Improving EDS

Brynjar Morka Mæhlum

November 24, 2022

Contents

Chapter 1

Introduction

The main goal of this project is to improve EDS analysis. There are multiple ways to do this. One way is to make the analysis more transparent, which would make it easier to understand and use. A second option is to improve the input parameters of the analysis by control checking the instrument with a known sample. A third way is to make the quantitative analysis more accurate, which would improve EDS analysis. In this project, the main focus have been trying to improve the transparency of the analysis. Thus, the problem statement was formulated as:

> Problem statement: **How can the transparency of the steps in the EDS analysis be improved, so that the analysis is easier to understand and use?**

With this problem statement, the following sub-problems were formulated:

1. How accurate it the out-of-the-box quantification in AZtec and HyperSpy?
2. What are done with the data at the different steps in the analysis when using HyperSpy?
3. How can the peaks and the background be modelled in a way that is easy to understand?
4. How is the spectrum calibrated, and is AZtec different than HyperSpy?
5. How does different background models affect the quantitative analysis done in HyperSpy?
6. When does the analysis fail, both in AZtec and HyperSpy?

(Question for Ton: Should I have a short paragraph here about the status of EDS analysis today?)

One of the main problems with the existing software for EDS analysis is that they are like black boxes. The manufacturer of EDS sensors, like EDAX, Hitatchi, Thermo Fisher Scientific, and Oxford Instruments **(Question for Ton: Cite this?)**, provide their own software for analyzing EDS data. These software packages are black boxes, because the code is hidden, and the users does not know what is happening inside the software. In other words, the user pushes

some buttons to start the analysis of their sample, and then the software does some "magic" to analyze the data and produce some results. The user has few options to change the analysis to fit their needs. Many users tend to accept the results **(Question for Ton: Need to cite something here?)** from the software without questioning them, even though the analyzation "magic" differs between software packages and might be unreliable. The manufacturer Hitachi have trouble with separating Cs from ??, and their solution is to neglect the existence of Cs **(Brynjar: Find citation)**. The manufacturer Oxford Instruments provides the software AZtec which have trouble with ??, and their solution is ?? **(Brynjar: Find something, eg. zero peak)**. Even if the software is not wrong, the user might not understand the results, and the user have no way to change the analysis to fit their needs. Some might say that using e.g. AZtec is EDS analysis for dummies. One could solve this problem with an increase in transparency, because that would make it easier to understand EDS analysis and easier for users to adapt the analysis to their own needs.

(Brynjar: Paragraph about Dispersion, offset, energy resolution?)

(Brynjar: Paragraph about Other parameters of EDS analysis?)

(Brynjar: Paragraph about Improving quantitative EDS analysis?)

Chapter 2

Results

2.1 Introduction

The results are presented in this chapter. The sections follow the structure of the sub-problems of the main problem statement formulated in Chapter 1.

2.2 Initial quantification

The initial quantification was done on the data from the GaAs wafer in AZtec and in HyperSpy as out-of-the-box as possible. The results are presented in Table 2.1. The wafer is a 1:1 alloy of gallium and arsenic, so the atomic percent of Ga and As should be 50% and 50% respectively.

Table 2.1: Initial quantification of the GaAs wafer. The ratio in the wafer is 1:1, so the correct ratio is 50% and 50%, because the results are in atomic percent. **(Brynjar: Put in the actual results here. Use both HyperSpy linear and model fitted results?)**

V_{acc}	AZtec		HyperSpy	
	Ga	As	Ga	As
5 kV	50 %	50 %	50 %	50 %
10 kV	50 %	50 %	50 %	50 %
15 kV	50 %	50 %	50 %	50 %
30 kV	50 %	50 %	50 %	50 %

2.3 Analysis steps in HyperSpy

(Question for Ton: Is this interesting to write about? The problem here is that the text is both method, some results and kind of discussion. What do I do with that? I want to keep it, but also restructuring it.)

The next sub-problem was to find out what is done with the data at the different steps in the analysis when using HyperSpy. In these steps it is assumed that the user have done qualitative analysis and want to do quantitative analysis on a set of elements. The analysis in AZtec is done as a black box, so it is not possible to see what is done with the data at the different steps. All variables inside croccodile need to be set by the user, e.g. `<element_list>` would be set to `['Ga', 'As']` for the GaAs wafer. An example notebook with quantification of the GaAs wafer is attached in APPENDIX. **(Brynjar: Make a notebook with GaAs quantification in HyperSpy, with the data somehow.)**

2.3.1 Loading the data and specifying the elements

```
s = hs.load(<filepath>, signal="EDS_TEM")
s.set_elements(<element_list>)
```

The first step in the analysis is to load the data as a HyperSpy `signal` type, and specifying the signal as TEM. The `signal` type is a class in HyperSpy that contains the data and the metadata, and it has methods for analysis. The `signal` type must be specified as TEM, because the `signal` type for SEM is very limited and does not have a method for quantification. When using .emsa files from AZtec, as is done in this project, the metadata contains some relevant and some irrelevant information. The information relevant later in this project is: acceleration voltage, dispersion, zero offset, energy resolution Mn $K\alpha$, After loading, it is possible to plot the data with `s.plot()`.

2.3.2 Removing the background linearly

```
bw = s1.estimate_background_windows(windows_width=<number>)
iw =  s1.estimate_integration_windows(windows_width=<number>)
```

The next step is to remove the background, which with the above code is done by a linear fit. The background can be removed through model fitting, which is covered in Section 2.3.4. The variable `windows_width` sets how wide the windows are for the background and integration, measured in FWHMs. A good starting value for `windows_width` is 2, but it should be tested by the user with a plot to see if the background will be removed correctly. The estimated windows can be plottet with:

```
s.plot(xray_lines=True, background_windows=bw, integration_windows=iw)
```

2.3.3 Quantification after linear background removal

```
s_i = s.get_lines_intensity(background_windows=bw, integration_windows=iw)
```

```
k_factors = [<k-factor 1>, <k-factor 2>]
quant = s.quantification(s_i, method='CL', factors=k_factors)
print(f'E1: {quant[0].data[0]:.2f} \%, E1: {quant[1].data[0]:.2f} \%')
```

The quantification is done with the four lines of code above, where the last one prints the results. The first line gets the intensity of the peak corresponding to the lines of the specified element. HyperSpy selects automatically which lines to use for quantification. To see which lines are used, the `s_i` variable can be printed. The second line sets the k-factors. The k-factors in this project have been the one from AZtec, which are theoretically estimated. The third line does the quantification, where the method is specified. The method is the Cliff-Lorimer method, described in detail in Mari Skomedal's master thesis [1, Sec. 2.2.3]. HyperSpy has a method for quantification with the zeta factor method. The zeta method requires the value for the beam current, which was not measured in this project. [1]

2.3.4 Removing the background with model fitting

Another way to remove the background is to fit a model to the data. This step would be done right after loading the data. If the raw data contains a zero peak, as is the case for most Oxford instrument EDS detectors, the zero peak needs to be removed before fitting the model. The zero peak is removed by skipping the first n channels, where n=30 works well with the data from the GaAs wafer. The model fitting is done with the following code:

```
s = s.isig[<zero_peak>:]
m = s.create_model(auto_background=False)
m.add_polynomial_background(order=12)
m.add_family_lines(<list_of_element_lines>)
m.plot()
```

The lines above removes the zero peak, create a model from the `signal` s, adds a 12th order polynomial, add the lines of the elements in the `signal`, and plot the model. This model is not fitted, it is just a generated spectrum with the lines of the elements. Eventually, the method `create_model()` can take the boolean argument `auto_add_lines=True`, which will automatically detect the elements in the sample. The model consists of a number of components, which can be accessed with `m.components`. The components are all the gaussian peaks in the spectrum, in addition to the background as a 12th order polynomial. The order of the polynomial can be changed, but it should be tested by the user to see if it is a good fit. Further, the model must be fitted.

```
m.fit()
m.plot()
```

[1] Results from the zeta methon can be converted to the cross section method, see the "EDS Quantification" documentation in HyperSpy.

The first line fits the model to the data to the components and the second line plots the model. HyperSpy have a own option for fitting only the background. Since the background is one of the components in m, it is fittet with the code line above.

2.3.5 Quantification after model fitting

```
m_i = m.get_lines_intensity()

k_factors = [<k-factor 1>, <k-factor 2>]

quant = s.quantification(s_i, method='CL', factors=k_factors)

print(f'E1: {quant[0].data[0]:.2f} \%, E1: {quant[1].data[0]:.2f} \%')
```

The quantification after model fitting is done in the same way as in Section 2.3.3, but with intensity from the model instead of the signal. When modelling GaAs, the model can add the intensity from both K-lines and L-lines. Since AZtec only gives the k-factors for either the K-lines or the L-lines, the user must remove the lines without k-factors before quantification.

2.3.6 Calibrating the spectrum with the HyperSpy model

```
m.calibrate_energy_axis(calibrate='scale')

m.calibrate_energy_axis(calibrate='offset')
```

The two lines above calibrates the spectrum with the HyperSpy model and updates the dispersion and zero offset. The metadata in the `signal` s is updated with the new calibration. Thus, doing the previous step with quantification after model fitting can give a more correct quantification.

2.4 Peak and background modelling

The next sub-problem was to find out how the peaks and the background are modelled in a way that is easy to understand. The model was buildt without HyperSpy, with the idea of making every step easier to understand. The model was used to be able to remove the background and be able to calibrate the spectrum. The model was compared to the HyperSpy model. The model could be used to quantify the elements in the sample, but this was not done in this project. **(Brynjar: Do I want to do this?)**

The first step in creating a model is to identify the peaks. The peaks are assumed to be gaussian curves. The initial way of identifying peaks was that the user manualy identified the peaks. Later the peaks were identified with the function `find_peaks()` from the `scipy.signal` package. Different peak prominence were tested, and the peak prominence of 0.01 gave the best results.

The second step is to make a gaussian in each peak and one polynomial for the background. To do the fitting, the components need an initial guess. The background needs a coefficient for each order of the polynomial. Each gaussian need to have a mean, a standard deviation, and a height. The mean is the peak position. The standard deviation is the width of the peak, where FWHM $= \text{std} * 2 * \sqrt{2 * \ln 2}$ [2]. The height is the amplitude of the peak. The easiest way to get the initial guesses for the gaussians is to normalize the data and set all three parameters to 1. In the normalization the highest peak was set to 1, and the rest of the peaks were scaled accordingly. The best way to get the initial guesses for the background is to clip out the peaks with linear interpolation and fit a polynomial. The initial guesses for the background is then the coefficients of the polynomial. With the initial guesses, the whole model is ready to be fitted.

The third step is to fit the model to the data. Using the `curve_fit()` function from the `scipy.optimize` package, the model is fitted to the data. The function `curve_fit()` uses the Levenberg-Marquardt algorithm to fit the model to the data. The function `curve_fit()` returns the optimal parameters for the model. Fitting both the gaussians and the background at the same time makes the fitting more stable. One of the first iterations, where the user manually inputted the peaks, the fitting tended to partially fail. The issue was that the fitting only was done on the peaks. To minimize the error in the fitting, one of the gaussian curves with a low amplitude was moved and got a huge standard deviation, which compensated the background. This was fixed by fitting both the gaussians and the background at the same time. Doing this made the fitting both better and it failed less often.

2.5 Calibration

The next sub-problem was to calibrate the data with a self produced Python script. With a fitted model of the spectrum, the calibration can be done. Calibration can both be done on raw data with channels on the x-axis and on poorly calibrated data with energy on the x-axis. The dispersion is calculated with **??**. Table Table 2.2 shows calibration from AZtec, HyperSpy, and the self produced Python script.

Table 2.2: Different calibration values. The AZtec calibration is reffered to as the uncalibrated value. The dispersion is calculated with **??**. The offset is calculated with **??**. The own calibration was done on Ga L_α and As K_α from the 30 kV measurement on the GaAs wafer.

Calibration method	Dispersion, [keV/channel]	Zero offset [channels]
AZtec	50 %	50 %
HyperSpy	50 %	50 %
Own calibration	50 %	50 %
30 kV	50 %	50 %

[2] FWHM defined at: https://en.wikipedia.org/wiki/Full_width_at_half_maximum

2.6 Background models

The next sub-problem was to find out how different background models affect the quantitative analysis done in HyperSpy, and how well different order polynomials fit the background. The background models were tested on the spectrum of GaAs, and later also on **(Brynjar: TODO: other spectra. Also make a table here with results)**. The background was modelled as a polynomial of different orders. To quantify the different background models, the residuals were calculated. The residuals are the difference between the data and the model. **(Brynjar: [])**use root-mean-square error? The TABLE XXXX **(Brynjar: [])**make table shows the residuals for the different order background models. The best orders were visually inspected. A later idea was to model the background as a spline, which is a piecewise polynomial. The spline is a piecewise polynomial with a smooth transition between the pieces. The spline was not tested in this project, but it could be a good alternative to the polynomial background model.

2.7 Analysis failure

The next sub-problem was to find out when the analysis fails, both in AZtec and HyperSpy.

(Question for Ton: Section about normalization too?)

Bibliography

[1] Mari Sofie Skomedal. Improving quantitative EDS of III-V heterostructure semiconductors in low voltage STEM. Master's thesis, Norwegian University of Science and Technology, 2022.