Dissertation

Francesco Tamburi

October 2023

Todo list

Chapter 1

Intro

1.1 Positron Annihilation Lifetime Spectroscopy

Positron annihilation as a means of investigating lattice imperfections traces its roots back to the late 1960s, when the link between positron lifetimes and lattice vacancies was first proposed. [?] As the lifetime of a positron is a function of the electron density at the annihilation site, and lattice defects (e.g. open vacancies) represent a local decrease in electron density, longer lifetimes are observed for positrons trapped in these defects. [?] The presence of multiple positron lifetimes within a sample material, thus, indicates the presence of lattice defects within that material.

Positron Annihilation Lifetime Spectroscopy (PALS) is the study of these lifetimes, with the aim of determining the number and types of defects present in a sample material. While other techniques to study lattice defects exist, the main advantage of PALS is the high sensitivity of the technique to open-volume defects.[1] This property, along with the non-destructive nature of the technique, has led to its application in various fields of study.[?] [?]

Positron lifetimes can be determined experimentally by measuring the time interval between the emission of gamma photons that signal the production and annihilation of the positron. Commonly used positron sources are weakly radioactive isotopes such as ^{22}Na, which simulataneously emits a 1.27 MeV γ-photon and a positron as a part of its β^+ decay. As the positron encounters the sample material, it quickly loses its kinetic energy in a process known as "thermalization" as it penetrates into the sample. The positron's wavefunction then diffuses into the material and finally annihilates, producing two 511 keV γ-photons.

A scintillator-photomultiplier setup converts the γ-rays into analog electrical pulses, which are then processed by discriminators, distinguishing the 1.27 MeV from the 511keV pulses. The former is used as a "start" signal to begin charging a capacitor, and the latter stops the charging process. The capacitor acts as a sort of "electronic stopwatch", converting the time delay into an amplitude signal. To ensure a linear time-amplitude relationship, the stop signal is delayed by a fixed amount. Each signal is stored in the memory of a multichannel analyzer, where the channel numbers represent time and every count represents a single annihilation event. The resulting lifetime spectrum contains more than 10^6 annihilation events.

[1]Thorough comparisons of defect sensitive techniques, as well as their respective advantages and disadvantages, are available in literature, but detailed analysis of these is beyond the scope of this project.

1.2 The lifetime spectrum

The resulting spectrum contains $k+1$ lifetime components, corresponding to k different defect types and the positron lifetime in the defect-free bulk (τ_b). From these we get the time-dependent positron decay spectrum $D(t)$ – the probability for a positron to still be alive at a given time t [?] – given by the expression

$$D(t) = \sum_{i=1}^{k+1} I_i \exp\left(-\frac{t}{\tau_i}\right), \tag{1.1}$$

where τ_i and I_i represent the lifetime and intensity of each component, respectively.[2] The resulting positron lifetime spectrum $N(t)$ is given by the absolute value of the time derivative of the positron decay spectrum $D(t)$[3]:

$$N(t) = \left|\frac{\mathrm{d}D(t)}{\mathrm{d}t}\right| = \sum_{i=1}^{k+1} \frac{I_i}{\tau_i} \exp\left(-\frac{t+t_0}{\tau_i}\right). \tag{1.2}$$

The experimental spectrum is the convolution of the lifetime spectrum analytically described above and an Instrument Resolution Function (IRF). This resolution function is mainly determined by the scintillator+photomultiplier and takes the form of a sum of several Gaussian functions,

$$F(t) = \sum G_i(t), \tag{1.3}$$

with variations in the peak position, relative height and width of each Gaussian, $G_i(t)$. This gives us an experimental spectrum of the form

$$D_{exp}(t) = \int_{-\infty}^{+\infty} D(t-t')F(t')dt'. \tag{1.4}$$

1.3 PALSfit

Analysis of experimental spectra to extract physically meaningful parameters can be done using various computer programs. Most software does this by fitting a model function to the experimental spectrum, using some sort of least-squares method.

One common disadvantage of all least-squares based programs is that the user must input the number of lifetime components in the spectrum. This often means starting with a single component and adding more until the variance stops decreasing.

PALSfit is a software package based on the least-squares method. Using the POSITRONFIT module, it can determine lifetimes and intensities from lifetime spectra. PALSfit Version 3, or more simply PALSfit3, is the latest version of this package, which this project is aimed at evaluating. Any subsequent references to PALSfit refer to this version of the software.

[2]When $k=0$ (i.e. there are no defect components), $\tau_i = \tau_b$ (the bulk lifetime) and $I_i = 100\%$

[3]Due to the delay introduced in the time-amplitude conversion, $t = t + t_0$

Chapter 2

Annihilation Spectra

2.1 The positron lifetime

The lifetime of a positron, from production to annihilation, tends to be quite short, usually measured in the picosecond to nanosecond time scale. Nevertheless, during this brief lifetime, we can identify a few key stages and different paths that the positron may take.

A significant fraction of the positrons produced (up to 10-15%) never penetrates the sample material, annihilating instead in the source or source-sample interface. This can occur, for example, due to the backscattering of the positrons as they encounter positively charged atomic nuclei.

Thermalization, the process that slows down positrons before they diffuse into the sample, occurs via different mechanisms, depending on both the type of material and the energy of the positron. For example, at high energies, atomic ionization is dominant, while at low energies (< 1 eV in metals and ~ 1 eV in semiconductors) scattering off phonons dominates. Regardless of the specific mechanisms, however, the thermalization process occurs within the first few picoseconds.

Once thermalization occurs, the positron is free to diffuse through the sample lattice. This is the longest stage and also the one where defect trapping predominantly occurs. During diffusion, the wavefunction of the positively charged positron is primarily concentrated in the intersitial space between atoms due to the repulsion of positively charged atomic nuclei. In a perfect lattice, the positron delocalizes into a Bloch state with $k_+ = 0$. [mention of band structure?]

Trapping occurs when a vacancy-type defect is encountered and a localized positron state forms at the defect. The rate at which this occurs, called the positron trapping rate κ is proportional to the defect concentration C. Specifically

$$\kappa = \mu C, \tag{2.1}$$

where μ is termed the positron trapping coefficient, and is constant for a given defect.

2.2 Trapping Model(s)

Trapping models allow us to deduce defect concentration and type based on annihilation spectra. Key assumptions are that positron trapping during thermalization can be safely ignored and that defects within the sample are homogeneously distributed. From trapping models, we can derive a series of rate equations based on the time-dependent positron diffusion equation. {expand diffusion?}

2.2.1 Single defect trapping

The simplest trapping model assumes a single, open volume defect type, such as a vacancy. After thermalization, positrons either annihalate in the defect-free bulk at a rate $\lambda_b = \frac{1}{\tau_b}$ or within an open-volume defect, with trapping rate κ_d ($\propto$ the defect concentration C). As the electron density within the defect is lower than that of the bulk, the corresponding annihilation rate λ_d must be smaller than λ_b.

This gives us the following equations:

$$\frac{\mathrm{d}n_b(t)}{\mathrm{d}t} = -(\lambda_b + \kappa_d)n_b(t), \tag{2.2}$$

$$\frac{\mathrm{d}n_d(t)}{\mathrm{d}t} = -\kappa_d n_d(t) + \kappa_d n_b(t), \tag{2.3}$$

that describe the change in the number of positrons in the bulk (n_b) and the defect (n_d), repectively, at time t. As we have assumed no trapping during thermalization, if we define N_0 as the number of positrons at $t = 0$, we get our starting conditions, i.e. $n_b(0) = N_0$ and $n_d(0) = 0$. The solution to eq.s 2.2 and 2.3 gives us the decay spectrum of the positrons

$$D(t) = \frac{\lambda_b - \lambda_d}{\lambda_b - \lambda_d + \kappa_d} e^{-(\lambda_b+\kappa_d)t} + \frac{\kappa_d}{\lambda_b - \lambda_d + \kappa_d} e^{-\lambda_d t}. \tag{2.4}$$

Taking eq. 1.1, and setting $k = 1$ (as we're dealing with a single defect type) we can make the appropriate subtitutions and get:

$$D(t) = I_1 \exp\left(-\frac{t}{\tau_1}\right) + I_2 \exp\left(-\frac{t}{\tau_2}\right), \tag{2.5}$$

where

$$\begin{aligned} \tau_1 &= \frac{1}{\lambda_b + \kappa_d} \quad , \quad \tau_2 = \frac{1}{\lambda_d}, \\ I_1 &= 1 - I_2 \quad , \quad I_2 = \frac{\kappa_d}{\lambda_b - \lambda_d + \kappa_d}. \end{aligned} \tag{2.6}$$

By taking the absolute value of the time derivative of the decay spectrum, in similar fashion to eq. 1.2, we get the positron lifetime spectrum

$$N(t) = \left|\frac{\mathrm{d}D}{\mathrm{d}t}\right| = \frac{I_1}{\tau_1} \exp\left(-\frac{t}{\tau_1}\right) + \frac{I_2}{\tau_2} \exp\left(-\frac{t}{\tau_2}\right). \tag{2.7}$$

Looking closely at eq. 2.7, we can see that there are two lifetime components in the spectrum. From eq. 2.6 we can observe that the trapping rate κ_d affects the first lifetime component and the relative intensity of the two components, but not the second lifetime component, which arises from positron annihilation within the defects. This means that while τ_2, also called the defect-related lifetime $\tau_d = 1/\lambda_d$, is simply the positron lifetime within the defect, τ_1 is not the positron lifetime within the defect-free bulk $\tau_b = 1/\lambda_b$. As $\tau_1 \leq \tau_b$, τ_1 is also called the "reduced bulk" lifetime. Additionaly, as τ_d is independent of defect concentration, it is taken to be an indication of the open volume of the defect.

2.2.2 Additional complexity

Often experimental spectra are more complex than can be described by the single defect model. They are likely to contain more than one defect type and often additional effects must be taken into consideration, such as detrapping, where a portion of the trapped positrons escape back into the bulk. The single defect model, however, can be easily extended to deal with this additional complexity.

To do so, we assume that the different defects do not interact, and thus we can write $k+1$ rate equations, instead of just the two in the single defect model, giving us

$$\frac{\mathrm{d}n_b}{\mathrm{d}t} = -\left(\lambda_b + \sum_{i=1}^{k} \kappa_i\right) n_b(t) + \sum_{i=1}^{k} \delta_i n_{di}(t), \tag{2.8}$$

for the bulk, where δ_i is the detrapping rate for a given defect type, and

$$\frac{\mathrm{d}n_{di}}{\mathrm{d}t} = \kappa_i n_b(t) - (\lambda_{di} + \delta_i) n_i(t), \tag{2.9}$$

for each defect type. The boundary conditions are simply $n_b(0) = 1$ and $n_i(0) = 0$ for all values of i. From this, we procede in much the same way as the single defect.

When dealing with a multi-defect spectrum, a useful parameter is the average lifetime $\overline{\tau}$. This is simply the average of the lifetime components, weighted by intensity, or

$$\overline{\tau} = \sum_{i=1}^{k+1} I_i \tau_i. \tag{2.10}$$

In any lifetime spectrum the average lifetime is simply the centre of mass of the spectrum. When fitting a multi-lifetime spectrum using a least-squares method, this can be easily determined by fitting a single lifetime component. Additionally, if we know the bulk lifetime of a defect free material, we can use the average lifetime of a sample of that material to determine the presence of defects, as in that case $\overline{\tau} > \tau_b$.

Another factor to consider when fitting an experimental spectrum is saturated positron trapping. In a sample with high defect concentration, the average spacing may be much smaller than the average diffusion length of the positron in the bulk. In such a case all positrons might be trapped, resulting in a single-component spectrum, with $\tau_d = \overline{\tau}$, and a defect concentration that cannot accurately determined.

Chapter 3

Methods

Outline: Explain PALSfit and PALSsim, graphs were made in Python using matplotlib.

Results

3.1 First analysis

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 3.1 and Figure 3.1). 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 3.2a, we can observe how the values of τ_1 outputted by the program change, as the simulated value of τ_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 3.2b that, for these values in particular, PALSFIT struggles to determine τ_1 in the 50-50 and 80-20 case.

Unlike for τ_1, where the simulated value of the lifetime is fixed, the simulated value of τ_2 (our point of comparison) changes in between spectra. To make the data easier to visualize, rather than the fitted value of τ_2, the difference between the result and the original is plotted instead (see Figure 3.2c). In the figure we can see that, aside from the 20-80 spectrum for $\tau_2 = 220$, 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 3.2a, 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 3.2c, 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 3.2d - 3.2f 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 3.2d,

Table 3.1: Instrument resolution function

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

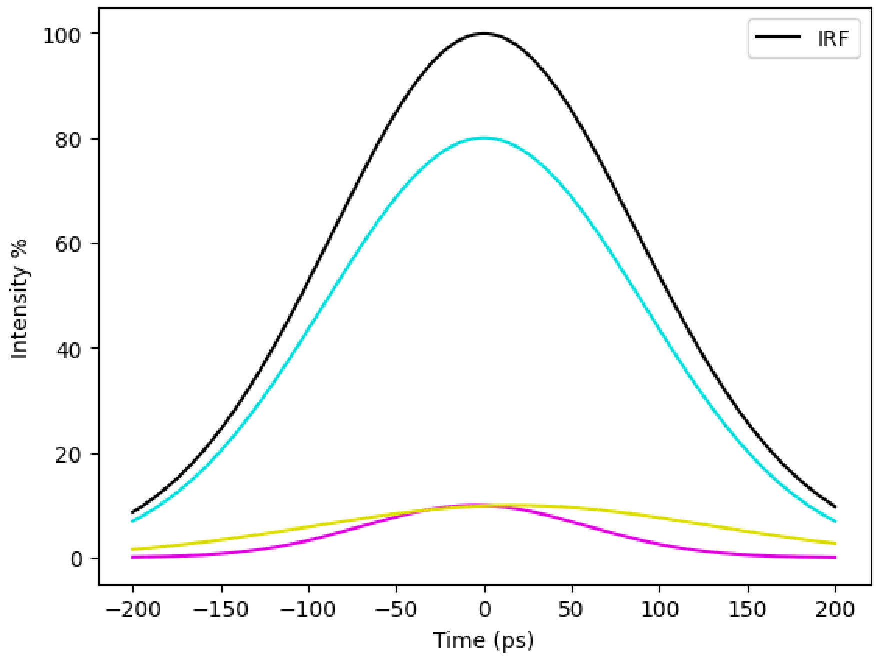


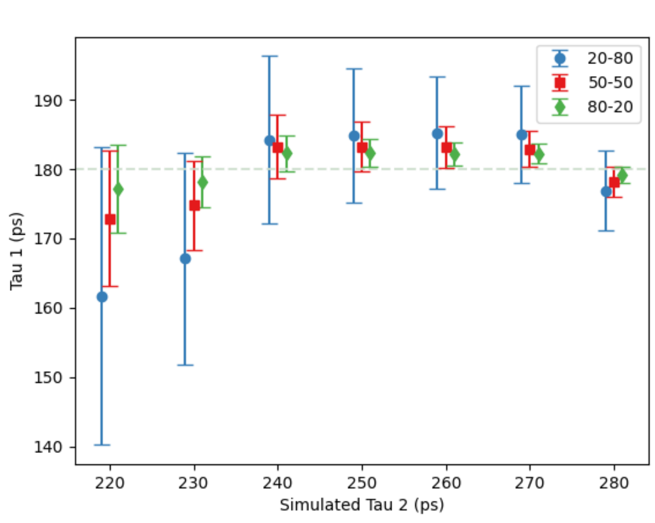
Figure 3.1: instrument resolution function

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.

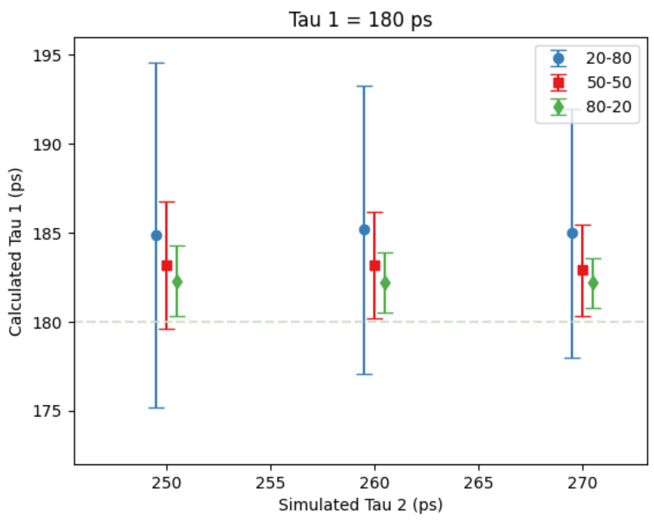
On the whole, PALSFIT seems to have performed well, but not perfectly. As the first lifetime, τ_1, was kept constant at 180ps, the next step would be to change τ_1 and see how that affects our results.

expand paragraph

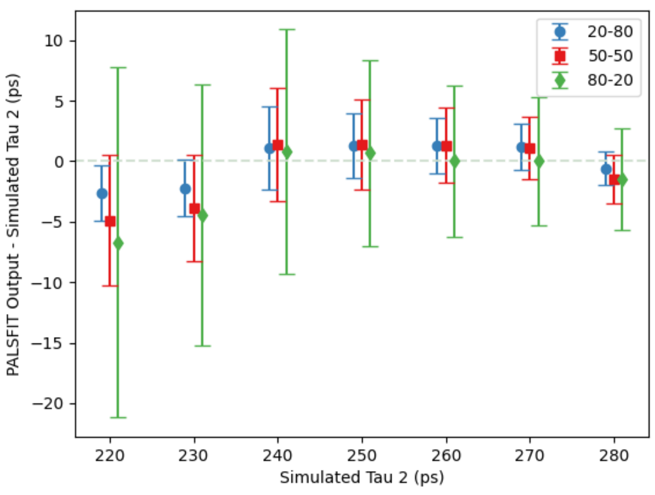
Figure 3.2



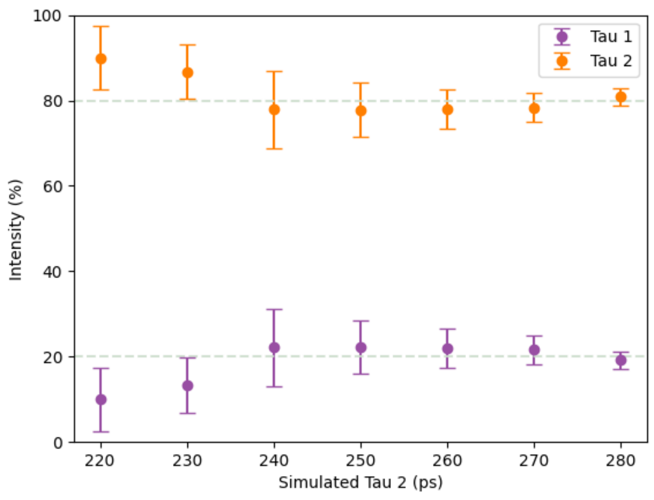
(a) fixed $\tau_1 = 180ps$



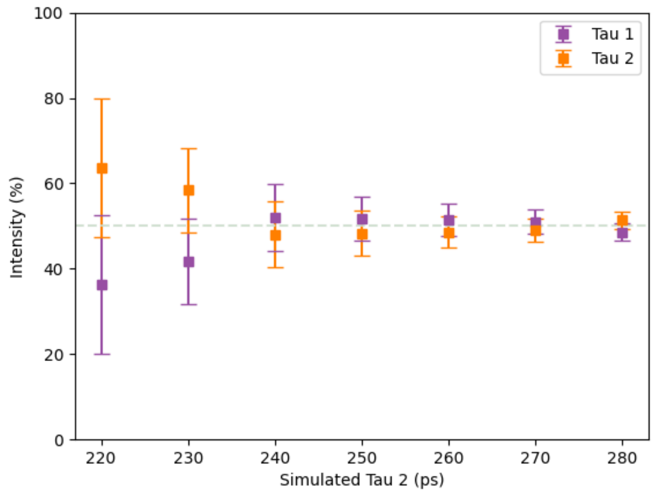
(b) close-up $\tau_1 = 180ps$



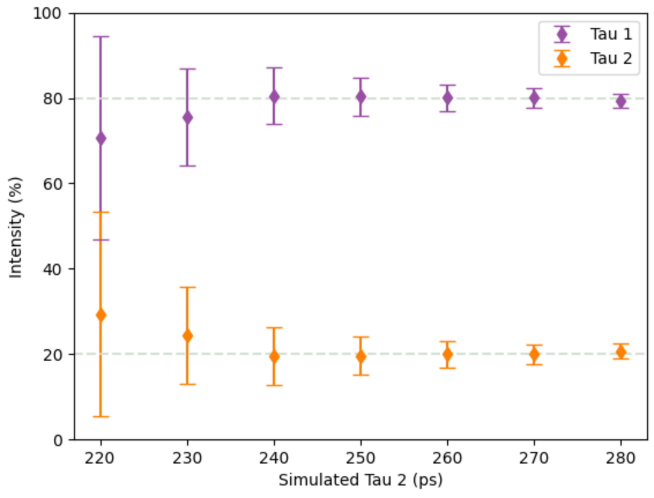
(c) τ_2 difference



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



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



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

3.2 Modifying τ_1

A similar procedure was performed for τ_1=150 and τ_1=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 τ_1 = 150, this meant a τ_2 range of 190-250ps, and for τ_1 = 220, this meant a corresponding τ_2 range of 260-320ps.

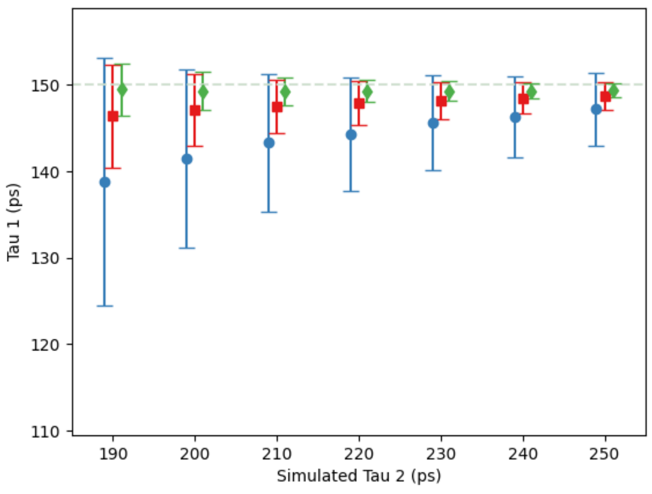
This was first done for τ_1 = 150. As can be seen in Figure 3.3, the results for this batch are all within error, with both accuracy and precision getting better as the lifetime separation increases, in line with what would be expected.

The other batch, meanwhile, where τ_1 was set to 220ps, was not as successful. The results can be seen in Figure 3.4, but in general, PALSFIT struggled with fitting the lowest values for τ_2 and the error bars are noticably larger. The program even gives nonsensical results when τ_1 = 260ps and the relative τ_1-τ_2 intensity is set to 80%-20%, as can be seen in Figures 3.4a, 3.4b and 3.4e.

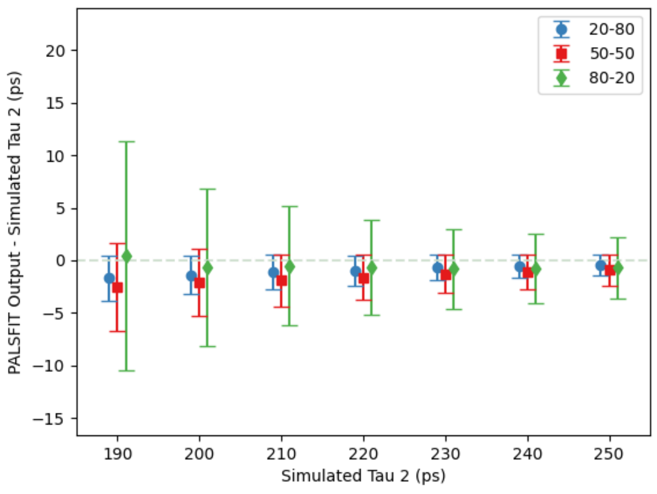
In order to compare all three datasets, the absolute difference between the fitted and simulated values of each relevant variable, τ_1, τ_2 and intensity was plotted. Additionally, plots for their respective standard deviations were generated. The resulting plots are represented in Figures 3.5-3.7. Two general observations are apparent from analysis of these figures:

The first is that the τ_2 = 180ps data seems much more scattered than the other two datasets, which both follow a relatively clear increase in precision and accuracy as the lifetime separation increases. The second is that as τ_1 decreases, PALSFIT seems better able to resolve the two lifetimes, with both the difference from the true value and the size of the error bars being the smallest when τ_1 = 150ps.

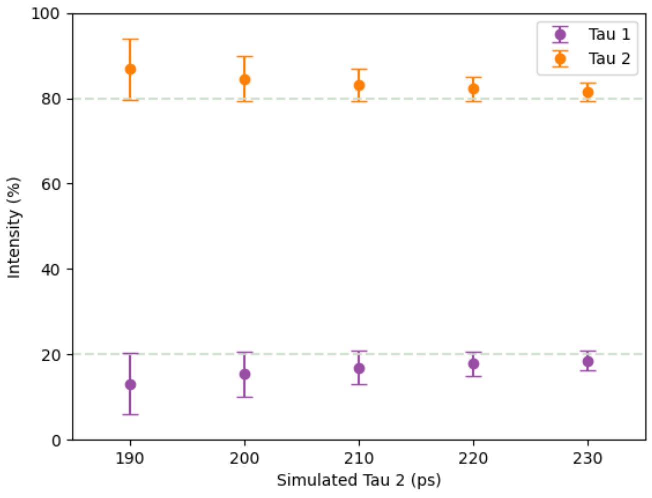
Figure 3.3



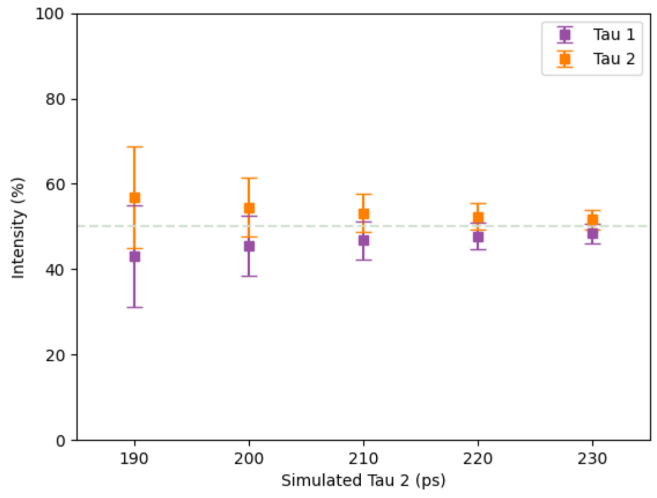
(a) fixed $\tau_1 = 150ps$



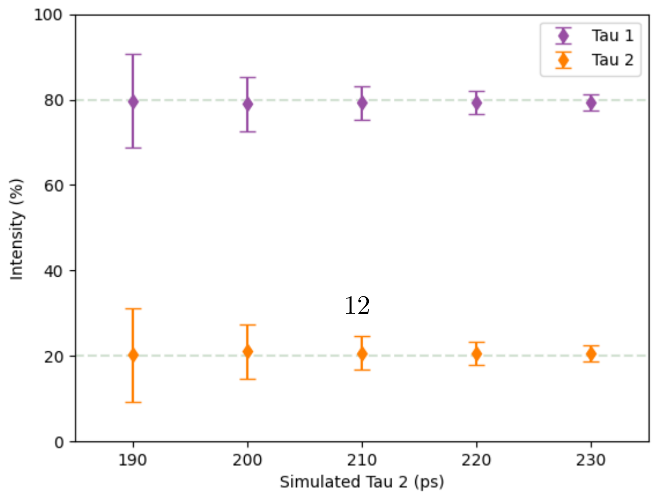
(b) τ_2 difference



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

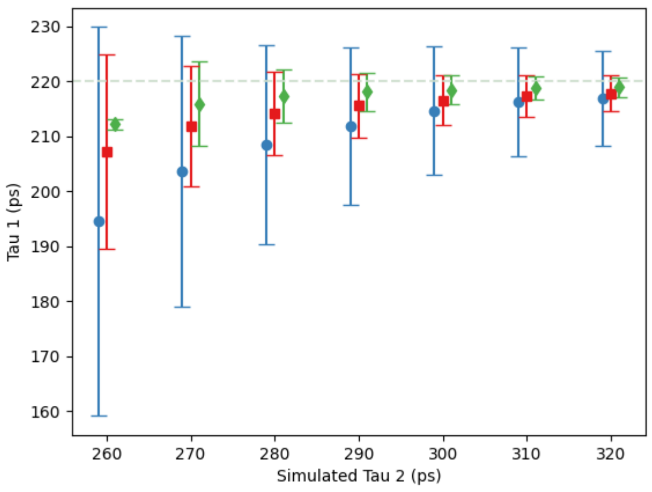


(d) $\tau_1 = 50\%, \tau_2 = 50\%$

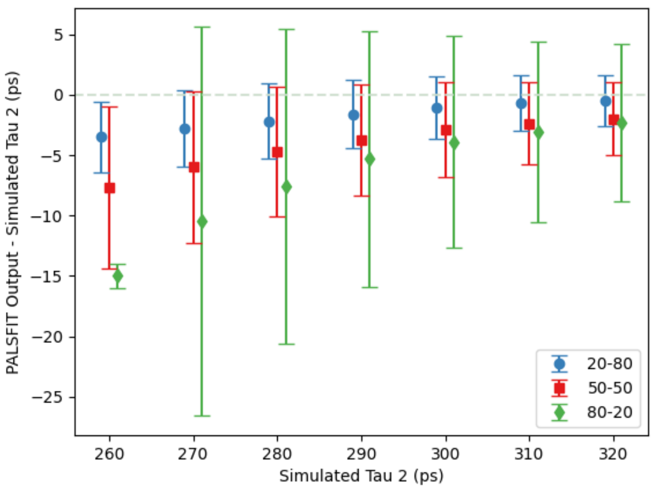


(e) $\tau_1 = 80\%, \tau_2 = 20\%$

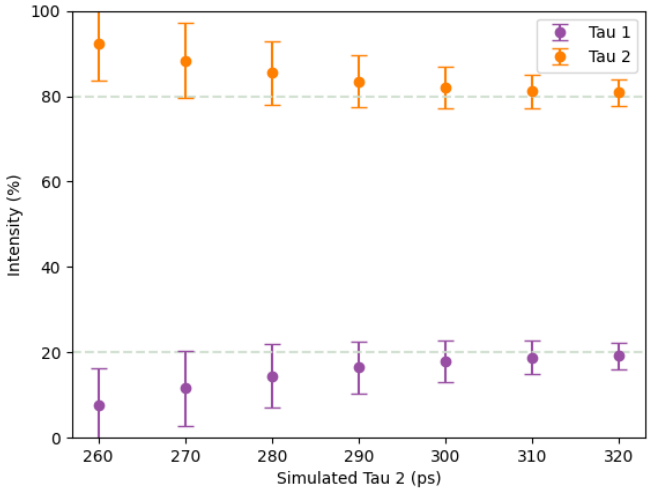
Figure 3.4



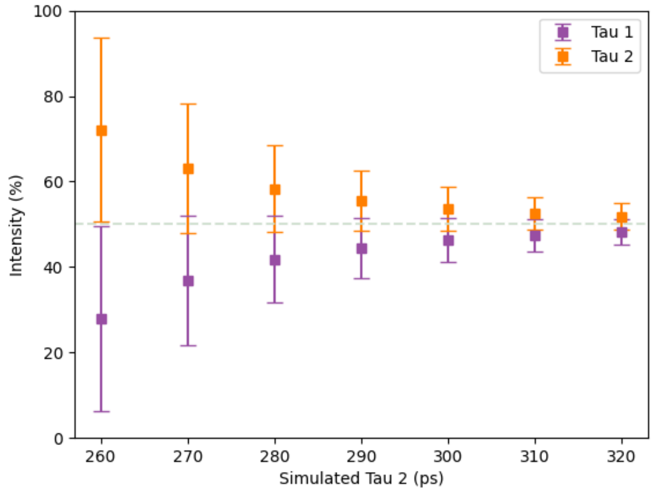
(a) fixed $\tau_1 = 220ps$



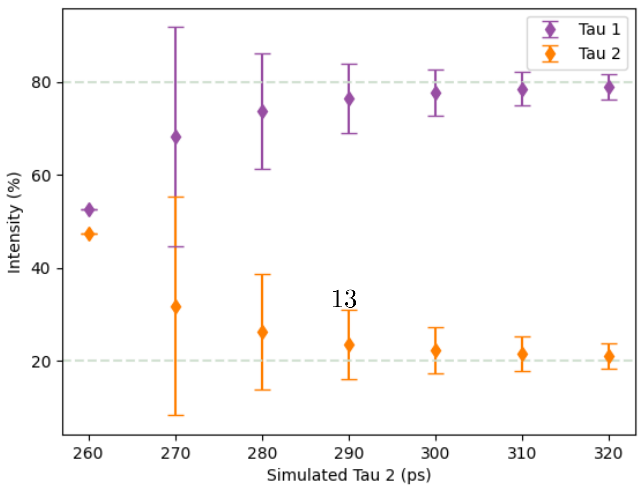
(b) τ_2 difference



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

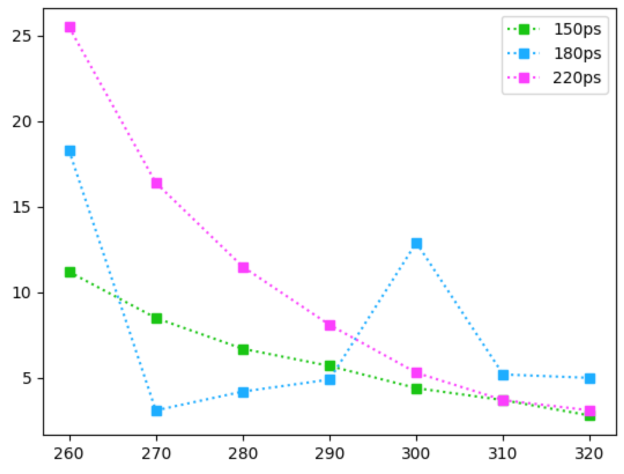


(d) $\tau_1 = 50\%, \tau_2 = 50\%$

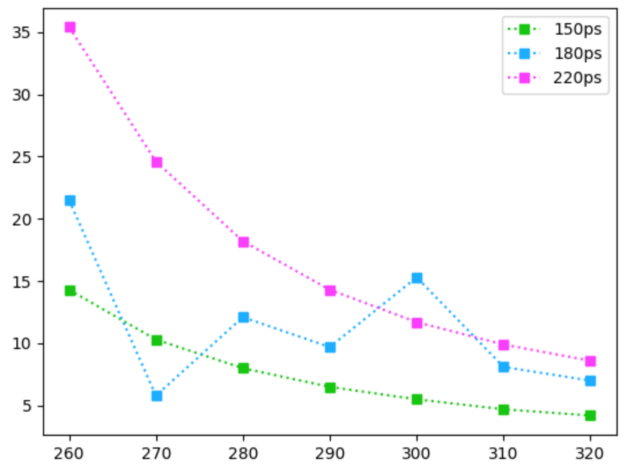


(e) $\tau_1 = 80\%, \tau_2 = 20\%$

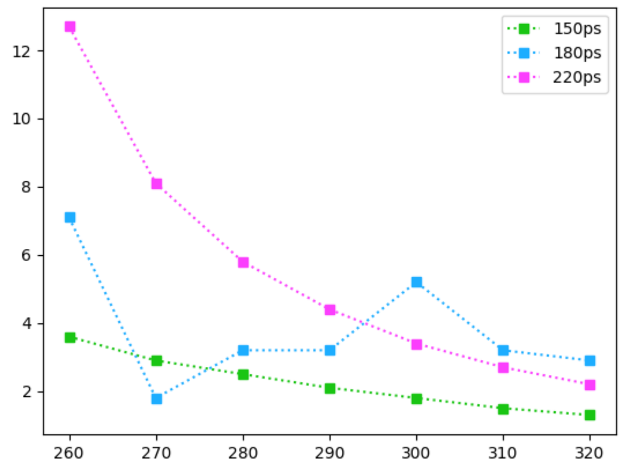
Figure 3.5: τ_1, rows = 20-80, 50-50, 80-20, columns = diff, std dev



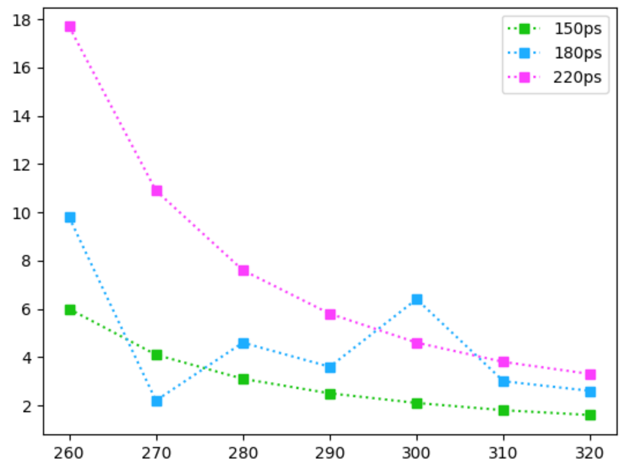
(a)



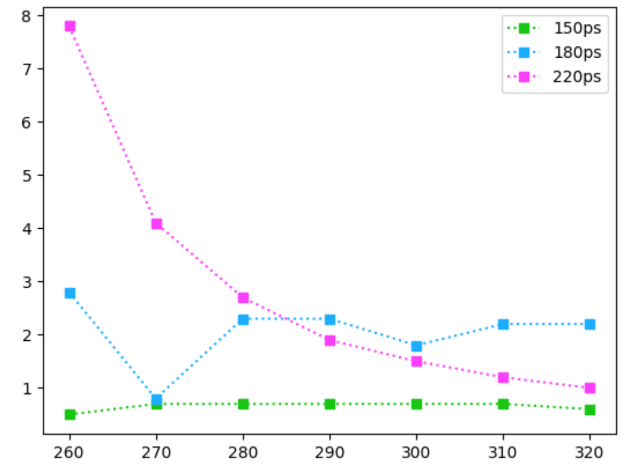
(b)



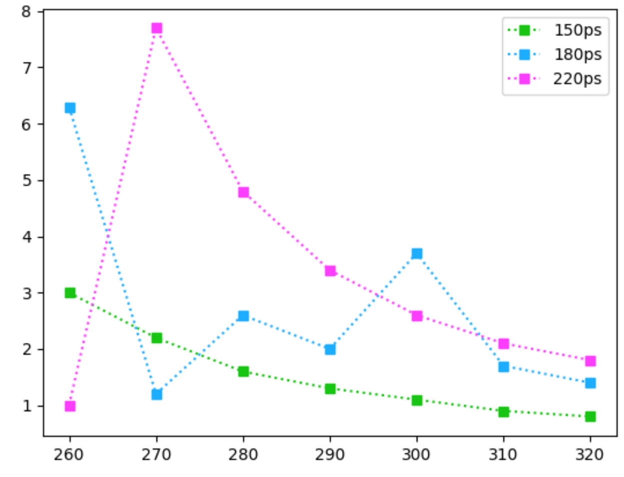
(c)



(d)

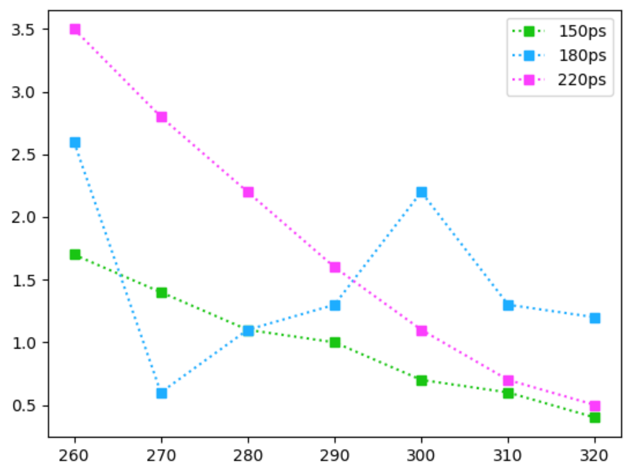


(e)

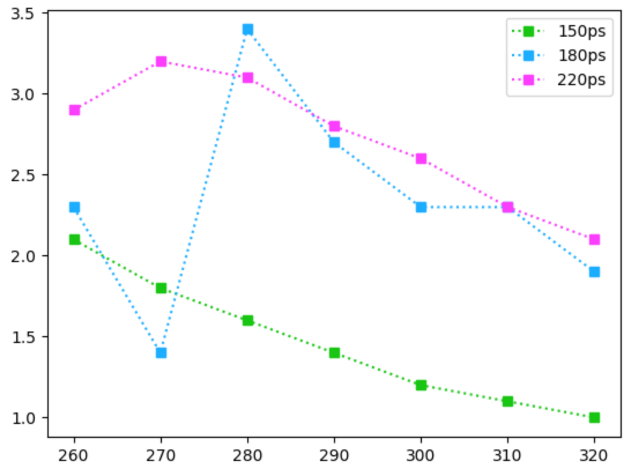


(f)

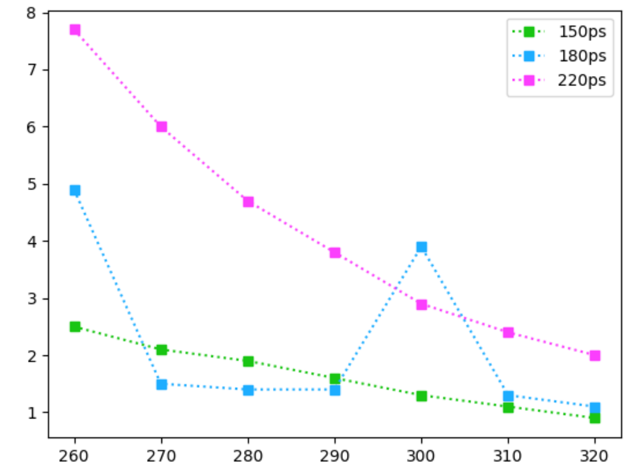
Figure 3.6: τ_2, rows = 20-80, 50-50, 80-20, columns = diff, std dev



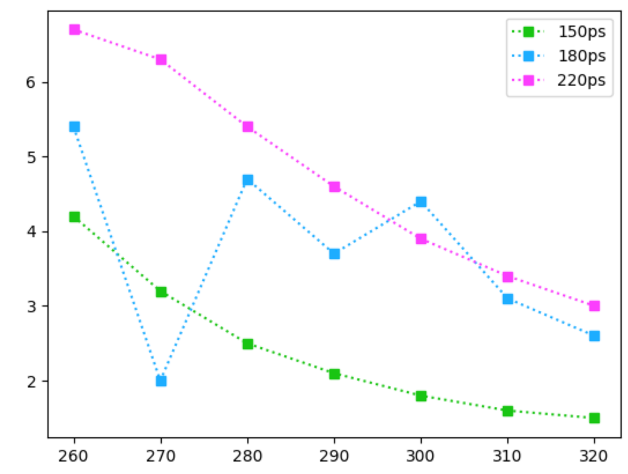
(a)



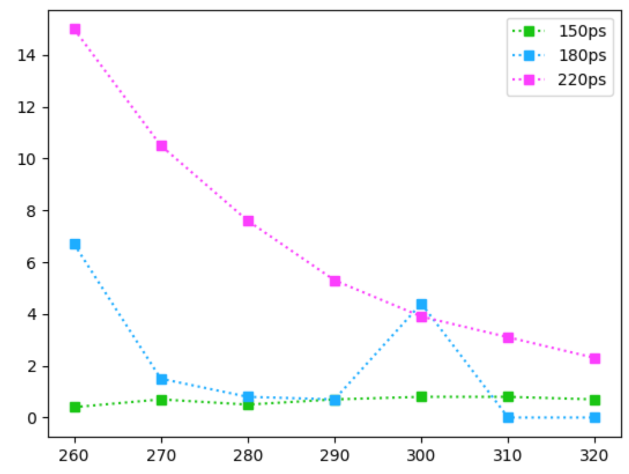
(b)



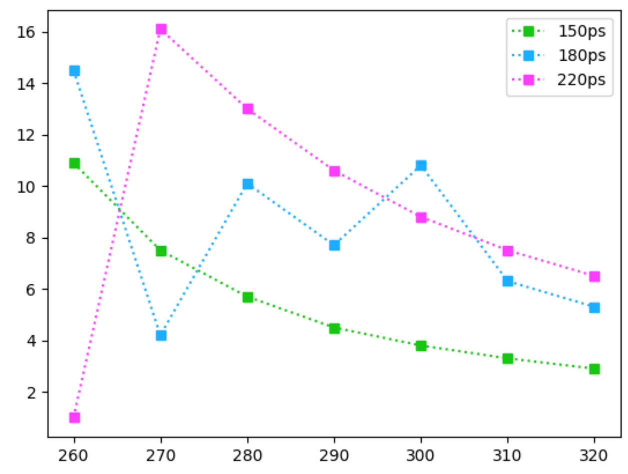
(c)



(d)

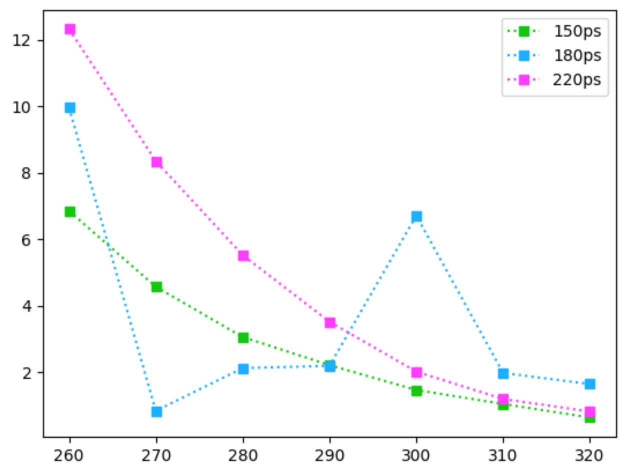


(e)

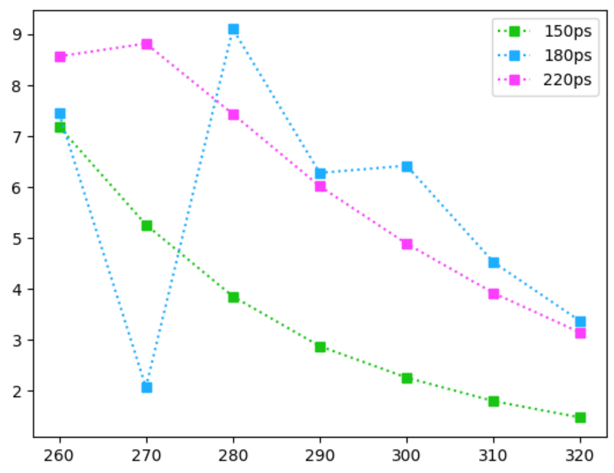


(f)

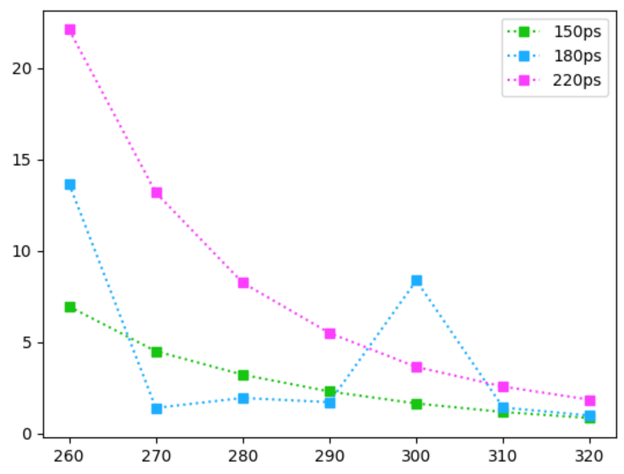
Figure 3.7: intensities, rows = 20-80, 50-50, 80-20, columns = diff, std dev



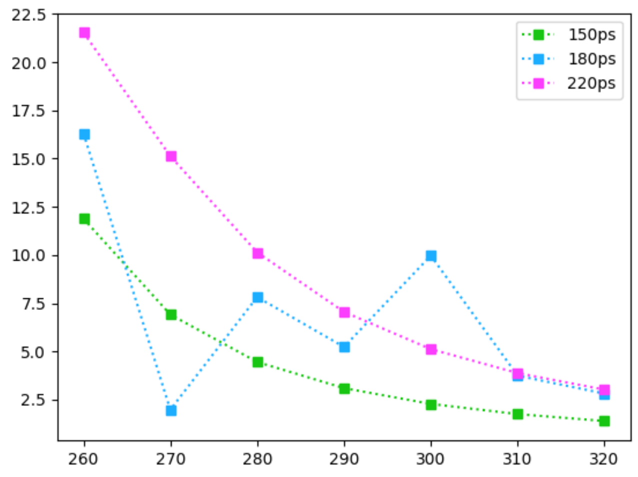
(a)



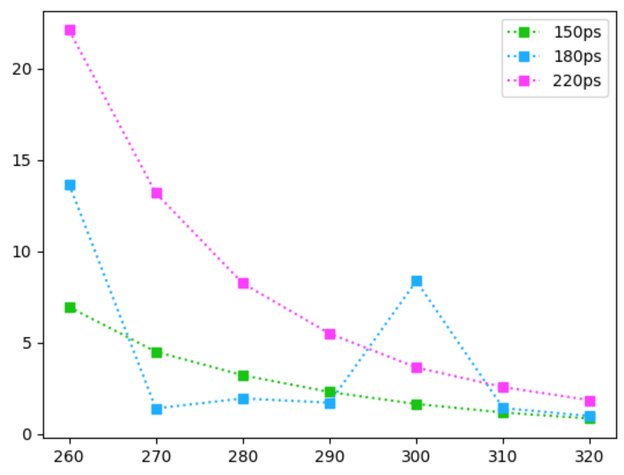
(b)



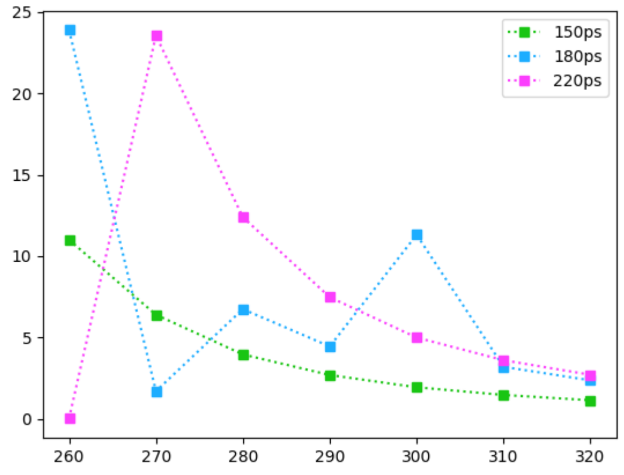
(c)



(d)



(e)



(f)

3.3 Single Gaussian IRF

To investigate how the instrument resolution function affects the software, the three Gaussian resolution function in use so far can be simplified to a single Gaussian (refer to . . .) . As a sanity check, we can compare how the program performs with a single Gaussian resolution function versus the previous three Gaussian resolution function. Setting $\tau_1 = 150$ps, as this previously gave us the best results, we get the data in Figure 3.8. We can then compare this to Figure 3.3 and see that the single Gaussian resolution function produces remarkably similar results, demonstrating the single Gaussian approximation to be valid.

reference to justification, placed earlier

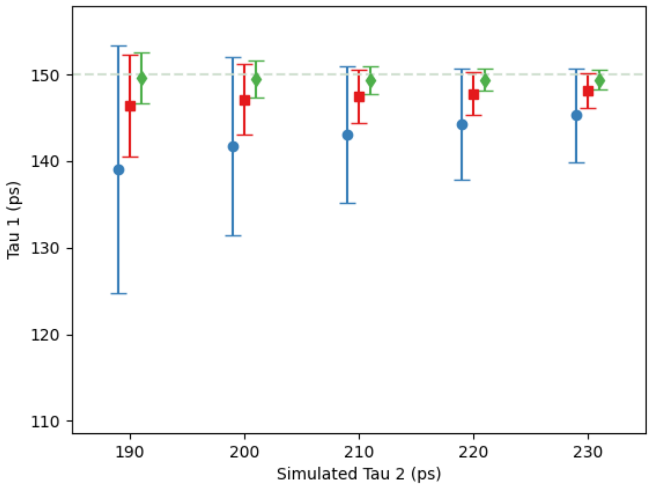
With that established, spectra were generated and fitted for FWHM values of 220ps, 180ps, 150ps and 100ps, with τ_1 set to 150ps and τ_2 ranging from 180-230ps. In general, the trend we expect is the same increase in precision and accuracy with increasing τ_2 that we see previously, but also an increase in both with narrower resolution functions. We would see this as multiple, separate, decreasing trendlines, ordered so that those corresponding to wider resolution functions were always higher. The results we get are shown in Figures 3.9-3.11, and we can analyse them in two parts.

If we just look at the subfigures on the left ((a),(c) and (e) for each figure), which represent the difference between the fit and original data, we see that for the 20%-80% and 50%-50% data the data follows the expected trend. When the τ_1 intensity is set to %80, however, we notice that accuracy doesn't seem to follow the expected relationship with the width of the resolution function. In many cases we have an outright reversal of the predicted relationship and in each figure we have significant crossing between the trendlines.

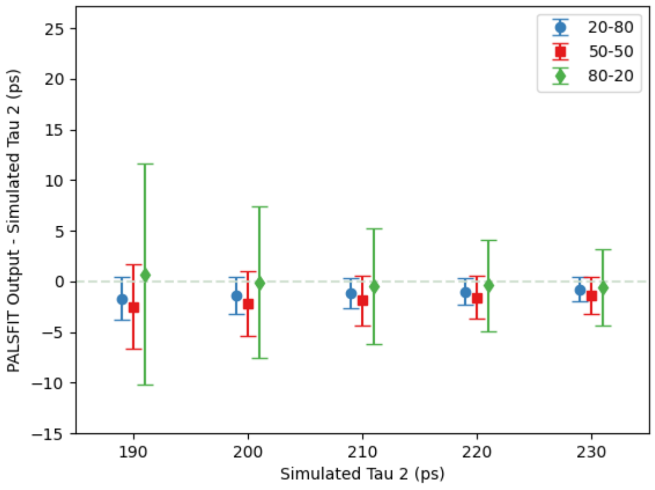
Looking at the subfigures on the right, representing the standard deviation of the fitting given by PALSFIT, we again see three outliers where the predicted relationship between precision and resolution function width is reversed, in Figures 3.10b, 3.10d and 3.11b, though in the case of 3.10d, this reversed relationship quickly reverts to the expected one as τ_2 increases.

possibly expand

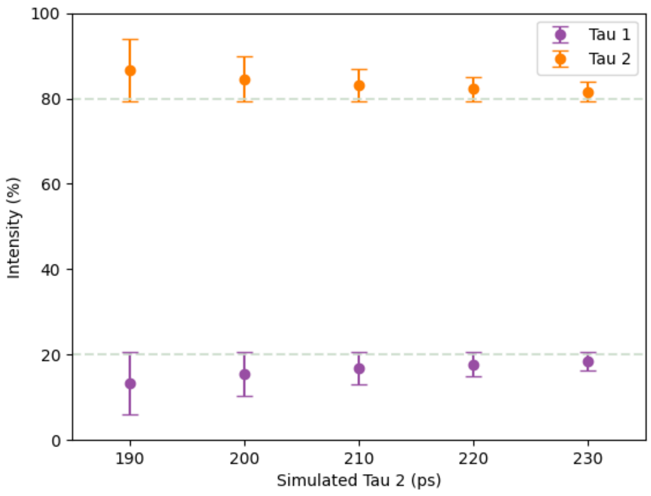
Figure 3.8



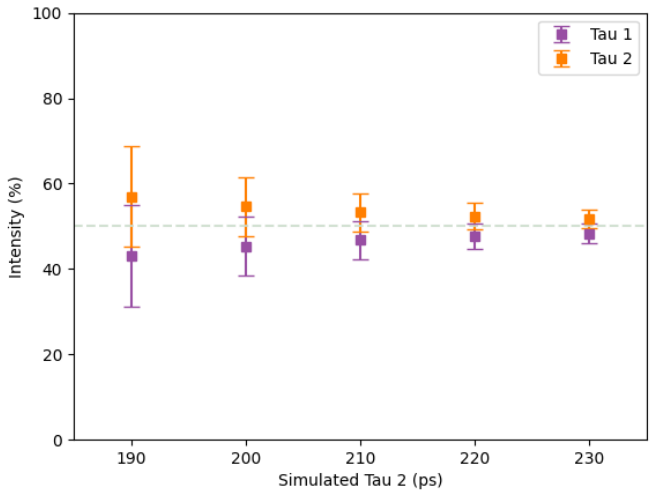
(a) fixed $\tau_1 = 150ps$



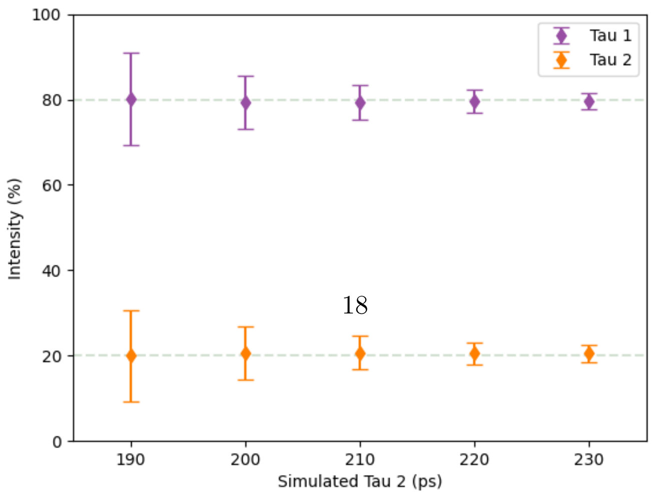
(b) τ_2 difference



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

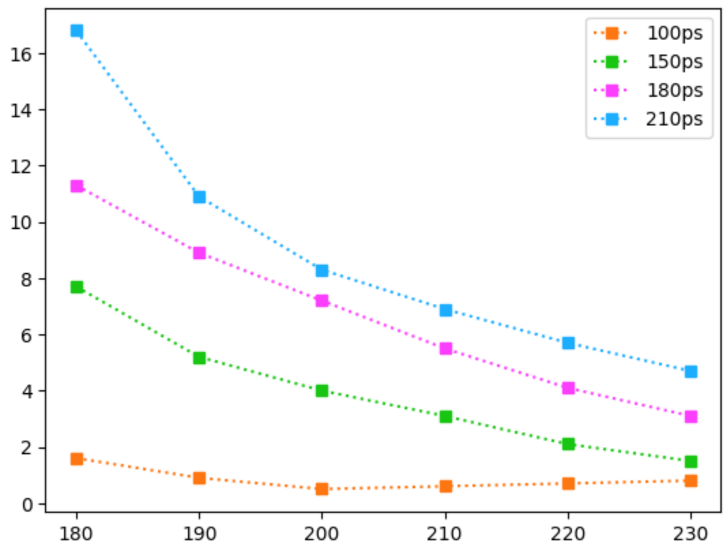


(d) $\tau_1 = 50\%, \tau_2 = 50\%$

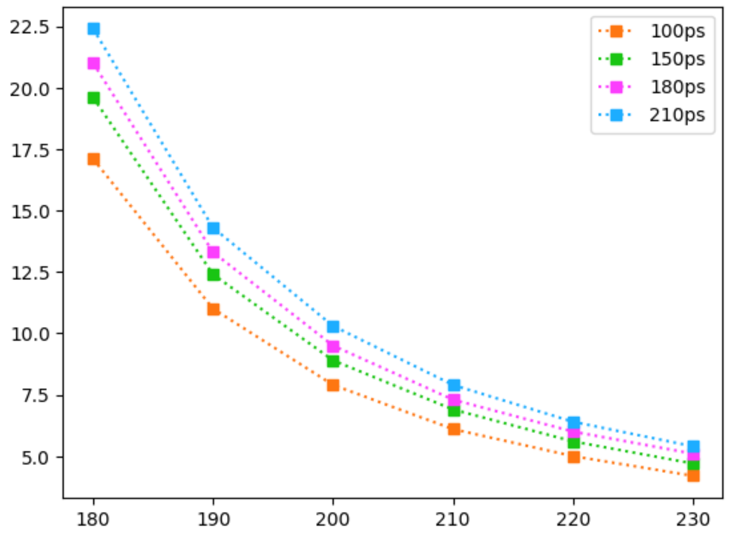


(e) $\tau_1 = 80\%, \tau_2 = 20\%$

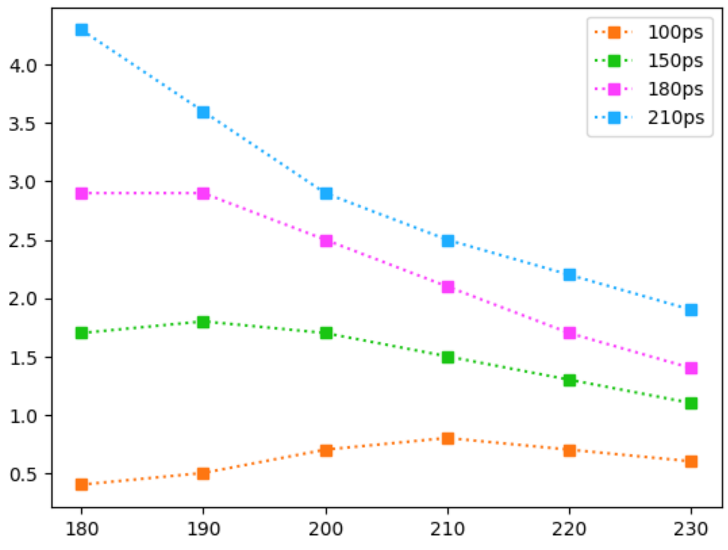
Figure 3.9: τ_1, rows = 20-80, 50-50, 80-20, columns = diff, std dev



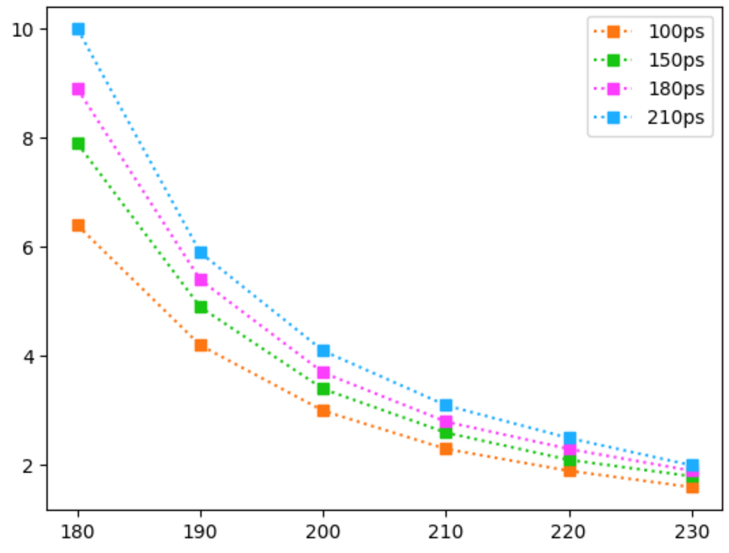
(a)



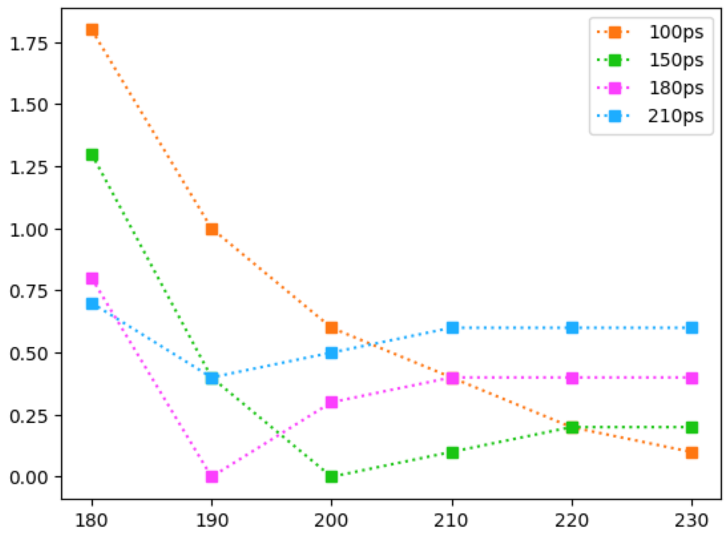
(b)



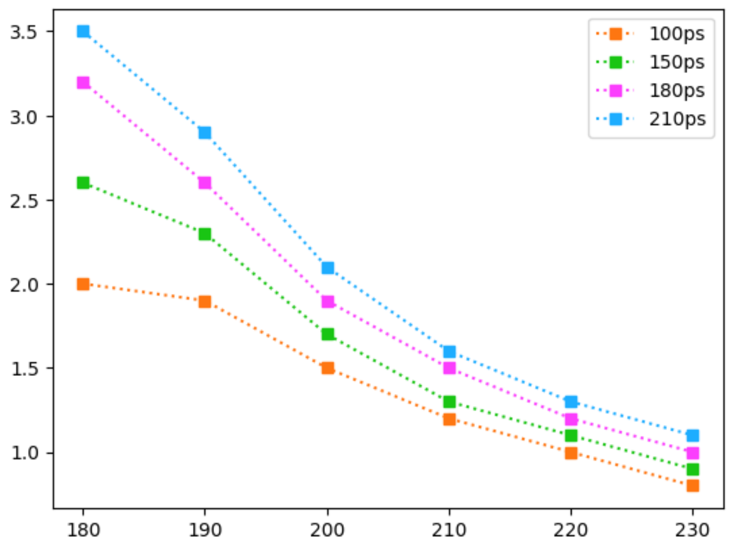
(c)



(d)

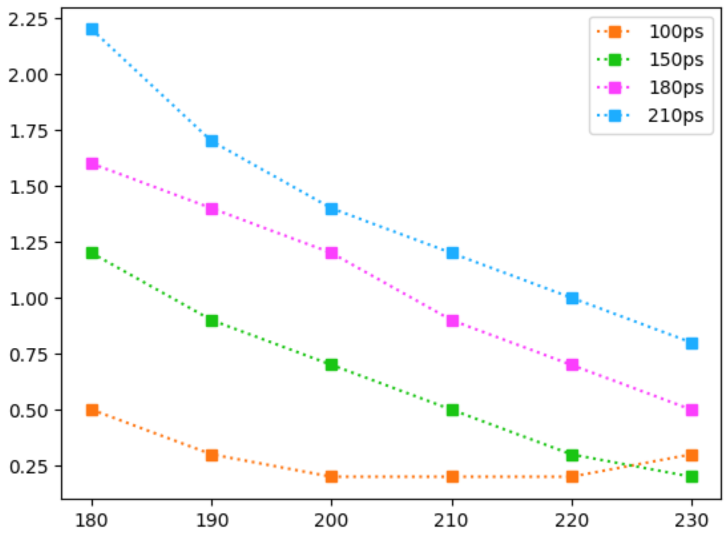


(e)

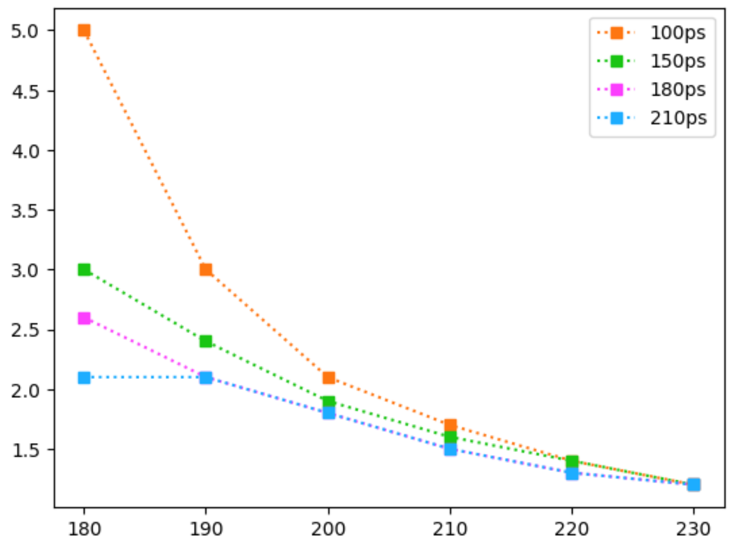


(f)

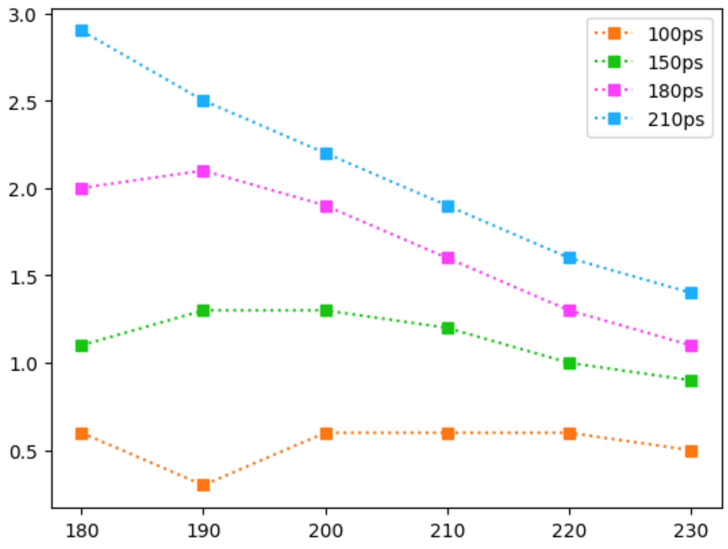
Figure 3.10: τ_2, rows = 20-80, 50-50, 80-20, columns = diff, std dev



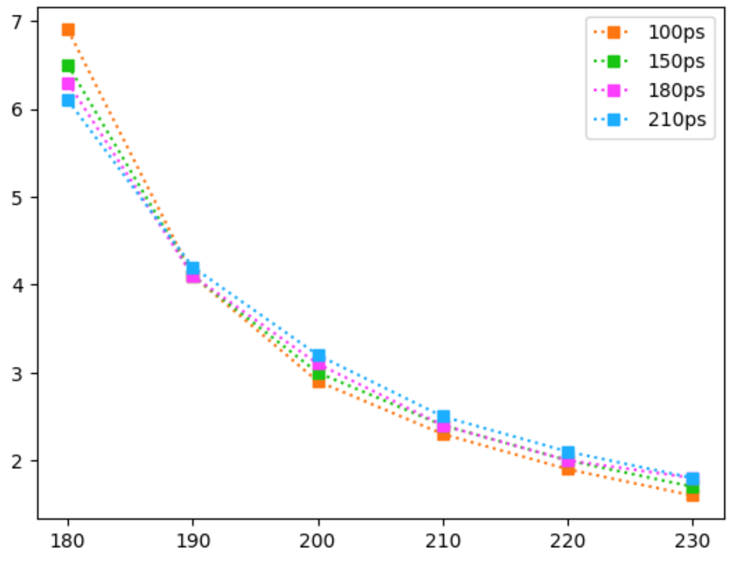
(a)



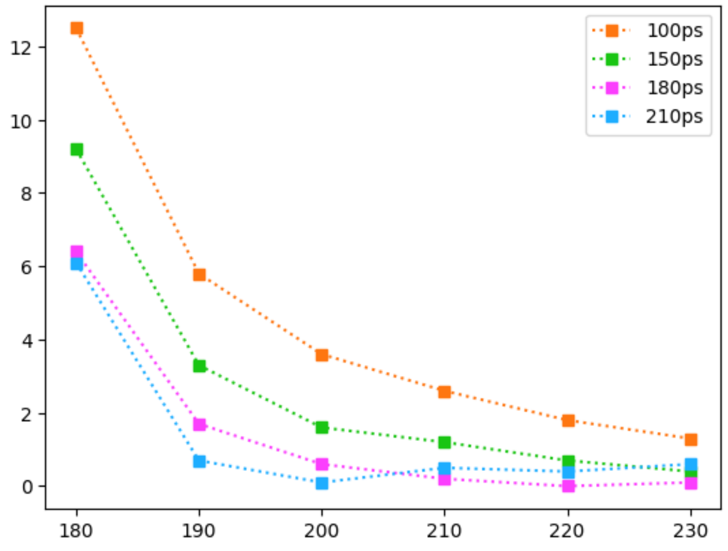
(b)



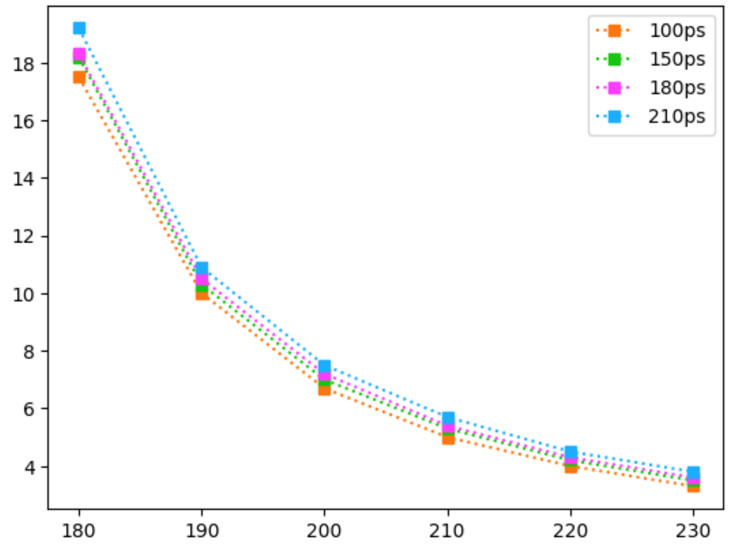
(c)



(d)

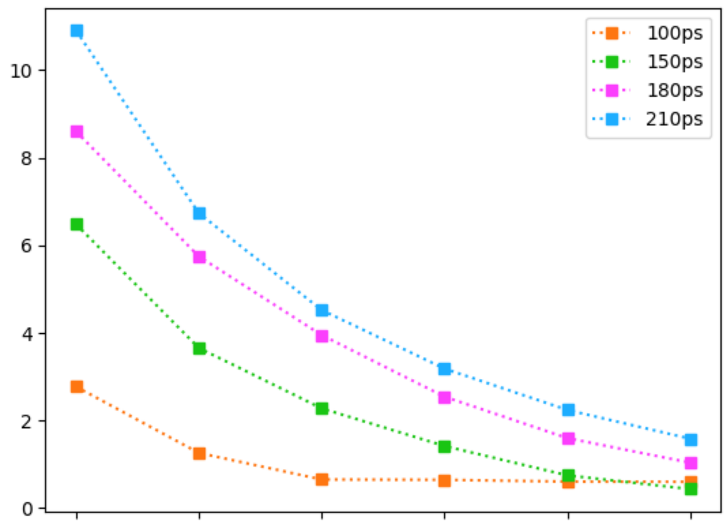


(e)

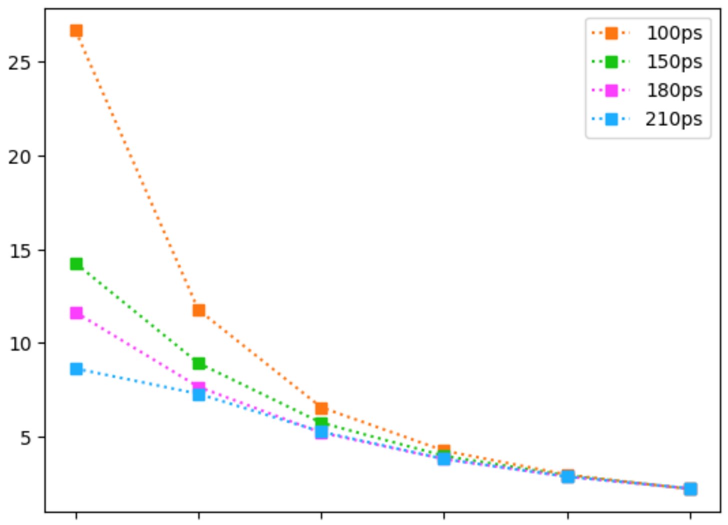


(f)

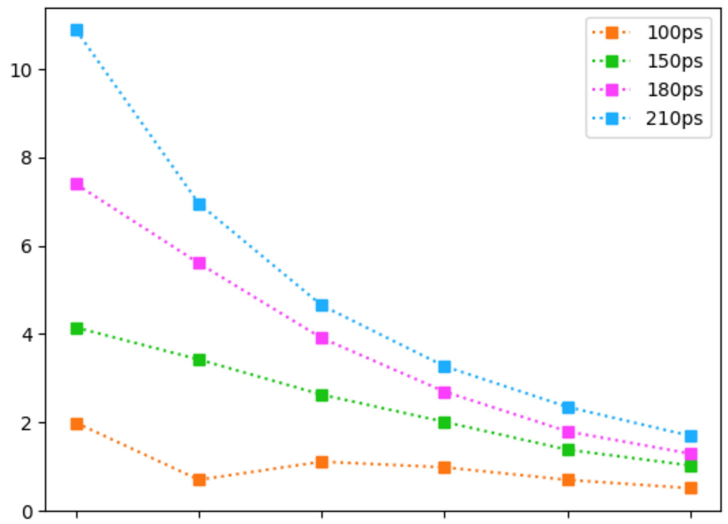
Figure 3.11: intensities, rows = 20-80, 50-50, 80-20, columns = diff, std dev



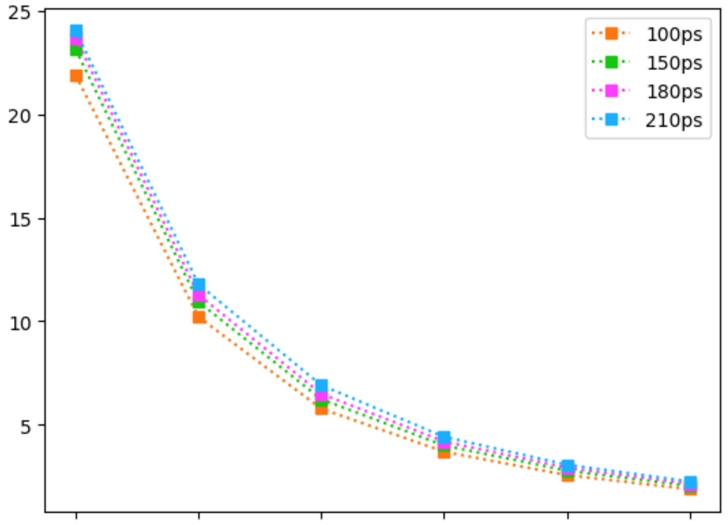
(a)



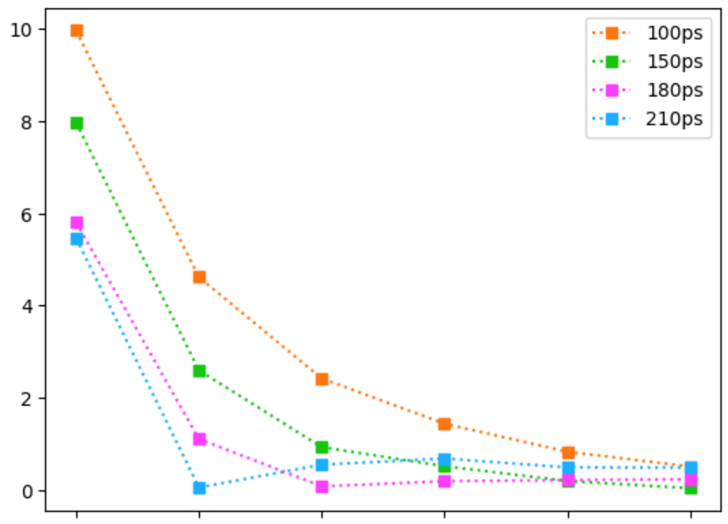
(b)



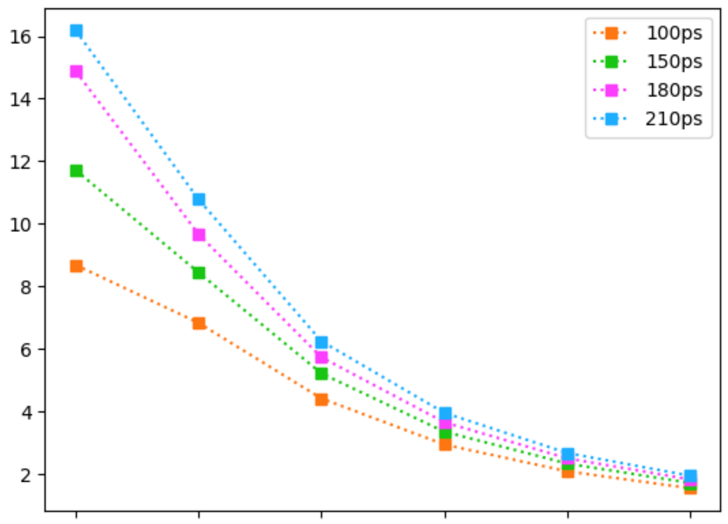
(c)



(d)



(e)



(f)

3.4 Number of counts

Another variable worth examining is the total number of counts in a given spectrum. When generating spectra using PALSSIM, this is given as the area of the spectrum without the background and a separate background value. Both were kept constant in all previous runs, with the area set to 5.65×10^6 and the background value set to 8.5. Expectations are that a higher number of counts would make spectrum analysis easier. To test this hypothesis, spectra with generated for τ_1 = 150ps, τ_2 = 180-230ps and a 210ps FWHM single gaussian resolution function, tripling the background area to 1.7035×10^7.

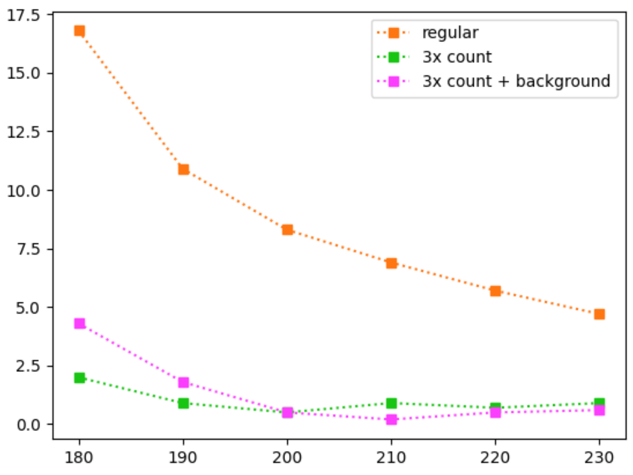
If we just increase the total area of the spectra and maintain the same background value as before, however, we inadvertently end up increasing the signal to noise ratio. While this would be desirable, in a practical setting an increase in total number of counts most likely corresponds to an increase in background noise. As such, spectra were also generated with both the tripled area and a proportional 3x increase in the background value (set to 25.5).

The generated spectra were then analyzed with PALSFIT, and the results are summarized in Figures 3.12-3.14, with a "regular" run for comparison. The regular run is simply the one represented in Figure 3.8, which used the original, unaltered area and background value.

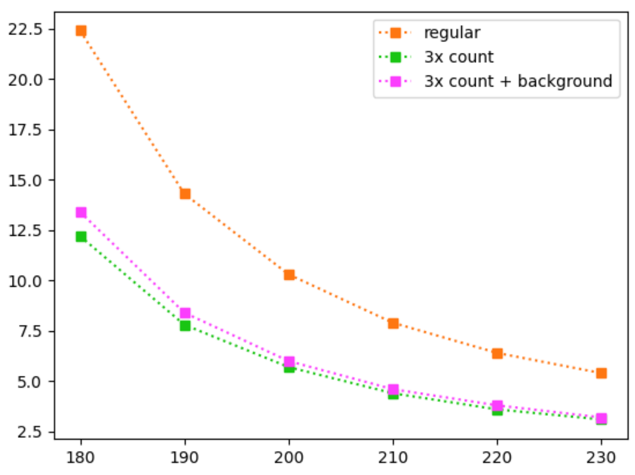
Looking first at the standard deviations (subfigures (b), (d) and (f), on the right side of the page), we see that, as expected, the data from the regular run is in most cases significantly less precise than the other two runs. Additionally, the data with the proportionally scaled background value seems to be slightly less precise than its unscaled counterpart, though in many cases just barely so, if at all. Exceptions tend to occur, however, for the shortest lifetime separation, when τ_2 is set to 180ps.

Looking at the left side of the figures ((a), (c) and (e)), we see that for the most part the first trend seem to hold true, with the regular run being significantly less accurate than the other two. The exception to this seems to be when the relative τ_1-τ_2 intensity is 80%-20%, in which we often have a reversal of expectations, at times with some significant crossover of the relevant trendlines. While, at a glance, the normal and scaled background triple area runs are still for the most part relatively close, with the proportional background looking less accurate overall, this isn't as universally true as it is when looking at the standard deviation. Specifically in Figures 3.12a, 3.13a and 3.14a, when τ_1 = 20% and τ_2 = 80%, PALSFIT seems to have performed worse with the non-scaled background, at least for significant portions of the graphs.

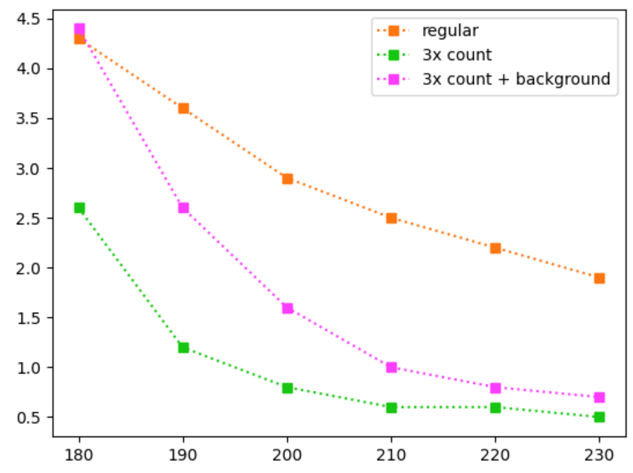
Figure 3.12: τ_1, rows = 20-80, 50-50, 80-20, columns = diff, std dev



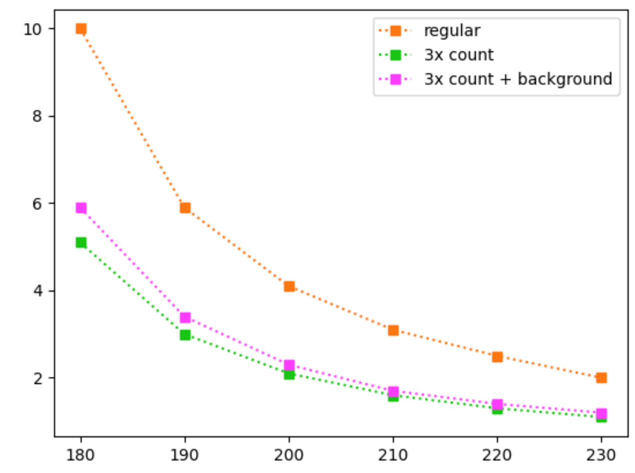
(a)



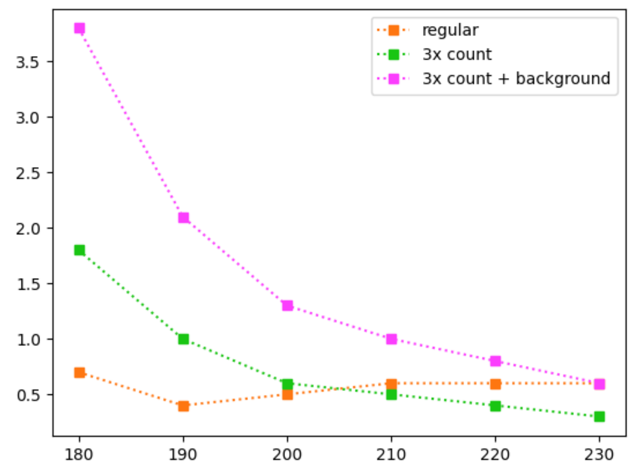
(b)



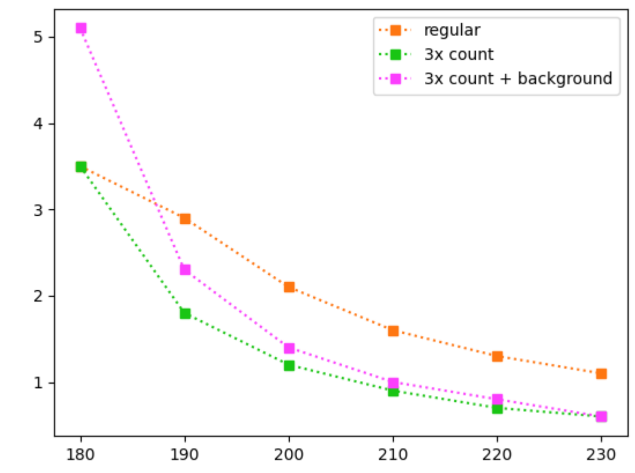
(c)



(d)

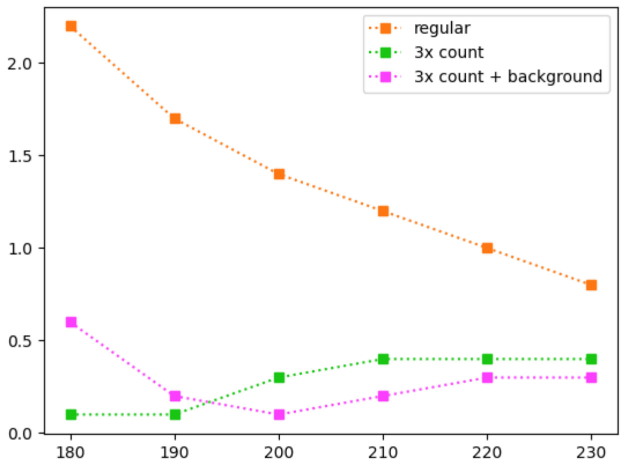


(e)

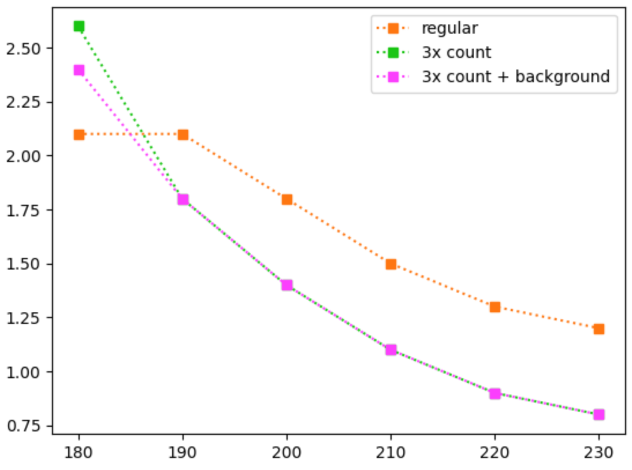


(f)

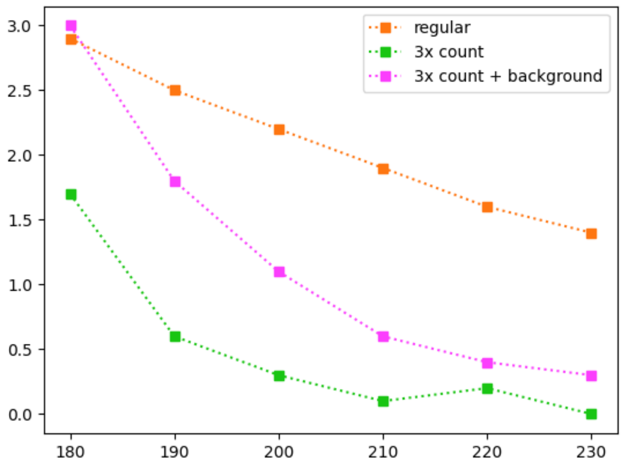
Figure 3.13: τ_2, rows = 20-80, 50-50, 80-20, columns = diff, std dev



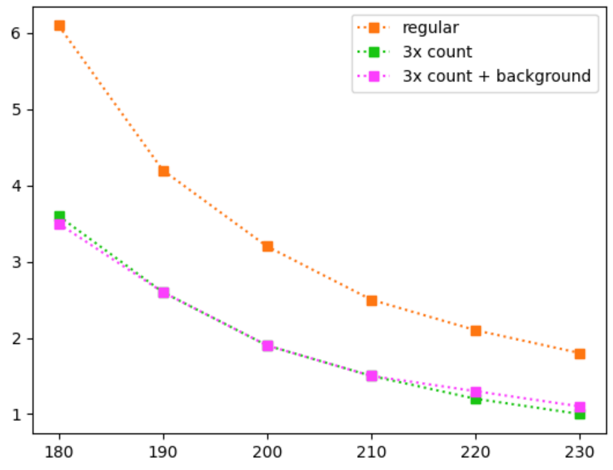
(a)



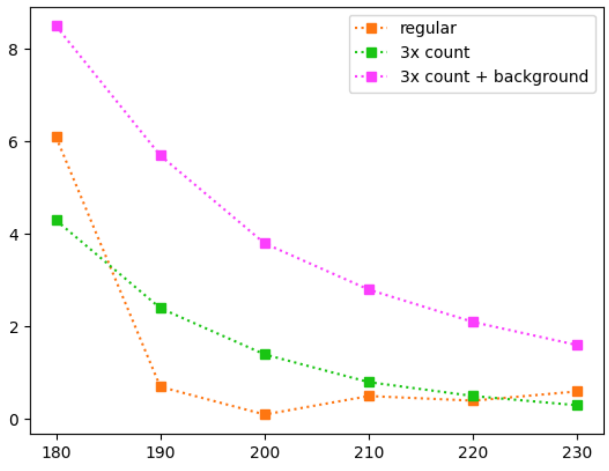
(b)



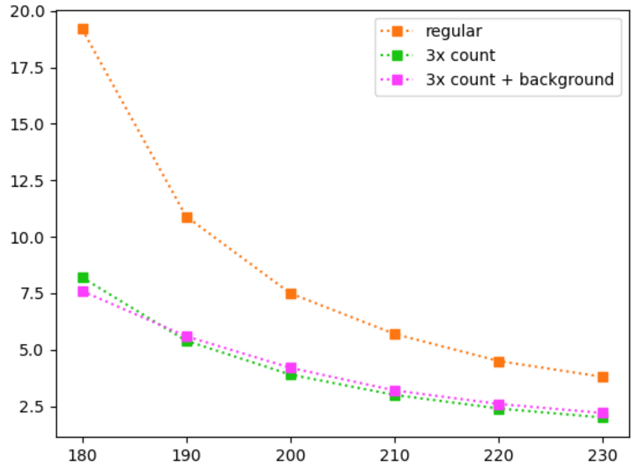
(c)



(d)

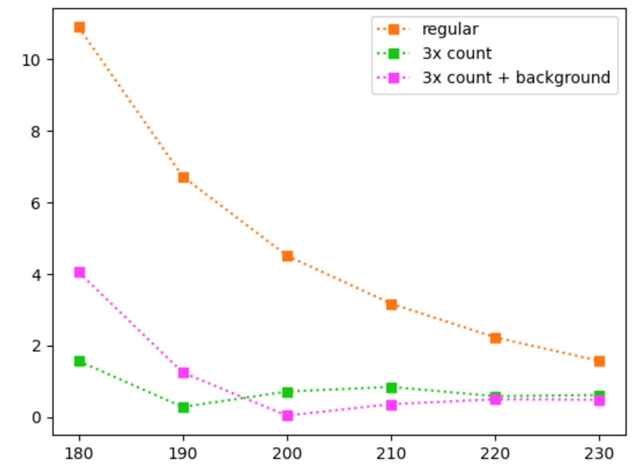


(e)

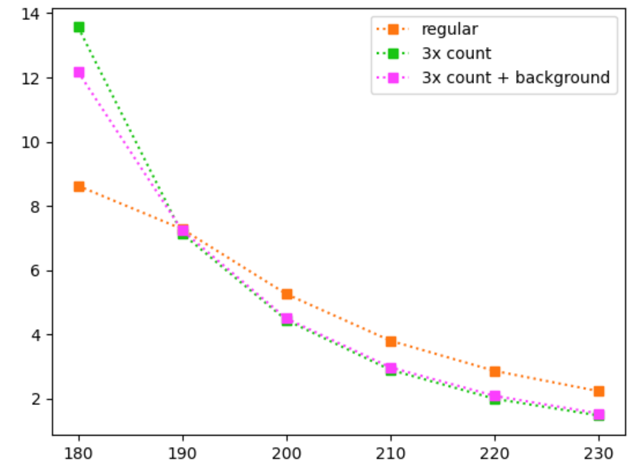


(f)

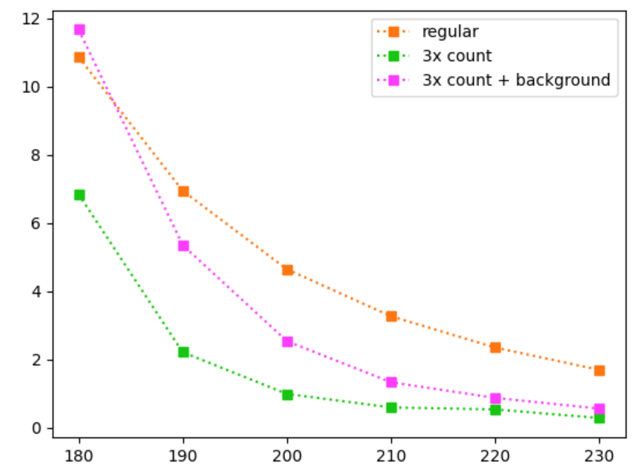
Figure 3.14: intensities, rows = 20-80, 50-50, 80-20, columns = diff, std dev



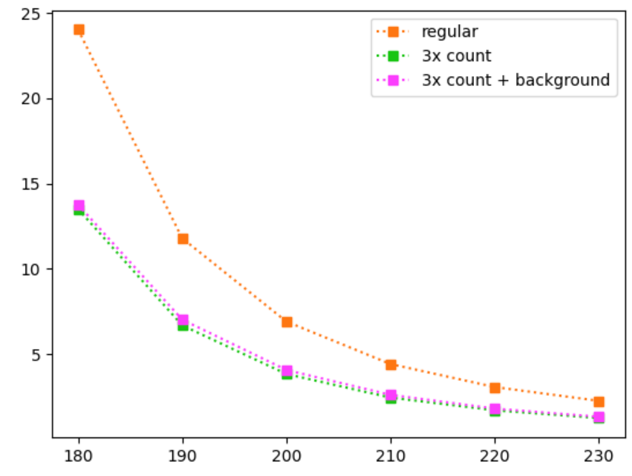
(a)



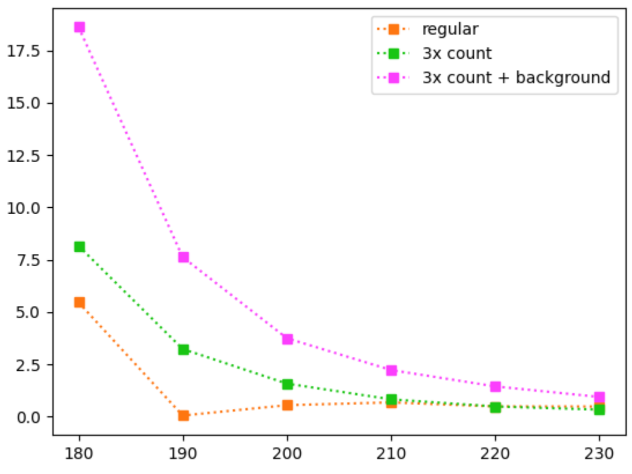
(b)



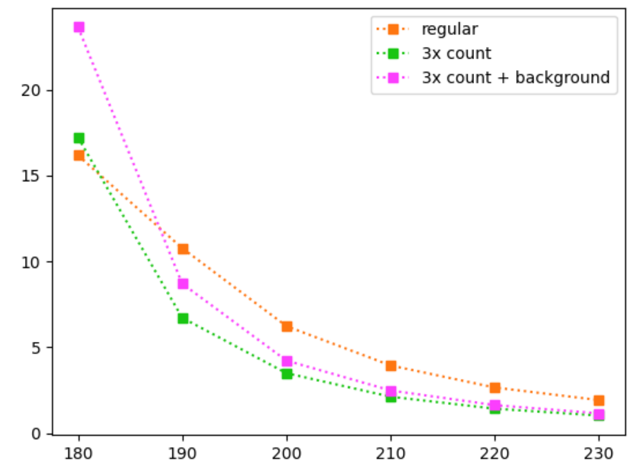
(c)



(d)



(e)



(f)

3.5 Realistic data

Analysis of real world data often differs from how we've conducted our analysis so far. One key factor is that usually the number of lifetimes present in a real life spectrum isn't known beforehand. As PALSFIT requires the user to input the number of fitted lifetimes, this might lead to situations in which the user predicts the wrong number of lifetimes, thus it might be useful to explore how the software behaves in such a situation.

To do so, three materials (?) were modeled, with three lifetime components. All three have two shorter components that differed between them, and a long lifetime component set to 2.6ns with a 0.15% intensity that was kept equal. The relative intensities of the two shorter lifetime components were adjusted to the following values:

relative		absolute	
τ_1	τ_2	τ_1	τ_2
99.5%	0.5%	99.3508%*	0.4993%*
90%	10%	89.865%	9.985%
80%	20%	89.88%	19.97%
50%	50%	49.925%	49.925%
20%	80%	19.97%	9.985%

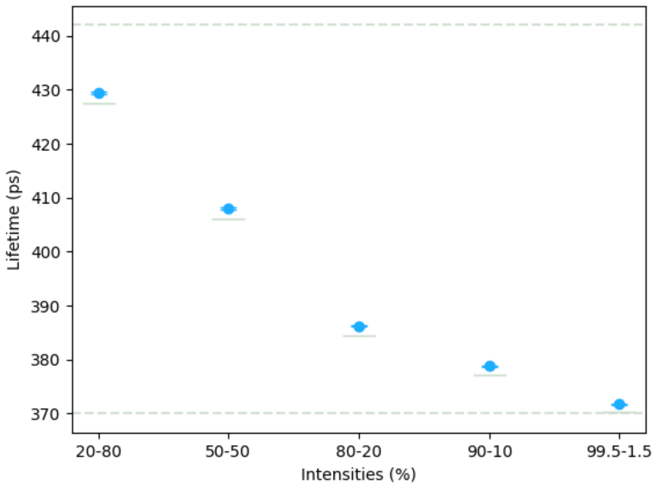
(a) intensities

τ_1(ps)	τ_2(ps)	τ_3(ns)
370	442	2.6
355	444	2.6
348	440	2.6

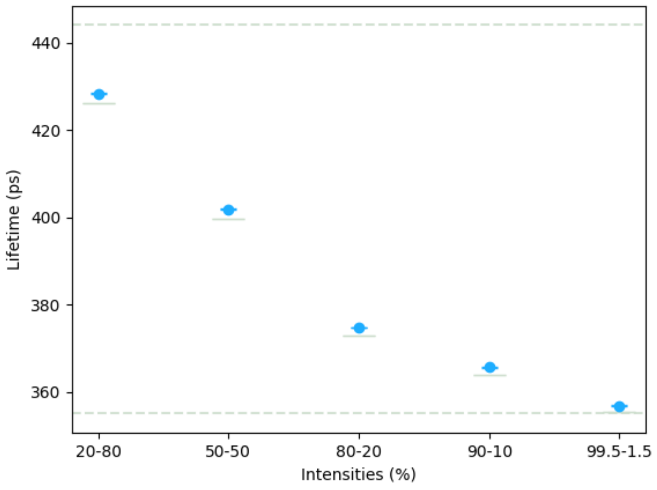
(b) lifetimes

When a single lifetime is fitted, the resulting lifetime seems to tend towards a weighted average of the two shorter lifetimes, as can be seen in the following figures:

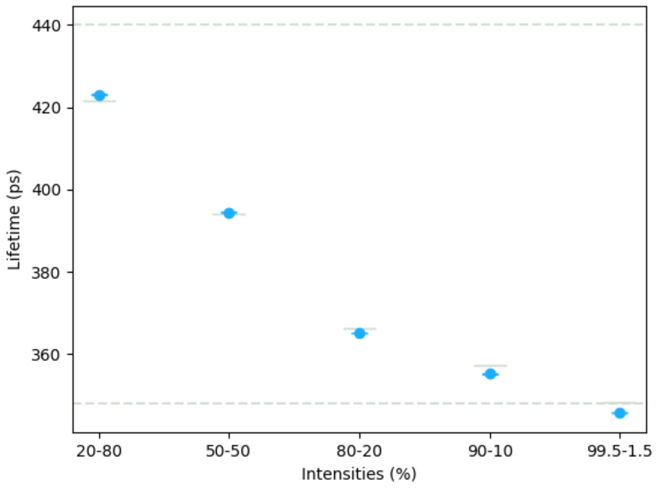
Needs to be finished.



(a) $\tau_1 = 370$, $\tau_2 = 442$

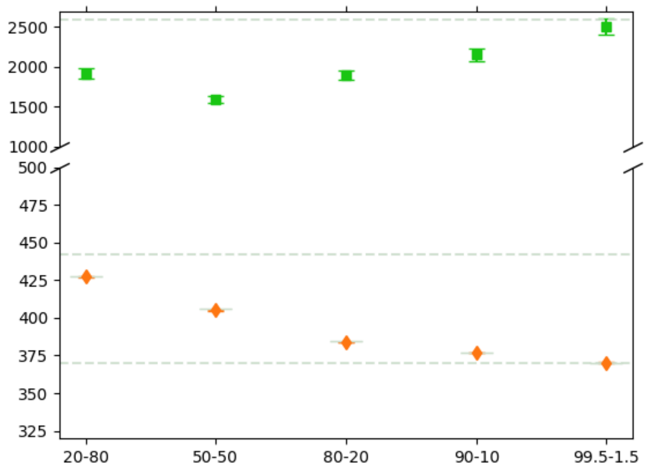


(b) $\tau_1 = 355$, $\tau_2 = 444$

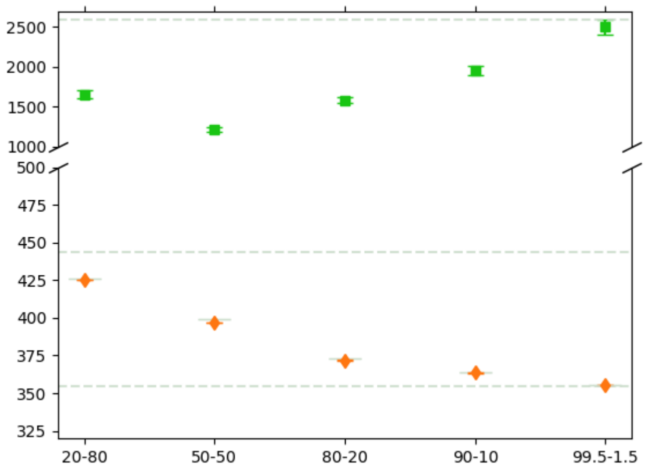


(c) $\tau_1 = 348$, $\tau_2 = 440$

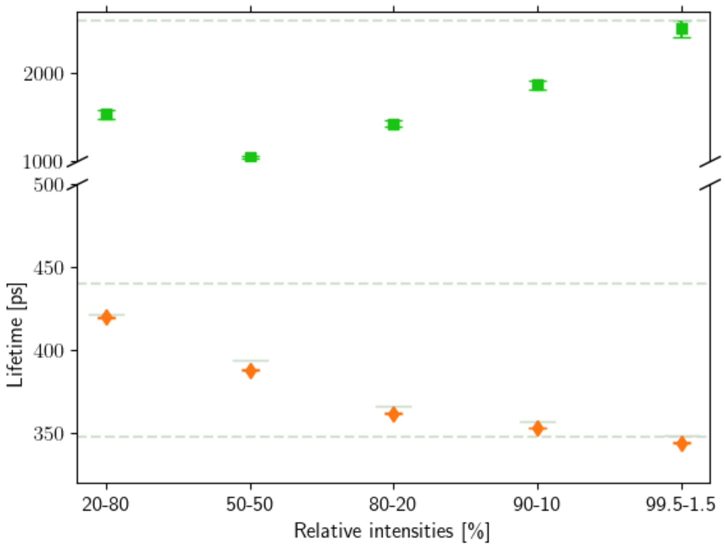
Figure 3.15: single lifetime fit



(a) $\tau_1 = 370$, $\tau_2 = 442$

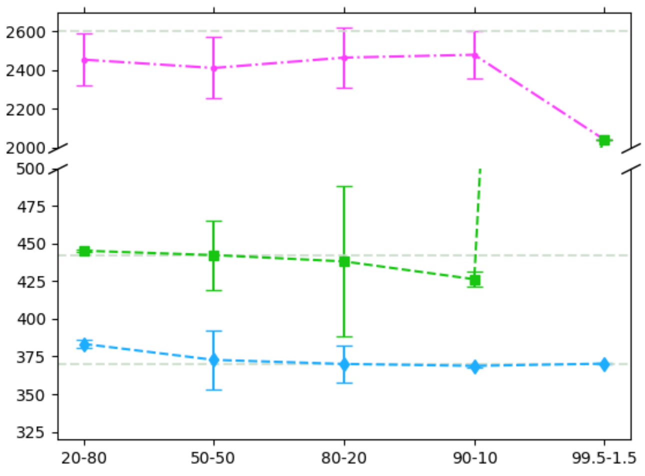


(b) $\tau_1 = 355$, $\tau_2 = 444$

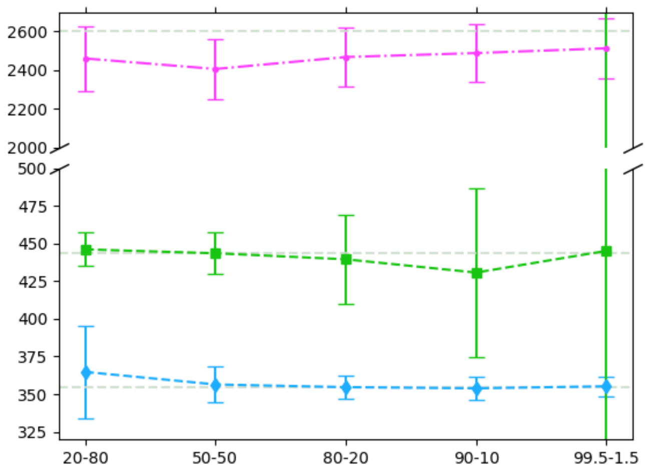


(c) $\tau_1 = 348$, $\tau_2 = 440$

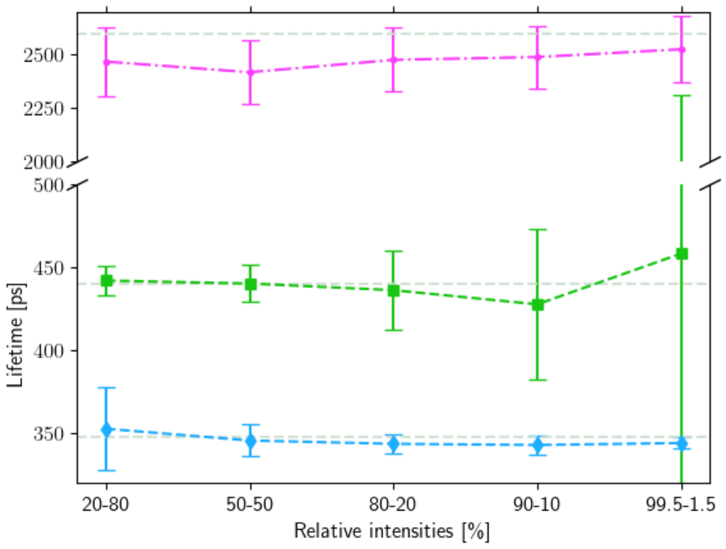
Figure 3.16: two lifetime fit



(a) $\tau_1 = 370$, $\tau_2 = 442$



(b) $\tau_1 = 355$, $\tau_2 = 444$



(c) $\tau_1 = 348$, $\tau_2 = 440$

Figure 3.17: three lifetime fit

3.6 Using the STM to vary τ_1:

Using STM, vacancy concentration rate was varied for a given value of defect lifetime τ_2. This was done for $\tau_2 = 370, 442$. Vacancy conc. rate was varied as such:

Table 3.3: Vacancy Conc.

Conc.
1E15
2E15
4E15
6E15
8E15
1E16
2E16
4E16
6E16
8E16
1E17
2E17
4E17
6E17
8E17
1E18

Needs to be typed up