University of Dundee

School of Science and Engineering

Physics

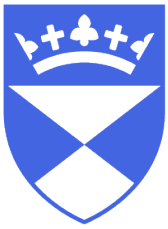
Analysis of PALS spectra using PALSfit

Francesco Tamburi

PH40006 BSc Physics project report

Academic year: 2023-2024

Supervisor: David J. Keeble



University of Dundee

Declaration of authorship

The following work has been completed by 13/04/2024 and is wholly my own work, unless stated otherwise. It has not been submitted for assessment elsewhere and I have not knowingly allowed it to be used or copied by any other person. I confirm that I have read and understood the School policy on Academic Dishonesty.

Candidate's signature: Francesco Tamburi Date 13/04/2024

Contents

Abstract

Positron Annihilation Lifetime Spectroscopy is an experimental technique used probe a sample material for defects in its structure. Positron lifetime is a function of electron density and so multiple lifetimes indicate areas of decreased density, such as defects in the lattice structure. Positron lifetimes are measured in a material and compiled into a lifetime spectrum. Decomposing the spectrum into individual lifetime components is mathematically complex, and as such is solved computationally. PALSfit3 is a software package developed for this purpose.

This project uses PALSfit3 to analyze a series of simulated spectra, comparing the fitted results to the simulated parameters. The aim of the project is to find patterns in where and how the program fails, in the hopes of establishing practical guidelines that may aim in analyzing experimental spectra. A variety of spectra were fitted and analyzed in a series of trials and a few key observations were made along the way. These observations ultimately represent potential avenues for further investigation. To that end, the development of tools to simulate, fit and analyze multiple lifetime spectra is encouraged.

Chapter 1

Introduction

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.[3] 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 the material sciences.[4][5]

1.1.1 The Positron Lifetime

Weakly radioactive isotopes, such as ^{22}Na, are commonly used to generate positrons. As part of its β^+ decay, ^{22}Na simultaneously produces a positron and a 1.27 MeV γ photon. Upon encountering the sample material, positrons quickly lose their kinetic energy in a process known as "thermalization".

[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.

The thermalization process isn't uniform, occurring via different mechanisms depending on the sample material and energy of the positron. At high energies, for example, atomic ionization is a dominant process, while at low energies (< 1 eV in metals and ~ 1 eV in semiconductors), phonon scattering is the most important process. Regardless of specific mechanisms in play thermalization only occurs in the first few picoseconds of a positron lifetime.

After thermalization, the positron is free to diffuse through the sample material. During the diffusion process, the wavefunction of the positively charged positron spreads out into a highly delocalized Bloch state, primarily concentrated in the interstitial space between atomic nuclei in the material. It's in this phase that the positron spends the bulk of its lifetime and where defect trapping predominantly occurs.[2] How far a positron diffuses determines how many atoms are probed for positron traps, and thus how sensitive the PALS process is to defects.

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

$$\kappa = \mu C, \tag{1.1}$$

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

Annihilation can occur in the delocalized state or the localized trapped state, producing two 511 keV γ-photons. In both cases, lifetimes are usually on the order of hundreds of picoseconds.

1.1.2 Positronium

Positrons can also enter a third state, however, where, instead of delocalizing into the bulk or being trapped in a defect, the positron enters a bound state with an electron, forming an exotic atom known as positronium (Ps). Annihilation lifetimes for positronium depend on the spin alignment between the two particles and are around 140ps for the less common, parallel spin, ortho-Ps, or 1-5ns for the more common, antiparallel spin, para-Ps.

1.1.3 Measuring Lifetimes

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. The

[2]Some trapping can occur during the thermalization process. The effect of this on positron lifetimes, however, has been proven to be negligible.

experimental setup for doing so begins with a coupled scintillator and photomultiplier, which converts the γ-rays into analog electrical pulses. These are then processed by discriminators, distinguishing the 1.27 MeV (in the case of ^{22}Na) 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.

1.1.4 The Spectrum

A lifetime 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[2] – given by the expression

$$D(t) = \sum_{i=1}^{k+1} I_i \exp\left(-\frac{t}{\tau_i}\right), \tag{1.2}$$

where τ_i and I_i represent the lifetime and intensity of each component, respectively.[3] The resulting positron lifetime spectrum $N(t)$ is given by the absolute value of the time derivative of the positron decay spectrum $D(t)$[4]:

$$N(t) = \left|\frac{\mathrm{d}D(t)}{\mathrm{d}t}\right| = \sum_{i=1}^{k+1} \frac{I_i}{\tau_i} \exp\left(-\frac{t+t_0}{\tau_i}\right). \tag{1.3}$$

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

$$F(t) = \sum G_i(t), \tag{1.4}$$

with variations in the peak position, relative height and width of each Gaussian, $G_i(t)$. When the individual Gaussians are close enough to each other, $F(t)$ can be approximated by a single Gaussian.

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

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

The experimental spectrum thus takes the form

$$D_{exp}(t) = \int_{-\infty}^{+\infty} D(t - t')F(t')dt'. \tag{1.5}$$

In addition, experimental spectra also contain a level of background noise and contributions from positron annihilation within the source.

1.2 The Standard Trapping Model

The goal of PALS is to extract meaningful information relating to defects in a sample material. The Standard Trapping Model (STM) links positron lifetimes to the number of defect types and their concentration. In the STM, the spectrum is modeled using a series of differential equations, based on annihilation and trapping rates, called rate equations. Two key assumptions of the model are that

(i) Positron trapping during thermalization can be safely ignored,

(ii) Defects within the sample are homogeneously distributed.

1.2.1 Single Defect Trapping

The simplest trapping model assumes a single, open volume defect type, such as an atomic vacancy. After thermalization, positrons either annihilate in the defect free bulk at a rate $\lambda_b = 1/\tau_b$ or are trapped in an open-volume defect, with a trapping rate $\kappa_d \propto$ the defect concentration C. As the electron density within the defect is lower than that of the bulk, the defect annihilation rate $\lambda_d = 1/\tau_d$ must be smaller than λ_b.

This information is used to write the rate equations:

$$\frac{\mathrm{d}n_b(t)}{\mathrm{d}t} = -(\lambda_b + \kappa_d)n_b(t), \tag{1.6}$$

$$\frac{\mathrm{d}n_d(t)}{\mathrm{d}t} = -\kappa_d n_d(t) + \kappa_d n_b(t), \tag{1.7}$$

which describes 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 1.6 and 1.7 gives us the decay spectrum of the positrons

$$D(t) = \frac{\lambda_b - \lambda_d}{\lambda_b - \lambda_d + \kappa_d} e^{-(\lambda_b + \kappa_d)t} + \frac{\kappa_d}{\lambda_b - \lambda_d + \kappa_d} e^{-\lambda_d t}. \tag{1.8}$$

Taking eq. 1.2, and setting $k = 1$ (as we're dealing with a single defect type) we can make the appropriate subtitutions and get:

$$D(t) = I_1 \exp\left(-\frac{t}{\tau_1}\right) + I_2 \exp\left(-\frac{t}{\tau_2}\right), \tag{1.9}$$

where

$$\begin{aligned} \tau_1 &= \frac{1}{\lambda_b + \kappa_d} \quad , \quad \tau_2 = \frac{1}{\lambda_d}, \\ I_1 &= 1 - I_2 \quad , \quad I_2 = \frac{\kappa_d}{\lambda_b - \lambda_d + \kappa_d}. \end{aligned} \tag{1.10}$$

By taking the absolute value of the time derivative of the decay spectrum, in similar fashion to eq. 1.3, we get the positron lifetime spectrum

$$N(t) = \left|\frac{\mathrm{d}D}{\mathrm{d}t}\right| = \frac{I_1}{\tau_1} \exp\left(-\frac{t}{\tau_1}\right) + \frac{I_2}{\tau_2} \exp\left(-\frac{t}{\tau_2}\right). \tag{1.11}$$

1.2.2 Observations and Additions

Looking closely at eq. 1.11, we can see that there are two lifetime components in the spectrum. From eq. 1.10 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 and as we increase the defect concentration (and thus increase the trapping rate κ_d) τ_1 and its associated intensity $I_1 \to 0$.

This limiting case, known as saturation trapping, happens when the defect concentration is high enough that the average spacing between defects is much smaller than the positron diffusion length. As a result, virtually all positrons are trapped, and the reduced bulk component disappears entirely from the spectrum.

An additional variable that can be useful to know is the average lifetime τ_{av}. This is calculated by taking the average of the lifetime components, weighted by intensity, or by finding the centre

of mass of the spectrum.

1.2.3 Multiple defects

The single defect model can be easily extended in order to model more complex spectra, where more than one defect type is present. This is simply done by assuming no interaction between different defects and writing additional rate equations for each additional defect type. Thus, instead of having two rate equations, we have $k+1$ equations, with $k =$ the number of defects, or

$$\frac{\mathrm{d}n_b}{\mathrm{d}t} = -\left(\lambda_b + \sum_{i=1}^{k} \kappa_i\right) n_b(t) \tag{1.12}$$

for the bulk and

$$\frac{\mathrm{d}n_{di}}{\mathrm{d}t} = \kappa_i n_b(t) - \kappa_i n_i(t) \tag{1.13}$$

for each defect type, with $i = 1, \ldots, k$.

1.3 Spectrum Decomposition and Project Aims

Decomposition of the lifetime spectrum is an example of an "inverse problem", where observations (the spectrum) are used to determine the causal factors that produced them (the components that make up the spectrum). The complexity of these problems means there are limits on if and how they can be solved.

Lifetime decomposition is commonly done by using some sort of least-squares method to fit a model function to an experimental spectrum. One disadvantage of this approach is that the number of lifetime components in the model function must be estimated before performing the fit. Working around this limitation often means, then, starting with a single component fit and increasing the number of components until the variance of the fit stops decreasing.

The aim of this project is to simulate and fit PALS spectra, comparing the data used to generate the spectra to the data returned from a using a least-squares fitting process. The hope is to use the knowledge gained to establish practical guidelines that may aid in analyzing experimental spectra.

Chapter 2

Method

This project consists of a series of trials aimed at evaluating different aspects of the fitting process. Each trial involves first simulating and fitting multiple spectra, then analysing the results of the fits. The former was accomplished using the PALSfit software package and the associated PALSsim program, while for the latter custom Python scripts were written to group the fits together and create relevant plots, using the Matplotlib library.

Each trial can be further grouped into one of three categories. The first investigates how changing the lifetimes and intensities of a two lifetime spectrum affects the fit. The second looks at experimental parameters, in particular what can be changed about PALS experiments to improve fitting performance. The third focuses on modeling spectra that closer reflect real experimental spectra.

In this chapter is a description and overview of the tools used to generate, fit and analyse the spectra that make up each trial.

2.1 The PALSfit Bundle

PALSfit is a Windows based software package, developed for the analysis of positron lifetime spectra, based on a non-linear least squares approach. Included are two modules, POSITRONFIT and RESOLUTIONFIT, which are used to deconvolve the lifetime components and resolution functions (respectively) from the lifetime spectra. As the scope of the project limited to lifetime component analysis, only the former will be used. PALSfit Version 3, or more simply PALSfit3, is the latest version of this package, which this project is aimed at evaluating. All subsequent references to "PALSfit" are to this specific version of the software.

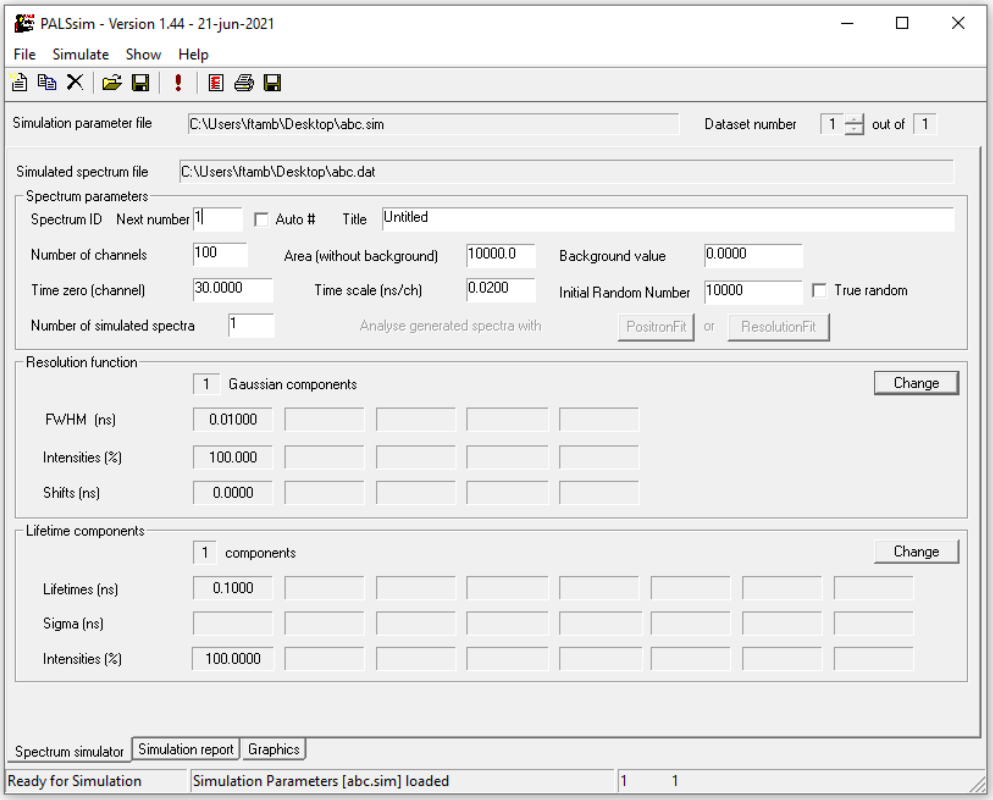


Figure 2.1.1: PALSsim window

Included with PALSfit is the PALSsim program, which generates lifetime spectra based on user input. PALSsim makes it possible to quickly generate spectra with full control over relevant parameters such as lifetimes, resolution function, background noise, etc. All spectra to be analysed in the project were produced using PALSsim.

2.1.1 PALSsim

As can be seen in Fig.2.1.1, PALSsim groups user input data into three sections: Spectrum parameters, Resolution function and Lifetime components.

In the *Lifetime components* section, the number of components in the spectrum and their associated values can be modified. Only the *Lifetimes* and *Intensities* fields will be used for each component throughout this project, as we'll assume each lifetime component to be discrete. Up to eight components can be simulated, but the maximum number of components used in the project will be three.

The *Resolution function* section deals with the instrument resolution function of the simulated spectrum. The resolution function is given by a sum of up to five Gaussian functions $G(t)$ with

the width, relative height and peak position of each Gaussian component being controlled by the *FWHM*, *Intensities* and *Shifts fields* respectively.

For the majority of this project, a three gaussian resolution function will be used, with the appropriate parameters taken from an experimental spectrum and indicated in Table 2.1.1. For a visual representation of the IRF, see Fig. 2.1.2.

The variables found within the *Simulation parameters* section affect the spectrum as a whole. Here we find parameters controlling the number of channels, time per channel, number of counts, t_0 and background noise. Relevant parameters and default values used are as follows:

- Number of channels: 1000
- Area (without background): 5650000
- Background value: 8.5
- Time zero (channel): 1400
- Time scale (ps/ch): 5

All other parameters were left unchanged.

PALSsim generates five files every time a new spectrum is simulated. The file with the .dat extension contains the simulated spectrum, organized as a table of values for each channel. A .sim file stores the simulated values, allowing them to be reloaded into the program. A simulation report is generated every time a simulation is ran that contains data about the spectrum and saved in a .out file. Lastly two control files with the .pfc and .rfc extensions are generated to be read by PALSfit. Loading one or the other control file into PALSfit determines whether the POSITRONFIT or RESOLUTIONFIT module is used to analyse the spectrum (.pfc for the former and .rfc for the latter).

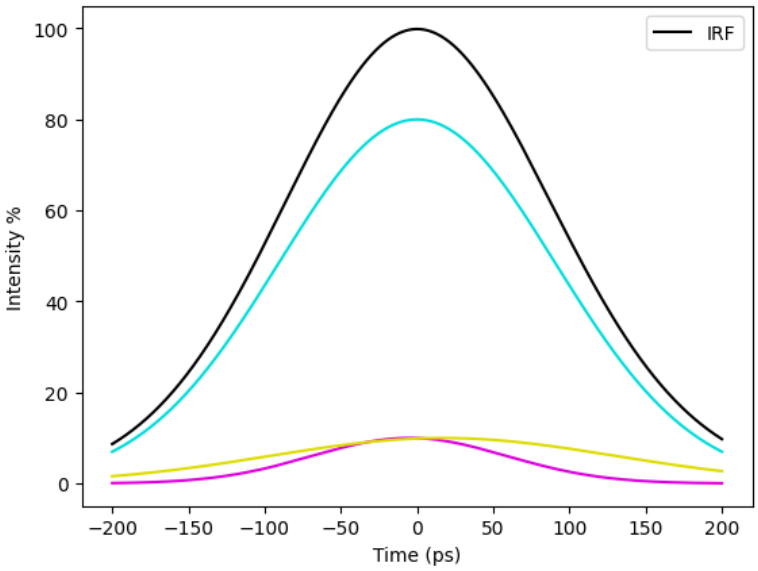


Table 2.1.1: IRF values

FWHM (ps)	Intensity (%)	Shift (ps)
213.2	80	0
150.2	10	-5
265.7	10	17

Figure 2.1.2: Visual representation of the IRF, generated using Python and Matplotlib

2.1.2 PALSfit

Opening a .pfc file with PALSfit should automatically load the spectrum and all relevant fitting values. The *Spectrum setup* window can be opened by clicking any of the two *Change* buttons in the *Spectrum* tab (selected by default when the program is opened). Here the *Default Ranges* button can be clicked to ensure that the appropriate range is fitted and the spectrum can be saved to the control file by checking the relevant checkbox.

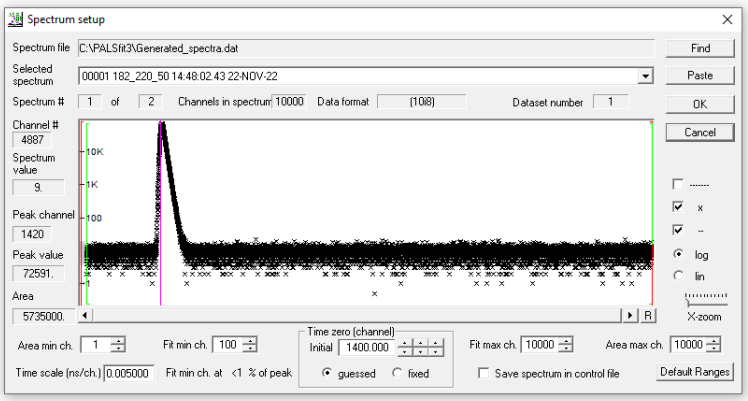


Figure 2.1.3: Spectrum setup window

Seven other tabs are present at the bottom of the program. The *Resolution function* tab shows a graphical representation of the resolution function used during fitting, and allows the user to modify the Gaussian components that make up the IRF. In the *Lifetime and Corrections* tab, the number of lifetime components and the initial values used in fitting them can be modified.

Meanwhile, the *Text output* tab displays the result of the last fit. All other tabs can be safely ignored, at least in the context of this project.

A fit is performed by pressing the *Analyse* button at the top of the window or pressing the F5 key. This generates a series of files in an output folder, contained in the same directory as the .pfc file. Of relevance are the .out file and .csv file that contain the results of the fit.

2.2 Python Scripts

Custom Python scripts were developed for the following purposes:

2.2.1 Grouping fit results

As each trial involves generating and fitting multiple spectra, the results of multiple fits had be collected into a single document for analysis. To this end, files were grouped into batches and subfolders containing spectra to be analysed together. Once generated and fitted, the fit results for each subfolder were contained in a single output folder. Two Python scripts were initially used to collect the data into a single file with all relevant information for plotting, that later were consolidated into one. The first script joined the multiple .csv files produced by PALSfit and the second added information on the original, simulated values of the lifetime components to the file.

2.2.2 Plotting fit data

All plots were produced using Python library Matplotlib. These can be grouped into a few different types, with an overview provided below as a reference to aid in interpretation.

An example of the first type, used for two-lifetime spectra, can be seen in Figure 2.2.1. On the x-axis is always the simulated value of τ_2. On the y-axis is either τ_1, with the (fixed) simulated value of τ_1 indicated with a grey dashed line, or the difference between the fitted and simulated values of τ_2, with the grey line at $y = 0$. Each datapoint represents a different spectrum. For each simulated τ_2, there are three datapoints, indicating the relative intensities of the two lifetime components, marked in the legend. Error bars are given by the standard deviation calculated by PALSfit.

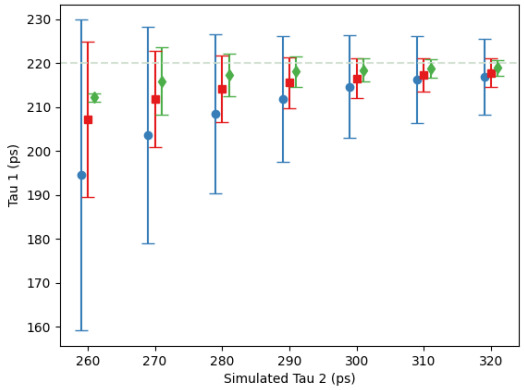


Figure 2.2.1: $\tau_1 = 220$ ps (fixed), $\tau_2 =$ 260-320 ps

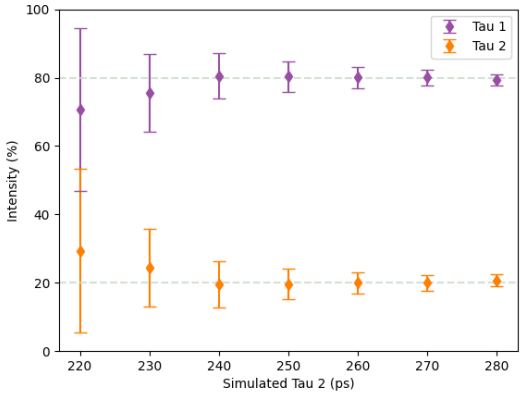


Figure 2.2.2: $I_1 : I_2 = 80\% : 20\%$, $\tau_2 =$ 260-320 ps

Closely related is the second type, shown in Figure 2.2.2 and also used for two-lifetime fits. The simulated τ_2 is still on the x-axis, but lifetime intensity is on the y-axis, instead. The grey lines trace the simulated values for I_1 and I_2 and each spectrum is represented by two vertically stacked points, representing the fitted values of I_1 and I_2, as indicated in the legend.

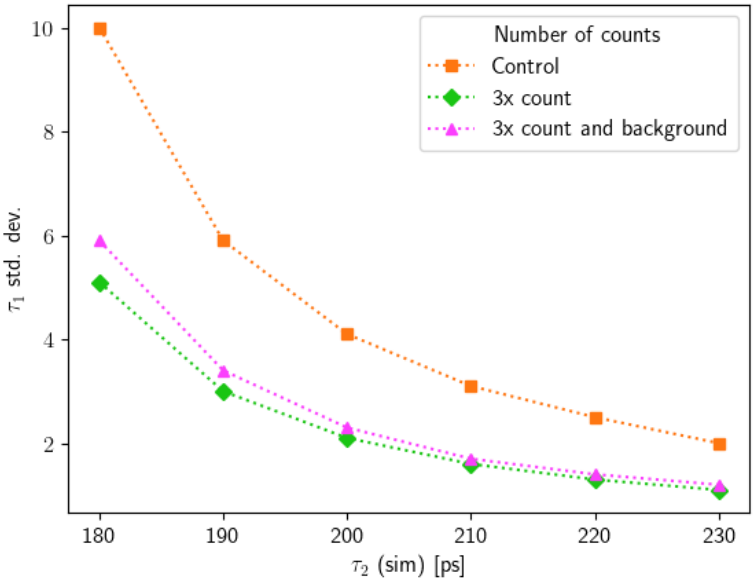


Figure 2.2.3: Std. dev. τ_1, $I_1 : I_2 = 50\% : 50\%$

In the third type a single fit result is being tracked on the y-axis, across the same intensity ratio, and with the x-axis representing (again) simulated τ_2. Figure 2.2.3 is an example where the variable being tracked is the standard deviation of τ_1, the intensity ratio is 50%-50%, and each colored line represents a different number of counts in the spectrum, indicated in the legend. Like the first type, each datapoint corresponds to an individual spectrum. Each spectrum is made up of two lifetime components, as in the previous types.

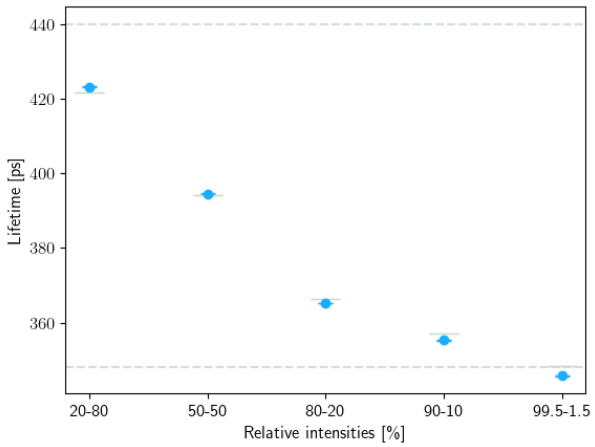


Figure 2.2.4: One lifetime fit, $\tau_1 = 348$ps, $\tau_2 = 440$ps

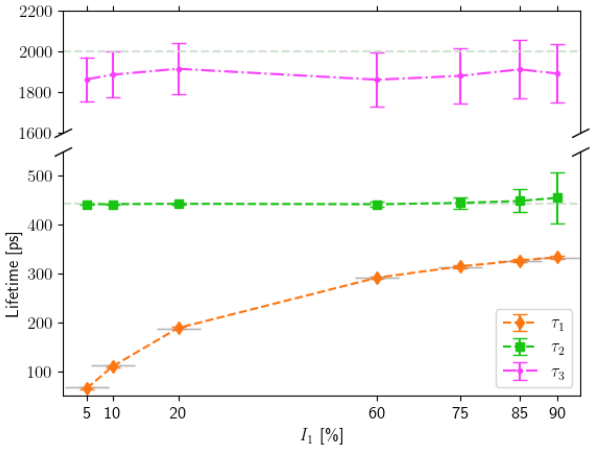


Figure 2.2.5: Three lifetime fit, Standard trapping model, $\tau_b = 339$ps, $\tau_d = 442$ps

For tracking multiple lifetimes across a range of intensities, the fourth type will be used. This has intensity on the x-axis, and on the y-axis is lifetime. Grey dashed lines indicate important fixed values. Short grey lines are used to indicate a value being compared against. In Figure 2.2.4, for example, the dashed line indicate two lifetimes and the short grey lines represent the intensity-weighted average lifetime. In Figure 2.2.4, the short grey lines indicate the reduced bulk lifetime.

2.2.3 Generating a single Gaussian resolution function

As one of the trials requires varying the width of the resolution function, a script was developed to approximate the three Gaussian resolution function used throughout the paper (see Table 2.1.1) with a single Gaussian resolution function. This same script was also used to produce Figure 2.1.2. Outlined is the procedure used. 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:

$$g(x) = a \exp\left[-\frac{-(x-b)^2}{2c^2}\right] \tag{2.1}$$

While a and b map easily to the intensity and shift columns in Table 2.1.1, 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, the following expression was used:

$$\text{FWHM} = 2\sqrt{2\ln(2)} \approx 2.355c \tag{2.2}$$

giving the resulting resolution function

$$g_s(x) = \sum_{i=1}^{3} a_i \exp\left(-\frac{(x-b)^2}{2c^2}\right) \tag{2.3}$$

As this is a single Gaussian resolution function, intensity is simply 100%. FWHM and shift can be determined analytically, by solving eq. eq:composite-res, but as a plot of $g_s(x)$ would be generated for use in this document, the choice was made to just extract these variables numerically using Python. To find the shift, the largest y-value was picked and its corresponding x-value was found. To find the FWHM the difference was calculated between the two values of x for which $g_s(x)$ is closest to half of the IRF. The resulting resolution function has a FWHM of ∼210 ps and a shift of ∼0 ps.

2.2.4 Batch simulation and fitting

A command-line version of PALSfit called PATFIT is available for use, found on the PALSfit website, as an auxiliary program. By using a Python library for interfacing with command-line programs, a script was developed that uses PATFIT to fit multiple spectra. This script then takes the .csv files produced by these fits and groups them into a single file. A separate script was also developed that allows for multiple .sim files to be generated, which can then be simulated using PALSsim. Using these two scripts still requires the user to simulate each .sim file individually, then link the spectrum file and adjust the fitting ranges on PALSfit. This can be tedious, however, and a single script to automate the entire process could be both useful and possible to code, though deemed beyond the scope of this project.

Chapter 3

Trials and Results

3.1 Lifetime separation, fixed τ_1

The aim of the first trial was to investigate how the separation between lifetime components affected the fit. PALS spectra containing two lifetime components were generated, corresponding to a two defect material in saturation trapping. The first lifetime, τ_1, remained fixed at 180 ps while the second lifetime, τ_2, decreased from 280 ps to 220 ps, in 10 ps intervals.

For each pair of lifetimes, intensity was varied to simulate three diferent relative defect concentrations. The values chosen for the first lifetime intensity I_1 were 20%, 50% and 80%, with $I_2 = 100\% - I_1$.

Expectations were that:

(i) The overall fit would improve as lifetime separation, $\tau_1 - \tau_2$, increases.

(ii) The fit would improve for a given lifetime component as the intensity of that component increases.

In Figure 3.1.1 is the fit for the first lifetime component τ_1. As would be expected, the accuracy of the fit increases as both the separation between lifetimes and the intensity of I_1 increases. The size of the error bars indicates the precision of the fit, and seems to follow the same expected trend.

A similar picture emerges when looking at τ_2, as can be seen in Figure 3.1.2. Here, as the simulated value of τ_2 that is being compared to the fitted τ_2 is constantly changing, y-axis plots

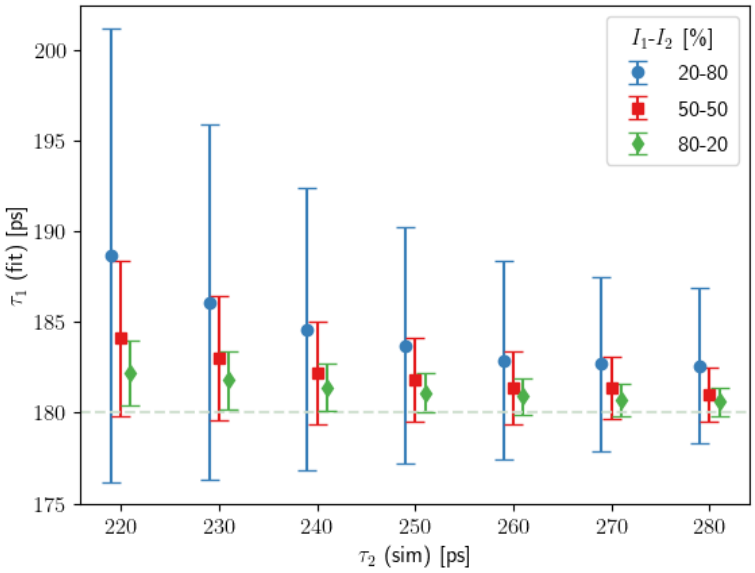


Figure 3.1.1: Fitted τ_1

the difference between the two values. Nevertheless, we see that both accuracy and precision increase with lifetime separation and I_2 this time, which makes sense as the second lifetime is the observed variable.

In both cases, the fitted values of the component tend to be higher than their simulated counterparts. Outliers emerge when looking at the 80-20 data in the 220-240ps range, for both τ_1 and τ_2. This is harder to spot in the former, though, unless zoomed in (see Figure 3.1.3).

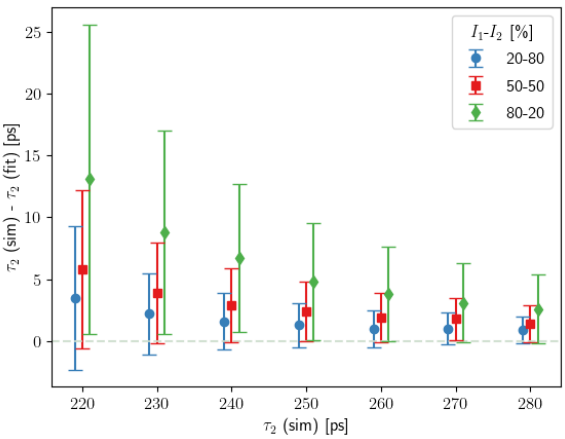


Figure 3.1.2: τ_2 (Fitted−Simulated)

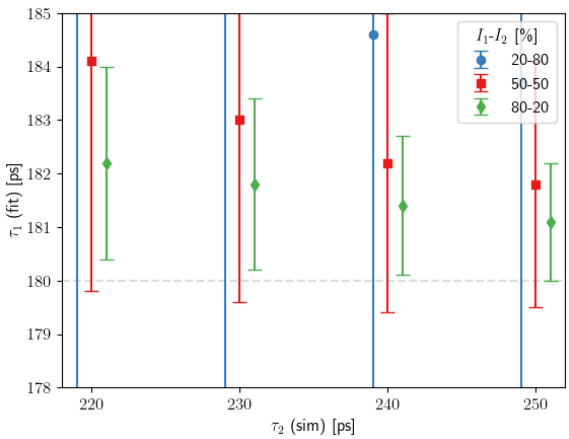


Figure 3.1.3: Fitted τ_1, $\tau_2 = [220 - 250]$

Plotting intensities (Figures 3.1.4, 3.1.5 and 3.1.6), the same expected trends are observed. The error bars seem to get smaller, though, as the intensity of the first component I_1 increases. As a consequence, the error bars of the first three datapoints in Figure 3.1.6 end up falling short

of their simulated values.

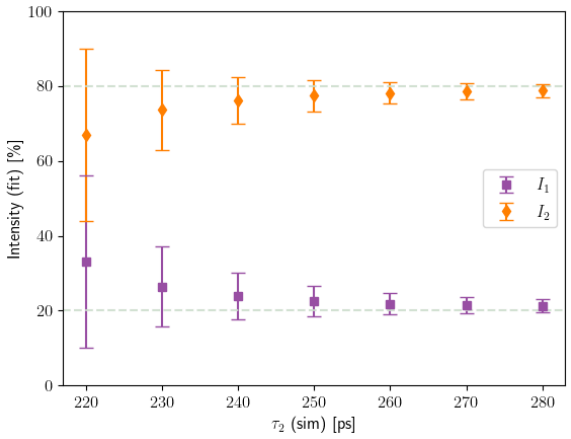


Figure 3.1.4: $I_1 = 20\%, I_2 = 80\%$

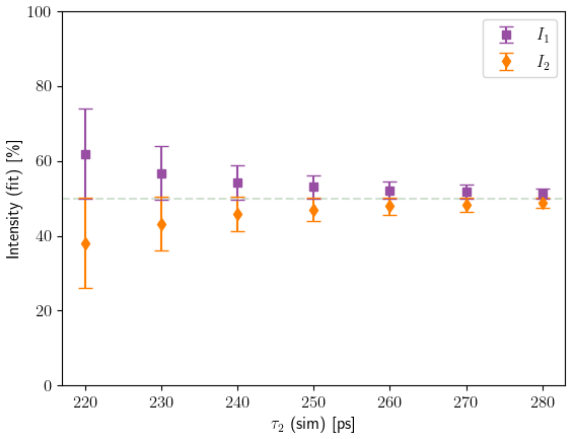


Figure 3.1.5: $I_1 = 50\%, I_2 = 50\%$

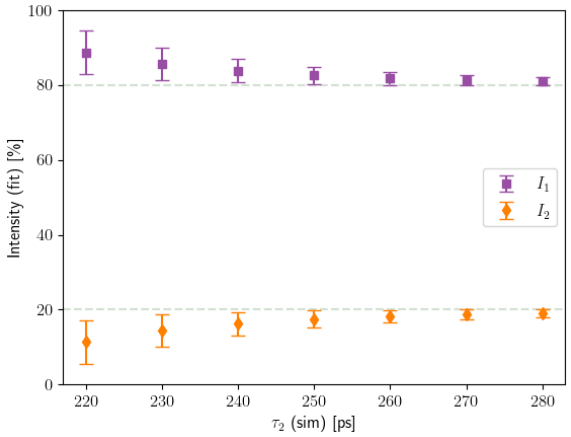


Figure 3.1.6: $I_1 = 80\%, I_2 = 20\%$

3.2 Lifetime separation, varying τ_1

This trial builds on the previous, investigating the effect of varying τ_1. Adding to the fits generated in the previous trial, two more sets of spectra were generated.

In the first of these, every lifetime was shortened by 30ps. This meant that the first lifetime τ_1 was set to 150ps and τ_2 ranged from 190-250ps, while keeping the relative spacing consistent. The spectra were fitted, and the same plots generated in the previous trial were also generated for this set of spectra. The full set of plots are available in Appendix A.1.

The second set of spectra lengthened every lifetime by 40ps, so τ_1 became 220ps and τ_2 spanned 260-340ps. The same procedure was followed and the full set of plots are available in A.2. The I_1-I_2=20%-80%, $\tau_2 = 260ps$ spectrum is of interest, as the fit gives a highly inaccurate result with a very low standard deviation. This can be seen particularly well in Figure 3.2.1, and is notable in that, so far, a decrease in accuracy has always corresponded to a decrease in precision.

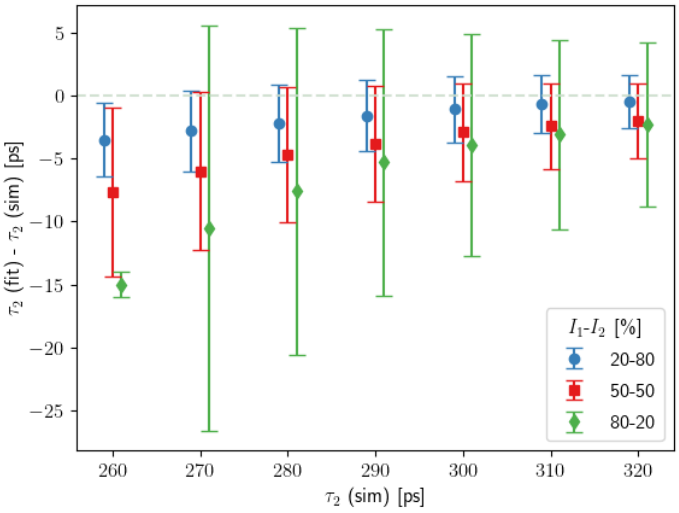


Figure 3.2.1: τ_2 (Fitted−Simulated), $\tau_1 = 220$ps

Figure 3.2.1 also shows an overall trend that is present in the lifetime fits for both the $\tau_1 = 220$ps and 150ps data. In both datasets, PALSfit tends to underestimate the lifetime fits, converging to the simulated values from the bottom. This is in contrast to the 180ps data (see Figures 3.1.1 and 3.1.2), where the fits converged from the top. Other than that, trends in lifetime separation and component intensity are observed to be similar to the first trial.

Plots were generated to compare the three datasets, tracking both accuracy and precision. These can be found in Appendix A.3. The odd bump in precision for the 80-20 spectrum

mentioned earlier is clearly visible in the 80%-20% standard deviation plots (Figures A.3.6, A.3.12 and A.3.18).

Overall though, while there is variation between the plots, the general trend seems to be that a decrease in τ_1 corresponds to an improvement of the fit, as can be seen in Figure 3.2.2. In addition, the standard deviations for the 180ps and 150ps data tend to be closer to each other than to the 220ps data, a trend that doesn't seem as present in the fitted variables.

Comparing the two lifetime components, the fits for τ_2 tend to be better overall than the τ_1 fits. Looking at intensities, higher values for I_1 yields more accurate results, while standard deviation seems to be highest for the 50%-50% data.

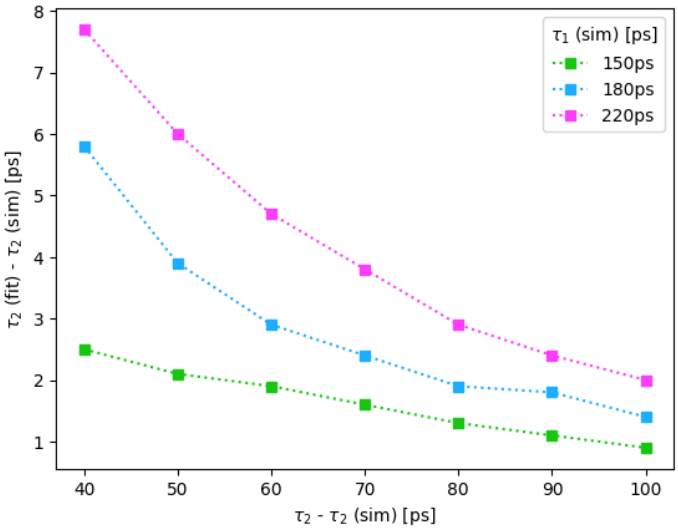


Figure 3.2.2: τ_2 (Fitted−Simulated)

3.3 Resolution function width

The aim of this trial was to investigate how the width of the resolution function affects the fit. While a narrower resolution function should result in a better fit, the idea was to quantify this improvement, as developing instrumentation that produce narrower resolution functions would require serious investment in time and resources.

As controlling the width of the resolution function is easier for a single Gaussian IRF, using the procedure outlined in 2.2.3, a 1-Gaussian approximation to the resolution function used in the previous trial was found. This resolution function was used to generate a series of spectra with fixed $\tau_1 = 150$ps and $\tau_2 = 180$-230ps. The results of the fit can be found in Appendix A.4.

This was compared against the $\tau_1 = 150$ps fit from the last trial. The difference between the results of the two fits was deemed negligible.

Additional spectra were then generated and fitted using progressively narrower resolution functions, with FWHM = 180, 150 and 100. A comparison of the results is available in Appendix A.5.

At a glance, an interesting pattern seems to emerge, based on the relative intensity of the components. When the intensity ratio I_1-I_2 = 20%-80% or 50%-50%, the narrower resolution functions produce the most accurate results (as in Figure 3.3.1). However, when the I_1-I_2 = 80%-20%, this trend seems to reverse (see Figure 3.3.2).

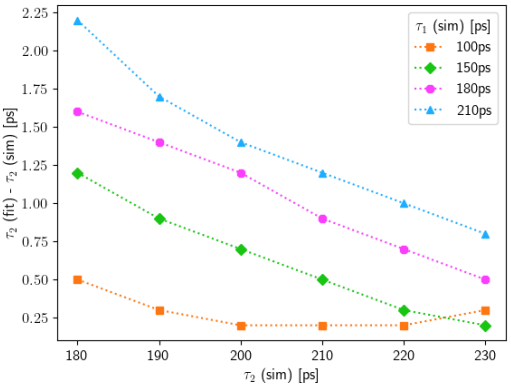


Figure 3.3.1: I_1-I_2=20%-80%

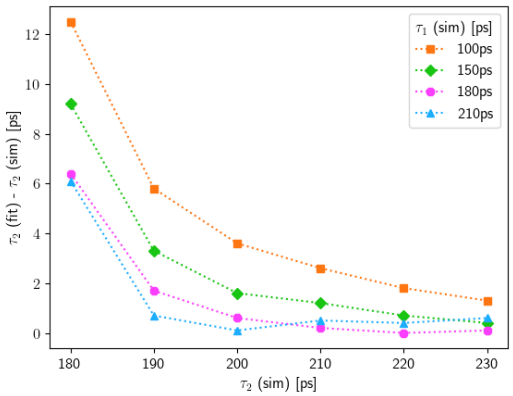


Figure 3.3.2: I_1-I_2=80%-20%

This reversal also occurs in some of the standard deviation plots, but the underlying pattern is less clear. The standard deviations do tend to quickly cluster together, though, as lifetime separation increases, indicating that the effect of the resolution function on the standard deviation is small.

Paying attention to the numbers, the overall effect of narrowing the resolution function is mixed, with the biggest improvements seen when fitting τ_1 with I_1-I_2 = 20%-80% (Appendix A.5.1), and when fitting the intensities (Appendix A.5.13 and A.5.15).

3.4 Number of counts

The goal of this trial is to examine how the total count number affects the fit. 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 trials, with the area set to 5.65×10^6 and the background value set to 8.5. Expectations were that a higher number of counts would make spectrum analysis easier. To test this hypothesis, spectra were generated

with $\tau_1 = 150$ps, $\tau_2 = 180$-230ps and a 210ps FWHM single Gaussian resolution function, while tripling the background area to 1.7035×10^7. This allows for direct comparison with the data used in the previous trial, as a control.

Increasing the total area of the spectra and maintaining the same background value, however, increases the signal-to-noise ratio. As the easiest way to increase count number in practice is to increase experimental run-time, a more realistic scenario involves an equivalent increase in background noise. To model these, the same spectra were generated with both the tripled area and a proportional 3x increase in the background value (set to 25.5).

The generated spectra were then fitted with PALSFIT, and the results are summarized in Appendix A.6. Overall, increasing the total number of counts seemed to improve both the accuracy and the precision of the fits. As expected, this improvement was diminished when the background noise level was raised to match the increased count. In most cases though, the fit with the tripled count and the scaled background noise was much closer to the tripled count without background scaling than it was to the control, especially as lifetime separation increased. An example of this can be seen in Figure 3.4.1.

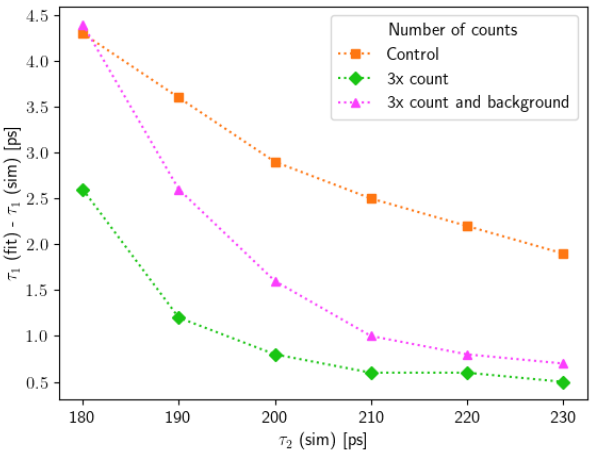


Figure 3.4.1: τ_2, I_1-I_2 = 50%-50%

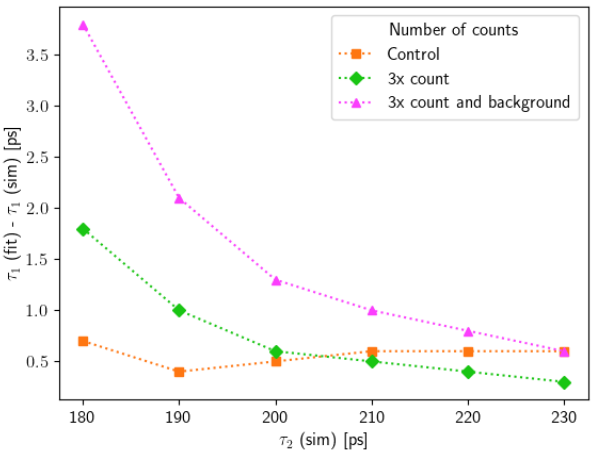


Figure 3.4.2: τ_2, I_1-I_2 = 80%-20%

Like in the last trial, though, the I_1-I_2 = 80%-20% fits show a reversal of the trend, as can be seen in Figure 3.4.2. Unlike the previous trial, the reversal is not present in the standard deviations.

3.5 Three lifetime components

The goal of this trial was to model three-lifetime spectra, based on three experimental samples. The first two lifetimes would be defect lifetimes and the third was much longer and weaker than

the former, characteristic of Para-Ps. During simulation, the third lifetime was kept constant at 2.6ns and an intensity of 0.15%. The sample lifetimes are given in Table 3.5.1.

Table 3.5.1: Lifetimes

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

Table 3.5.2: Intensities

relative		absolute	
I_1	I_2	I_1	I_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%

The relative intensities between the first two lifetime components was varied, simulating different defect concentrations, as shown in Table 3.5.2. On the left side of the table are the relative intensities chosen. Due to the presence of the third component, the values for I_1 and I_2 had to be adjusted to those in the "absolute" column. To closer emulate the experimental process, each spectrum was fitted with one, two and then three components. The result of the fits can be found in Appendix A.7.3.

The one-lifetime fits are found to roughly follow the average lifetime τ_{av}. An example can be seen in Figure 3.5.1. Of note is that PALSfit returns very small error bars.

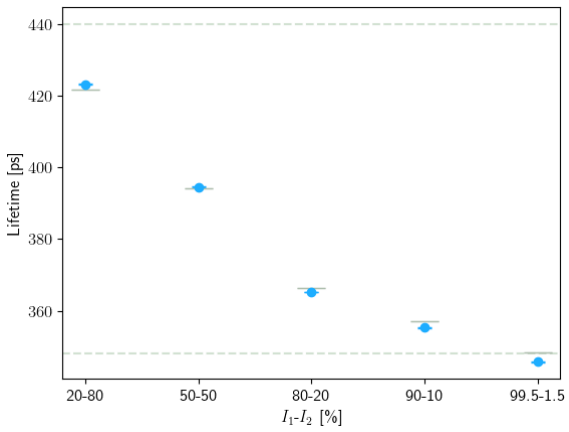


Figure 3.5.1: One lifetime fit, $\tau_1 = 348$ps, $\tau_2 = 348$ps

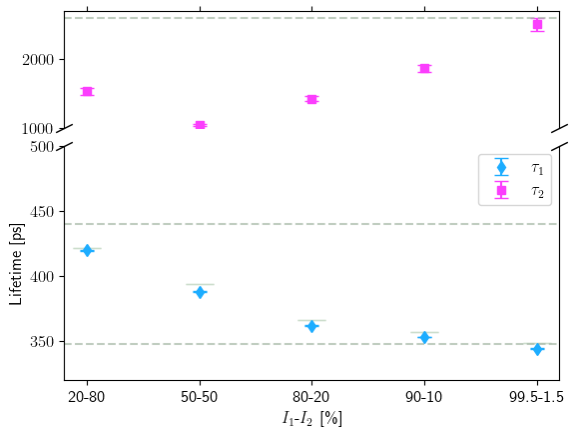


Figure 3.5.2: Two lifetime fit, $\tau_1 = 348$ps, $\tau_2 = 348$ps

In the two-lifetime fits produced, the first component seems to follow τ_{av}, and try to fit the Ps lifetime. This can be seen in Figure 3.5.2. Looking at just the second component, we see that it trends closer to the simulated 2.6ns component as the difference between I_1 and I_2 grows. We can make sense of this physically as the three component spectrum trending closer to a

two-component spectrum.

The intensity for the first lifetime is near-100% and the second is near-0%, as can be seen in the appendix. The tiny error bars seen in the one-lifetime fit also make a return, in both the intensity and lifetime plots.

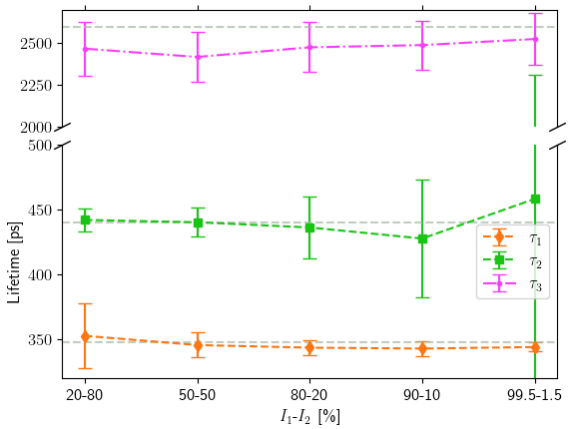


Figure 3.5.3: Three lifetime fit, $\tau_1 = 348$ps, $\tau_2 = 348$ps

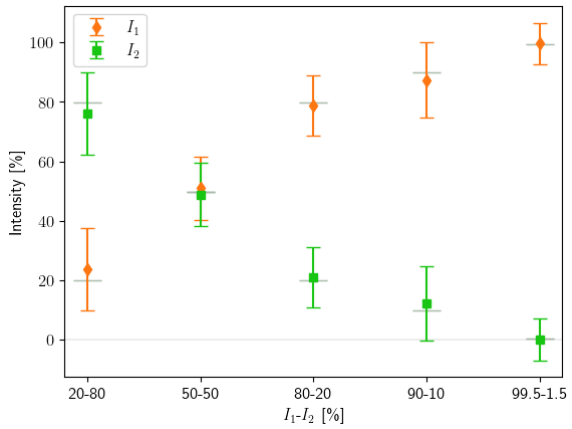


Figure 3.5.4: Intensities, three lifetime fit, $\tau_1 = 348$ps, $\tau_2 = 348$ps

Finally, when looking at the three-lifetime fits, such as the ones shown in Figures 3.5.3 and 3.5.4, the error bars come back and the three fitted components follow their simulated lifetime and intensity values. Datapoints where the fit fails are characterized by either massive or miniscule error bars, and tend to be concentrated towards the extremes, with regards to intensity.

Additionally, the program seems to have a tendency to underestimate the third lifetime. Finally, as expected, the sample with the both the smallest lifetime separation and the longest first component ($\tau_1 = 370$ps and $\tau_1 = 370$ps) has the worst showing, overall.

Chapter 4

Conclusion

A total of five trials were performed with PALS spectra that were simulated and fitted using PALSsim and PALSfit. The first two looked at how the interaction between lifetime components and intensities of two-lifetime spectra affected the fitting process. The second two gauged to what extent changing two experimental parameters, the width of the instrument timing resolution function and the number of counts, might improve fits. The third studied how the addition of a third component, corresponding to the effect of para-Ps formation, and the choice in number of components affected the fit.

4.1 Findings

A number of conclusions may be drawn from the first two trials. As expected, lifetime separation and relative intensity are both significant factors in how well two components can be resolved, and the effect of these can be seen in both the accuracy and precision of the fit. Larger lifetime separation leads to an improvement in fitted values, while higher intensity of a given component positively affects fitting results for that component's associated lifetime value, to the detriment of the values for other components. The value of the first lifetime component in relation to its separation also seems to play a role in how well the components can be resolved. Lower values of τ_1 tend to correspond to better fits, overall. In addition, when looking at two-component spectra, the lifetime values for the second component seem to be somewhat easier to fit. These findings seem to extend throughout the trials.

Of interest is how the relative intensities of the components seem to affect the trials more generally, or at least when looking at the two component trials. Trends in the 20%-80% and 50%-50% data tend to reverse when looking at the 80%-20% data. This effect extends to the

second two trials, too. Frustratingly, the source of this reversal remains a mystery, demanding further investigation as to whether this is an error in data processing, or a real effect. Another factor that seems consistent throughout the two component trials is a better fit for τ_2 overall.

Moving on to the second two trials, the findings indicate that while both a decrease in resolution function width and an increase in count number seem to improve the fit, in general, the latter seems to yield more consistent results. The improvement seems to be in both accuracy and precision, and persists even when accounting for the proportional increase in background noise. An interesting direction might be to test what happens if the signal-to-noise ratio is decreased as the count number increases, and to what degree the latter can make up for the former.

The one component fit in the last trial seems to follow the average lifetime. So too does the first component of the two lifetime fit, with the second component attempting to fit the low intensity positronium lifetime. Finally, in the three component fit, of note is that PALSfit seems to consistently underestimate the third lifetime. A general takeaway, which can be applied to any spectrum, is to pay particular attention to the standard deviations. A large standard deviation is obviously undesirable, but a very small one, especially when one defect component is strong, seems to be indicative of an inaccurate fit.

4.2 Difficulties

The biggest challenge over the course of the project seemed to be a lack of tools to quickly generate, fit and compare a large number of spectra. PALSfit admittedly does allow multiple spectra to be linked to a single control file, but sometimes doing so would result in buggy or inconsistent results. As a consequence, the choice was made early on to use separate control files for every spectrum, which made the process of generating and fitting them more laborious. While some development of the aforementioned tools was done over the course of the project, an effort towards further developing these tools to make them widely available and easy to use might be a useful undertaking.

An example of where the lack of appropriate tools might have been a limiting factor during this project is in the final trial, that didn't make it into the dissertation. The aim of this trial was to use the standard trapping model to simulate a material with an increasing defect concentration, and so a spreadsheet was used to calculate the appropriate lifetime components. Some sort of tool that could import values from this spreadsheet and use them to generate and fit the appropriate spectra would have been quite useful. The fits were generated in the end, but a lack of the time necessary to figure out and develop a way to visualize the data then led

to the decision to abandon work on this trial. Here too, access better tools could have sped the process up and made the difference.

4.3 Final assessment

The aim of this project was to fit simulated PALS spectra to gain knowledge that could aid in interpretation of experimental spectra. Ultimately, this was partially successful and some potentially useful information was gained, as illustrated in the findings. More information could potentially be gained from access to more data, with the limiting factor being tools to generate, process and visualize that data. The final conclusion is that there is still work to be done in this space.

References

1. R. Krause-Rehberg and H.S. Leipner, Positron Annihilation in Semiconductors: Defect Studies (Springer-Verlag, Berlin Etc., 1999).

2. A. Dupasquier and A.P. Mills, Positron Spectroscopy of Solids: Proceedings of the International School of Physics "Enrico Fermi" Course CXXV, Varenna on Lake Como, Villa Monastero, 6-16 July 1993 (IOS Press, Amsterdam, 1995).

3. I.K. MacKenzie, T.L. Khoo, A.B. McDonald, and B.T. McKee, Physical Review Letters 19, 946 (1967).

4. F.A. Selim, Materials Characterization 174, 110952 (2021).

5. A.R. Abraham and P.M.G. Nambissan, Design, Fabrication, and Characterization of Multifunctional Nanomaterials 123 (2022).

Appendix A

Figures

A.1 Two components, $\tau_1 = 150$ps, $\tau_2 = 190$-250ps

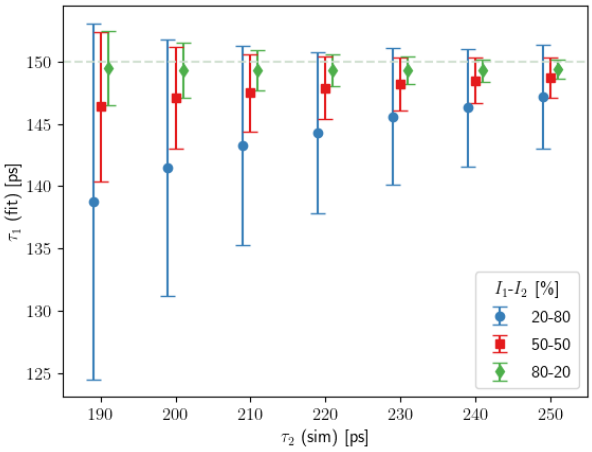


Figure A.1.1: Fitted τ_1

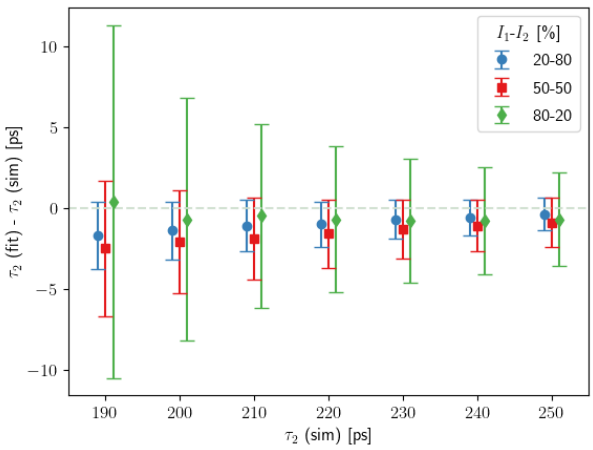


Figure A.1.2: τ_2 (Fitted−Simulated)

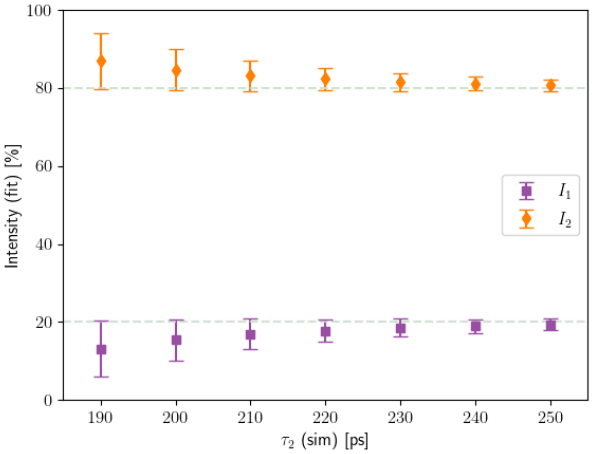


Figure A.1.3: $I_1 = 20\%, I_2 = 80\%$

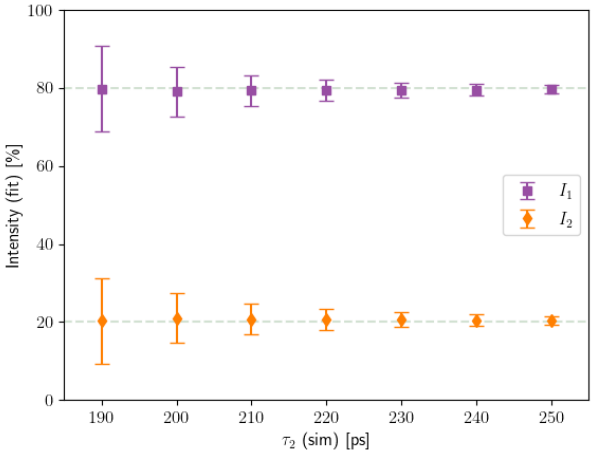


Figure A.1.4: $I_1 = 80\%, I_2 = 20\%$

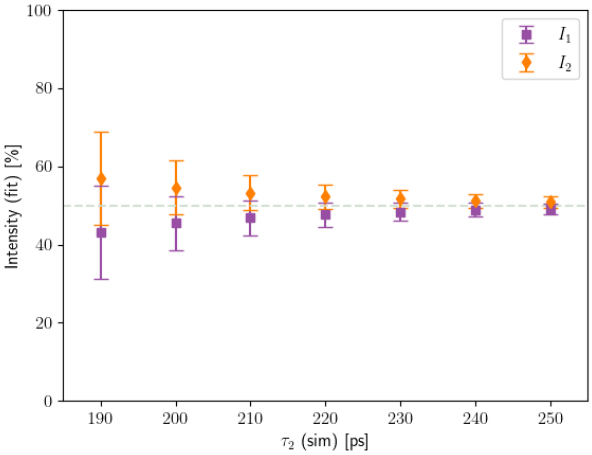


Figure A.1.5: $I_1 = 50\%, I_2 = 50\%$

A.2 Two components, $\tau_1 = 220$ps, $\tau_2 = 260$-340ps

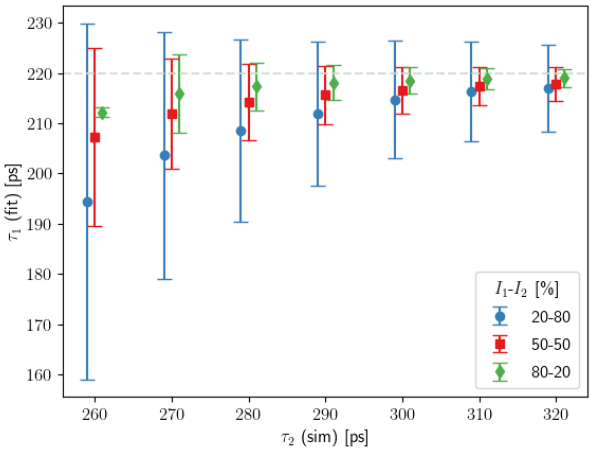


Figure A.2.1: Fitted τ_1

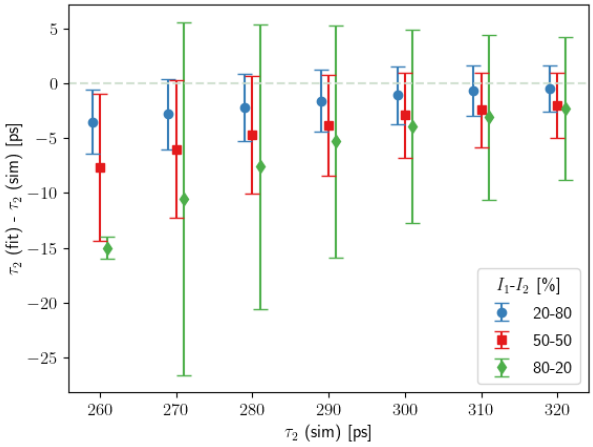


Figure A.2.2: τ_2 (Fitted−Simulated)

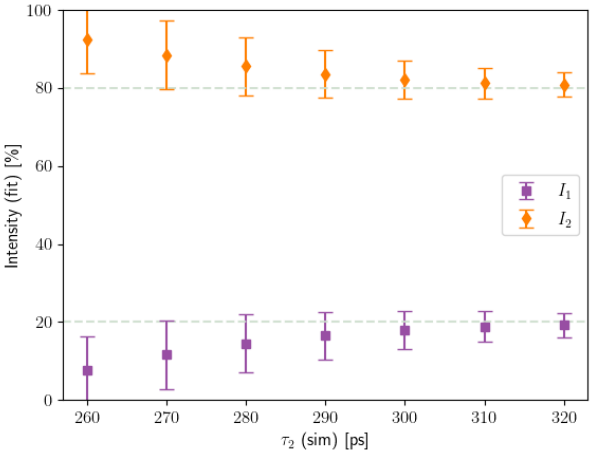


Figure A.2.3: $I_1 = 20\%, I_2 = 80\%$

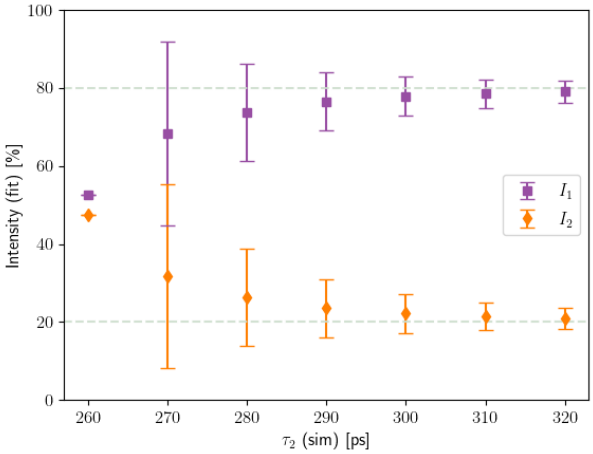


Figure A.2.4: $I_1 = 80\%, I_2 = 20\%$

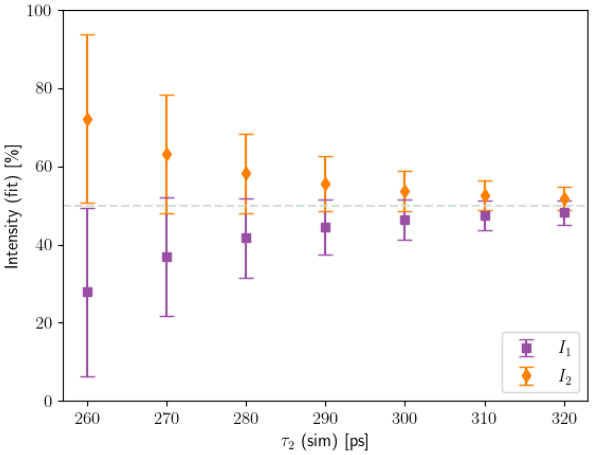


Figure A.2.5: $I_1 = 50\%, I_2 = 50\%$

A.3 Comparison two component fit, $\tau_1 = 150, 180, 220$

First component, τ_1

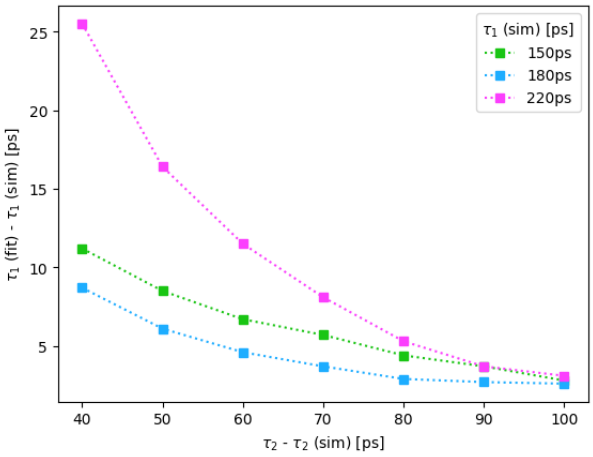


Figure A.3.1: I_1-I_2 = 20%-80%

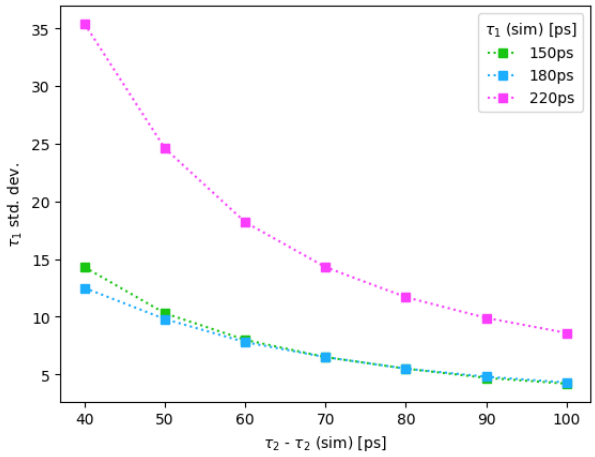


Figure A.3.2: Std. dev. I_1-I_2 = 20%-80%

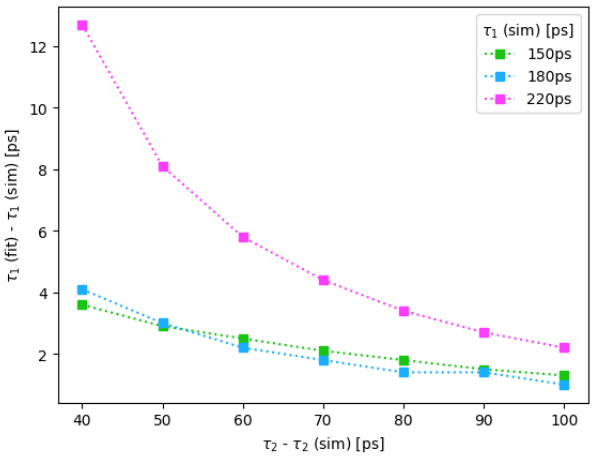


Figure A.3.3: I_1-I_2 = 50%-50%

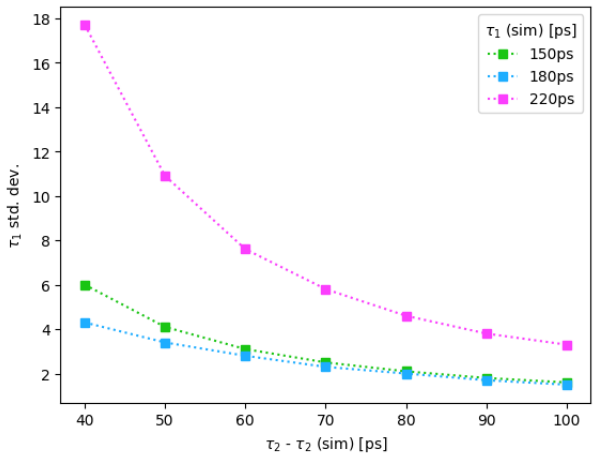


Figure A.3.4: Std. dev. I_1-I_2 = 50%-50%

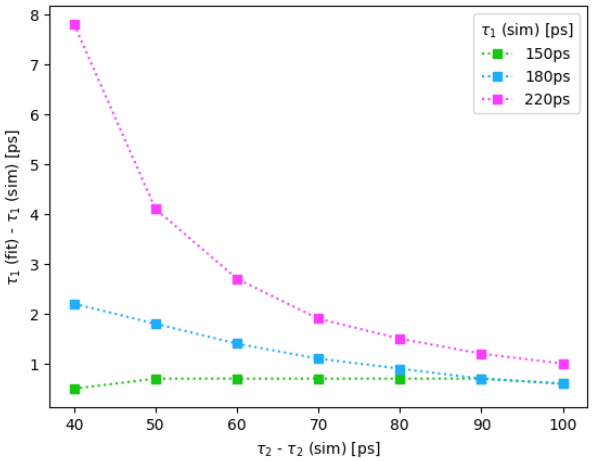


Figure A.3.5: I_1-I_2 = 80%-20%

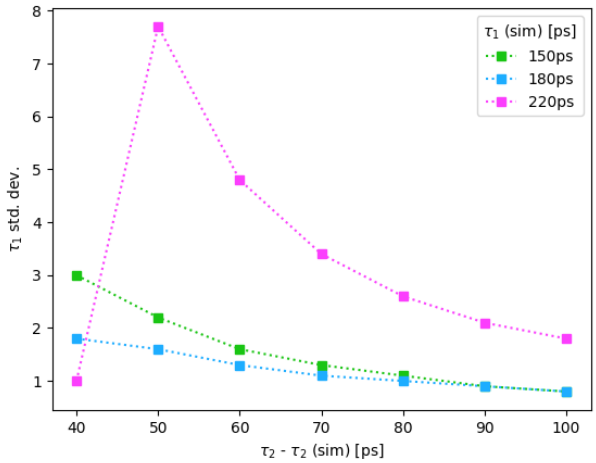


Figure A.3.6: Std. dev. I_1-I_2 = 80%-20%

Second component, τ_2

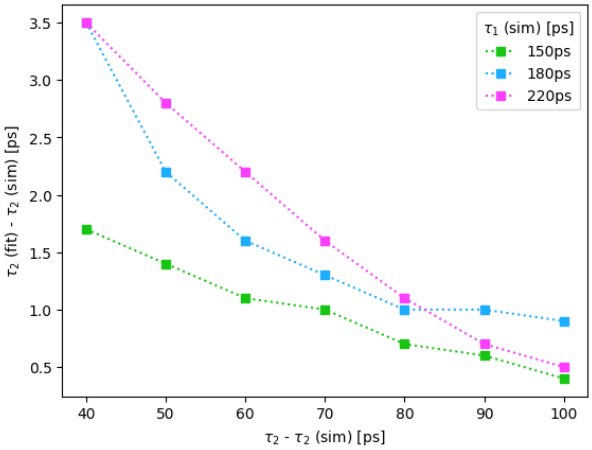


Figure A.3.7: I_1-I_2 = 20%-80%

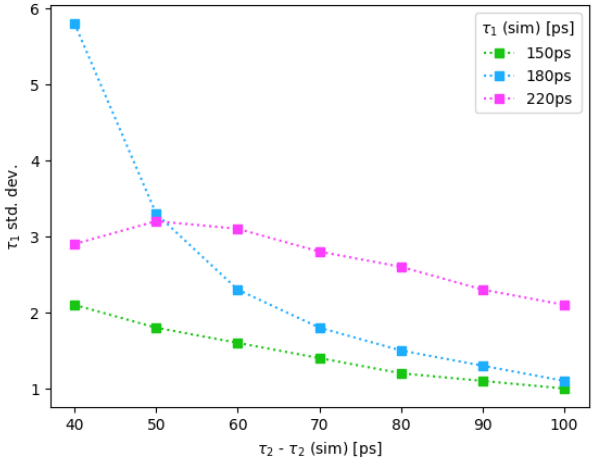


Figure A.3.8: Std. dev. I_1-I_2 = 20%-80%

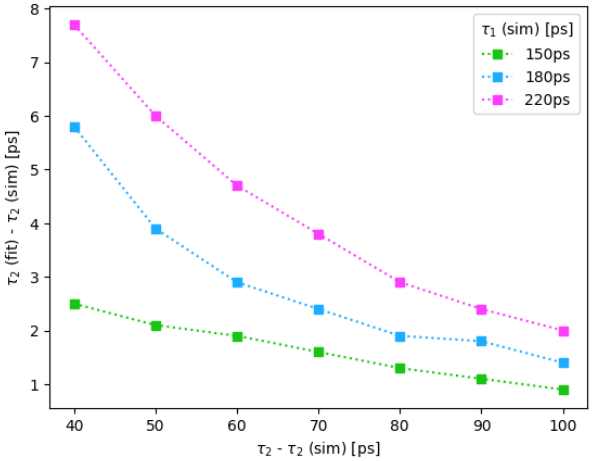


Figure A.3.9: I_1-I_2 = 50%-50%

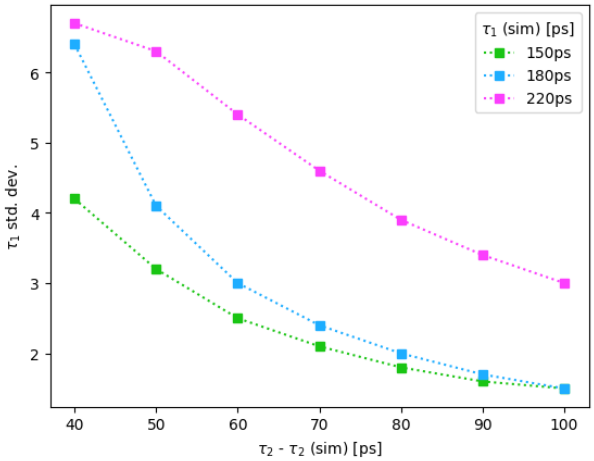


Figure A.3.10: Std. dev. I_1-I_2 = 50%-50%

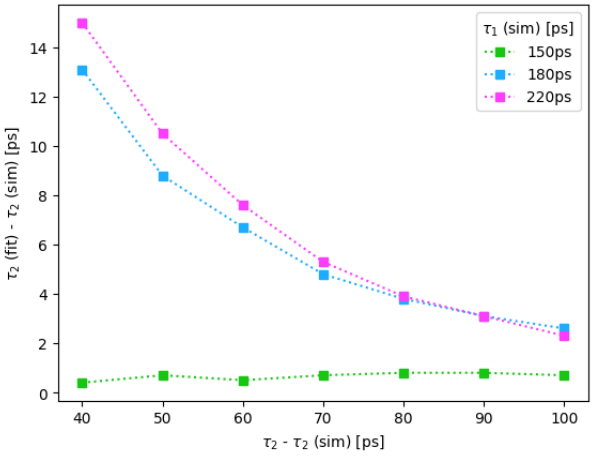


Figure A.3.11: I_1-I_2 = 80%-20%

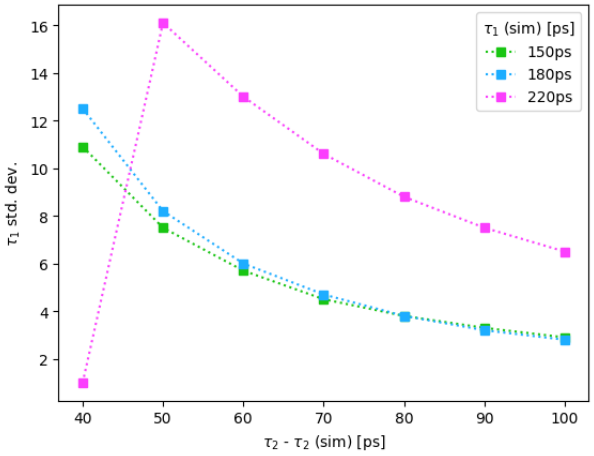


Figure A.3.12: Std. dev. I_1-I_2 = 80%-20%

Intensity

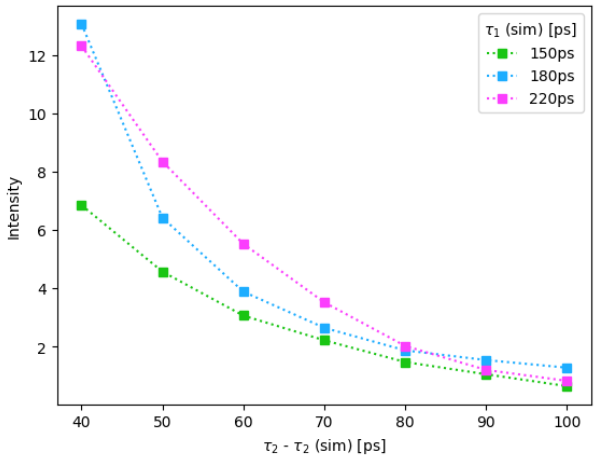


Figure A.3.13: I_1-I_2 = 20%-80%

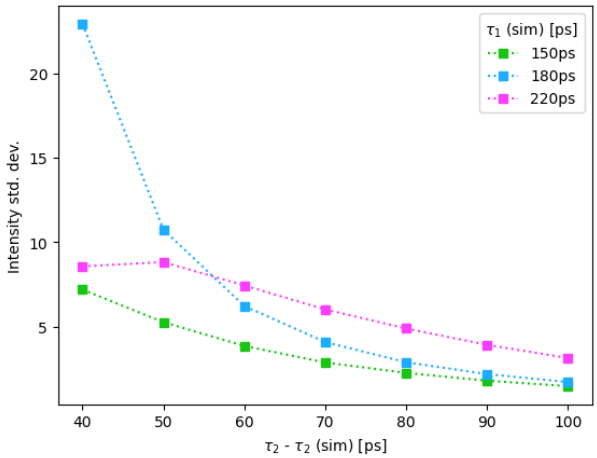


Figure A.3.14: Std. dev. I_1-I_2 = 20%-80%

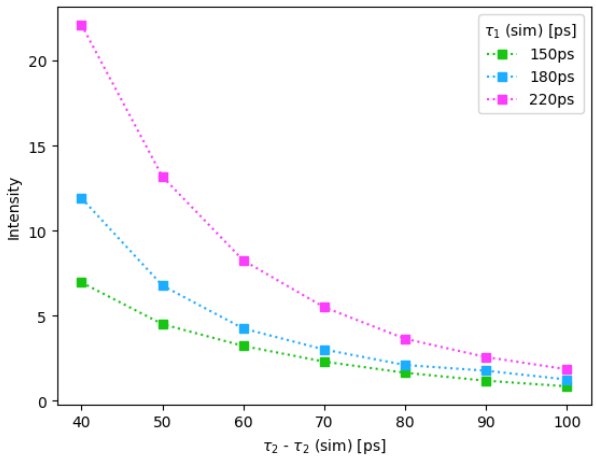


Figure A.3.15: I_1-I_2 = 50%-50%

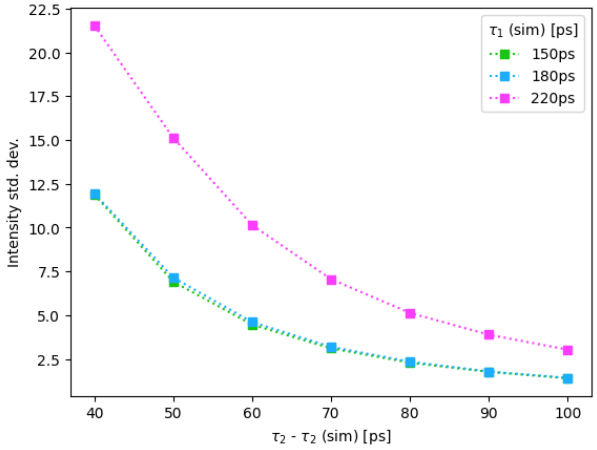


Figure A.3.16: Std. dev. I_1-I_2 = 50%-50%

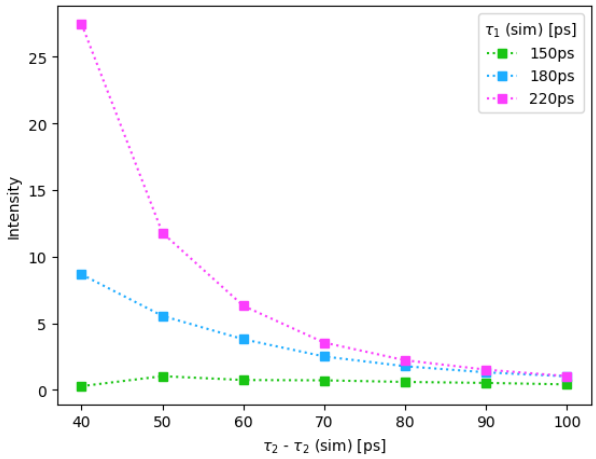


Figure A.3.17: I_1-I_2 = 80%-20%

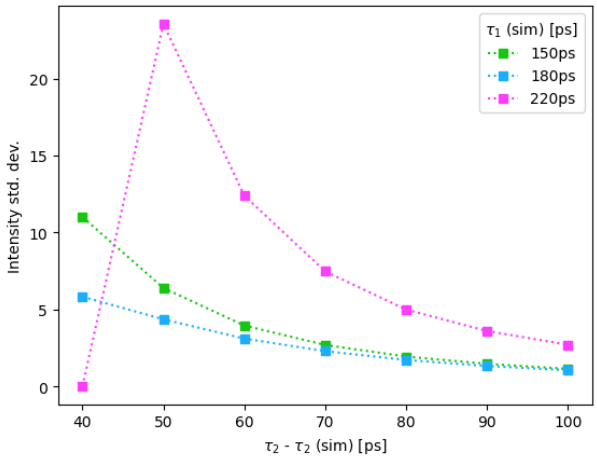


Figure A.3.18: Std. dev. I_1-I_2 = 80%-20%

A.4 Two components, Single Gaussian IRF, τ_1 = 150ps, τ_2 = 180-230ps

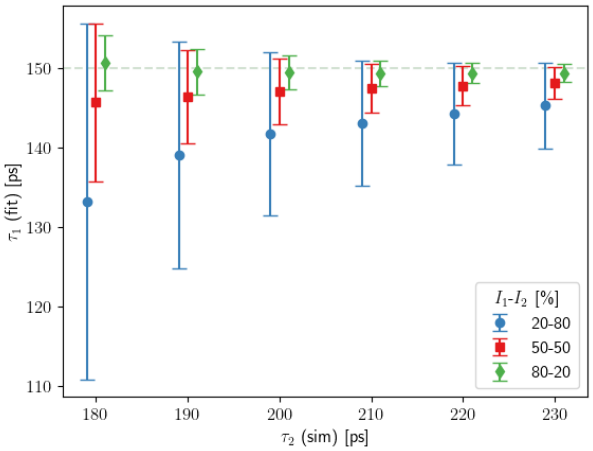


Figure A.4.1: Fitted τ_1

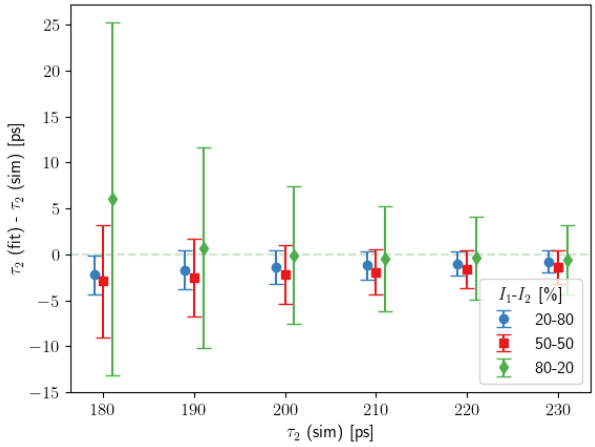


Figure A.4.2: τ_2 (Fitted−Simulated)

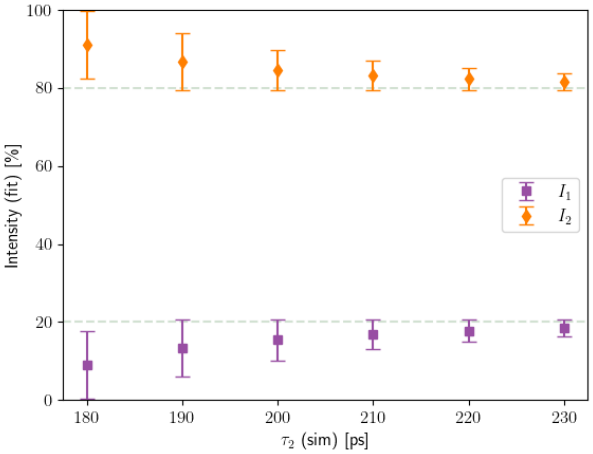


Figure A.4.3: $I_1 = 20\%, I_2 = 80\%$

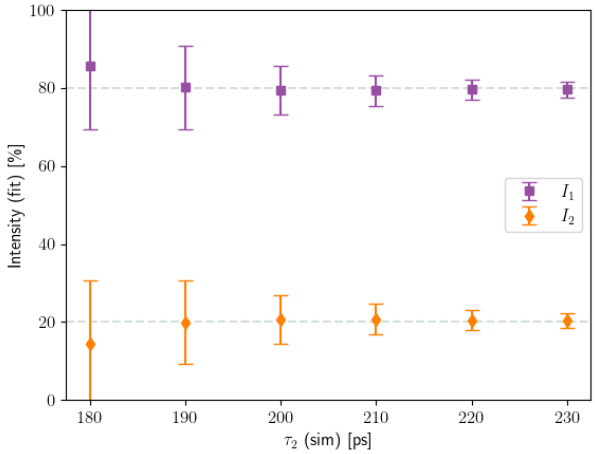


Figure A.4.4: $I_1 = 80\%, I_2 = 20\%$

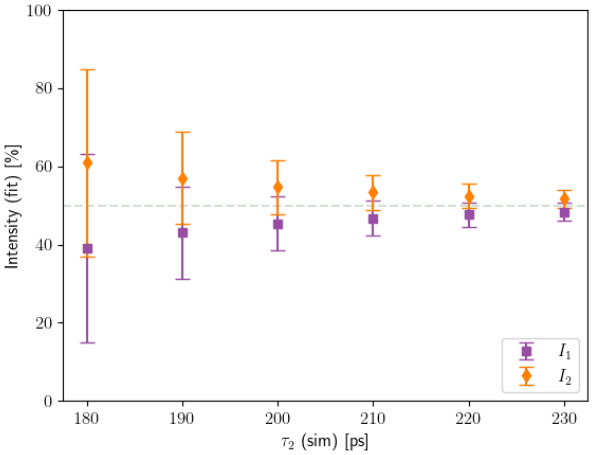


Figure A.4.5: $I_1 = 50\%, I_2 = 50\%$

A.5 Comparison two component fit, single IRF, FWHM = 210,180,150,100

First component, τ_1

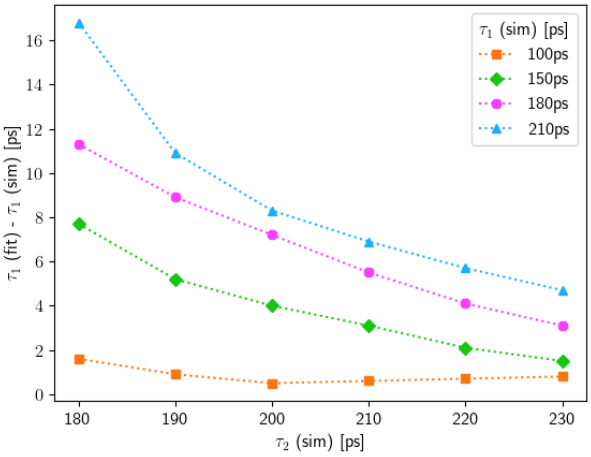


Figure A.5.1: I_1-I_2 = 20%-80%

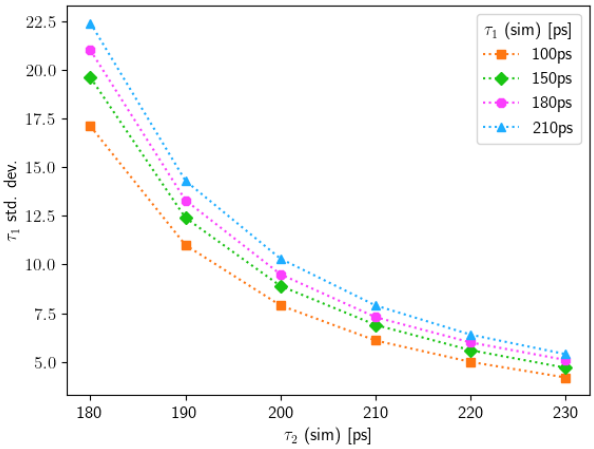


Figure A.5.2: Std. dev. I_1-I_2 = 20%-80%

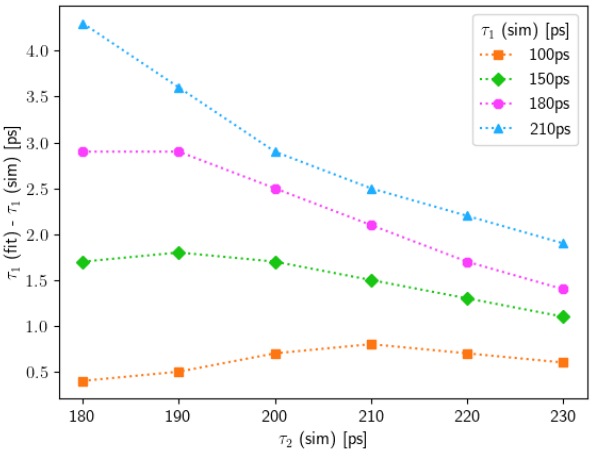


Figure A.5.3: I_1-I_2 = 50%-50%

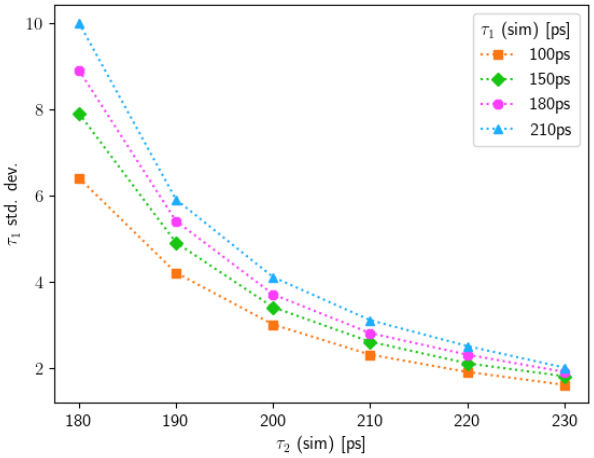


Figure A.5.4: Std. dev. I_1-I_2 = 50%-50%

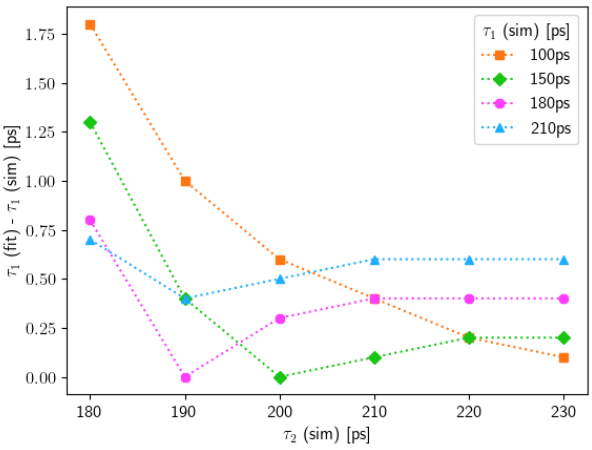


Figure A.5.5: I_1-I_2 = 80%-20%

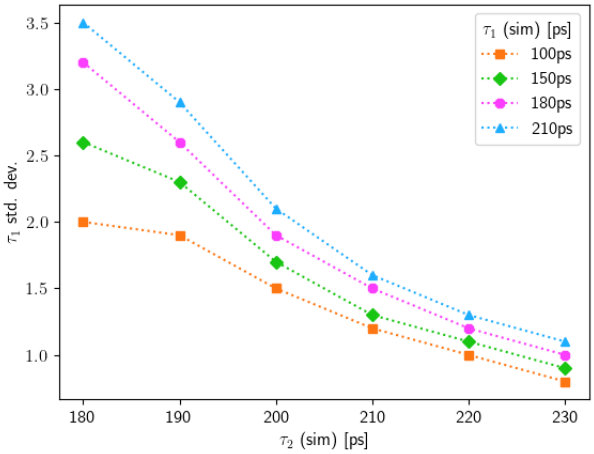


Figure A.5.6: Std. dev. I_1-I_2 = 80%-20%

Second component τ_2

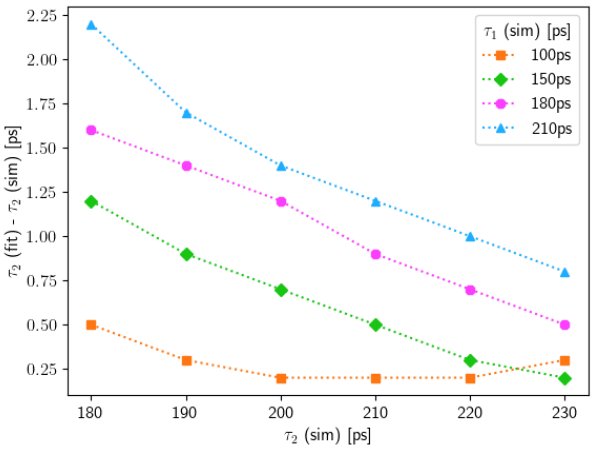


Figure A.5.7: I_1-I_2 = 20%-80%

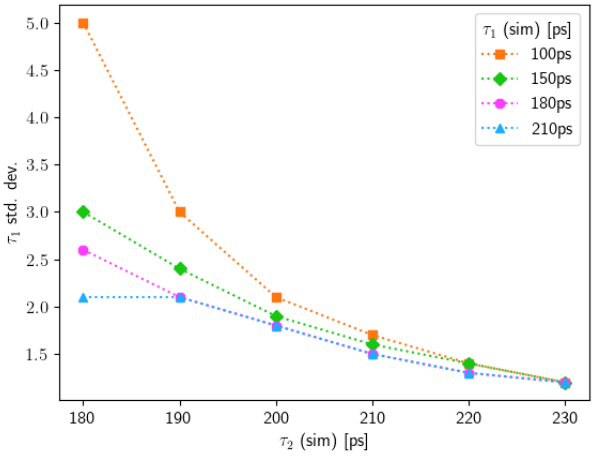


Figure A.5.8: Std. dev. I_1-I_2 = 20%-80%

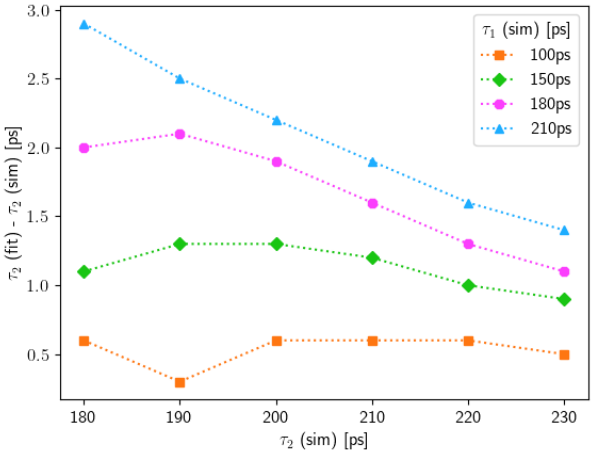


Figure A.5.9: I_1-I_2 = 50%-50%

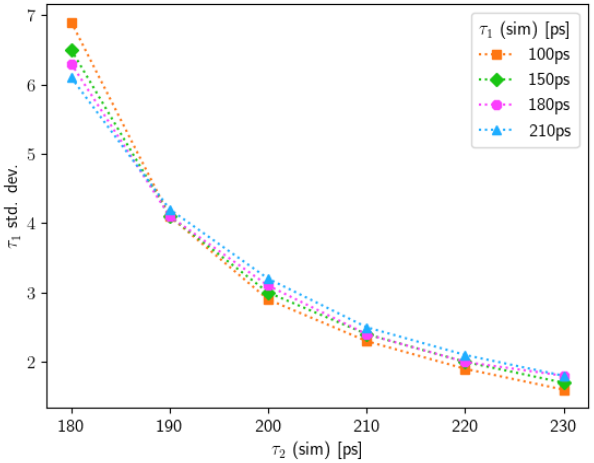


Figure A.5.10: Std. dev. I_1-I_2 = 50%-50%

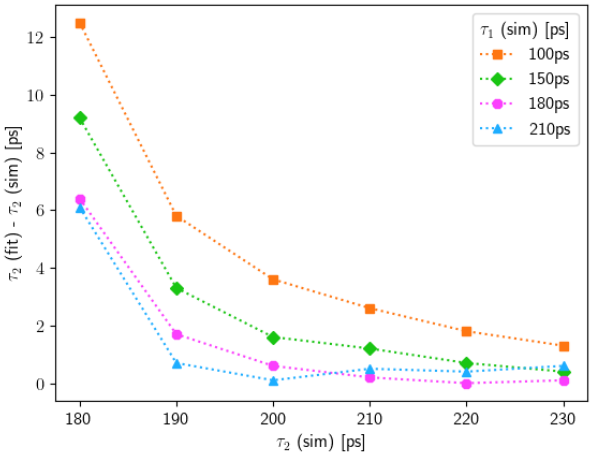


Figure A.5.11: I_1-I_2 = 80%-20%

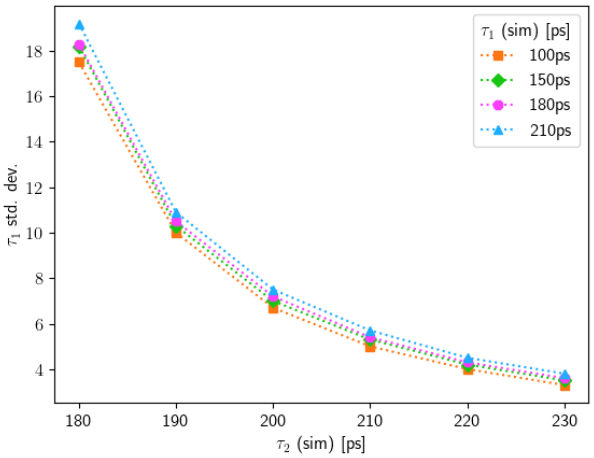


Figure A.5.12: Std. dev. I_1-I_2 = 80%-20%

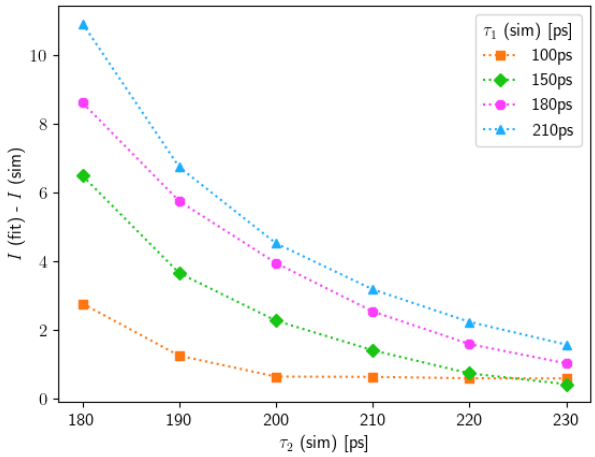


Figure A.5.13: I_1-I_2 = 20%-80%

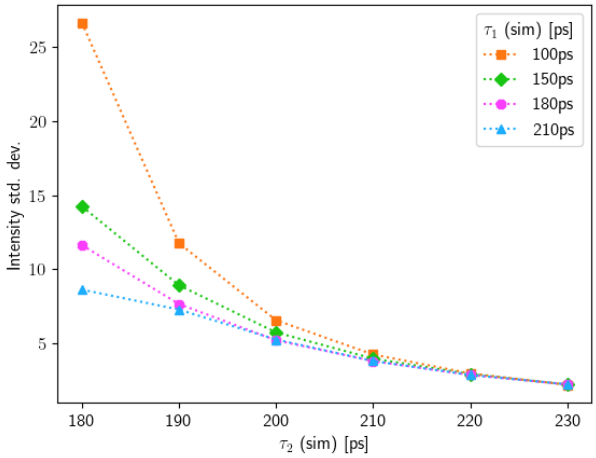


Figure A.5.14: Std. dev. I_1-I_2 = 20%-80%

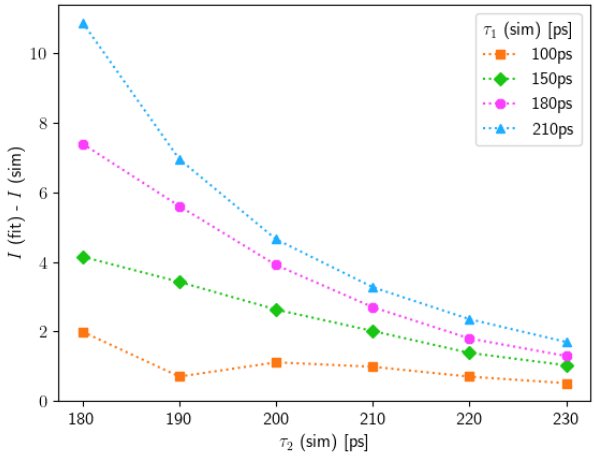


Figure A.5.15: I_1-I_2 = 50%-50%

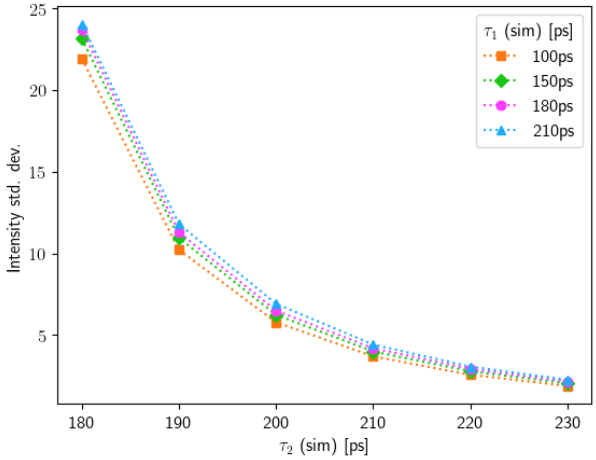


Figure A.5.16: Std. dev. I_1-I_2 = 50%-50%

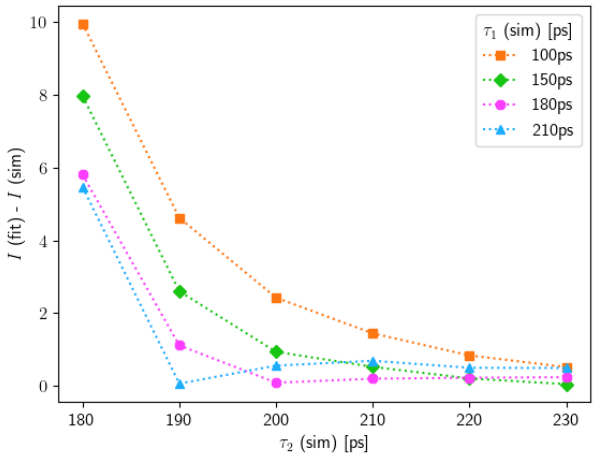


Figure A.5.17: I_1-I_2 = 80%-20%

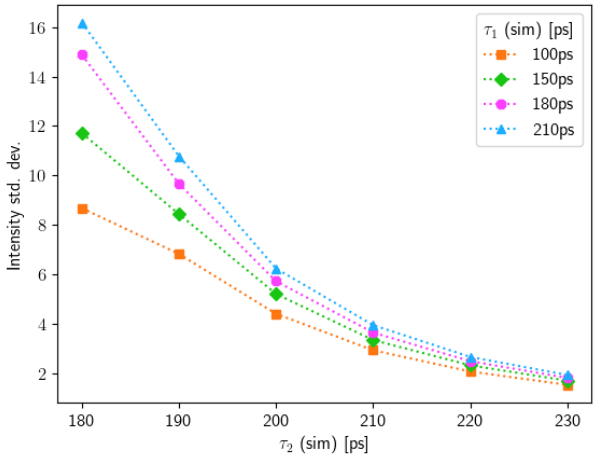


Figure A.5.18: Std. dev. I_1-I_2 = 80%-20%

A.6 Comparison two component fit, single IRF, 3x Number of counts

First component, τ_1

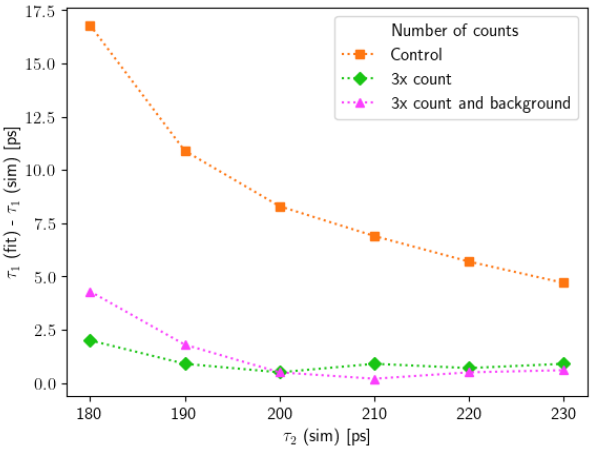


Figure A.6.1: I_1-I_2 = 20%-80%

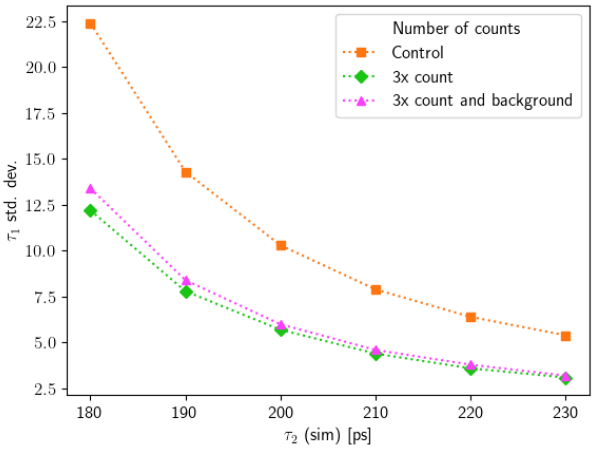


Figure A.6.2: Std. dev. I_1-I_2 = 20%-80%

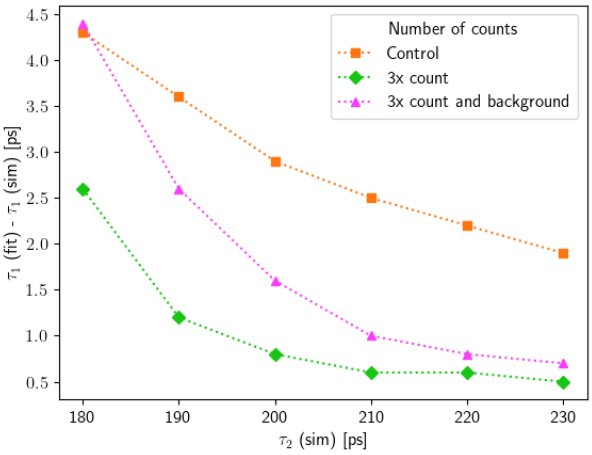


Figure A.6.3: I_1-I_2 = 50%-50%

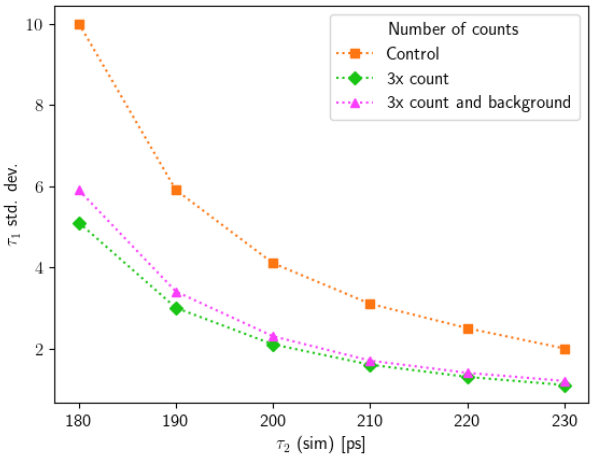


Figure A.6.4: Std. dev. I_1-I_2 = 50%-50%

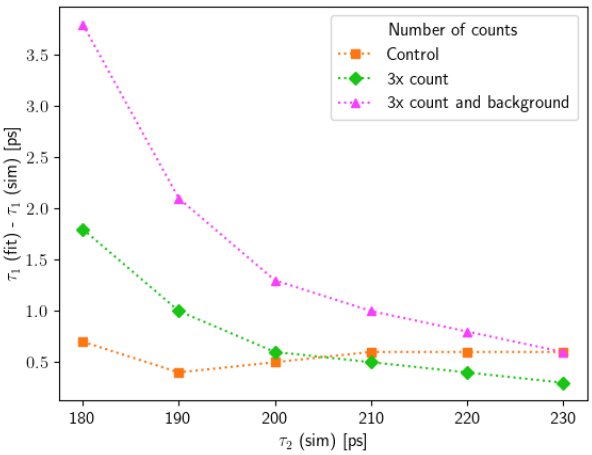


Figure A.6.5: I_1-I_2 = 80%-20%

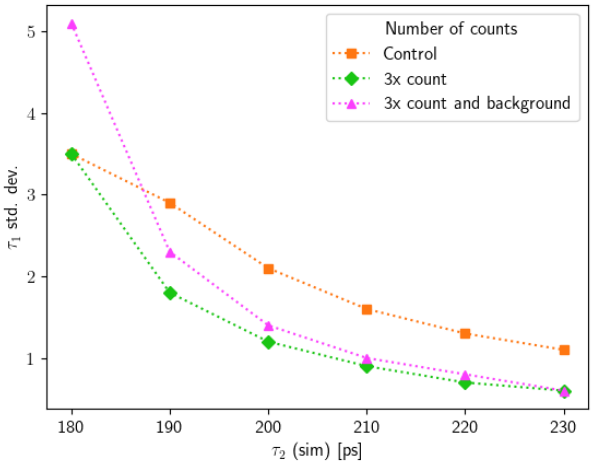


Figure A.6.6: Std. dev. I_1-I_2 = 80%-20%

Second component, τ_2

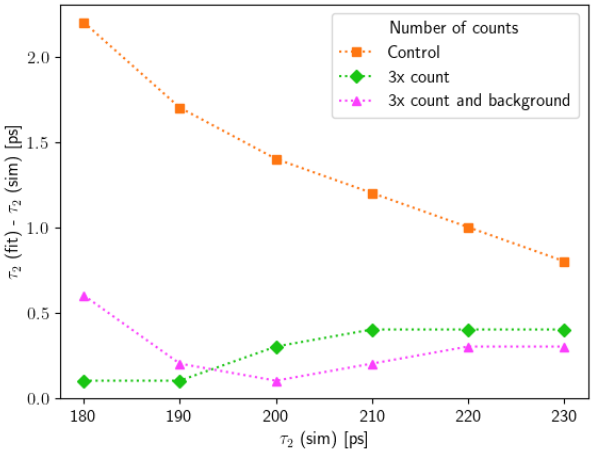


Figure A.6.7: I_1-I_2 = 20%-80%

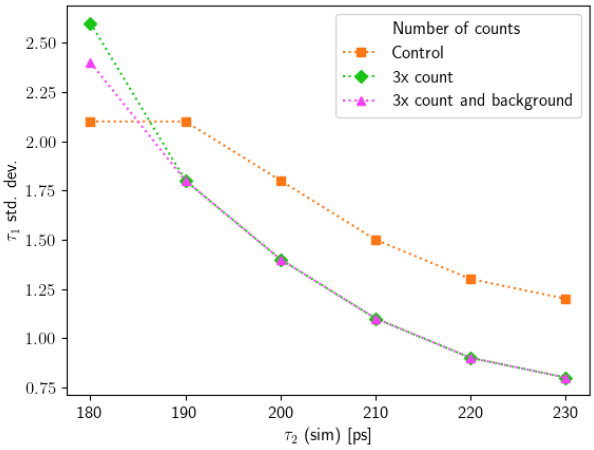


Figure A.6.8: Std. dev. I_1-I_2 = 20%-80%

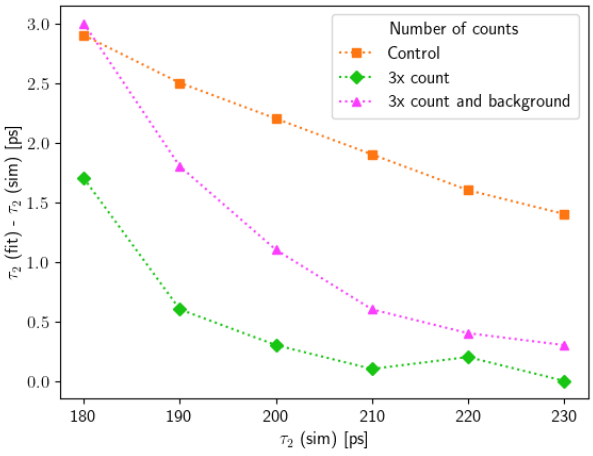


Figure A.6.9: I_1-I_2 = 50%-50%

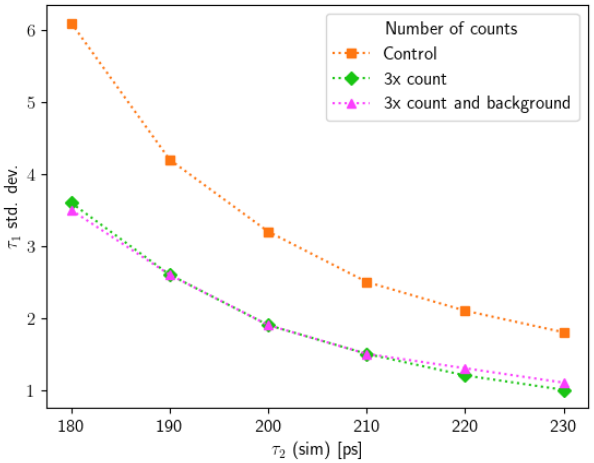


Figure A.6.10: Std. dev. I_1-I_2 = 50%-50%

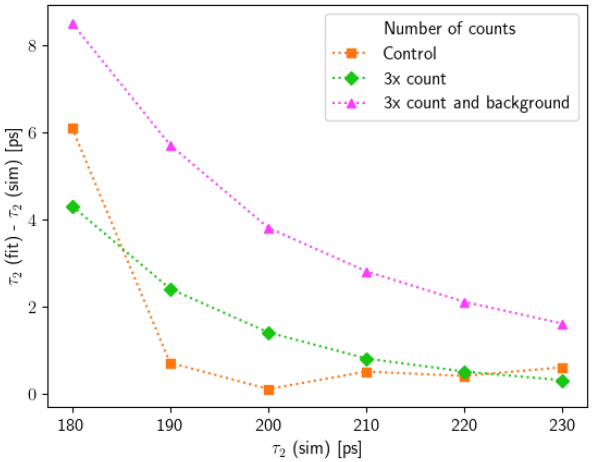


Figure A.6.11: I_1-I_2 = 80%-20%

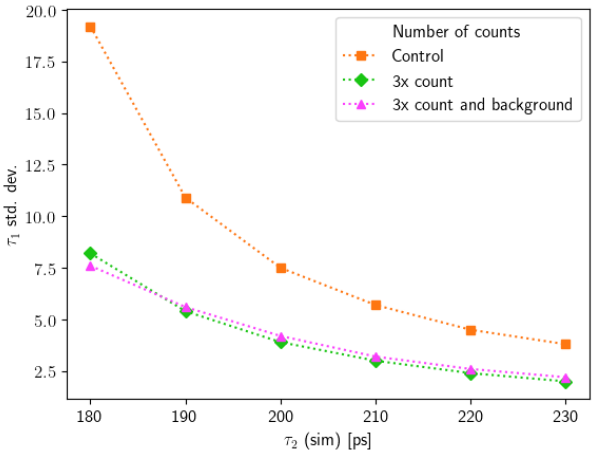


Figure A.6.12: Std. dev. I_1-I_2 = 80%-20%

Intensity

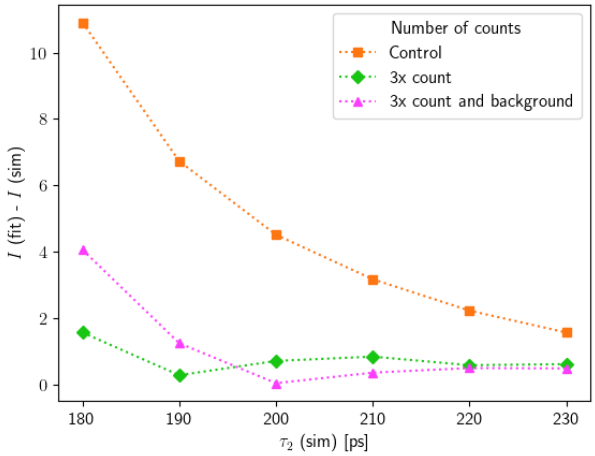


Figure A.6.13: I_1-I_2 = 20%-80%

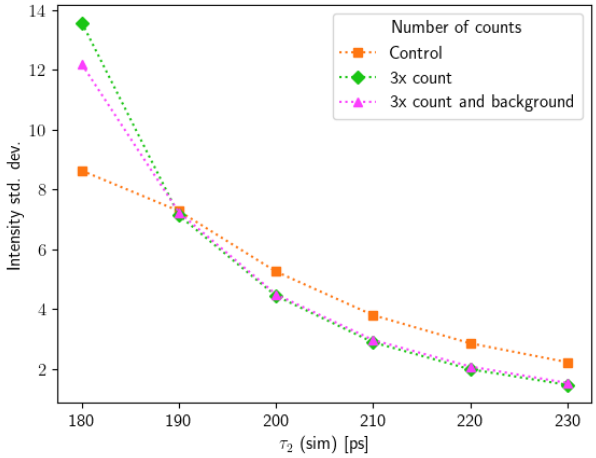


Figure A.6.14: Std. dev. I_1-I_2 = 20%-80%

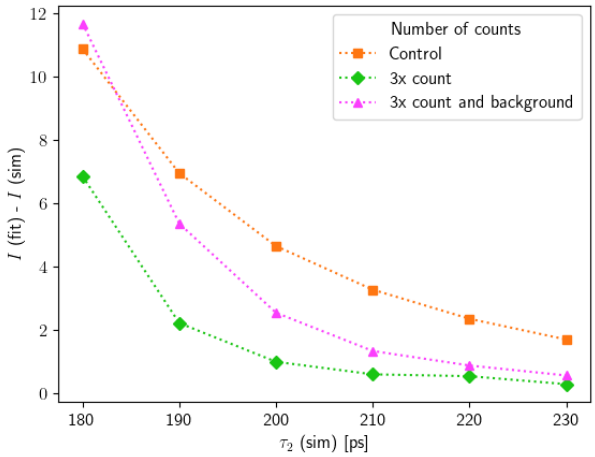


Figure A.6.15: I_1-I_2 = 50%-50%

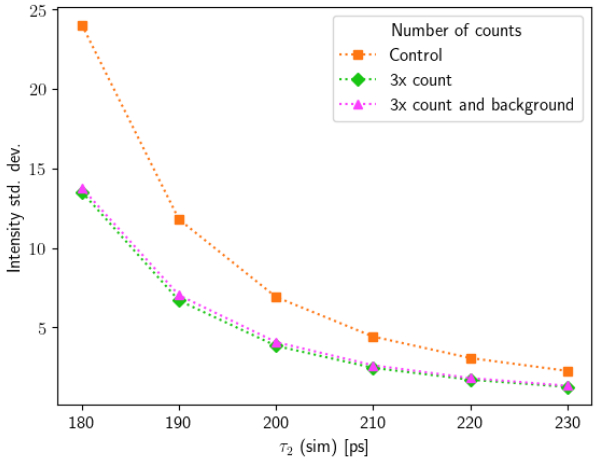


Figure A.6.16: Std. dev. I_1-I_2 = 50%-50%

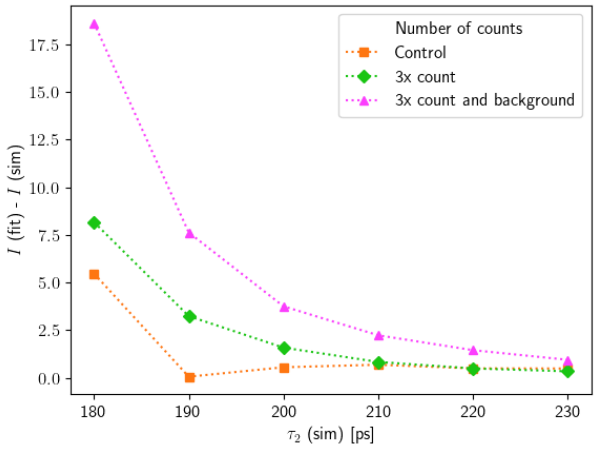


Figure A.6.17: I_1-I_2 = 80%-20%

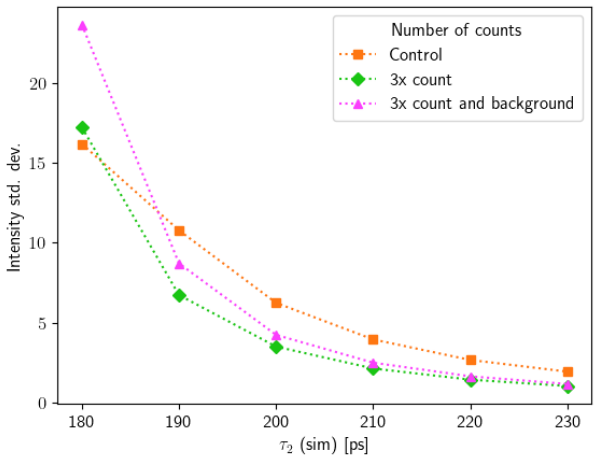


Figure A.6.18: Std. dev. I_1-I_2 = 80%-20%

A.7 Three component fit, $\tau_3 = 2.6$ns

A.7.1 $\tau_1 = 370, \tau_2 = 442$

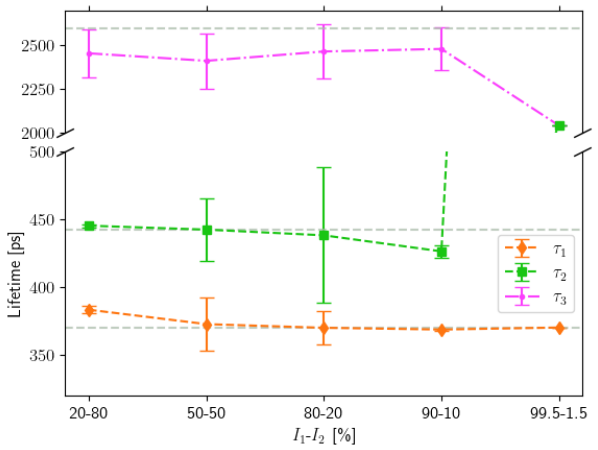


Figure A.7.1: Three lifetime fit

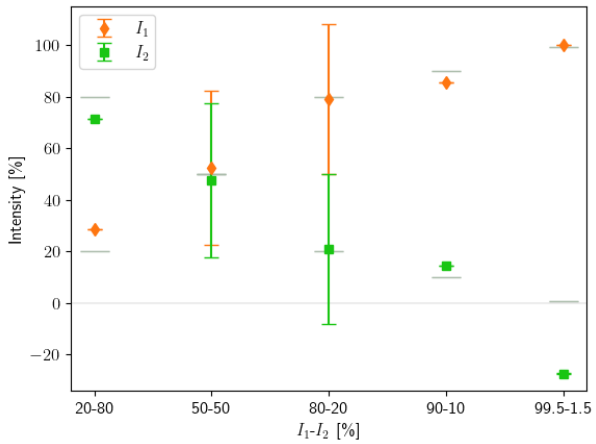


Figure A.7.2: Intensities, Three lifetimes

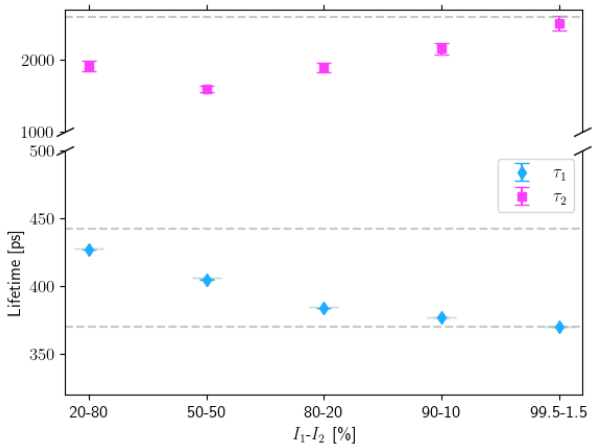


Figure A.7.3: Two lifetime fit

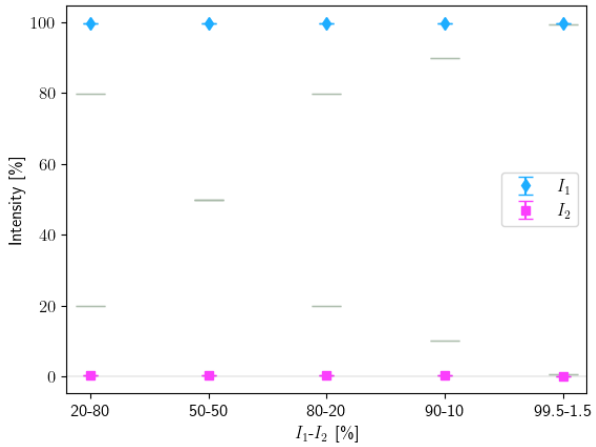


Figure A.7.4: Intensities, Two lifetimes

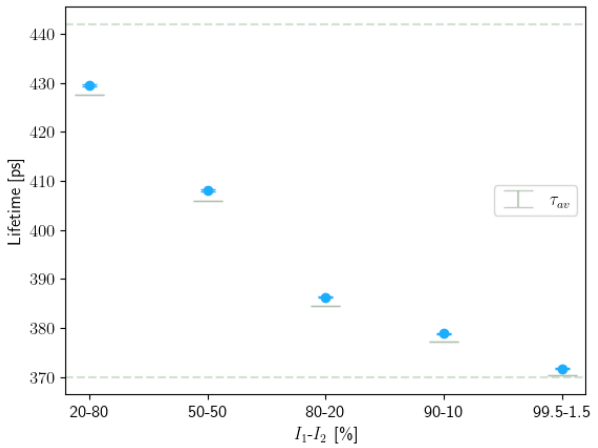


Figure A.7.5: One lifetime fit

A.7.2 $\tau_1 = 355, \tau_2 = 444$

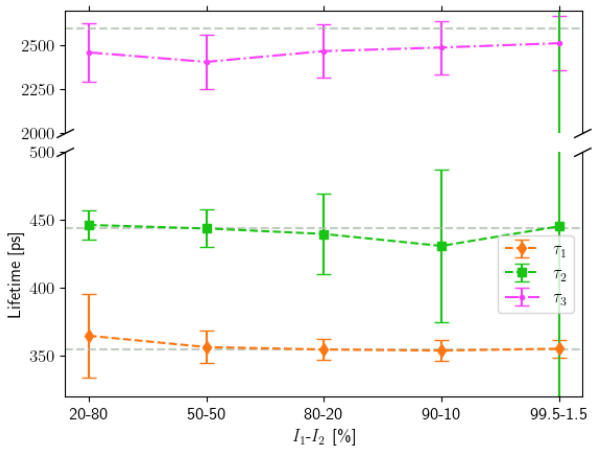


Figure A.7.6: Three lifetime fit

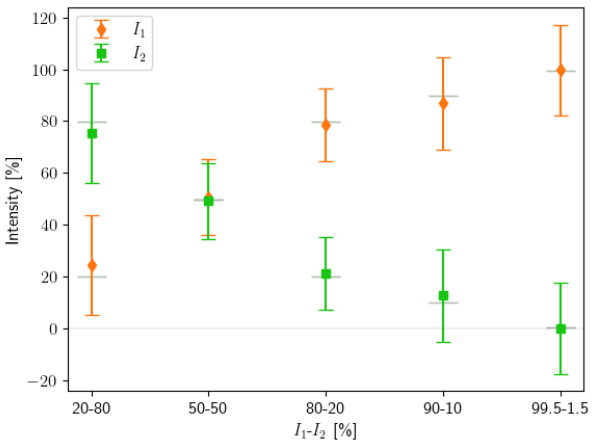


Figure A.7.7: Intensities, Three lifetimes

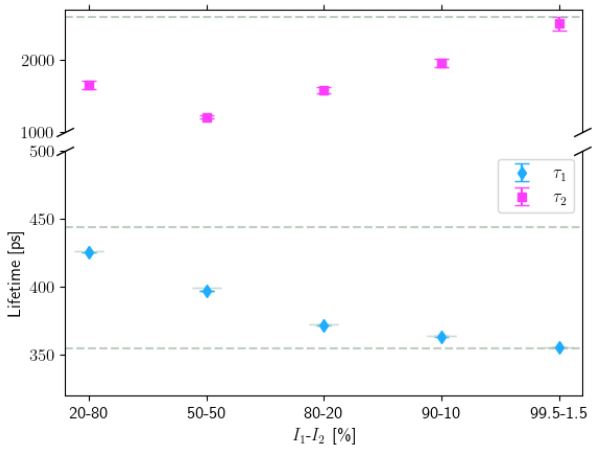


Figure A.7.8: Two lifetime fit

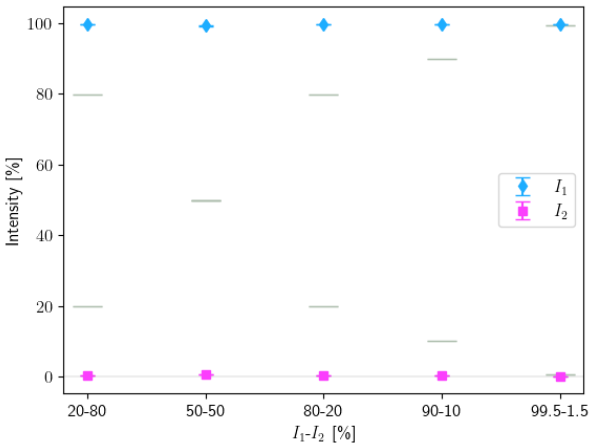


Figure A.7.9: Intensities, Two lifetimes

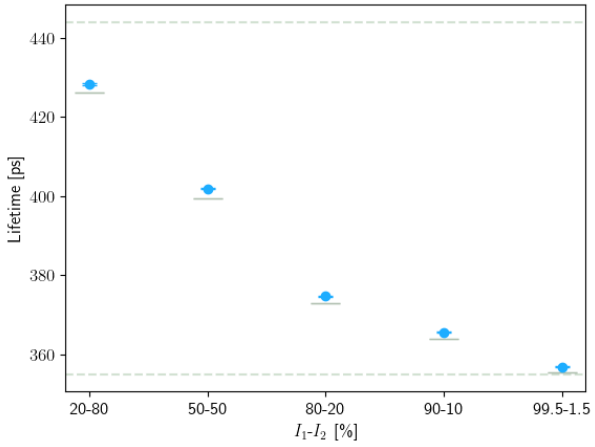


Figure A.7.10: One lifetime fit

A.7.3 $\tau_1 = 348, \tau_2 = 440$

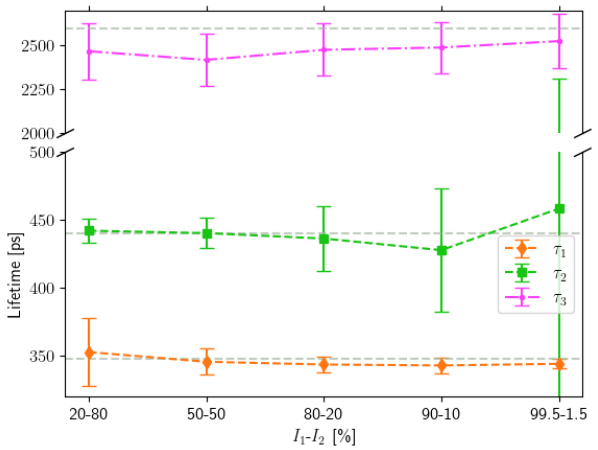


Figure A.7.11: Three lifetime fit

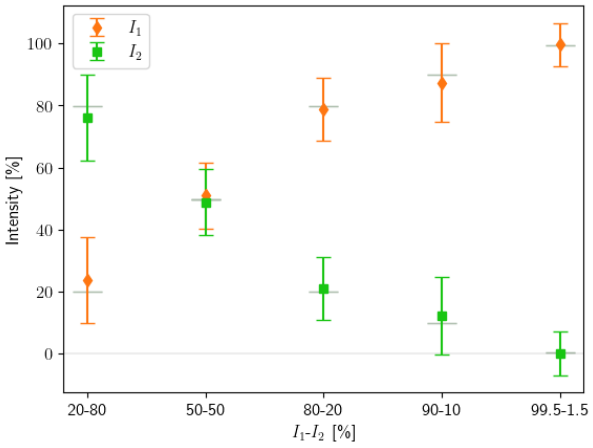


Figure A.7.12: Intensities, Three lifetimes

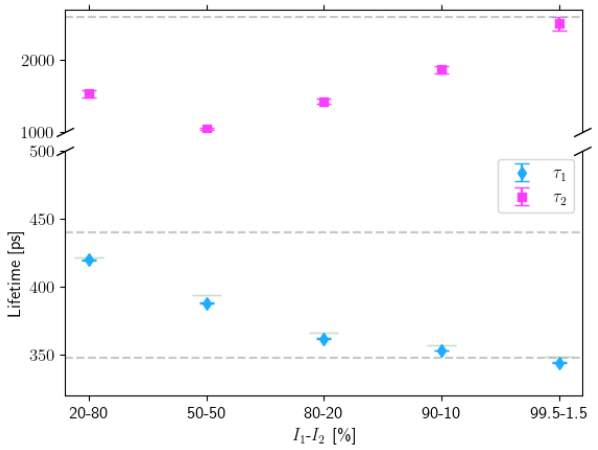


Figure A.7.13: Two lifetime fit

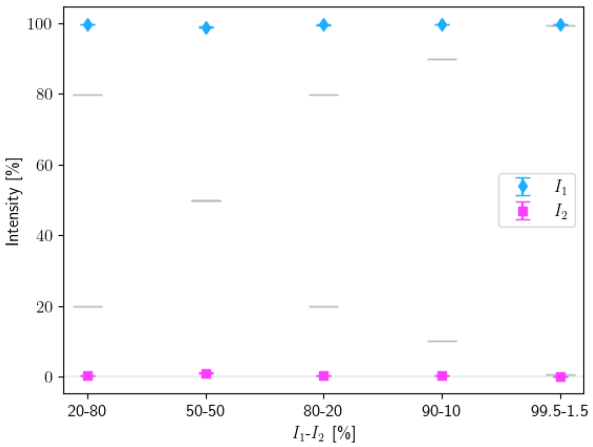


Figure A.7.14: Intensities, Two lifetimes

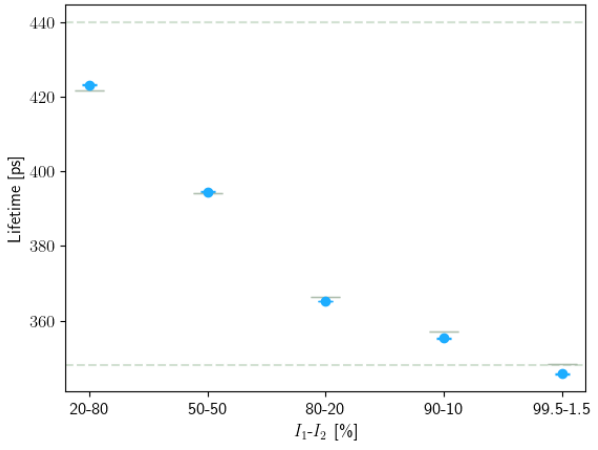


Figure A.7.15: One lifetime fit

Appendix B

Code

All code used during this project can be found in the Github repository at this link: https://github.com/francescotamburi/Palsfit