Estimation of the average spike width of a typical FEL spectrum

Sara Kjær

April 2020

Contents

1 Introduction

In recent years, free electron lasers (FELs) have made significant developments in generating ultra-short and ultra-bright laser pulses, capable of probing entirely new energy scales. To produce such pulses, a beam of free electrons is accelerated close to the speed of light. It is then passed through an undulator, which is an array of magnets forcing the beam to move in a transverse sinusoidal pattern. Consequentially, the beam emits synchrotron radiation, which is amplified through mirrors placed on either side of the undulator, and pulses are emitted in periodic intervals. X-ray free electron lasers (XFELs), which are FELs operating in the X-ray regime, use this technique to generate pulses with a duration ranging down to the femtosecond scale. When such bright pulses become ultra short, it allows for very high peak intensities, creating conditions enabling a whole new range of experiments.

An attractive use of XFEL pulses is in so-called Diffraction before Destruction experiments. Here, a molecule or another kind of specimen is probed with the 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 signal and X-ray spectroscopy at unprecedented temporal scales. In other words, new experiments in many fields have been made possible with the conditions created at XFELs.

The SwissFEL laser [1] is an XFEL located at the Paul Scherrer Institute in Villigen, Switzerland. Recently constructed, it opened up for user experiments in 2018, and is with its 740 metres of beamline an attractive tool for experiments in various fields. It currently lases at 1 to 70 Å and, similarly to most other XFELs worldwide, produces pulses of duration down to a few femtoseconds. To further expand the experimental opportunities that come with the conditions created at XFELs, the aim is to extend its temporal range down to the attosecond scale, that is below 1 fs. Where this is entirely possible at SwissFEL, it has proven extremely difficult to directly measure the duration of pulses in this regime due to restrictions coming from the experimental set-up itself. Characterisation of the temporal profile of the pulses is nonetheless necessary in order to understand the conditions created for a given experiment.

A previously used method to determine pulse duration has been to analyse the spectral information in the frequency domain. That is because the width of a spike in the frequency domain is directly related by a Fourier transform to the duration of the pulse. This method was first demonstrated by Inubushi et al. [2], and can, if widths are accurately measured, reconstruct the temporal profile of the pulse. There are however a few shortcomings of this method. For one, it is a difficult process to automate due to a wide variety in features across spectra. Secondly, information regarding the energy chirp of the pulse is lost when using this method, and it is therefore always desirable to measure the temporal profile directly. In attempting to expand measurements to the attosecond scale, results can be cross-checked using the spectral method. It is therefore the objective of this project to develop a robust algorithm capable of determining the number of spikes in a spectrum, and their average widths, as to provide verification of the direct measurements.

This report outlines the physics behind the problem, the steps taken in estimating average spike widths, and the structure of the final algorithm. Provided are also provisional results for spectral datasets from past runs, on which the model was tested. The goal is that the algorithm be implementable in an experiment aiming to measure pulses at the attosecond scale. Restricted beamtime demands for a highly efficient model, which can be easily used by the experimenter. It is therefore taken into consideration how fast the algorithm processes data, whether it can run in parallel on multiple 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 for hard X-ray radiation and a newly built one for soft X-ray radiation. Measurements of sub-femtosecond pulses will take place in the hard X-ray beamline, which is also where all data used to test the algorithm comes from. It emits radiation at 6-7 keV, and its spectral information is recorded by a photon single-shot spectrometer (PSSS), which is placed 59 metres after the laser's undulator. The PSSS consists of two main parts: a thin diamond which splits off the first order light, and a regular crystal spectrometer characterising the remaining light by diffracting it through a piece of crystal. This gives energy spectra with spikes occuring at around 4-8 keV, depending on the radiation strength used in the experiment [1].

Aside from the spectrometer, there are two other ways to record the pulse duration: using a passive streaker or directly streaking the photon pulses. The former is the method which will be used in the upcoming experiment. A passive streaker has been installed at SwissFEL after the undulator line, which can detect sub-femtosecond pulses generated by slotted foil. Slotted foil is an alternative to compression of the electron beam, where both techniques have the purpose of making the pulse short. More information on the experimental set-up required to produce and detect sub-femtosecond pulses at XFELs can be found in [3]. The full layout of the SwissFEL accelerator is shown in Figure 1.

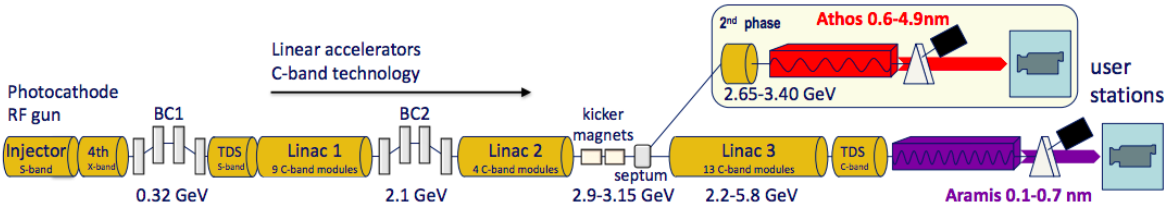
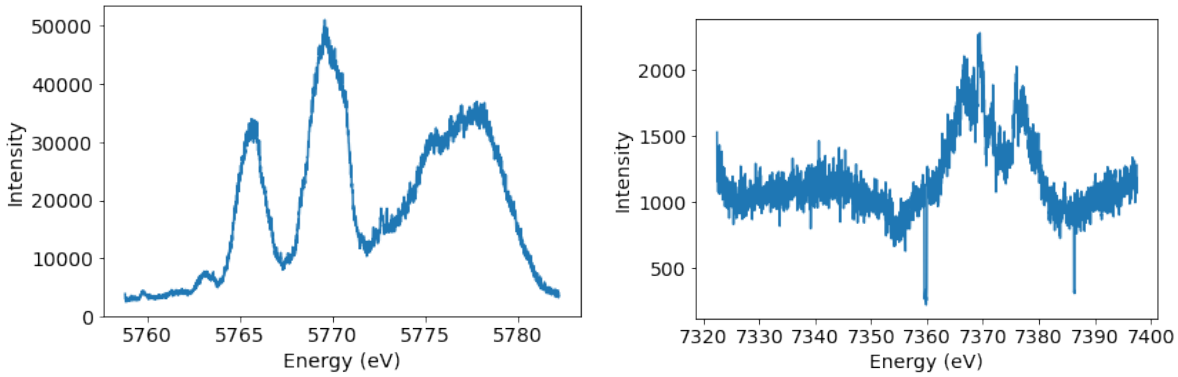


Figure 1: The accelerator layout of SwissFEL, graphics from [1]

2.2 Energy spectra

The energy spectra produced by the spectrometer are recorded and stored. Two examples of spectra from different experimental runs are shown in Figure 2. It is clear to see from this figure that spectral features vary widely between spectra, due to different experimental conditions. Where Figure 2a has quite distinct features and clearly contains 3 spikes with FWHMs around 2-5 eV, the properties of the spectrum in Figure 2b are more unclear. It could be the case that what we see is one very broad spike, but we may just as well interpret it as 4 narrow spikes that significantly overlap with one another. The difference in noise levels and relative intensity we see between the spectra give rise to the indistinct features of Figure 2b, which immediately poses another challenge for algorithm construction. Since noise is an unavoidable property of the spectra, classification of spectra becomes very difficult, even when they are inspected manually. An algorithm's conclusions therefore depend on the definitions of e.g. a 'spike' we feed to it, which also gives rise to difficulty in cross-checking results.

This issue arises mostly for noisy datasets: since features are easy to verify for spectra like that in Figure 2b,



(a) Typical low-noise spectrum

(b) Typical high-noise spectrum

Figure 2: Typical XFEL energy spectra from different runs with different energy scales. Intensity is relative, hence the vertical axis is given in arbitrary units.

constructing a reliable algorithm for a dataset with similar spectra would be easy. General features of entire datasets emerge when comparing average spectra across datasets. In Figure 3, the average spectra of 9 different threestage LINAC runs are plotted (phase 0-8, starting from below). It is clear to see that phases 2-5 would be the datasets easiest to construct a verifiable algorithm for. These average spectra have more universal features, which indicates low noise levels and high relative intensity. 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 purposes of attempting measurements at the attosecond scale, the most suitable datasets can be selected for comparison.

2.3 Overlapping spikes

As mentioned in the context of Figure 2b, some spectra contain overlapping spikes. Assuming that spikes follow a roughly Gaussian profile, spikes are classified as overlapping when the Gaussian approximations of individual spikes intercept considerably above zero. A raw spectrum with Gaussian approximations with significant overlap is plotted in Figure 4. Clearly, when modelling the spikes on Gaussian profiles, the sum of all individual Gaussian plots should approximately add up to the full spectrum. In this case, the approximations have been accurate, and the widths of the Gaussian plots will be comparable to the widths of the spikes in the raw spectrum. With overlaps, however, discrepancies between the added Gaussian profiles and the raw spectrum arise. Consequently, widths are not comparable. These cases must be taken into account in the construction of the algorithm, and adjustments must be made to correctly measure the widths.

2.4 Relation to temporal domain

Spike width and pulse duration are related by a simple Fourier transform. That implies that the shorter the spike width, the longer the pulse duration, and vice versa [2]. Since information on the chirp is lost when using the spectral method to measure pulse duration, the pulse duration can only be reconstructed within the boundaries of zero-chirp and maximum chirp. We observe the minimum duration, τ_{min}, in the absence of a chirp, and the maximum possible duration, τ_{max}, in the case of the maximum chirp. The formulae used

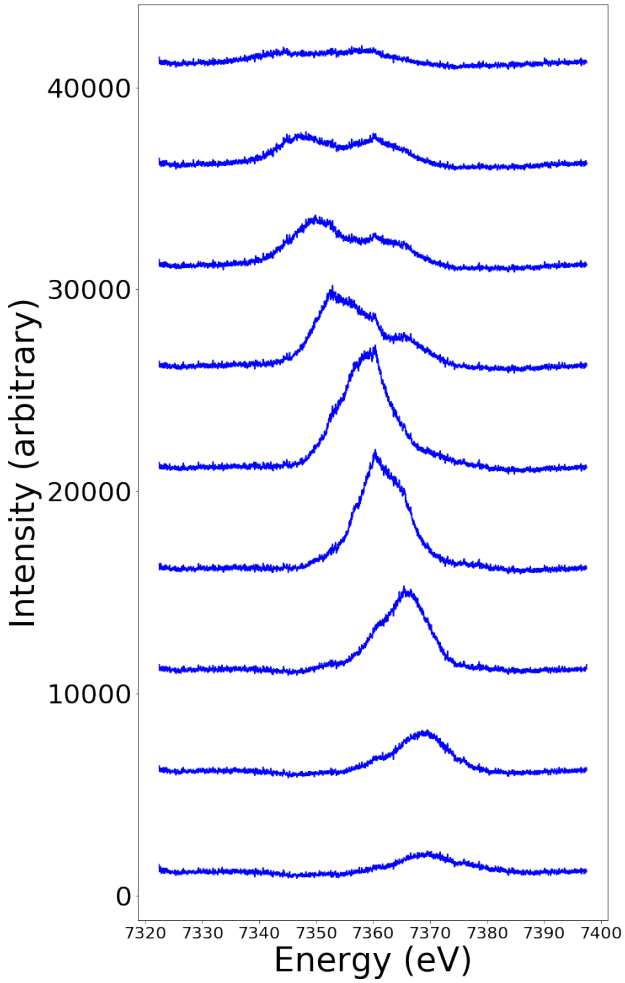


Figure 3: Average spectra for a threestage LINAC run with phases 0-8 (starting from below)

to deduce minimum and maximum pulse duration are:

$$\tau_{min} = \frac{4\hbar \log 2}{f_p}$$

$$\tau_{max} = \frac{4\sqrt{2}\hbar \log 2}{f_p}$$

where f_p is the weighted average spike width in eV, and τ_{min} and τ_{max} are computed in seconds, conventionally subsequently converted to attoseconds.

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, which allows the user to input a file and prompt analysis of spike data. The model is

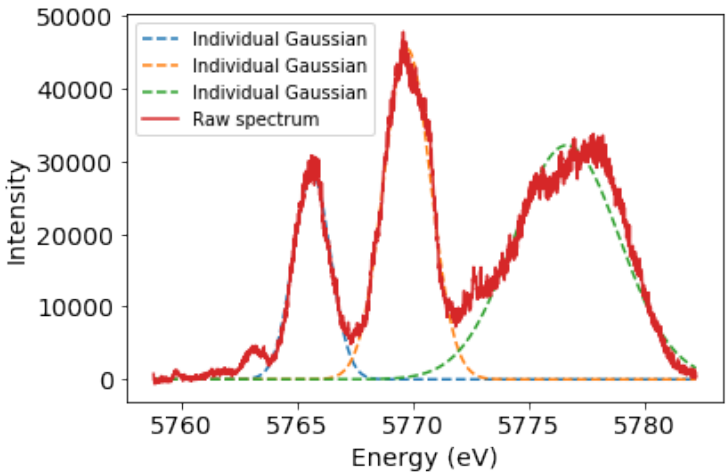


Figure 4: Gaussian approximations to spikes

based on spectral data from previous SwissFEL runs, each dataset containing between 100 and 1000 spectra, compressed into 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 pseudo-class. Parallel to the model itself are classes for data analysis. These are prompted from the visual interface (Appendix A), and returns visual outputs to the user. An overview of the entire architecture of the model can be found in Appendix B.
Following from the discussion in section 2.2, the model depends on some user-defined parameters for spike identification. These include required peak prominence, nearest neighbour distance, peak height, parameters for data filtering, and condition for classifying spikes as overlapping. The following parameters have been found to be suitable for all datasets on which the model has been tested:

- Minimum spike height: 20% of maximum spike intensity
- Minimum spike prominence: 10% of maximum spike intensity
- α parameter for data filtering: 1
- β parameter for data filtering: 700
- Lowpass filter degree: 5
- Overlap condition: 5% of maximum spike intensity

The use of these parameters will become clear in the following section. Parameters have been made explicitly tunable through the class *parameters.py*, and can thus be adjusted to make the model more or less sensitive.

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 2, 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 [4], which takes two parameters: degree and cutoff frequency. By

inspection, suitable values for a standard dataset were initially found at a order of 5 and a cutoff frequency of 0.005. An example of a spectrum and its filtered spectrum is plotted in Figure 5. Here, the noise is also plotted, and the smoothed spectrum has been shifted to have its minimum at zero, as to increase the accuracy of subsequent normal approximations.

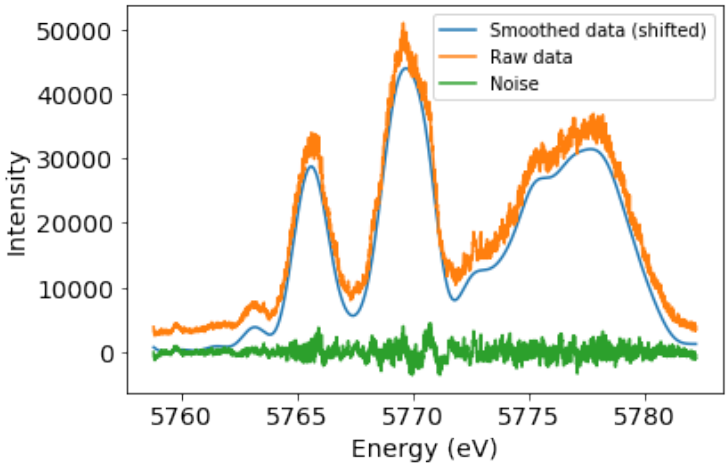


Figure 5: A raw spectrum plotted with its lowpass function and noise

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 filtered spectrum for noisier datasets. A feedback loop is therefore implemented to optimise the parameter for each spectrum. The function was optimised based on a parameter, $\bar{r}$, which is defined as follows:

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

where α and β are tunable parameters, 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 between the raw and smoothed spectra with the horizontal noise of the smoothed data, as to account for overfitting. The loop starts with a very low cutoff frequency of 0.0001, giving a severly underfit lowpass function, and iteratively increases the cutoff by 0.001 until $\bar{r}$ reaches a minimum. α and β are respectively set to 1 and 700, which is appropriate for all datasets the model has been tested on. Figure 6 illustrates the iterative process, in this case yielding an optimised cutoff frequency of 0.05.

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 model.

3.3 Peak finding

The next step of the algorithm is to identify local maxima and determine which qualify as spikes. This is done in two steps. First, a standard SciPy peak finder [5] is applied 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

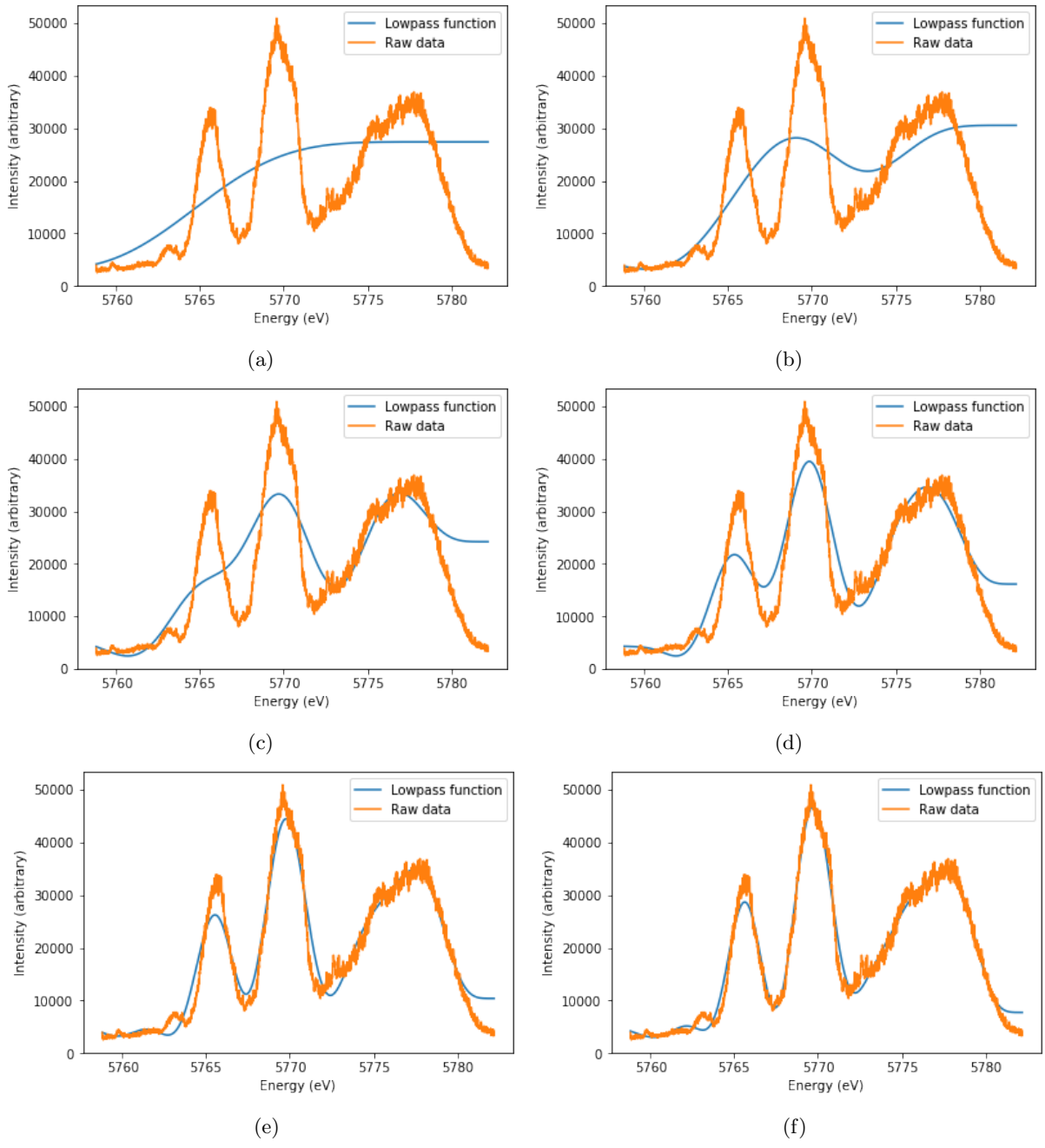


Figure 6: Progressive fitting of an energy spectrum under tuned lowpass parameters

such as distance from edge of spectrum can be specified. This class is easy to manipulate, and can thus be tuned for a more or less sensitive algorithm.
To prepare each spike for Gaussian approximation, the local minima 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

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 was used [6], 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 average rms standard deviation value over the entire spectrum. A filtered spectrum is plotted in Figure 7, with individual Gaussian approximations, along with detected minima and maxima also indicated.

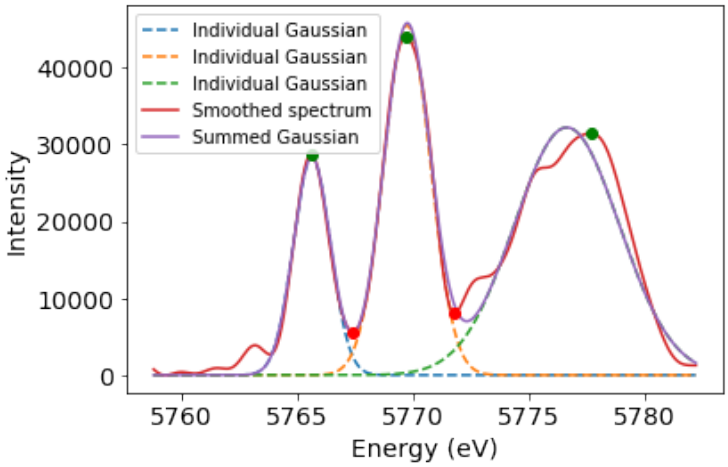


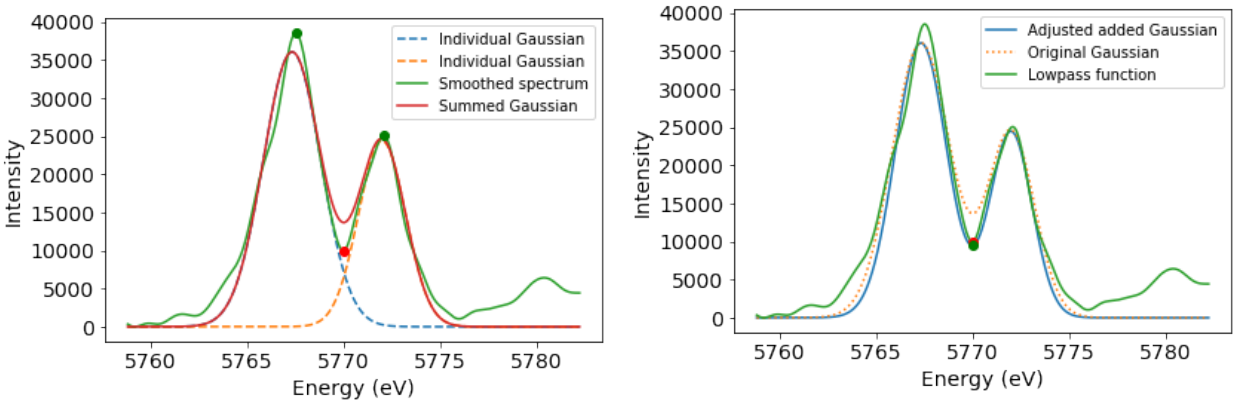
Figure 7: Gaussian approximations

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. 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.

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 the filtered spectrum, the

sigmas and amplitudes of the individual functions are decreased by 1% for each iteration. For spectral minima higher 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 8 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 16b provides the more accurate fit, and hence the more accurate spike widths.



(a) Original spectrum and approximations

(b) Adjusted Gaussian approximation

Figure 8: Overlapping spectra

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.

4 Results and discussion

4.1 Conclusions about pulse durations

Concretely, for the datasets on which the model has been tested, the conclusions are summarised in Table 1. Here, the average number of spikes per spectrum for the dataset, and their average widths are listed. This latter value is then used to compute τ_{min} and τ_{max} as outlined in Section 2.4. The "Spike number distribution"-column refers to how many spectra are registered as containing 1 spike, 2 spikes etc. The names of the datasets corresponding to each index are listed in Appendix D.

Dataset index	Average number of spikes	Average spike width (eV)	τ_{min} (as)	τ_{max} (as)	Spike number distribution ([1, 2, ..., n])
1	1.99	2.39	764	1080	[25, 50, 24]
2	2.01	1.99	917	1297	[53, 98, 44, 3, 0, 1]
3	2.03	2.53	721	1020	[51, 97, 48, 4]
4	2.31	1.98	922	1304	[26, 94, 71, 8]
5	2.53	1.91	956	1352	[23, 76, 74, 24, 2]
6	2.76	2.28	800	1131	[6, 73, 88, 29, 4]
7	2.16	4.30	424	600	[59, 69, 48, 12, 5, 1]
8	1.75	4.67	391	553	[91, 74, 28, 5, 1]
9	1.64	2.14	853	1206	[103, 73, 17, 5, 0, 1]
10	1.49	2.72	671	949	[119, 67, 11, 3]
11	1.68	2.91	627	887	[95, 74, 28, 2]
12	1.92	4.70	388	549	[60, 100, 36, 4]
13	1.88	2.66	686	970	[56, 117, 23, 4]
14	1.99	5.14	355	502	45, 119, 30, 6]
15	2.15	5.26	347	491	[38, 108, 43, 9, 1, 1]

Table 1: Model's conclusions for trial datasets

We can conclude that the average number of spikes centers around 2 for each dataset, whereas the average spike width varies more. More noisy datasets, i.e. low or high phases tend to have broader spikes, which could indicate that some features that are registered as spikes are in reality noisy datasets, which have slight "bumps" over a wide part of the spectrum. As discussed in Section 2.2, the most reliable datasets for the purpose of the experiment are the middle phases (datasets 4-5 and 10-13), which here show more consistency in spike width and number of spikes. As expected, the widths found by the algorithm imply pulse duration on the sub-femtosecond scale.

4.2 Model evaluation

Evaluating how accurately the model predicts spike widths essentially lies in the evaluation of 2 key parts of the model: the adaptive lowpass function, and the procedure for treating overlapping spectra. The lowpass function is optimised on a parameter that balances the r^2 value with horizontal smoothness. Since the desired balance must be determined by the designer of the model, there is no way to quantitatively optimise the balancing parameters α and β, and they must be tested visually. Various combinations of these parameters have therefore been applied to all datasets, and their plots inspected. The most appropriate combination found was $\alpha = 1$ and $\beta = 700$. The plots for some combinations are shown in Appendix C.

The second point of evaluation is the overlap procedure. The accuracy of this step is evaluated by comparing how many spectra were initially classified as overlapping, to how many spectra still do not satisfy the condition for maximum difference between minima. For spectra with a single overlap, all spectra will be within the condition, since the difference is only optimised at one point. Some spectra with more overlaps, however, are not re-fitted within the condition, since optimising over multiple minima proves a much more

difficult task, as discussed in Section 3.5. The algorithm to treat spectra with overlaps has been gradually improved, moving from a prototype effectively re-fitting 77% of spectra from a trial dataset, to the final version re-fitting 86% from the same dataset.
Looking at the performance for all available datasets, a clear decrease in the average percentage of spectra with overlaps is seen, going from 44.4% to 11.2%. 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 $r2$ value increases. A full overview of the effectiveness of the overlap treatment is given in Appendix D. This data is generated using the *modelanalysis.py* class.

4.3 Efficiency

The goal is that the model can be used during experiments to cross-check experimental data, such that the machine can be immediately calibrated should there be disagreement. 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 1.002 minutes. Each spectrum containing overlaps takes on average 0.35 seconds to re-fit. A similar version has been tested in Mathematica, taking roughly 3 minutes to analyse just 100 spectra, making it 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.

4.4 Adjustments

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.
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 for 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.
In conclusion, the model suffices to determine the temporal duration of XFEL pulses for datasets with mod-

erate noise levels, using the relation between the frequency and temporal domains. It has been demonstrated that sub-femtosecond pulses are indeed registered. 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 Andreas Adelmann, Alexander Malyzhenkov and Eduard Prat Costa for their advise, feedback and support during this project.

References

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

[2] 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.

[3] 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.

[4] 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).

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

[6] 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).

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, which will be processed automatically when inserted. Once the algorithm has run over the entire dataset, a panel appears with options for plotting the average spectrum (Figure 9), and getting a table with the summarised output (Figure 10).

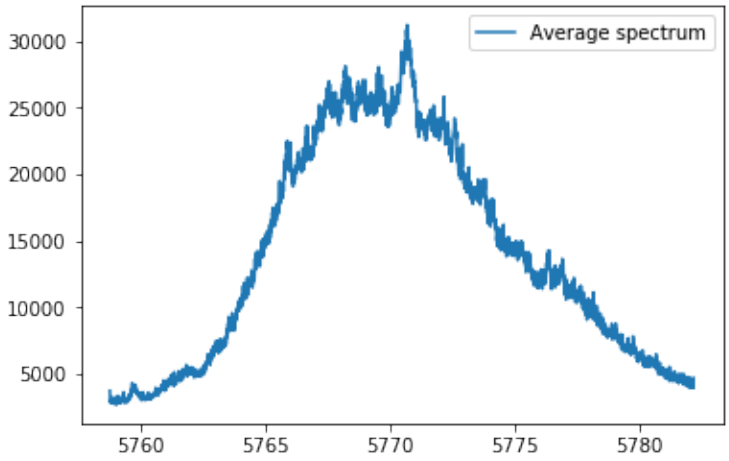


Figure 9

Index	Value
Dataset	20190518_173944_projection-Copy2.h5
# of 1,2,.. spikes	[14.,43.,36., 7.]
single spike widths	1.3779763499632147
2-spike widths	2.8816228338072145
overall spike widths	2.602901453311942
Average number of spikes in single spectrum	2.36

Figure 10

B Programme architecture

The diagram below (Figure 11 provides an overview of how classes are inter-connected in the model.

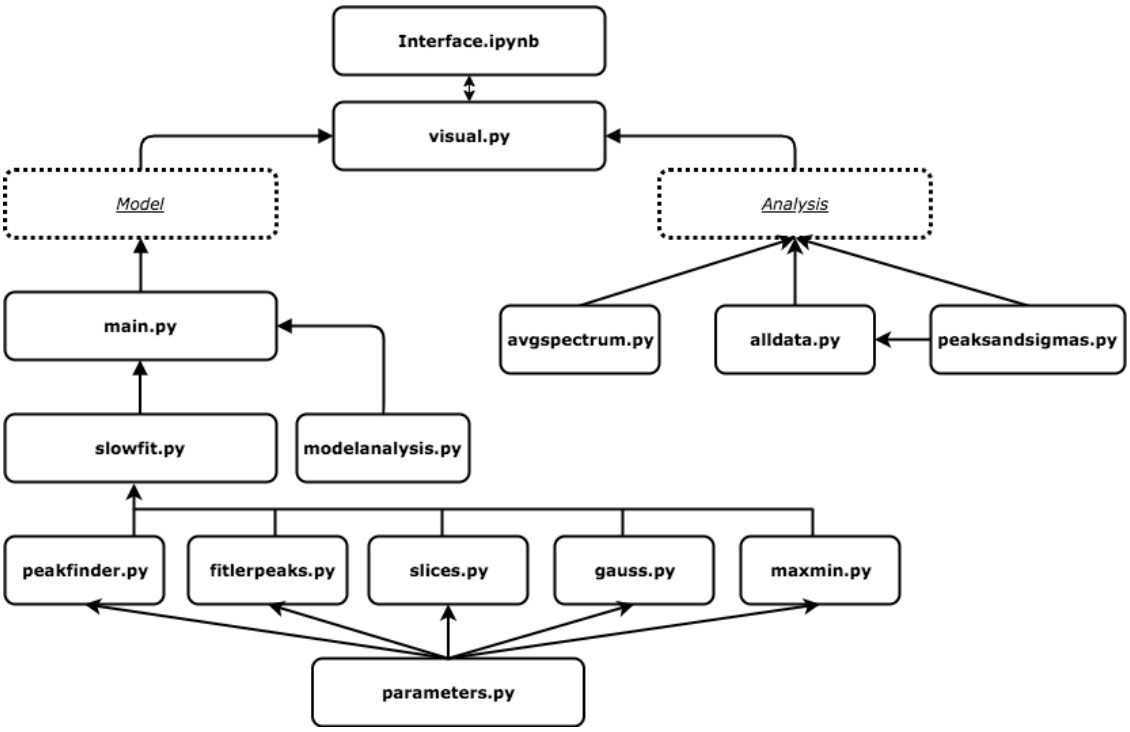
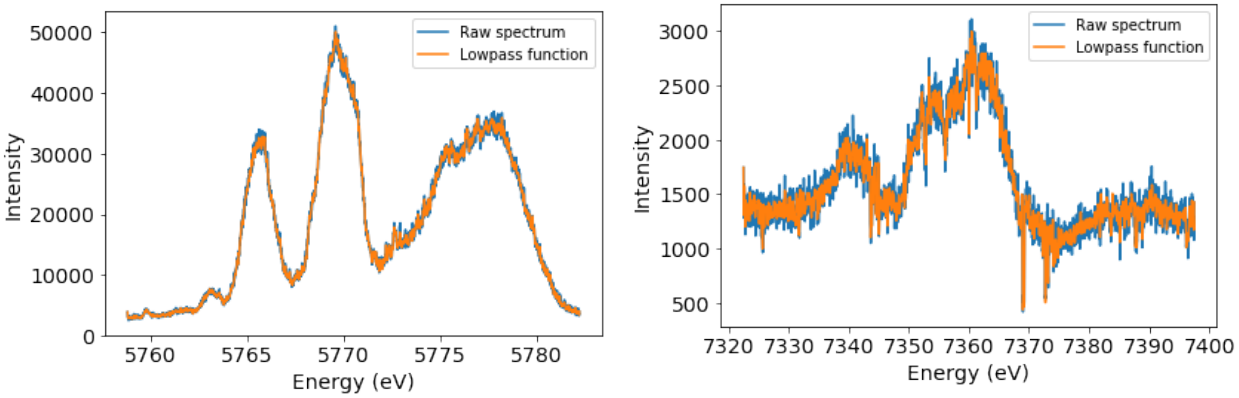


Figure 11: Programme architecture

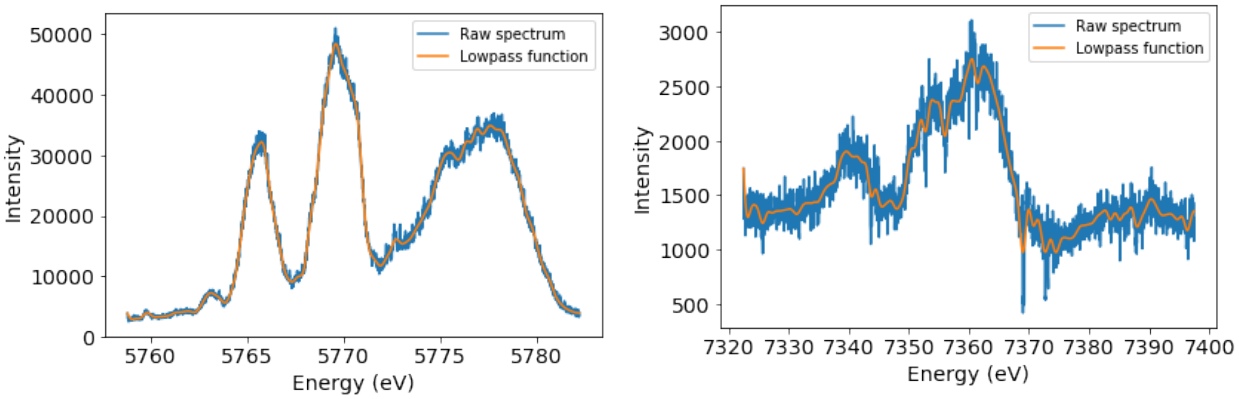
C Alpha and Beta tuning

Shown below are the final smoothed spectra for different combinations of the α and β parameters. Examples are shown for two spectra coming from different datasets, to demonstrate how the lowpass filter acts differently for various noise levels. With α consistently kept at 1, β is varied, starting from a severe over-fit, and increased to achieve more balanced lowpass functions. Figure **??** shows the performance of the filter with the final parameters included in the model.



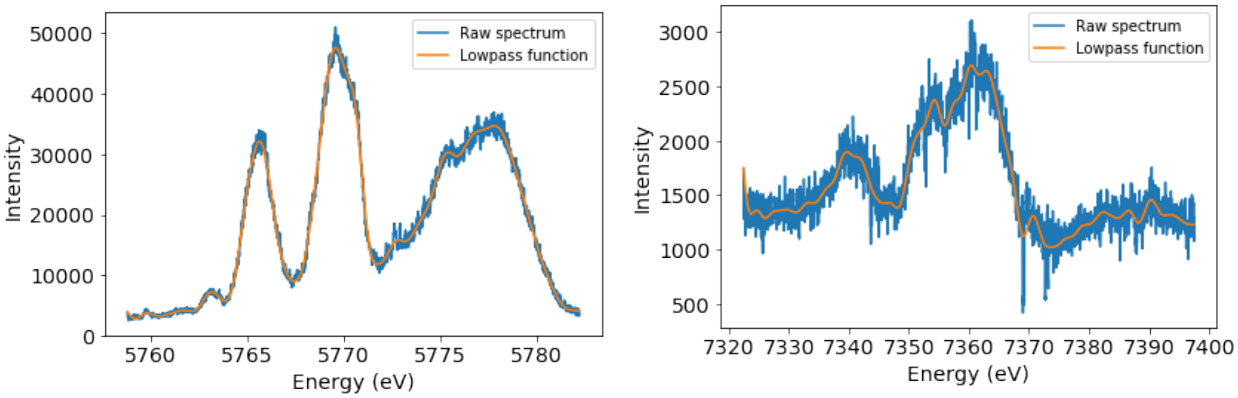
(a) (b)

Figure 12: $\alpha = \beta = 1$



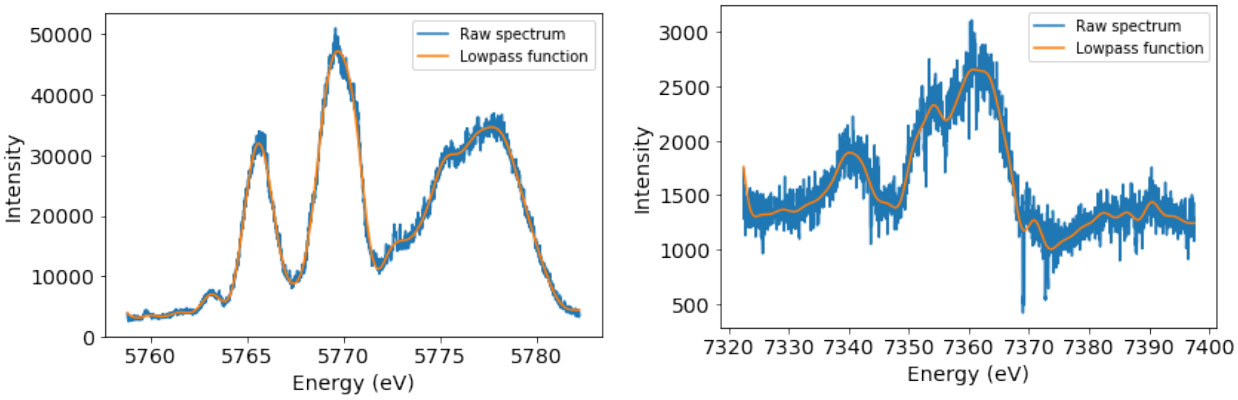
(a) (b)

Figure 13: $\alpha = 1, \beta = 100$



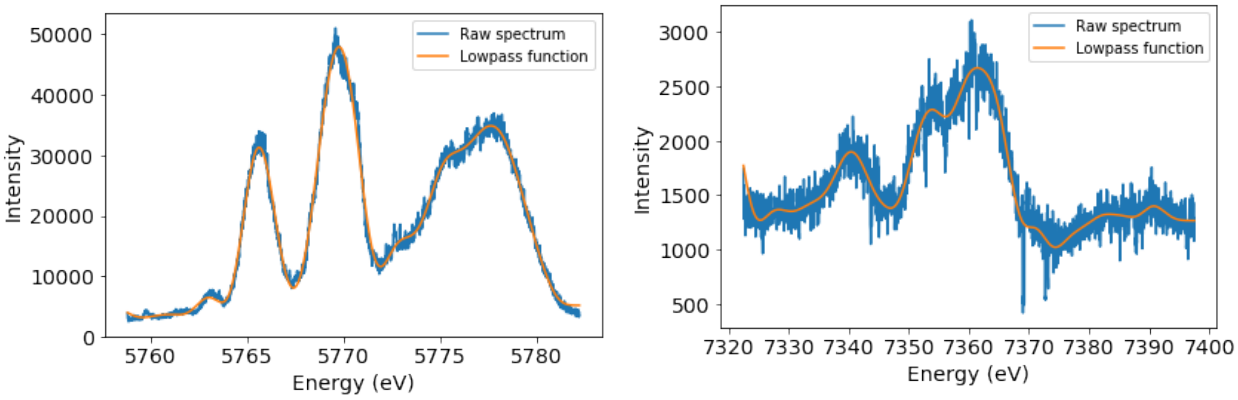
(a) (b)

Figure 14: $\alpha = 1, \beta = 300$



(a) (b)

Figure 15: $\alpha = 1, \beta = 700$



(a) (b)

Figure 16: $\alpha = 1, \beta = 1000$

D Overlap data

In Table 2 is the data from analysing the model's performance on correcting for overlapping spikes. Also included is an overview of the names of datasets trialled with the model (Table 3.

Dataset index	Spectra with overlaps, initial (%)	Spectra with overlaps, final (%)	Old average r2-value	New average r2-value
1	36.0	8.0	20495749478	21733648228
2	42.0	10.5	325452238	337104831
3	35.5	7.5	684979236	714028050
4	46.5	13.5	824670522	864847196
5	57.5	16	932620316	1013755905
6	63.5	22	667680201	696010474
7	61.5	15.5	147529429	150336436
8	39	6	226829116	229752368
9	34	3	371915340	376143301
10	27.5	5.5	484132440	489159773
11	34.5	5.5	478955852	490812397
12	41.5	9.5	290891330	301345286
13	43.5	10.5	222002780	230768008
14	44.5	12.5	160867437	164868670
15	59.5	23	110750966	113805113

Table 2

Index	Name of dataset
1	20190518_173944_projection.h5
2	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase$_0$.h5
3	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase$_1$.h5
4	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase$_2$.h5
5	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase$_3$.h5
6	2019-10-14_00-12-26_twostage_v3_PSSS_LINAC1_Phase$_4$.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