Estimation of the average spike width of a typical FEL spectrum and reconstruction of FEL pulse durations

Sara Kjær

May 2020

Semester Project in Physics

Supervisor: Dr. Andreas Adelmann
Project research advisors: Dr. Eduard Prat Costa, Dr. Alexander Malyzhenkov

Contents

Abstract

SwissFEL is an X-ray free electron laser (XFEL) facility at the Paul Scherrer Institute. It produces intense and short pulses used for research in multiple scientific fields. The duration of the pulses generated at SwissFEL for standard operation is of few femtoseconds or longer. Expanding this range down to the attosecond scale would allow for a wider range of experiments utilising X-ray radiation. Direct measurements of sub-femtosecond hard X-ray pulses in time domain are extremely difficult and have not been performed at SwissFEL yet. Alternatively, the information in the frequency domain available from spectrometers can be used to infer the XFEL pulse duration. The determination of the spike widths of the XFEL spectra can be linked to the pulse durations via a Fourier transformation. This report outlines the construction of an algorithm for spike width determination, and demonstrates its use to reconstruct pulse durations in the attosecond regime using data acquired at SwissFEL.

1 Introduction

Free-electron lasers (FELs) are modern research instruments used in multiple scientific fields. They consist of an RF gun, which generates the electron bunches, and subsequently passes them through a linac, where they are compressed and accelerated to close to the speed of light. The bunches then go to the undulators, which are arrays of alternately poled magnets, causing the electrons to move in a periodical pattern, effectively slowing them down. The electrons moving on a curved trajectory spontaneously emit synchrotron radiation. Photons travel with the speed of light in vacuum. This radiation, moving faster than electrons, interacts with downstream electrons, accelerate or decelerate them, grouping them into bunches. The characteristic separation scale between bunches is equivalent to the wavelength of the produced radiation. These groups of microbunched electrons start emitting radiation in the same phase, effectively exponentially amplifying the spontaneous radiation. This process is well-known as self-amplified spontaneous emission (SASE) [1][2][3].

Figure 1: A picture of the SwissFEL linac from inside the tunnel

X-ray free electron lasers (XFELs) working in the hard X-ray regime are based on the SASE process, since external driving sources currently do not exist. They generate ultra-bright, highly intense pulses with a standard duration of a few dozen femtoseconds. In recent years, it has become possible to extend the temporal range down to the attosecond scale, creating conditions enabling a whole new range of experiments. One of the attractive applications of short XFEL pulses is the so-called "diffraction before destruction" experiments [4]. Here, a molecule or another kind of specimen is probed with an X-ray laser pulse, which, due to its ultra-short duration, diffracts off the atoms before it destroys the structure of the molecule. The diffracted pulse is then recorded on a screen behind the specimen, providing new insights into its structure without effects of radiation damage. Other uses include nonlinear X-ray spectroscopy at unprecedented temporal scales [5].

SwissFEL [6] is an XFEL facility located at the Paul Scherrer Institute in Villigen, Switzerland. A segment of the facility is shown in Figure 1. Recently constructed and commissioned, it occupies a total of 740 metres, and opened up for user experiments in 2018. Similarly to most other XFELs worldwide, it produces pulses of duration down to several femtoseconds in its standard configuration. To further expand the experimental opportunities of SwissFEL, the aim is to extend its temporal range down to the attosecond scale (i.e. below 1 fs). Characterisation of the temporal profile of the pulses is necessary on a shot-to-shot basis in order to understand the conditions created for a given experiment. Although the generation of such pulses is possible at SwissFEL, it has proven extremely difficult to directly measure the duration of pulses in this regime in time domain.

An alternative method to determine the pulse duration is analysing the spectral information (as seen in Figure 2) in frequency domain [7][8]. This method is based on the fact that the width of a spike in frequency domain is directly related by a Fourier transform of the duration of the pulse. Although the phase information (energy chirp) is lost with this method, it is easier to implement since spectral information is readily available both at SwissFEL as well as most other XFELs. The objective of this project is to implement the spectral method for determining the duration of the short pulses at SwissFEL. In particular, we have

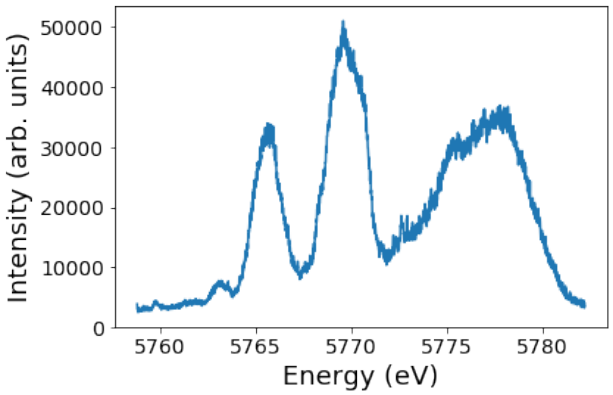


Figure 2: An example of SwissFEL energy spectrum.

developed a robust algorithm capable of determining the number of spikes in a spectrum and their average widths.

This report outlines the physics behind the problem, the steps taken in estimating average spike widths and reconstructing pulse durations, and the structure of the final algorithm. Provided are also provisional results for spectral datasets from past experimental runs, on which the model was tested. The goal is that the algorithm can be implemented in experiments aiming to measure pulses at the femtosecond and attosecond scale. Restricted beamtime demands for a highly efficient model, which can be easily used by the experimenter to perform a fast post-run analysis of the gathered data. It is therefore taken into consideration how fast the algorithm processes data, whether it can run in parallel on multiple processors or machines, and how it can be made the most user friendly.

2 Background

2.1 Experimental set-up and spectrometer

SwissFEL hosts two beamlines: one recently commissioned for hard X-ray radiation (ARAMIS) [6], and another under development for soft X-ray radiation (ATHOS) [9]. Generation and measurement of sub-femtosecond pulses take place in the hard X-ray beamline (Figure 3). Although radiation is generally emitted at 2-12 keV, the data used for this report comes from 5-8 keV runs. Its spectral information is recorded by a photon single-shot spectrometer (PSSS), which is placed 59 metres downstream of the laser's last undulator module. The PSSS [10], whose layout is sketched in Figure 4, operates at 4-13 keV and consists of four main components: a diamond grating, calibration foils, a profile monitor and a component of four bent Silicon crystals. The diamond grating splits off the ±1st diffraction order of X-rays, leaving the 0th order undisturbed for further experiments downstream. Typically, the +1st order diffracted X-rays are then the ones to be analysed by the spectrometer in the standard mode. In attosecond mode, on the other hand, the 0th order spectra are collected due to low signal. There are 5 calibration foils positioned immediately after the grating, which allows energy calibration of the spectrometer. The profile monitor ensures that the beam passed to the crystal has homogeneous intensity. This is necessary since machine instabilities cause jitter, which would result in disturbed measurements. This component is operated at other times than the spectrometer, and normalises the data of runs. Finally, the bent Silicon crystals allow for the beam to be Bragg-reflected, and the beam is passed to a detector placed 1m away from the crystal, away from the vacuum chamber. This generates the final energy spectrum with spikes occurring around 4-13 keV, as shown with an example spectrum in Figure 2. Further details on PSSS structure are laid out in [11].

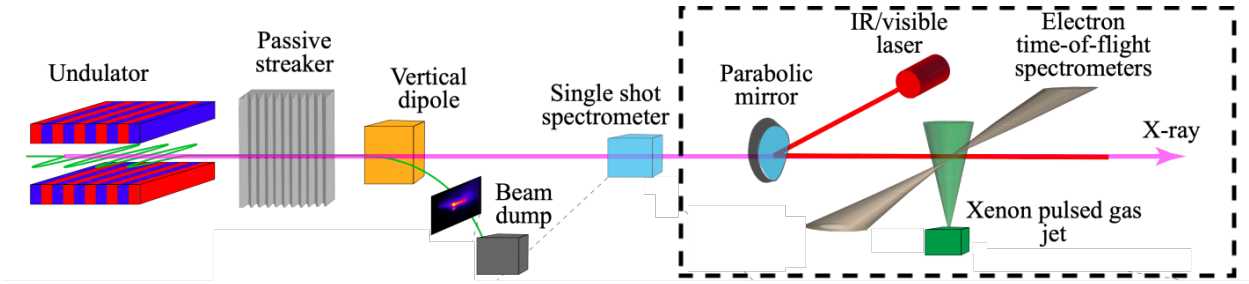


Figure 3: The layout of the SwissFEL post-undulator diagnostics, graphics from [12]. Key components that have already been installed are the undulator for generating laser-like X-rays, the passive streaker for measuring temporal profiles of beams, and the single shot spectrometer providing the spectral information. The dashed section depicts components to be installed for direct measurement in the time domain.

Aside from the spectral method for electron beams, there are two other ways to record the pulse duration: using a post-undulator diagnostics for an electron beam or directly measuring the photon pulses in time domain. The post-undulator diagnostic of an electron beam is possible with a passive streaker [13][14]. This method has been implemented at SwissFEL, where passive streaker has been installed and its commissioning is currently undergoing (see Figure 3). Utilising the device, SwissFEL scientists have recently detected short pulses generated with a slotted foil [15][16]. The slotted-foil approach is a method used as an alternative to strong non-linear compression [8][17], which is the most commonly used method for generating attosecond pulses. We also foresee the direct measurement of the produced attosecond hard X-rays in the time domain. SwissFEL experts plan to accomplish it using a method described in [18]. More details on realisation of this

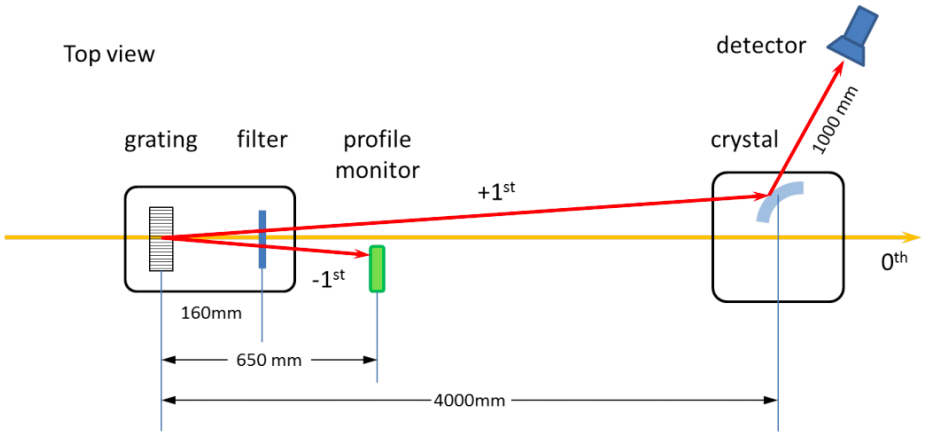


Figure 4: Schematic layout of the PSSS, as detailed in [10]. In order of appearance from left to right are components: diamond grating, calibration foils (filter), profile monitor, bent Silicon crystal.

approach can be found in [12].

2.2 Relation to temporal domain

X-ray pulses have a multi-spike structure in the time- and frequency domains due to the nature of the SASE-FEL process [1]. The widths of the spikes in frequency domain and the pulse duration are related by a Fourier transform, implying that the shorter the spike width in the frequency domain, the longer the pulse duration, and vice versa. The reconstructed pulse duration from the spectral widths depends on the energy chirp of the X-rays, which is a parameter that can not be retrieved from the spectra. The calculated pulse duration will thus vary from a minimum value, τ_{min}, assuming no energy chirp to a maximum value, τ_{max}, for the chirp that maximises the pulse duration. In linear approximation, the formulae used to compute minimum and maximum pulse durations for a single spike model are:

$$\tau_{min} = \frac{4\hbar \ln 2}{f_p} \times 10^{18} \tag{2.1}$$

$$\tau_{max} = \frac{4\sqrt{2}\hbar \ln 2}{f_p} \times 10^{18} \tag{2.2}$$

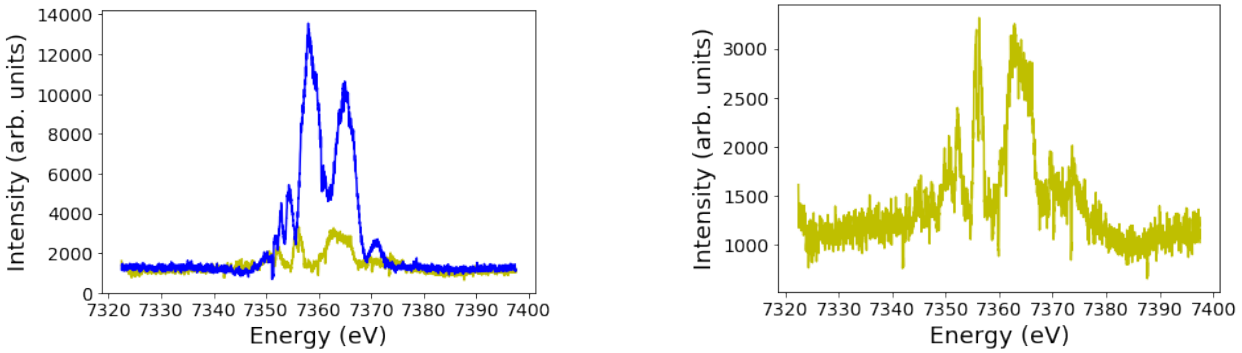
where $\hbar$ is the reduced Planck constant, f_p is the weighted average spike width in eV, and τ_{min} and τ_{max} are computed in attoseconds.

2.3 Energy spectra and analytical challenges

We collected and fitted the energy spectra acquired by the PSSS. The premise of the fitting is that the individual spikes in a spectrum follow a Gaussian distribution, which allows us to estimate widths by looking at their standard deviations. There are however a few challenges when analysing spectra from the PSSS, decreasing the accuracy of the conclusions of this method. Some of these challenges are outlined in this section.

2.3.1 Changing characteristics

Spectral characteristics can significantly vary from shot to shot. This is for example seen when comparing the two spectra from the same experimental run plotted in Figure 5. In blue, we see one with two distinct spikes, both at relatively high intensity, with FWHMs of a few eV. In yellow, as also seen in Figure 5b where it has been plotted on its own intensity scale, is a spectrum with more unclear features. It is difficult to conclude whether it contains one, very wide spike, or several narrower ones that overlap. With such varying characteristics, it is important that the model adjusts its fits and parameters for each spectrum that is analysed. As such, the model presented in this report has been made sensitive to changing spectral characteristics. In general, we still face a big challenge in terms of algorithm construction. Since spectral features unavoidably change, and are sometimes very ambiguous, classification of spectra becomes complicated, even when they are inspected on a case-to-case basis. An algorithm's conclusions consequentially depend on the definitions of e.g. a "spike" we feed to it, which also gives rise to difficulty in cross-checking results.



(a) High-SNR spectrum

(b) Low-SNR spectrum

Figure 5: Two typical SwissFEL spectra from the same run, but with very different SNR. In blue in subfigure (a) a spectrum with a high SNR, and in yellow with a low SNR. The spectrum in yellow is plotted separately on a proper scale in (b) to make its features more apparent. Spectra are from a SwissFEL run with threestage compression and phase 3 settings for the linac, recorded on 14/10/2019.

2.3.2 Signal-to-noise ratio

Unclear features generally arise from a low signal-to-noise ratio (SNR). If we look back at Figure 5, it is evident that the spectrum in Figure 5b has a significantly lower SNR than the blue spectrum of Figure 5a, which will make the analysis more difficult. The random noise is unavoidable, and comes from post-processing of the spectral data.

We can plot the average spectrum of each dataset the model receives, and determine whether datasets are noisy, by looking at the appearance of general characteristics. In Figure 6, the average spectra of 9 different threestage LINAC runs are plotted (phases 0-8, starting from below). It is evident that phases 2-5 have clearer features and relatively high SNR, which makes them the datasets that are the easiest to construct a verifiable algorithm for. Moreover, it can demonstrate the presence of 2-colour modes for phases 5-7, where we see two prominent peaks at energies around 7345 and 7360 eV. The comparison thus shows which types of runs can give the highest algorithm accuracy, and by extension, the most accurate comparison between direct temporal measurements and reconstructed temporal profile. As such, there remains a limit on the algorithm regarding spike classification for noisy datasets, but for the purpose of attempting measurements

at the attosecond scale, the most suitable datasets can be selected for comparison.

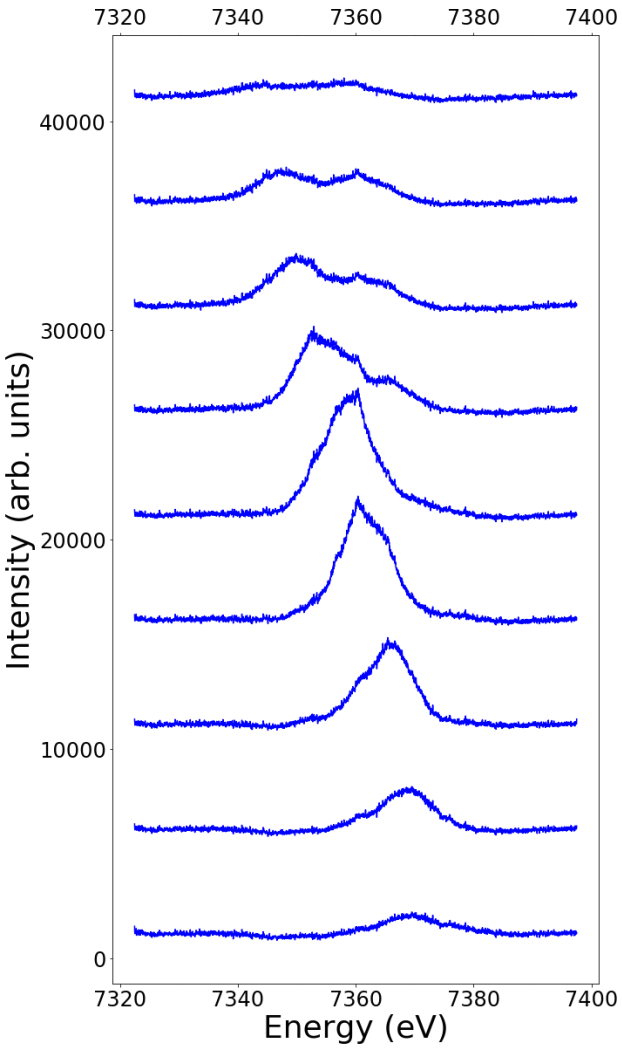
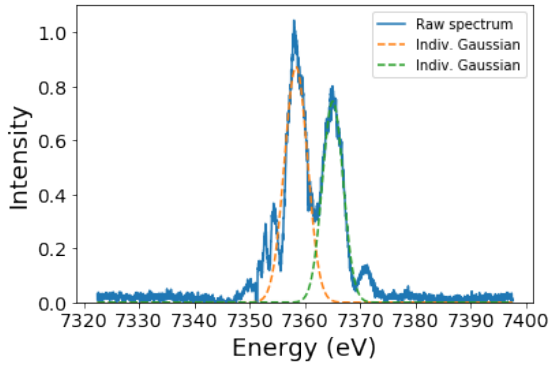


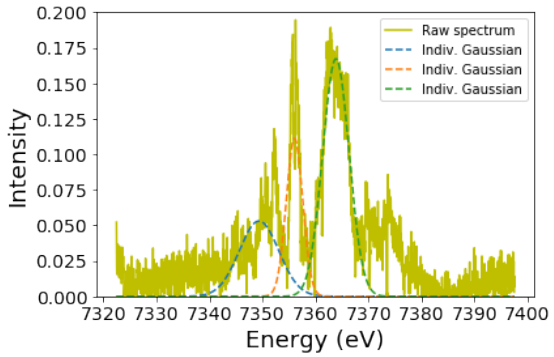
Figure 6: Average spectra for a SwissFEL run with 3-stage compression with phase settings 0-8 for the linac (with phase 0 at the bottom, and phase 8 at the top.)

2.3.3 Gaussian approximation and overlapping spikes

Our assumption that spikes follow a Gaussian profile implies that each spectrum contains a superposition of Gaussian modes. In Figure 7, we see individual Gaussian distributions applied to the two spectra that were also plotted in Figure 5. Before the Gaussian fits, the spectra have been shifted to zero, to make the approximation easier. This is because we see a certain amount of floor noise in each spectrum, such that the spectrum does not have a minimum intensity of zero. This is most likely pixel noise coming from the aperture recording the signal from the spectrometer. By shifting to zero, approximations can be made without assuming an offset. The intensities of the spectra in Figure 7 have furthermore been normalised to 1, making it clear that the spectrum in Figure 7b has a relatively low intensity compared to Figure 7a, making the conclusions about the latter spectrum more reliable than the former.

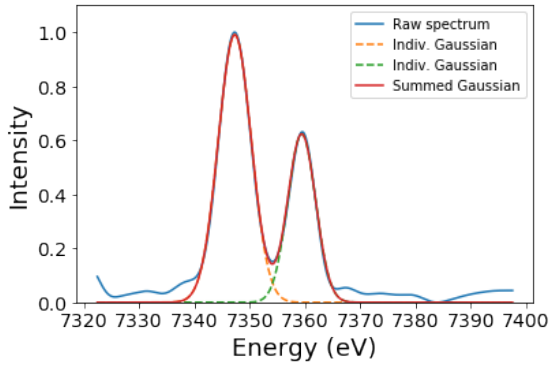


(a) Low-SNR spectrum

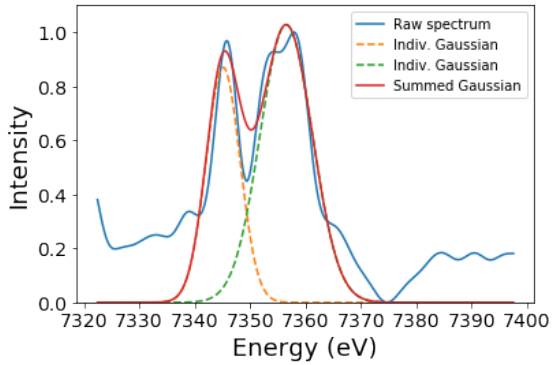


(b) High-SNR spectrum

Figure 7: The same two spectra as Figure 5, but normalised to 1 and shifted to 0, with Gaussian approximations plotted in dashed lines.



(a) Spectrum without significant overlaps



(b) Spectrum containing a significant overlap

Figure 8: Comparison of two spectra (each normalised to 1) and their summed Gaussian distributions. (a) contains no significant overlap, as opposed to (b) where a major overlap can be identified by the difference in minima between the summed Gaussian and the original smoothed spectrum. Further explanation of this procedure will follow in Section 3.5 and 3.6.

Spikes are classified as overlapping when the Gaussian approximations of individual spikes intercept. A raw spectrum with Gaussian approximations that are significantly overlapping is plotted in Figure 8b. When modelling the spikes on Gaussian profiles, the sum of all individual Gaussian plots should approximately fit the full spectrum. In the case of e.g. Figure 8a, the approximations seem to have been accurate, and the widths of the Gaussian fits will be comparable to the widths of the spikes in the raw spectrum. With significant overlaps, however, discrepancies between the summed Gaussian profiles and the raw spectrum arise, as seen in Figure 8b. Consequently, the widths of the Gaussian fits do not properly represent the widths of the actual spikes. Such cases must be taken into account in the construction of the algorithm, and adjustments made to correctly measure the widths.

3 Algorithm

3.1 Technical remarks

The algorithm is written using object oriented Python programming. It is accessed through a visual interface in Jupyter Notebook [19], which allows the user to input a file and prompt analysis of spike data. The model is based on spectral data acquired at SwissFEL, each dataset containing between 100 and 1000 spectra, saved in a .h5 file. A main script, *main.py*, loops through all spectra, and applies the algorithm to each. The algorithm follows a hierarchical structure, where a pseudo-class, *slowfit.py*, step-by-step calls upon a range of subclasses, each returning values to the class. Parallel to the model itself are classes for data analysis. These are prompted from the visual interface (Appendix A), and return visual outputs to the user. An overview of the entire architecture of the model is shown in Figure 9. A more detailed version of this diagram can be found in Appendix B.

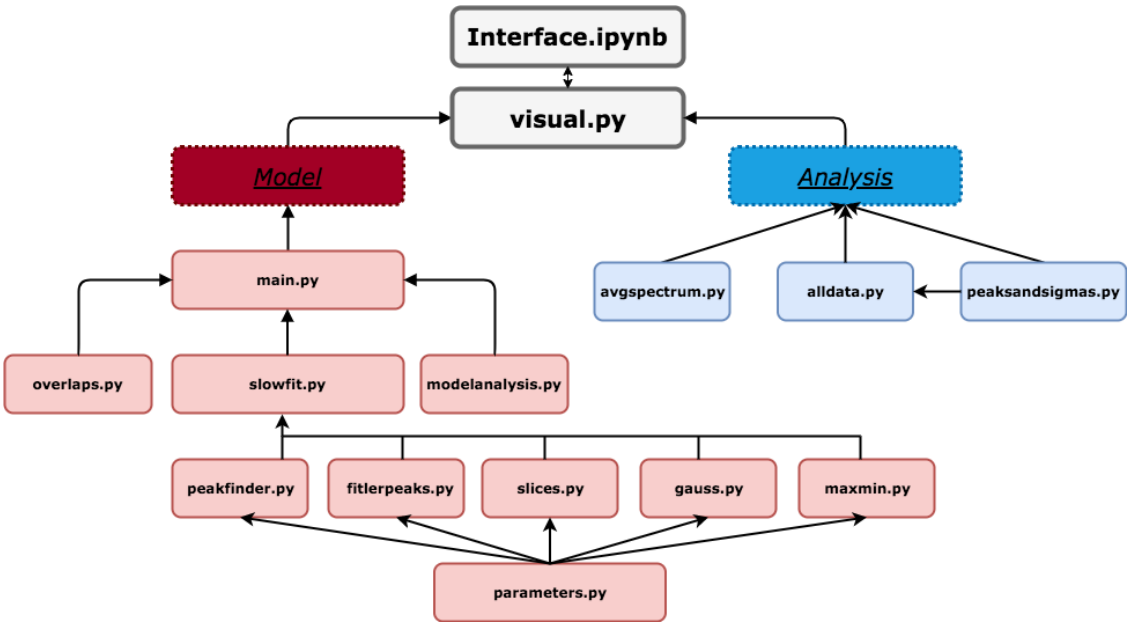


Figure 9: Programme architecture: In red, the algorithm itself, in blue, the modules for data analysis.

Following from the discussion in section 2.3, the model depends on some user-defined parameters for spike identification. We define first the required peak prominence, which takes into account both the peak height and its position relative to other peaks, as to determine how much the peak stands out. The nearest neighbour distance determines how close peaks may be to one another to be recorded. The restriction on peak height filters out small disturbances at low intensity. The overlap condition is used to classify spectra as containing overlaps or not, and the parameter α is related to filtering the data, and will be explained in more detail in Chapter 3. Parameters have been made tunable through the class *parameters.py*, and can thus be adjusted to improve the model for different datasets. The parameter values that were found to be appropriate for the model are:

- Minimum spike height: 20% of the maximum spike intensity
- Minimum spike prominence: 10% of the maximum spike intensity

- Minimum nearest neighbour distance: 5% of the horizontal energy range
- α parameter for data filtering: 700 (dimensionless units)
- Lowpass filter degree: 5 (dimensionless units)
- Overlap condition: 5% of maximum spike intensity

3.2 Data smoothing

To prepare the data for the peak finding algorithm, a first step is to smooth the data. As is evident from Figure 5, spectra will inevitably contain noise, and such microstructures may disturb peak detection. A lowpass filter is therefore implemented, yielding a smooth spectrum to use for further analysis. A well-suited filter is the Butterworth lowpass filter [20], whose action depends on two parameters: degree and cutoff frequency. The degree determines the form of the filter itself, i.e. the order of the polynomial that is used to compute the signal response. The cutoff frequency is the frequency at which the signal response starts to rapidly drop. Suitable values for a standard dataset were initially found at a degree/order of 5 and a cutoff frequency of 0.005. An example of a spectrum and its filtered data is plotted in Figure 10. Here, the noise is also plotted.

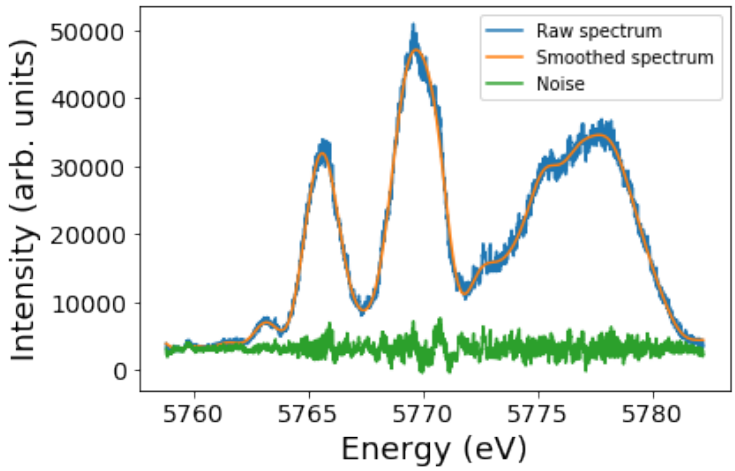


Figure 10: A raw spectrum plotted with its lowpass function and noise

Where an order of 5 is consistently appropriate, the initial value for the cutoff frequency gives rise to a mismatch between the raw and the smoothed spectrum for noisier datasets. A feedback loop is therefore implemented to optimise this parameter for each spectrum. The function was optimised based on a parameter, $\bar{r}$, which is defined as follows:

$$\bar{r} = \sqrt{\frac{\sum_{i=0}^{L}(raw[i] - smooth[i])^2 + \alpha \times \sum_{i=0}^{L}(smooth[i] - smooth[i+1])^2}{L}} \tag{3.1}$$

where α is a tunable parameter, L the length of the dataset, *raw* the dataset of the raw spectrum, *smooth* the smoothed dataset, and i the index of a data point. $\bar{r}$ thus balances the r^2-value, which is a measure widely used to determine the goodness-of-fit of regressions, between the raw and smoothed spectra with

the horizontal noise of the smoothed data, as to account for overfitting. The parameter α determines the weight of the horizontal smoothness, and is currently set to 700 which was found to be appropriate for all trial datasets. The loop starts with a very low cutoff frequency of 0.0001, giving a severely underfit lowpass function, and iteratively increases the cutoff by 0.001 until $\bar{r}$ reaches a minimum. The values of $\bar{r}$ across iterations are plotted in Figure 11. Figure 12 illustrates the iterative process, in this case yielding an optimised cutoff frequency of 0.0121. In Figure 12f, the algorithm has been taken through 5 further iterations, until iteration 17, resulting in an over-fit. Comparing the encircled areas of Figure 12e with those in Figure 12f, we see that there are many more micro-structures in the latter. This could for some datasets result in the detection of too many peaks. This is especially an issue for highly noisy datasets.

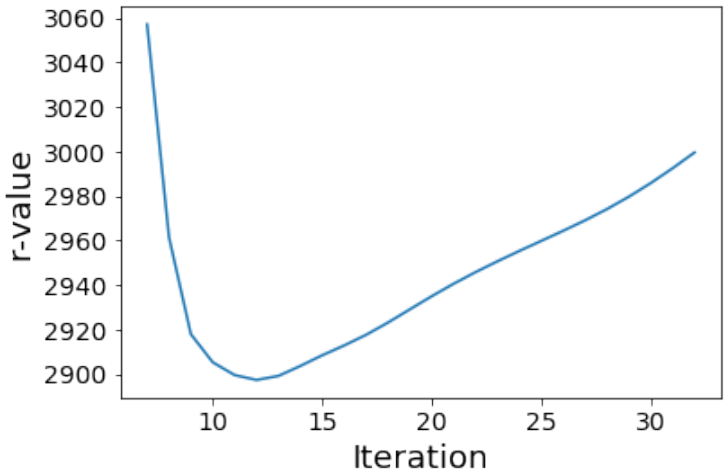


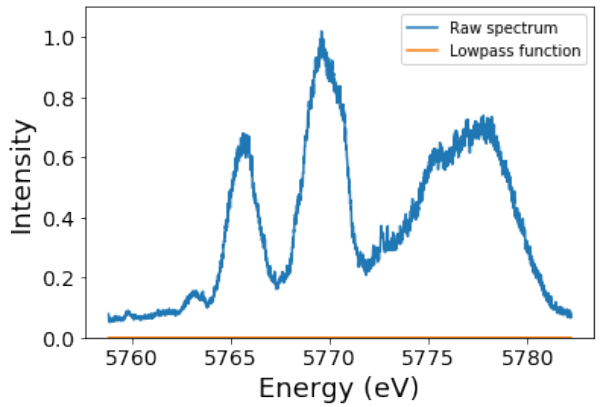
Figure 11: Progressive values of $\bar{r}$ for iterations 8-32. We see that the $\bar{r}$-value is indeed minimised at iteration 12.

The choice of $\alpha = 700$ was based on trial-and-error using different values. The purpose of the parameter is to balance the impact of smoothness and sensitivity to micro-structures in a way that is appropriate to all datasets. Since there is no default "optimal" balance of these two factors, its value must be determined by visual inspection. The goal is to determine a value for α, which, for all datasets, yields the right amount of "smoothness" upon the minimisation of $\bar{r}$. The loop described above was tested for all datasets with various values of α, and the final smoothed functions compared. Ultimately, there is a certain degree of freedom in choosing its value, where any value from 600 to 1000 would have been appropriate for our trial datasets. It is a matter of taste - here, α was chosen such that a few micro-structures remain, leaving it up to the peak finder (described in the subsequent section) to filter out local, low-prominence peaks. The tuning of α for two spectra from different datasets is demonstrated in Appendix C.

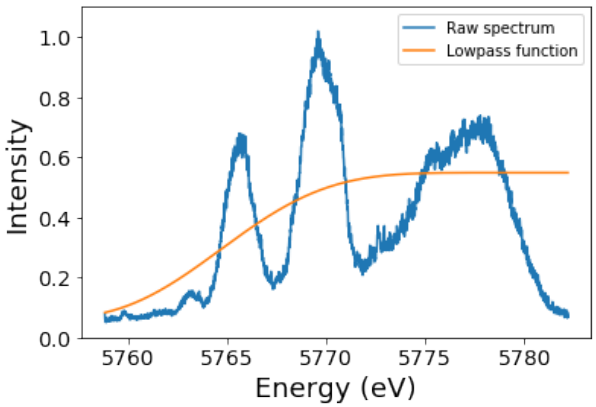
The adaptive lowpass filter was tested against an adaptive moving window average filter, and a median filter. No apparent improvement was observed from these alternative methods, and in the case of noisy datasets, they even yielded a significant increase in error. For this reason, the lowpass filter is implemented in the final version of the algorithm.

3.3 Peak finding

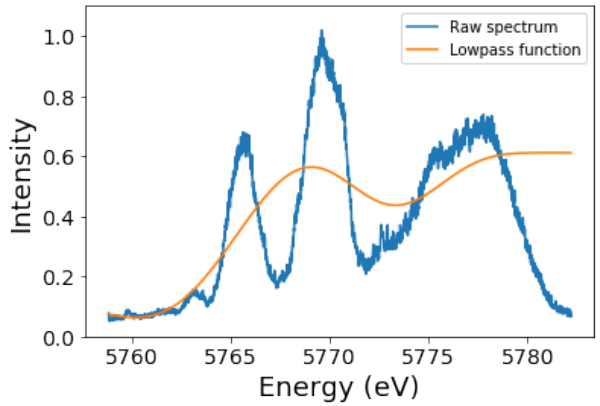
The next step of the algorithm is to identify the local maxima of the spectrum and determine which of them qualify as spikes. This is done in two steps. First, a standard SciPy peak finder [21] is applied



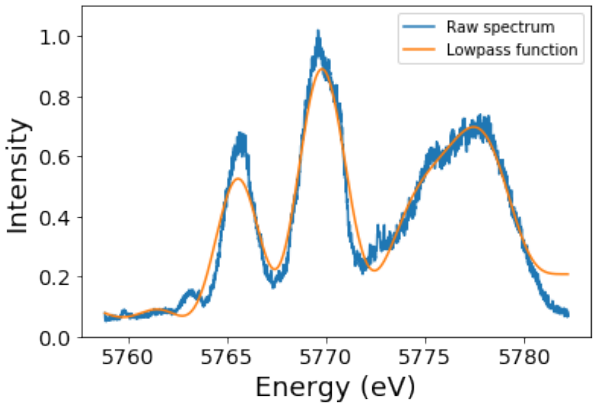
(a) Starting cutoff frequency (= 0.0001)



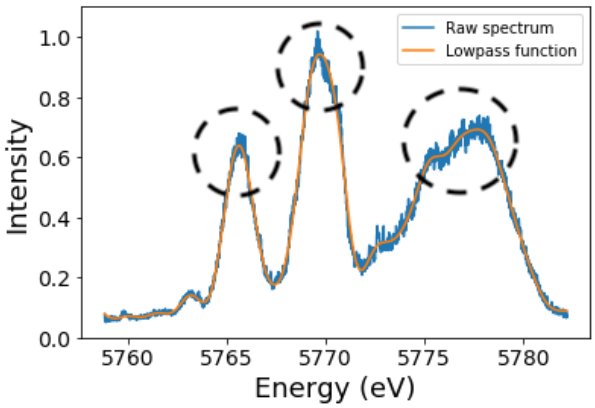
(b) 1st iteration cutoff frequency (= 0.0011)



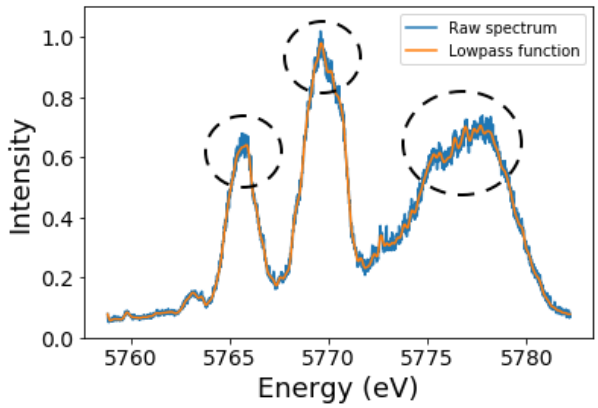
(c) 2nd iteration cutoff frequency (= 0.0021)



(d) 7th iteration cutoff frequency (= 0.0071)



(e) Final cutoff frequency, for which $\bar{r}$ is minimised (= 0.0121)



(f) Over-fitting cutoff frequency: 5 iterations further than final cutoff (= 0.0171)

Figure 12: Progressive fitting of an energy spectrum under tuned lowpass parameters. Intensities are normalised to 1. Only selected iterations are shown, with 12 iterations to get to the final cutoff frequency. The result after 5 further iterations is shown in subfigure (f). Significant impact of the lowpass cutoff frequency can be seen in the encircled areas, where peak detection could be disturbed due to an overfitting.

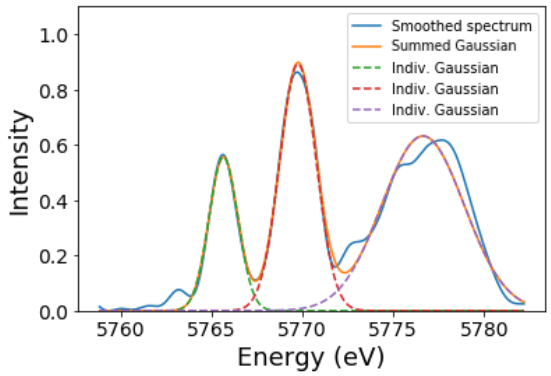
to the smoothed spectrum, with conditions on peak prominence, minimum height and nearest neighbour distance. These are all specified in the *parameters.py* class. Secondly, the data is passed through a filtering

function, where further conditions such as distance from edge of spectrum can be specified. This class takes the spectrum and peak positions as input, and can be modified for an improved algorithm.

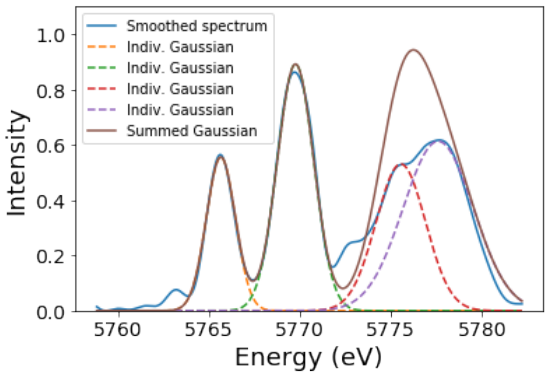
To prepare each spike for the Gaussian fit, the local minima, which are the minima between detected peaks, are furthermore localised, and the dataset sliced into its individual spikes. For each slice, the energy scale is preserved by assuming zero-valued intensity outside the spike itself. The spectra were furthermore assumed zero-valued at the edges, since some edge spikes carry long "shoulders", yielding inaccurate Gaussian fitting.

3.4 Gaussian approximation of data

With the data sliced into individual spikes, it is now possible to extract standard deviations from their approximated normal distributions. For this procedure, the "lmfit" Python package [22] was used, and the standard deviations accordingly extracted. For each spectrum, the final values returned are therefore 1) number of spikes in the spectrum, and 2) the weighted average standard deviation value over the entire spectrum. The latter is subsequently converted to the FWHM, and used to calculate the temporal duration. A filtered spectrum is plotted in Figure 13a, with individual Gaussian approximations.



(a) Three spikes fitted to spectrum. The summed Gaussian fits quite accurately to the data.



(b) Four spikes fitted to spectrum. The overlap results in a highly-peaked summed Gaussian.

Figure 13: Two approaches to fitting Gaussian distributions to an energy spectrum. (a): The spectrum is assumed to contain three spikes, one being significantly wider than the others. (b): The spectrum is instead fitted with four Gaussians, two of which are significantly overlapping. These two spikes have approximately the same width.

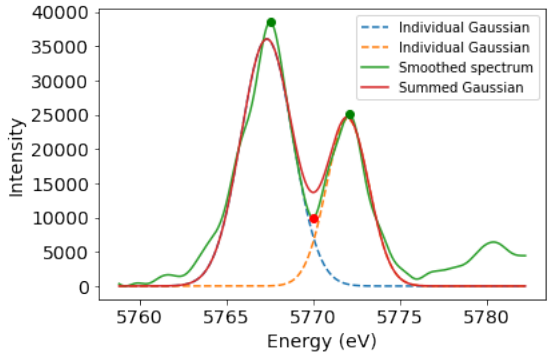
It is, of course, crucial that this Gaussian approximation be as accurate as possible. To test this, the individual Gaussian functions are summed and compared to the filtered spectrum. To inspect whether the plots overlap, the local minima of the summed Gaussian function and the filtered spectrum are compared. In the case of Figure 13a, we are thus looking at the two minima between spikes of both the blue and the orange lines. Should the difference at any minimum be above 5% of the maximum intensity (or, the condition set in the *parameters.py* class), the spectrum is flagged as containing overlaps, and treated by the following step of the algorithm.

It can also be the case that some overlaps are so significant that they are registered as one, very wide spike. An example is the spectrum we saw in Figure 13a, where the last spike contains a small bump before its peak. Where the initial Gaussian approximation does seem accurate, this could be interpreted as an extra spike, and approximated as in Figure 13b. Due to the massive overlap, the summed Gaussian deviates

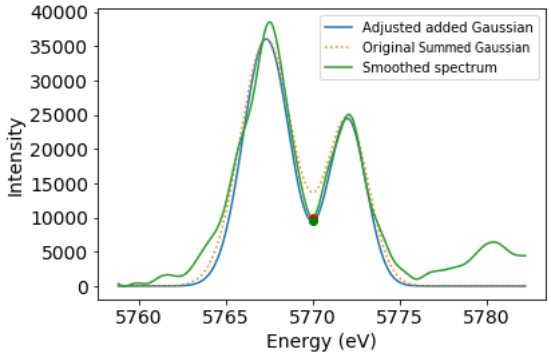
significantly from the real data, which must subsequently also be corrected for. The current model does not consider these cases, but a criterion on the spike widths could be implemented in future models, such that any spikes that are a lot wider than the average for the spectrum would be re-fitted with an extra Gaussian distribution.

3.5 Correction for overlaps

If there is a significant difference between the local minima of the summed Gaussian function and the filtered spectrum, the individual Gaussian functions are either too broad or too narrow. The objective is to iteratively re-construct the individual functions with a modified standard deviation until the sum adequately fits the filtered spectrum. Should the local minimum of the summed Gaussian be higher than the filtered spectrum, the sigmas and amplitudes of the individual functions are decreased by 1% for each iteration. For



(a) Original spectrum and individual Gaussian approximations.



(b) Adjusted Gaussian approximation, clearly fitting the data better.

Figure 14: Overlapping spectra. After registering the spectrum as containing overlaps (a), the Gaussians are adjusted, resulting in the spectrum shown in (b). Intensity is in arbitrary units.

Gaussian minima lower than that of the summed Gaussian function, the sigmas and amplitudes are iteratively increased by 1%. For each iteration, the sum is computed and the difference once again measured, and the loop breaks when a minimum is reached. The resulting standard deviations are then extracted, and the values previously extracted replaced. The r^2 value is also computed for subsequent analysis. Figure 14 demonstrates two spectra: (a) a spectrum identified as containing an overlap, and (b) the adjusted summed Gaussian function. It is clear to see that Figure 14b provides the more accurate fit, and hence the more accurate spike widths. In the re-fitting process, the heights of the Gaussian distributions are kept constant by multiplying their amplitudes by the same factor as the standard deviation. A potential further development to the model could be to also adjust the amplitudes to correspond exactly to the peaks before adjusting standard deviations. This could be beneficial for spectra like that in Figure 14, where the second Gaussian peak coincides perfectly with the corresponding spike, but the first Gaussian is better approximated with a larger amplitude.

Spectra with an identified overlap that contain more than 2 spikes are treated more carefully. All individual Gaussian functions must be re-fit, but under different conditions, since differences between minima can be of varying magnitudes. That is, refitting the functions around one overlap can impact the neighbouring overlap. Here, both the relative size of the differences, whether an individual spike is by the edge, and whether neighbouring spikes have been widened or narrowed are all factors that are taken into account. The full

algorithm for this procedure can be found in the *overlap.py* class.

The current algorithm re-fits most of the overlapping spectra such that they are within the defined overlap condition. With the difficulties arising from spectra with multiple overlaps, few spectra remain which are not re-fitted within the conditions. For our trial datasets, the amount of spectra with overlaps was originally 44.4%, and was decreased to 11.2% after the re-fitting procedure. This is a significant decrease, but could be optimised with a more detailed algorithm. The average r^2-value between the summed Gaussian and the filtered spectrum is also computed, showing an average increase of 3.5% after overlap treatment. This is expected, as the initial Gaussian fitting takes the most accurate approximation possible to the filtered spectrum. As the Gaussian distributions are refitted, they deviate from the filtered spectrum, and while the overlap becomes smaller, the r^2-value increases. Data for evaluating the efficiency of the overlap procedure is generated using the *modelanalysis.py* class, and outlined in Appendix D.

To perfectly re-fit the remaining spikes, the algorithm can be expanded to include more conditions, or loop over a wider range of combinations of possible standard deviations for the individual spikes. This would, however, be at the cost of efficiency, and the amount of algorithm steps for such a procedure increases exponentially with the number of spikes in a spectrum. Alternatively, one could also consider an Extremum Seeking (ES) feedback scheme, as detailed in [23]. This is a complex procedure to implement, but speeds up the optimisation procedure significantly.

4 Results and discussion

4.1 Results for pulse durations

For the trial datasets, the results are listed in Table 1. Here, the average number of spikes per spectrum for the dataset, and their average widths are listed.

Dataset index	Experiment configurations	Average number of spikes	Average FWHM weighted spike width (eV)	Photon energy (keV)	τ_{min} (as)	τ_{max} (as)	Spike number distribution (%) ([1, 2, ..., 6])
1	Twostage Phase 0	2.01	6.56	7.284	278	393	[26.5, 49.0, 22.0, 1.5, 0.0, 0.5]
2	Twostage Phase 1	2.03	6.14	7.284	297	420	[25.5, 48.5, 24.0, 2.0, 0.0, 0.0]
3	Twostage Phase 2	2.31	5.62	7.284	325	460	[13.0, 47.0, 35.5, 4.0, 0.0, 0.0]
4	Twostage Phase 3	2.53	6.12	7.284	298	422	[11.5, 38.0, 37.0, 12.0, 1.0, 0.0]
5	Twostage Phase 4	2.76	4.05	7.284	451	638	[3.0, 36.5, 44.0, 14.5, 2.0, 0.0]
6	Threestage non-linear	1.99	7.29	5.771	250	354	[25.0, 50.0, 24.0, 0.0, 0.0, 0.0]
7	Threestage Phase 0	2.16	8.63	7.360	211	299	[29.5, 34.5, 24.0, 6.0, 2.5, 0.5]
8	Threestage Phase 1	1.75	7.90	7.360	231	327	[45.5, 37.0, 14.0, 2.5, 0.5, 0.0]
9	Threestage Phase 2	1.64	7.05	7.360	259	366	[51.5, 36.5, 8.5, 2.5, 0.0, 0.5]
10	Threestage Phase 3	1.49	7.22	7.360	253	357	[59.5, 33.5, 5.5, 1.5, 0.0, 0.0]
11	Threestage Phase 4	1.68	7.54	7.360	242	342	[47.5, 37.0, 14.0, 1.0, 0.0, 0.0]
12	Threestage Phase 5	1.92	10.53	7.360	173	245	[30.0, 50.0, 18.0, 2.0, 0.0, 0.0]
13	Threestage Phase 6	1.88	10.56	7.360	173	244	[28.0, 58.5, 11.5, 2.0, 0.0, 0.0]
14	Threestage Phase 7	1.99	9.76	7.360	187	264	[22.5, 59.5, 15.0, 3.0, 0.0, 0.0]
15	Threestage Phase 8	2.15	9.42	7.360	194	274	[19.0, 54.0, 21.5, 4.5, 0.5, 0.5]

Table 1: Results for trial datasets. The photon energy is the central energy of the respective dataset. τ_{min} and τ_{max} are computed using the average spike width in Equations 2.1 and 2.2.

This latter value is then used to compute τ_{min} and τ_{max} as outlined in Section 2.2. Details of each run are

listed in the first column, and the full file names detailed in Appendix D. The "Spike number distribution"-column refers to how many spectra are registered as containing 1 spike, 2 spikes etc, here listed as percentage values. Since some spectra are registered as not containing any spikes, these values do not always add up to 100%.

The spike widths given are weighted. That is, for each spectrum, a relative weight has been put on each spike counted into the average width, depending on its amplitude. As such, the spikes with the greatest amplitudes are given the most significance when calculating the average width. We can conclude that the average number of spikes centers around 2 for each dataset, whereas the average spike width varies more. We clearly see that the pulses with twostage compression have relatively narrow spikes, implying longer pulse durations than the threestage compression runs. As expected, the widths found by the algorithm all imply pulse duration on the attosecond scale, where all estimates of both τ_{min} and τ_{max} are below 1 fs.

4.2 Efficiency

The goal is that the model can be used during running experiments, for example to verify pulse durations measured directly in temporal domain. Since beamtime is limited, the programme must be as efficient as possible. The average time taken for the programme to run through a dataset consisting of 200 spectra is about 1 minute. Each spectrum containing overlaps takes on average 0.35 seconds to re-fit. A similar version has been tested in Mathematica [24], taking roughly 3 minutes to analyse just 100 spectra. The Python algorithm being around 6 times faster than the Mathematica algorithm, it is made clear that Python is a much more suitable language for the purpose.

To speed up the process even more, the programme can easily be modified to run in parallel. Since it follows a hierarchical structure, the function in the pseudo-class can be duplicated, the dataset split into two, and the programme sent to process on two different cores. Alternatively, a GPU can also be considered for faster processing. Jupyter notebook integrates well with GPUs, and can thus be added directly at the level of the visual interface. This makes it simple to integrate the algorithm into an online system.

4.3 Possible improvements

While the model succeeds in finding appropriate spike widths by both detecting peaks, approximating their Gaussian distributions and re-fitting spectra containing overlapping spikes, there are still a few inaccuracies. The number of spectra containing overlaps is for example not decreased to zero for any of the datasets tested for. Furthermore, spikes detected to be relatively wide for some datasets also point towards a flaw with peak detection for noisy datasets.

The algorithm for treating overlaps can be improved by 1) having finer increments/decrements of the standard deviations and amplitudes, and 2) re-defining the algorithm all together. Currently, only few conditions are taken into account when fitting spectra with more than one overlap. While the readjustment of peaks does depend on the overlaps on either side, and whether neighbouring peaks have been made bigger or smaller, more feedback should be implemented.

The conditions for a spike to be detected using the peak finder are currently quite generous, since the algorithm is built to process any kind of dataset. This means that conditions, e.g. peak height, are in terms of percentages and not total height. This implies that spectra with very low intensities will still contain spikes according to the algorithm. In reality, such spectra sometimes contain mere noise data, yielding broad spike widths. If a certain intensity level can be expected for a given run of the experiment, absolute height

criteria can be included in the *parameters.py* class, filtering out spectra consisting only of noise. For datasets known to contain many more spikes than the ones used to test this model, the conditions for e.g. nearest neighbour distance could be loosened to make the model more apt for such datasets. It is also possible to reconstruct the pulse duration from the amount of spikes in a spectrum, since the number of spikes is inversely proportional to the pulse duration.

4.4 Conclusion

In conclusion, using the relation between the frequency and temporal domains, the model suffices to determine the temporal duration of XFEL pulses for datasets with moderate noise levels. It has been demonstrated that sub-femtosecond pulses are indeed generated and acquired. The noisier the spectra, however, the more uncertain the conclusions. This invites for more specific conditions in the peak detection and overlap processing parts of the algorithm, which will depend on attributes of the experiment itself, such as intensity expected and energy scale.

Acknowledgements

I would sincerely like to thank Dr. Andreas Adelmann, Dr. Alexander Malyzhenkov and Dr. Eduard Prat Costa for their feedback, support and guidance, and without whom I would not have been able to complete this project. I would furthermore like to thank Simon Koch and Dr. Pavle Juranic for useful technical discussions.

References

[1] Zhirong Huang and Kwang-Je Kim. "Review of x-ray free-electron laser theory". In: *Physical Review Special Topics - Accelerators and Beams* 10.3 (2007). DOI: 10.1103/PhysRevSTAB.10.034801.

[2] SwissFEL. *SwissFEL: how it works.* URL: https://www.psi.ch/de/swissfel/how-it-works. (accessed: 24.04.2020).

[3] C. Pellegrini, A. Marinelli, and S. Reiche. "The physics of x-ray free-electron lasers". In: *Rev. Mod. Phys.* 88 (1 Mar. 2016), p. 015006. DOI: 10.1103/RevModPhys.88.015006. URL: https://link.aps.org/doi/10.1103/RevModPhys.88.015006.

[4] HN Chapman, C Caleman, and N Timneanu. "Diffraction before destruction". In: *Phil. Trans. R. Soc. B* 369 (2014). DOI: 10.1098/rstb.2013.0313.

[5] Kenji Tamasaku et al. "Nonlinear Spectroscopy with X-Ray Two-Photon Absorption in Metallic Copper". In: *Phys. Rev. Lett.* 121 (8 Aug. 2018), p. 083901. DOI: 10.1103/PhysRevLett.121.083901. URL: https://link.aps.org/doi/10.1103/PhysRevLett.121.083901.

[6] C. Milne and T. Schietinger. "SwissFEL: The Swiss X-ray Free Electron Laser". In: *Applied Sciences* 720.7 (2017). DOI: 10.3390/app7070720.

[7] Y Inubushi et al. "Determination of the Pulse Duration of an X-Ray Free Electron Laser Using Highly Resolved Single-Shot Spectra." In: *Physical Review Letters* 109.14 (2012). DOI: 10.11103/PhysRevLett.109.144801.

[8] S. Huang et al. "Generating Single-Spike Hard X-Ray Pulses with Nonlinear Bunch Compression in Free-Electron Lasers". In: *Physical Review Letters* 119 (2016). DOI: 10.1103/PhysRevLett.119.154801.

[9] Rafael Abela et al. "The SwissFEL soft X-ray free-electron laser beamline: Athos". In: *Journal of Synchrotron Radiation* 26.4 (July 2019), pp. 1073–1084. DOI: 10.1107/S1600577519003928. URL: https://doi.org/10.1107/S1600577519003928.

[10] J. Rehanek et al. "The hard X-ray Photon Single-Shot Spectrometer of SwissFEL—initial characterization". In: *Journal of Instrumentation* 12.P05024 (2017). DOI: 10.1088/1748-0221/12/05/P05024.

[11] M. Makita et al. "High-resolution single-shot spectral monitoring of hard x-ray free-electron laser radiation". In: *Optica* 2.10 (2015). DOI: 10.1364/OPTICA.2.000912.

[12] P. Juranic et al. "Characterization of hard X-ray attosecond pulses at SwissFEL". In: *Research proposal* (available upon request).

[13] S. Bettoni et al. "Temporal profile measurements of relativistic electron bunch based on wakefield generation". In: *Phys. Rev. Accel. Beams* 19 (2 Feb. 2016), p. 021304. DOI: 10.1103/PhysRevAccelBeams.19.021304. URL: https://link.aps.org/doi/10.1103/PhysRevAccelBeams.19.021304.

[14] C. Behrens, F. Decker, and Y. et al. Ding. "Few-femtosecond time-resolved measurements of X-ray free-electron lasers". In: *Nat Commun.* (2014). DOI: 10.1038/ncomms4762.

[15] P. Emma et al. "Femtosecond and Subfemtosecond X-Ray Pulses from a Self-Amplified Spontaneous-Emission–Based Free-Electron Laser". In: *Phys. Rev. Lett.* 92 (7 Feb. 2004), p. 074801. DOI: 10.1103/PhysRevLett.92.074801. URL: https://link.aps.org/doi/10.1103/PhysRevLett.92.074801.

[16] Y. Ding et al. "Generating femtosecond X-ray pulses using an emittance-spoiling foil in free-electron lasers". In: *Applied Physics Letters* 107 (2015). DOI: 10.1063/1.4935429.

[17] Senlin Huang et al. "Generation of stable subfemtosecond hard x-ray pulses with optimized nonlinear bunch compression". In: *Phys. Rev. ST Accel. Beams* 17 (12 Dec. 2014), p. 120703. DOI: 10.1103/PhysRevSTAB.17.120703. URL: https://link.aps.org/doi/10.1103/PhysRevSTAB.17.120703.

[18] W. Helml et al. "Measuring the temporal structure of few-femtosecond free-electron laser X-ray pulses directly in the time domains". In: *Nature Photonics* 8 (2014). DOI: 10.1038/NPHOTON.2014.278.

[19] Project Jupyter. *Jupyter*. URL: https://jupyter.org/. (accessed: 25.04.2020).

[20] SciPy Community. *scipy.signal.butter*. URL: https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.butter.html. (accessed: 26.03.2020).

[21] SciPy Community. *scipy.signal.find_peaks*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html. (accessed: 26.03.2020).

[22] Matthew Newville and Till Stensitzki. *Non-Linear Least-Squares Minimization and Curve-Fitting for Python*. URL: https://lmfit.github.io/lmfit-py/. (accessed: 26.03.2020).

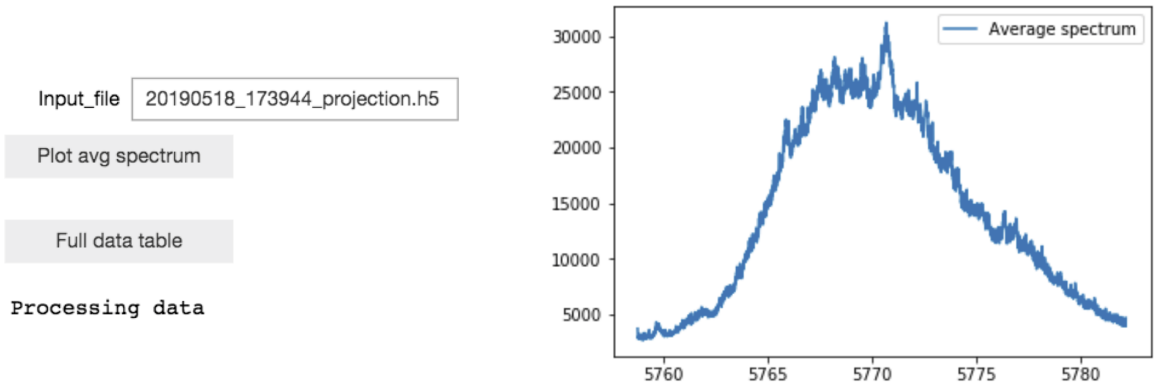
[23] Alexander Scheinker and David Scheinker. "Bounded extremum seeking with discontinuous dithers". In: *Automatica* 69 (July 2016), pp. 250–257. DOI: 10.1016/j.automatica.2016.02.023.

[24] Wolfram. *Wolfram Mathematica*. URL: https://www.wolfram.com/mathematica/. (accessed: 24.04.2020).

Appendices

A Visual interface

To apply the algorithm to a dataset and prompt analysis of the model's output, a visual interface is set up. It asks for an input file name, and a panel appears with options for plotting the average spectrum and getting a table with the summarised output. To plot the average spectrum is a relatively fast procedure. On the other hand, when "Full data table" is clicked, the algorithm is prompted, and the data will appear after all the data has been analysed. This takes roughly a minute, depending on the speed of the user's computer. The visual interface is shown, with the resulting plot and data table, in Figure 15.



Index	Value
Dataset	20190518_173944_projection.h5
# of 1,2,.. spikes	[25.,50.,24.]
single spike widths	1.3839898726334527
2-spike widths	2.893712135983803
overall spike widths	2.390471381533686
Average number of spikes in single spectrum	1.9898989898989898

Figure 15: The visual interface from which the algorithm is prompted.

B Programme architecture

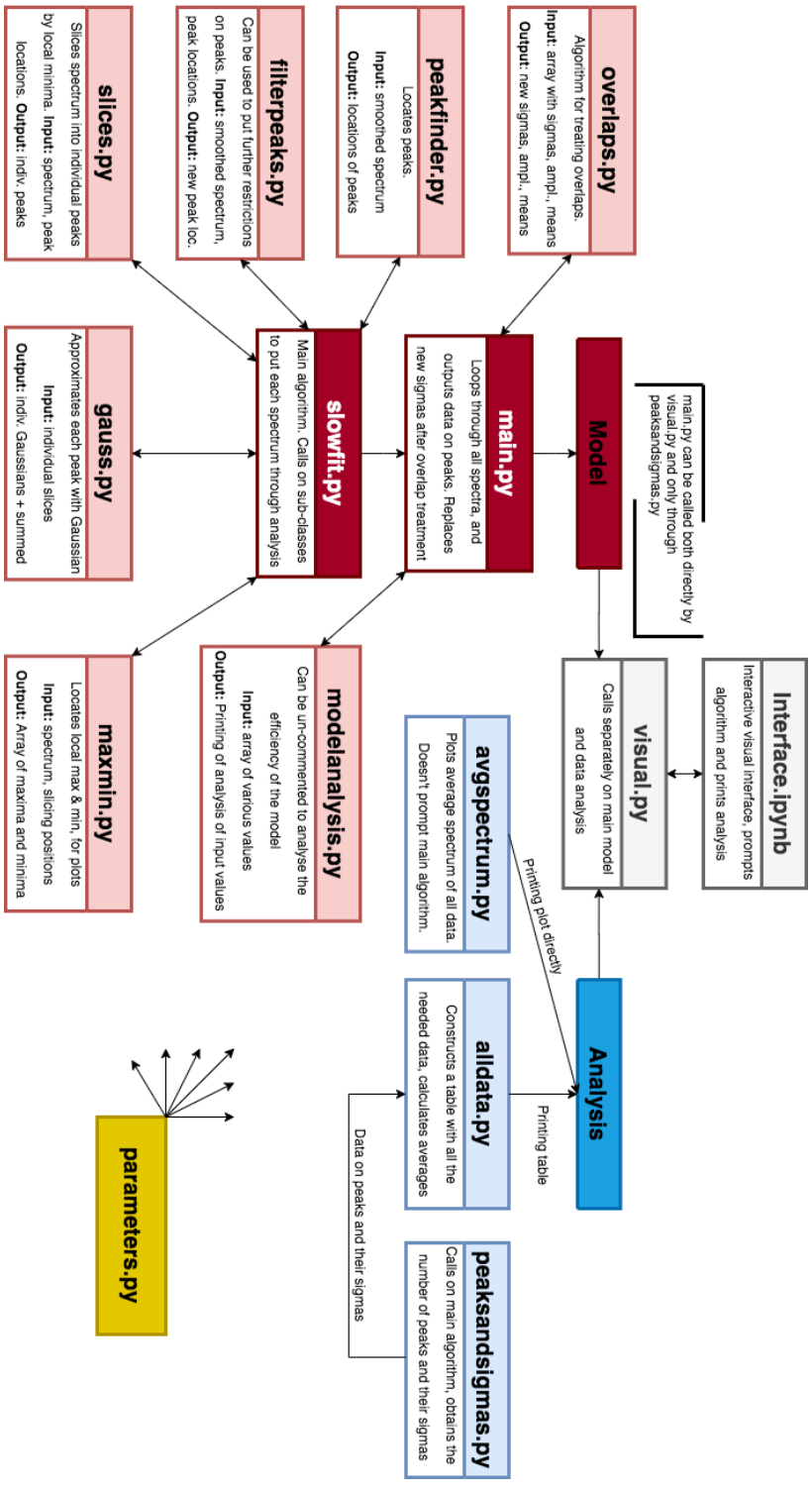
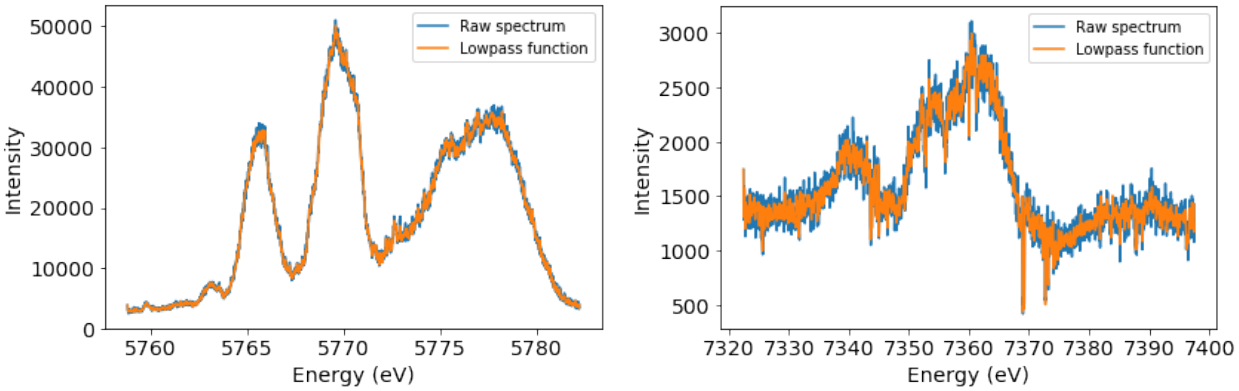


Figure 16: Programme architecture. In red: component for main algorithm. Blue: data analysis. Yellow: Parameter class

C Lowpass parameter tuning

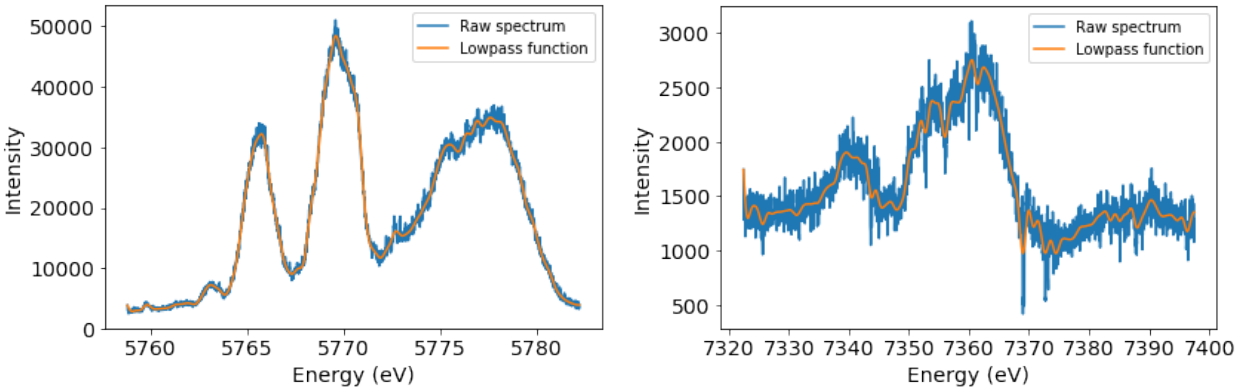
Shown below are the final smoothed spectra for different values of the α parameter. Examples are shown for two spectra coming from different datasets, to demonstrate how the lowpass filter acts differently for various noise levels. Subfigures (a) show the performance for a threestage non-linear compression run, whereas subfigures (b) show the performance for a threestage LINAC phase 0 run. α starts from a severe over-fit, and is increased to achieve more balanced lowpass functions. Figure 20 shows the performance of the filter with the final parameters included in the model.

As discussed in the main text, there is a fairly wide range of suitable values for the parameter α, and the chosen value provides enough smoothness in both cases, whilst preserving a few micro-structures.



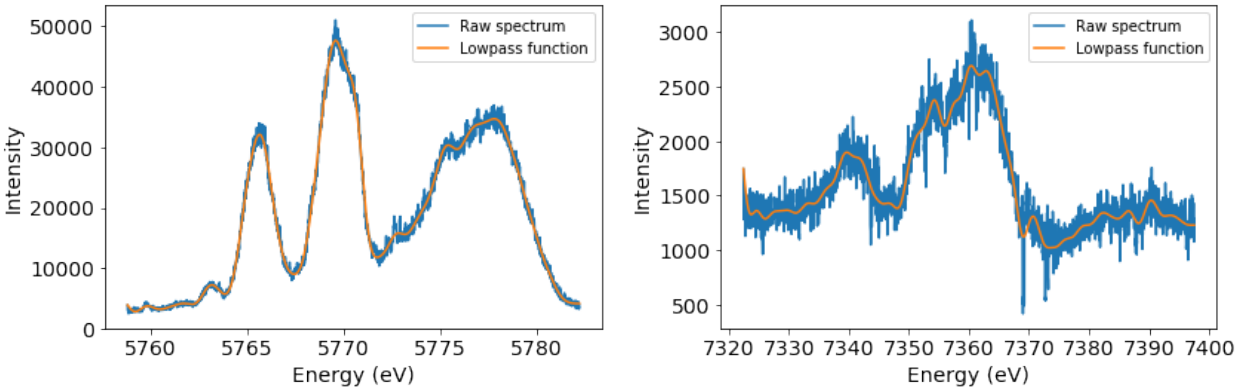
(a) (b)

Figure 17: $\alpha = 1$



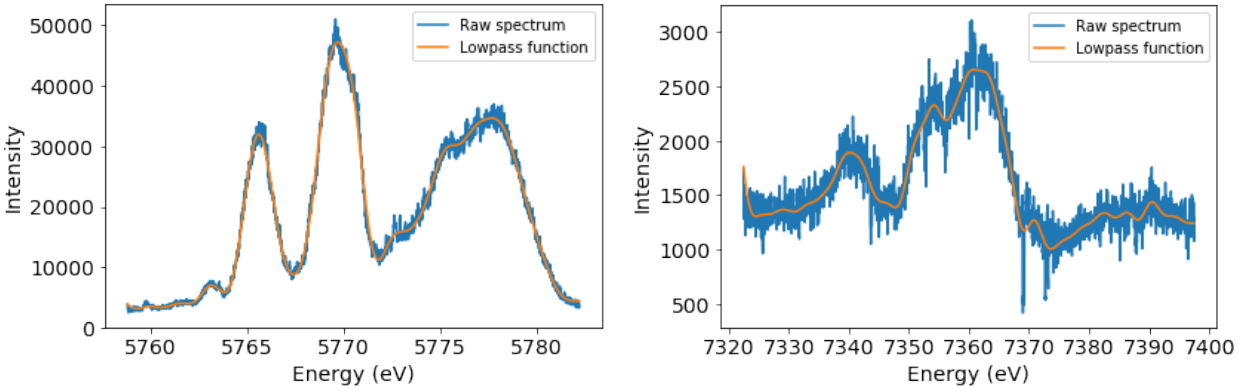
(a) (b)

Figure 18: $\alpha = 100$



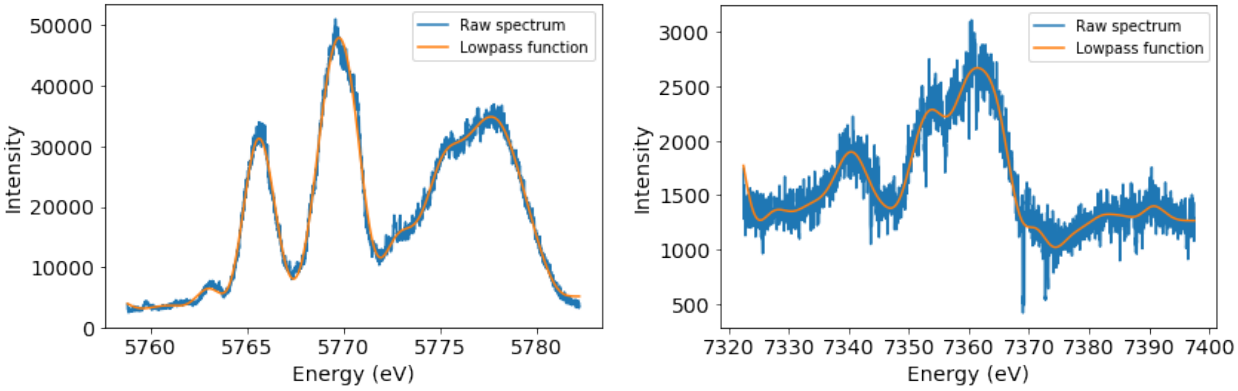
(a) (b)

Figure 19: $\alpha = 300$



(a) (b)

Figure 20: $\alpha = 700$



(a) (b)

Figure 21: $\alpha = 1000$

D Overlap data

In Table 2 is the data from analysing the model's performance on correcting for overlapping spikes. The r^2-values are normalised to 1=1,000,000,000. Also included is an overview of the names of datasets trialled

with the model (Table 3). The r^2-values differ, due to the re-fitting of the Gaussian profiles. We see that the new values are consistently higher than the old, which is expected. This is because the old values were minimised, and the generation of new distributions will yield a result that deviates from this.

Dataset index	Spectra with overlaps, initial (%)	Spectra with overlaps, final (%)	Old average r^2-value	New average r^2-value
1	42.0	10.5	0.325	0.337
2	35.5	7.5	0.685	0.714
3	46.5	13.5	0.825	0.865
4	57.5	16	0.933	1.014
5	63.5	22	0.668	0.696
6	36.0	8.0	20.5	21.7
7	61.5	15.5	0.148	0.150
8	39	6	0.227	0.230
9	34	3	0.372	0.376
10	27.5	5.5	0.484	0.489
11	34.5	5.5	0.479	0.491
12	41.5	9.5	0.291	0.301
13	43.5	10.5	0.222	0.231
14	44.5	12.5	0.161	0.165
15	59.5	23	0.111	0.114

Table 2

Index	Name of dataset
1	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase_0.h5
2	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase_1.h5
3	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase_2.h5
4	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase_3.h5
5	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase_4.h5
6	20190518_173944_projection.h5
7	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_0.h5
8	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_1.h5
9	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_2.h5
10	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_3.h5
11	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_4.h5
12	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_5.h5
13	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_6.h5
14	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_7.h5
15	2019-10-14_02-45-48_threestage_v1_PSSS_LINAC1_Phase_8.h5

Table 3: Dataset indices. The datasets are available on request at skjaer@student.ethz.ch.