2c. In the figure we can see that, aside from the 20-80 spectrum for τ_2 = 220, the software performs better than for τ_1.

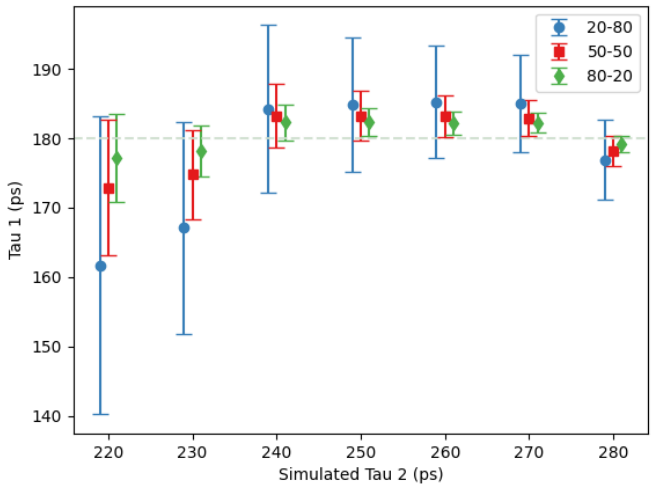
The error bars represent the standard deviation of – and thus the confidence of the program in – the lifetimes. From them, we see two factors that affect the size of the bars. The first is the relative intensities of the two intensities. In fact, in Figure 2a, which plots τ_1, the error bars are the smallest for the 80-20 data, where the shorter lifetime is more intense, and in Figure 2c, tracking τ_2, the opposite is the case and we have the 20-80 data is most precise. Observing all the figures, we see the second factor: as the time interval between the two lifetimes increases, the size of the error bars decrease.

In Figures 2d - 2f the fitted intensities of the two lifetimes are plotted against simulated τ_2, for each combination of simulated intensities. In the figures, we see that when the relative intensity of τ_2 was greater or equal to τ_1, then PALSFIT was able to calculate all the appropriate lifetime intensities. However, when the intensity of τ_2 was set to 80%, shown in Figure 2d, the software was unable to output the correct intensities for the first two simulated values of τ_2. Looking at our error bars, we can see the same relationship between their size and the lifetime separation mentioned earlier.

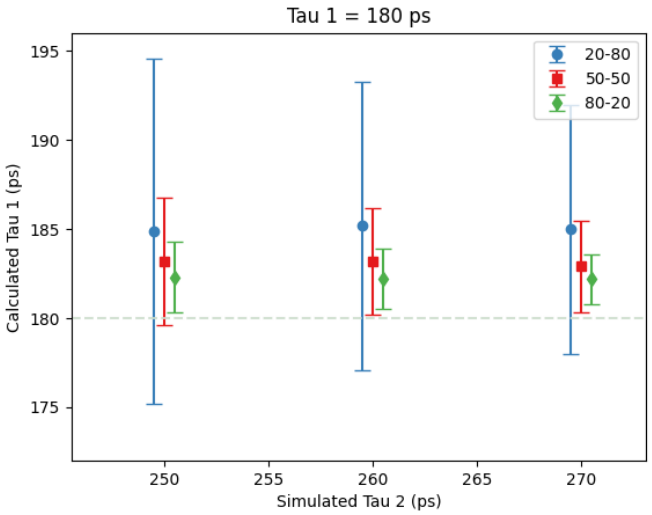
On the whole, PALSFIT seems to have performed well, but not perfectly. As the first lifetime, τ_1, was kept constant at 180ps, the next step would be to change τ_1 and see how that affects our results.

expand paragraph

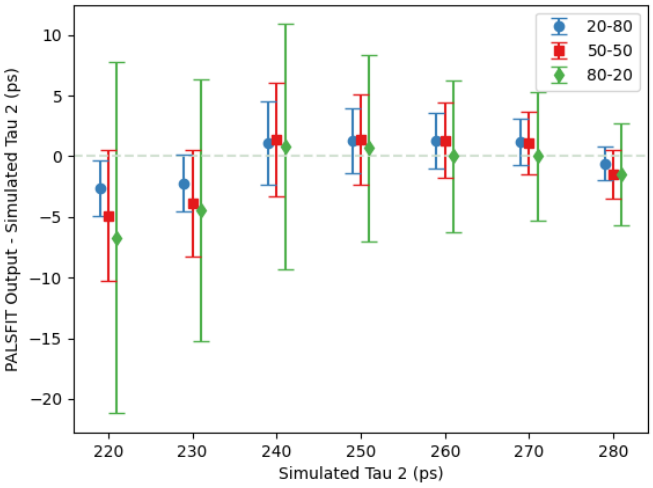
Figure 2



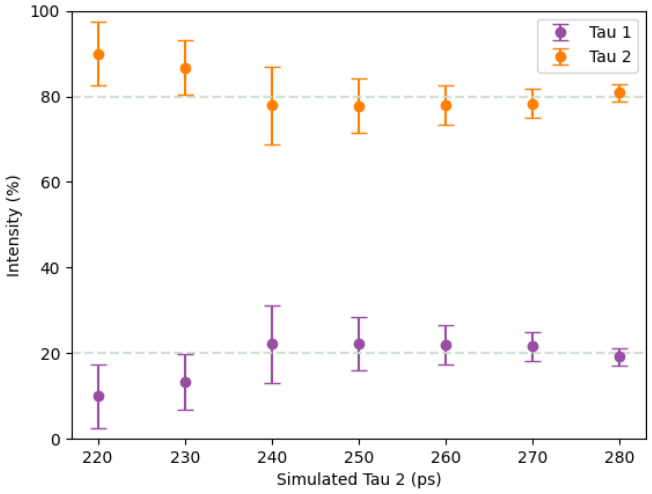
(a) fixed $\tau_1 = 180ps$



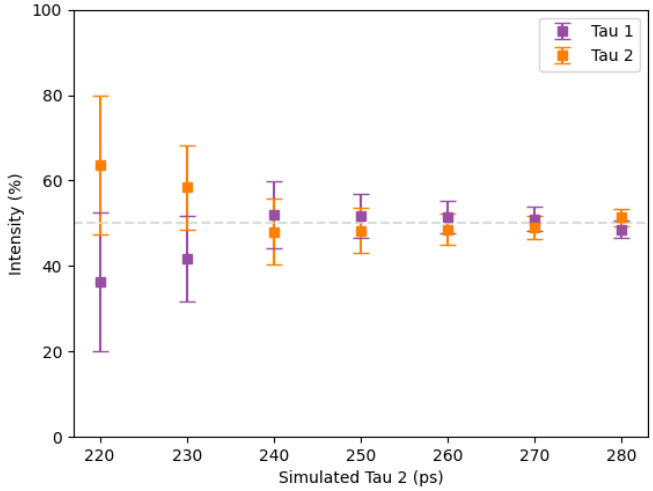
(b) close-up $\tau_1 = 180ps$



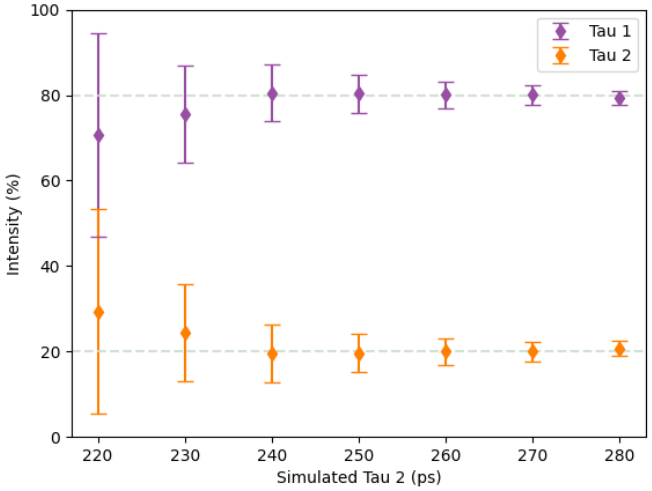
(c) τ_2 difference



(d) $\tau_1 = 20\%, \tau_1 = 80\%$



(e) $\tau_1 = 50\%, \tau_2 = 50\%$



(f) $\tau_1 = 80\%, \tau_2 = 20\%$

0.2 Modifying τ_1

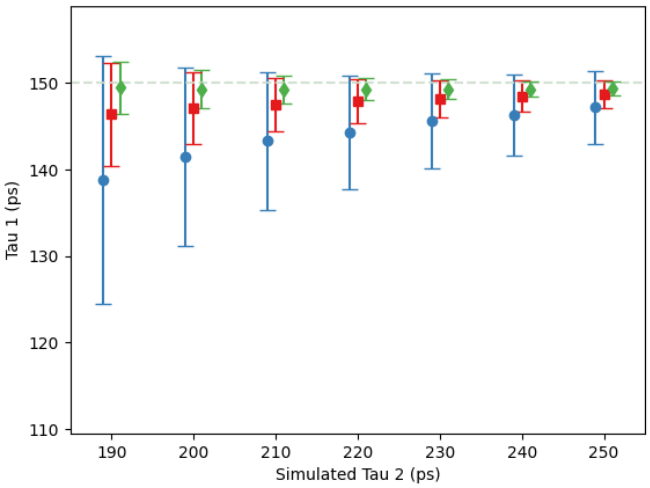
A similar procedure was performed for $\tau_1 = 150$ and 220. To keep the relative time interval the same in between batches, the corresponding spacing between τ_1 and the τ_2 range was kept consistent. For $\tau_1 = 150$, this meant a τ_2 range of 190-250ps, and for $\tau_1 = 220$, this meant a corresponding τ_2 range of 260-320ps.

This was first done for $\tau_1 = 150$. As can be seen in Figures 3a-3e, the results for this batch are all within error, with both accuracy and precision getting better as the lifetime separation increases, in line with what would be expected.

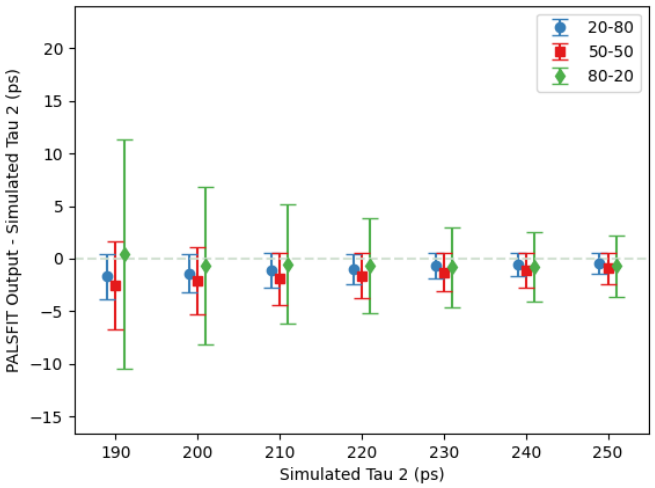
The other batch, meanwhile, where τ_1 was set to 220ps, was not as successful. The results can be seen in Figures 4a-3e, but in general, PALSFIT struggled with fitting the lowest values for τ_2 and the error bars are noticably larger.

rewrite from here on properly

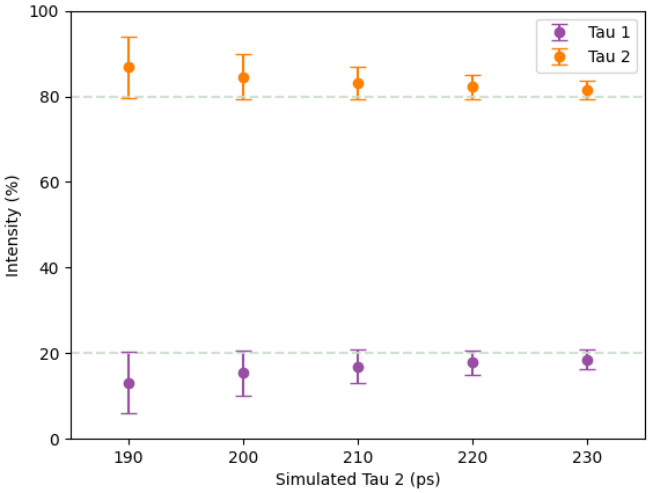
Figure 3



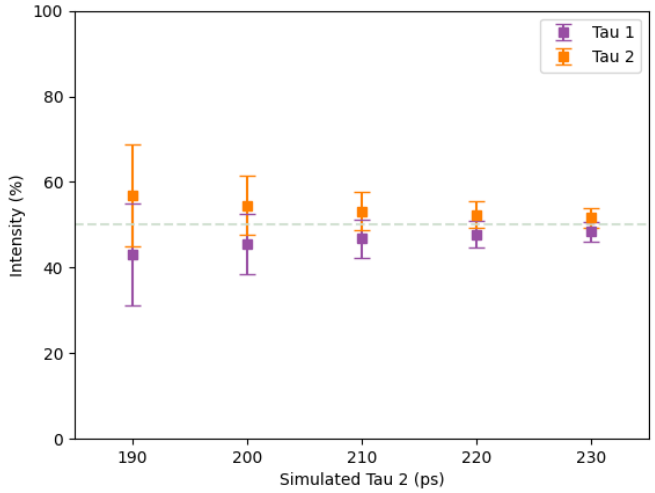
(a) fixed $\tau_1 = 150ps$



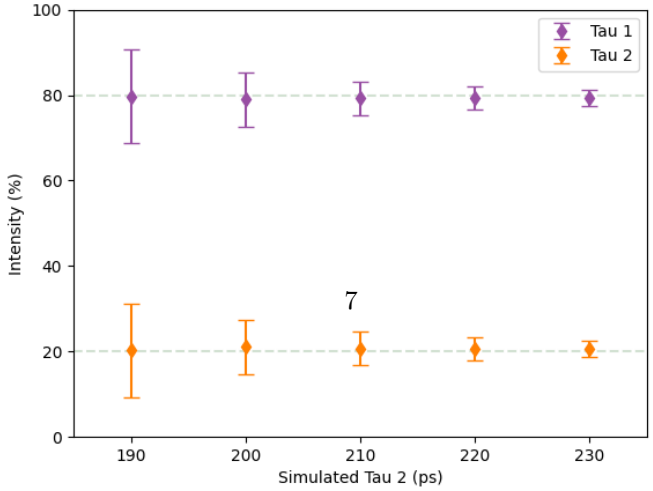
(b) τ_2 difference



(c) $\tau_1 = 20\%, \tau_2 = 80\%$

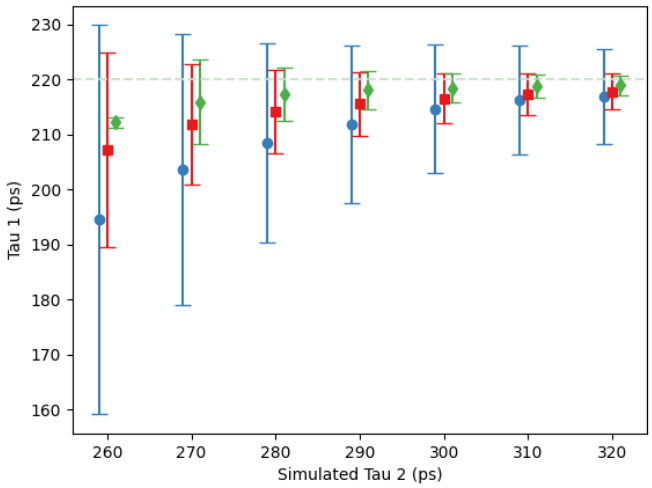


(d) $\tau_1 = 50\%, \tau_2 = 50\%$

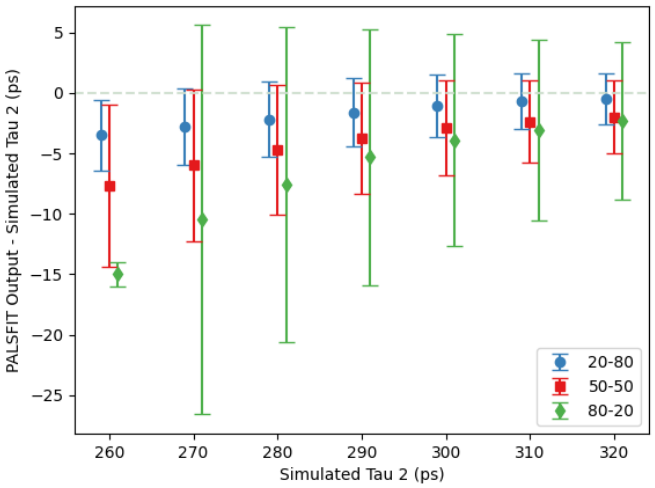


(e) $\tau_1 = 80\%, \tau_2 = 20\%$

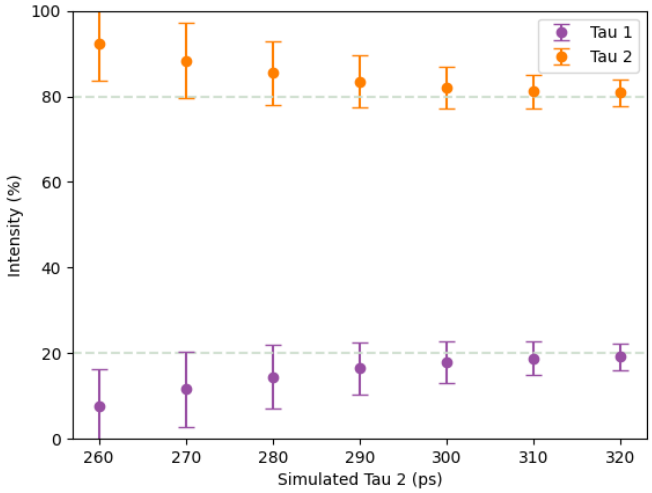
Figure 4



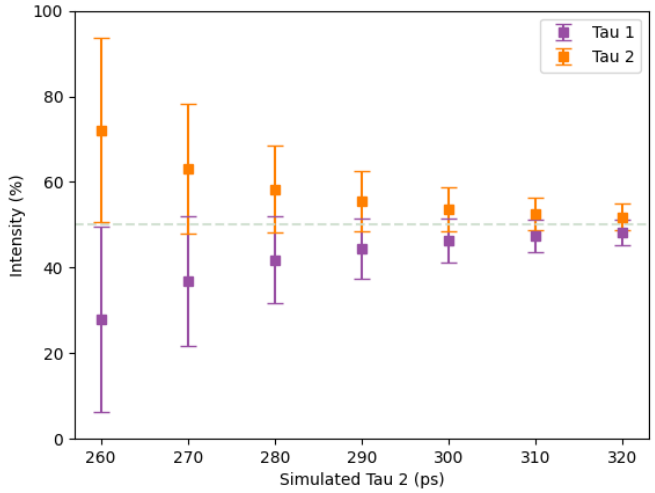
(a) fixed $\tau_1 = 220ps$



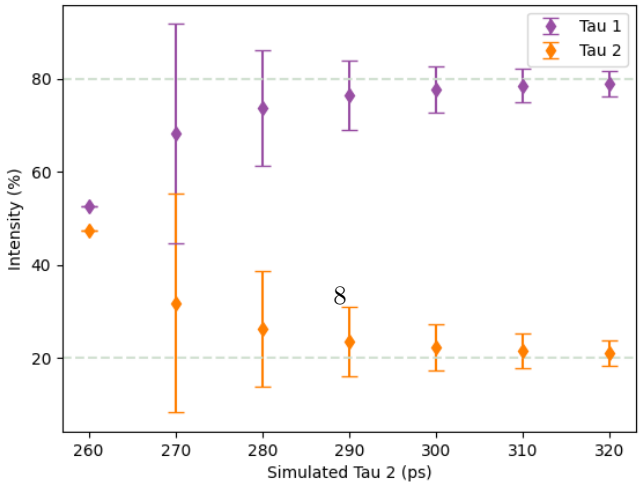
(b) τ_2 difference



(c) $\tau_1 = 20\%, \tau_2 = 80\%$

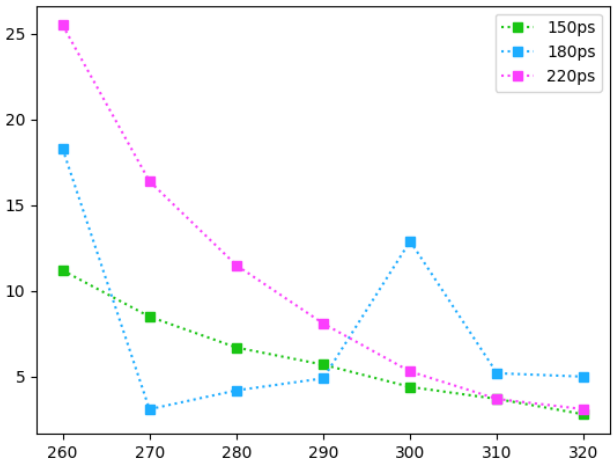


(d) $\tau_1 = 50\%, \tau_2 = 50\%$

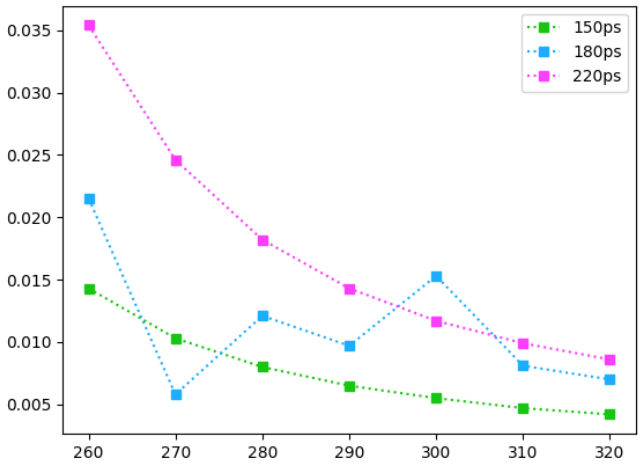


(e) $\tau_1 = 80\%, \tau_2 = 20\%$

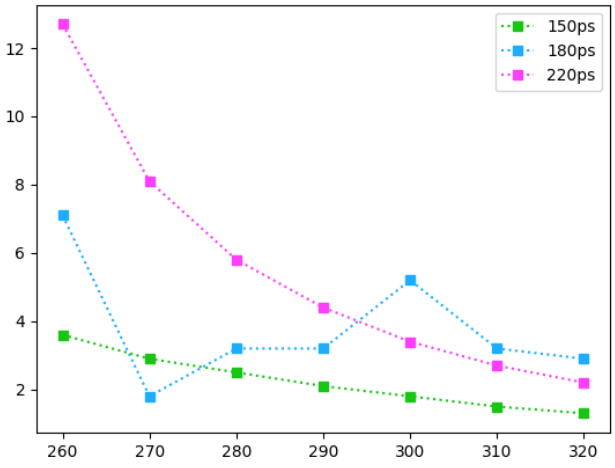
Figure 5: τ_1, rows = 20-80, 50-50, 80-20, columns = diff, std dev



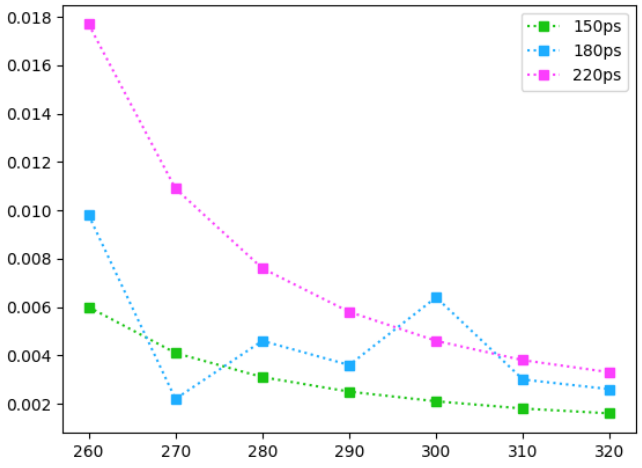
(a)



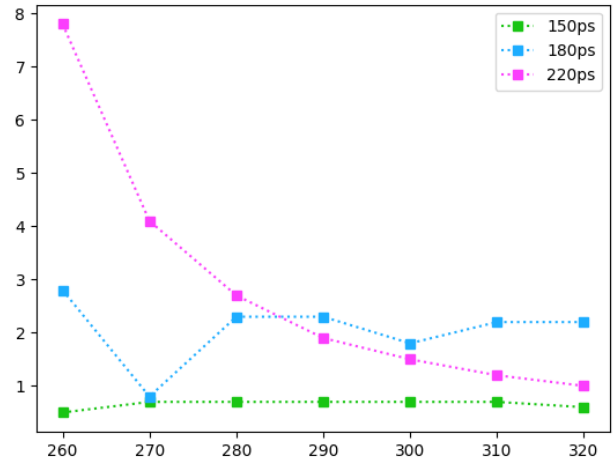
(b)



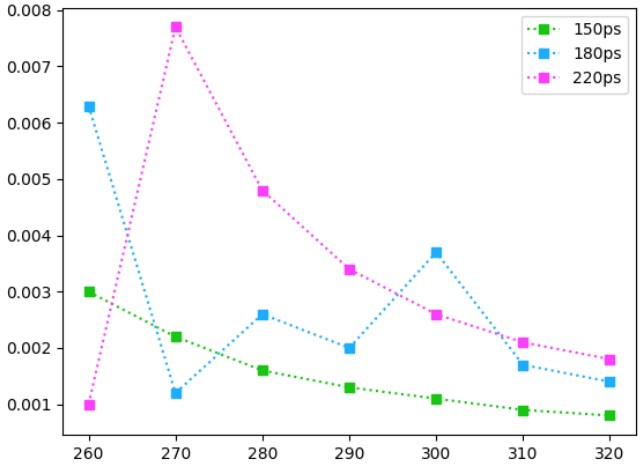
(c)



(d)

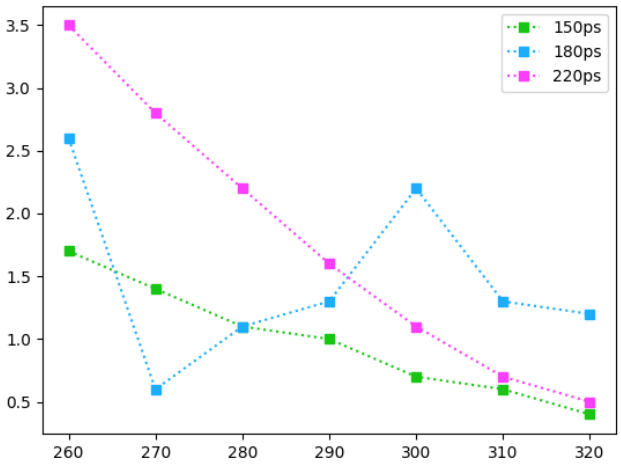


(e)

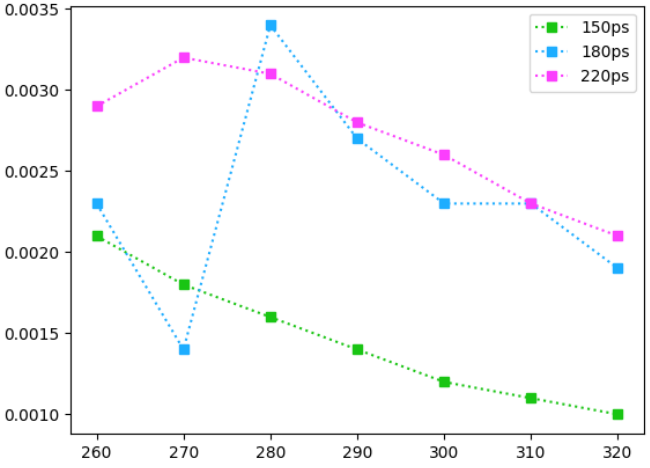


(f)

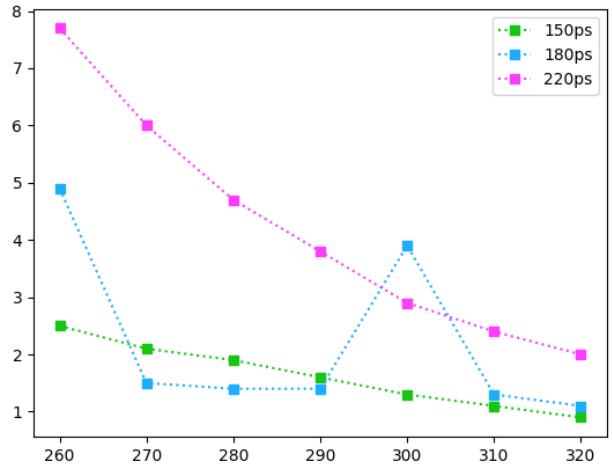
Figure 6: τ_2, rows = 20-80, 50-50, 80-20, columns = diff, std dev



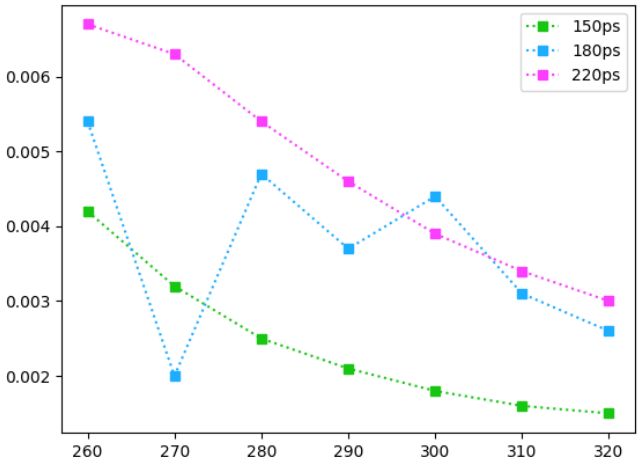
(a)



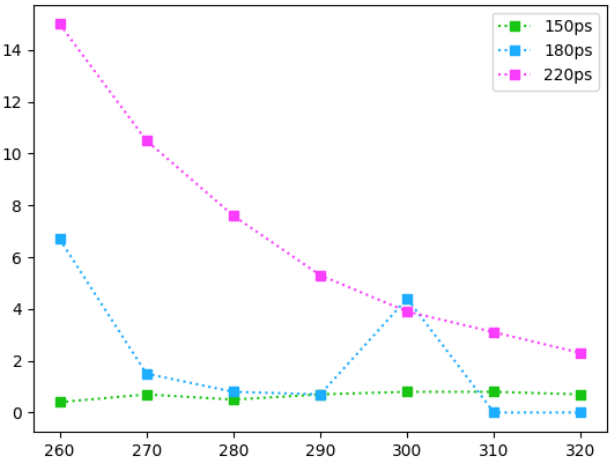
(b)



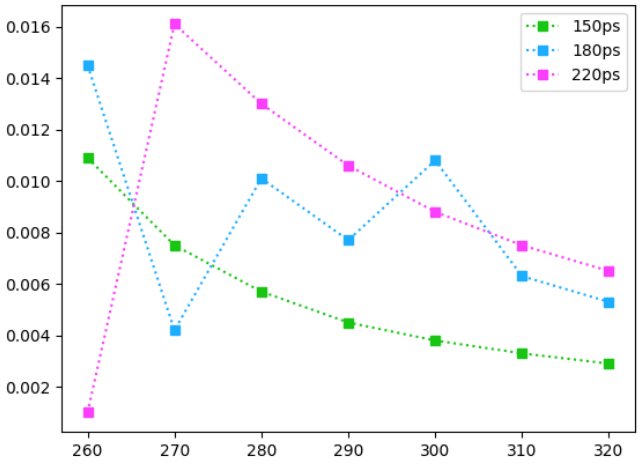
(c)



(d)

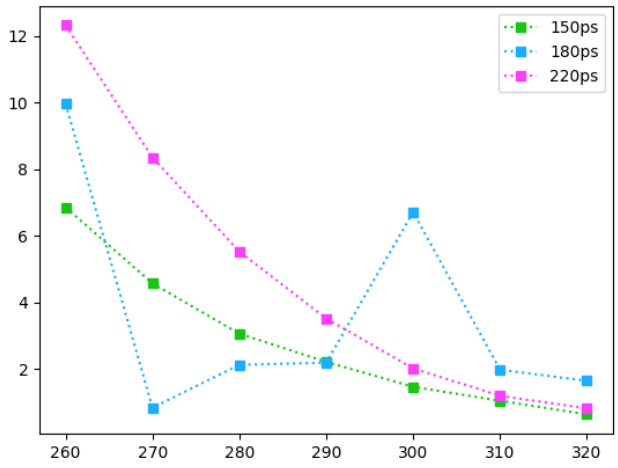


(e)

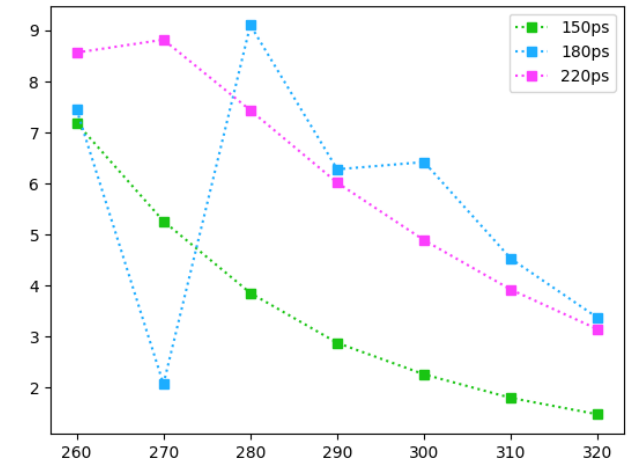


(f)

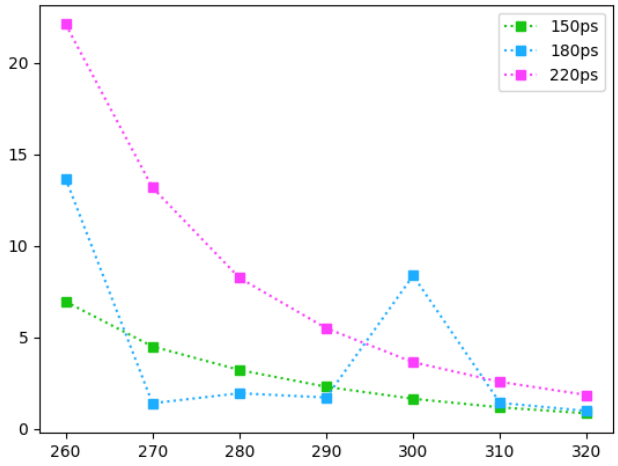
Figure 7: intensities, rows = 20-80, 50-50, 80-20, columns = diff, std dev



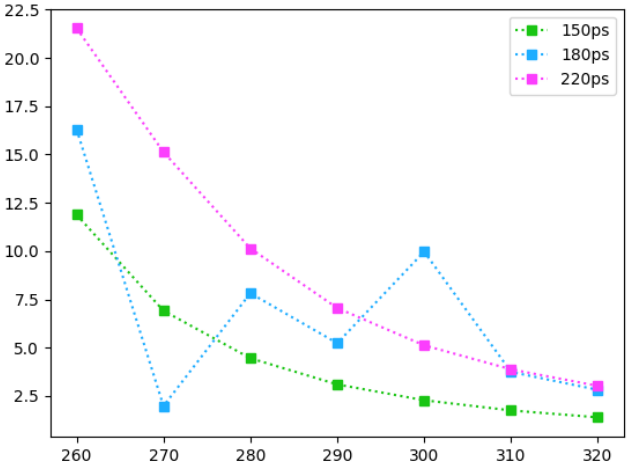
(a)



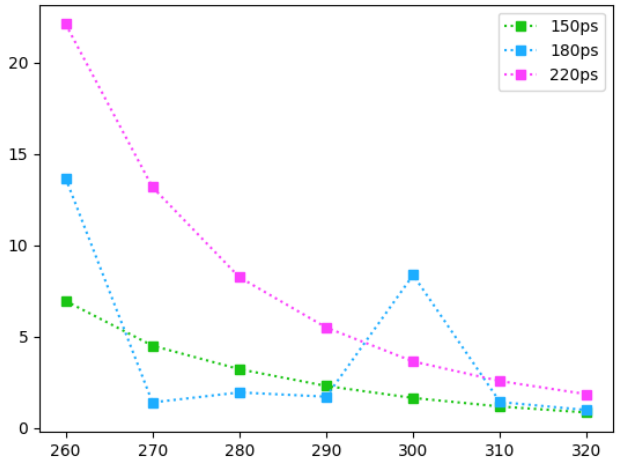
(b)



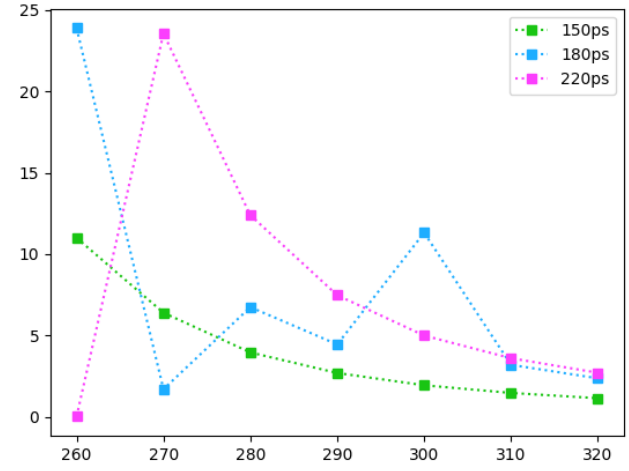
(c)



(d)



(e)



(f)

0.3 Instrument resolution function

Using Python, we can plot the three gaussian functions that compose the instrument resolution function. The equation for a Gaussian function with parameters a, b and c (corresponding to the height of the peak, the position of the peak and the width of the graph) is given by the equation:

$$g(x) = a \exp\left(-\frac{(x-b)^2}{2c^2}\right)$$

While a and b map easily to the intensity and shift columns in Table 1 , c is expressed as a standard deviation, and not the full-half-width-maximum given in the table. To convert from one to the other, we can use the following expression:

$$FWHM = 2\sqrt{2\ln(2)} \approx 2.355c$$

The resultant resolution function is, then, just the sum of the three gaussians $g_s(x)$, where

$$g_s(x) = \sum_{i=1}^{3} a_i \exp\left(-\frac{(x-b)^2}{2c^2}\right)$$

This resolution function looks quite similar to a gaussian function itself, and thus it seems sensible to investigate the feasability of approximating a more complex 3 gaussian resolution function, with a single gaussian.

The intensity of the single gaussian approximation is the easiest parameter to determine, as it's simply 100%. We could try to determine b and c analytically by solving the resulting $g_s(x)$ analytically, but, as we already have already generated a list of values for $g_s(x)$ when plotting the function, it's much easier to just extract the needed values numerically using Python.

Finding b is as easy as finding the largest element of the list, and its corresponding x value. As PALSSIM requires the FWHM rather than the standard deviation c, we just need to find the difference between the two values of x for which $g_s(x)$ is closest to half of the IRF.

Doing so gives us the following values to insert into PALSSIM:

FWHM (ps)	Shift (ps)	Intensity (%)
210	0	100

Table 2: Single gaussian Instrument resolution function

Plugging in these values into PALSSIM gives us very similar results to the three gaussian IRF, with slight differences in the first datapoint for some of the graphs. Overall, though, it does seems like we can reasonably approximate a more complex, three gaussian IRF with a single gaussian.

The next step would be to examine how the width of the instrument resolution function (IRF) affects the results. The values for τ_1 and τ_2 can be set to be kept the same between runs, with τ_1 fixed to 150ps and τ_2 ranging from 180-230ps, as we had the best results with these values. The full-width half maximum of our single gaussian IRF can be set to the following values: 100ps, 150ps, 180ps, 210ps.