Contents

Chapter 1

Results

1.1 Introduction

The results are presented in this chapter. The sections follow the structure of the sub-problems in **??**.

1.2 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 1.1. 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 1.1: 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.

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 %

1.3 Analysis steps in HyperSpy

(Question for Ton: Is this interesting to write about?)

(Question for Ton: This section is both presenting what and why we do the steps. That is leaning into the discussion. Should I separate 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.)**

1.3.1 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()`.

1.3.2 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 1.3.4. 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)
```

1.3.3 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]

1.3.4 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:

```
m = s.isig[<zero_peak>:].create_model(auto_background=False)

m.add_polynomial_background(order=12)

m.add_family_lines(<list_of_element_lines>)

m.plot()
```

The three lines above 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()
```

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.

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

1.3.5 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 1.3.3, but with intensity from the model instead of the signal. When modelling GaAs, the model adds both K-lines and L-lines. Since AZtec only gives the k-factors for the K-lines, the L-lines must be removed before quantification.

1.3.6 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.

1.4 Calibration

The next sub-problem was to calibrate the data with a self produced Python script.

1.5 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.

1.6 Background models

The next sub-problem was to find out how different background models affect the quantitative analysis done in HyperSpy.

1.7 Analysis failure

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

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.