An Approach towards Automated Spectral Analysis of Galaxy Data

Bachelorarbeit
zur Erlangung des Grades Bachelor of Science
an der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam

eingereicht bei:
Prof. Dr. Philipp Richter
Professur für Astrophysik
mit dem Schwerpunkt
Extragalaktische Astrophysik

von:
Moritz Itzerott
Eisenbergerstrasse 1
04626 Schmölln
Matrikel-Nr.: 796906

vorgelegt am:
24.01.2022

Contents

1 Abstract **1**

2 Kurzzusammenfassung **1**

3 Introduction **2**

4 Theoretical Background **5**

4.1 Data Analysis 5
4.2 Spectral Analysis 6
  4.2.1 Line Shapes and Broadening 8
  4.2.2 Measuring Equivalent Widths and Line Flux 11
4.3 Fitting a Model to Data 14
4.4 Extinction and Dust Attenuation 16

5 Galaxy Spectra and MUSE **19**

6 Designing the Algorithm **21**

6.1 Target Selection 21
6.2 Continuum Fitting 22
6.3 Spectral Analysis 23

7 Applying the Algorithm **25**

7.1 Data Import and Target Preselection 25
7.2 Continuum Fitting 27
7.3 Spectral Analysis 31
  7.3.1 Equivalent Width Estimation 33
  7.3.2 Line Flux Estimation 35
7.4 Data Output 36

8 Result **37**

8.1 Line Flux Measurements 37
8.2 Equivalent Width Measurements 40
8.3 Balmer Decrements 43
8.4 Redshift Measurements 46
8.5 Conclusion 48

9 Outlook **49**

10 Acknowledgements **51**

11 Appendix: The "ANGEL.py" Code **53**

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Zuhilfenahme anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.
Die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen sind als solche kenntlich gemacht.
Die „Richtlinie zur Sicherung guter wissenschaftlicher Praxis für Studierende an der Universität Potsdam (Plagiatsrichtlinie) - Vom 20. Oktober 2010“, im Internet unter `https://www.uni-potsdam.de/fileadmin/projects/ambek/Amtliche_Bekanntmachungen/2011/ambek-2011-01-037-039.pdf`, habe ich zur Kenntnis genommen.

Ort, Datum

Unterschrift

1 Abstract

The aim of this thesis is to acquire knowledge and techniques, necessary for an approach in automation of analysis of galaxy spectra. This requires expertise in programming and in the underlying physical processes of spectral features. Hence, starting with the physical and computational background, the design of a PYTHON-program for automated spectral analysis, as well as its application onto different data sets, is established and discussed in detail.
As a result of the data reduction process, this program produces a set of equivalent width(W_λ), line flux(F_λ) and redshift(z_G) measurements estimated from galaxy spectra of the MUSE-Wide Survey[Her+17a] and the MUSE Hubble Deep Field South[Bac+17] for both Balmer Hα and Hβ emission and the CaK absorption lines. Measurements of the CaH absorption line had to be abandoned, because of an overlapping Hϵ feature. Also, because of the low amount of sufficient CaK absorption in the used galaxy spectra, statistical analysis was not meaningful and therefore refrained from for this feature.
Although planned beforehand, the final PYTHON-tool had to include many exceptions and fail saves in order to account for damaged data or special cases in the spectrum. Therefore, it differs significantly in complexity from the initial design. This also includes a mandatory restructuring of every data set before introducing it to the routine, since it has been observed that astronomical data sets are not structured uniformly.
Comparing the resulting equivalent width and line flux data with a Balmer Decrement of $\mathrm{H}_\alpha/\mathrm{H}_\beta = 2.859$ shows, that measurements from the program lie in accordance to what the model predicts. Deviation towards higher decrements are later connected to interstellar extinction and reddening. It was even possible to cross-correlate specific Balmer Decrement measurements and orientation of the respective galaxy.

2 Kurzzusammenfassung

Ziel dieser Arbeit ist es, Wissen auf dem Gebiet der astrophysikalischen Datenanalyse zu erlangen. Da Algorithmen zunehmends die Analyse von Spektren übernehmen und die Größe astrophysikalischer Datensammlungen stetig wächst, wird diese Analyse in Form eines selbstgeschriebenen Programmes durchgeführt. Die dafür notwendigen Kenntnisse der Atom- und Astrophysik und Informatik, speziell der Programmiersprache PYTHON, wurden im Verlauf der Arbeit erwor-

ben. Mit diesem Wissen wurde das Programm vorerst geplant und anschließend umgesetzt, wobei es sich letztendlich sehr vom geplanten Entwurf unterscheidet, um mit unerwarteten Problemen, wie der uneinheitlichen Struktur von verschiedenen Datensammlungen und dem Auftreten von fehlerhaften oder unvollständigen Daten umgehen zu können.

Die resultierenden Daten umfassen die Äquivalentbreite W_λ, den Linienfluss F_λ und die Rotverschiebung z_G von Galaxien der MUSE-Wide Survey[Her+17a] und des MUSE Hubble Deep Field South[Bac+17] für die Linien Hα, Hβ und CaK. Daten der ursprünglich ebenfalls zu betrachtenden CaH-Linie konnten, aufgrund einer Überdeckung mit einer weiteren Linie, nicht verwendet werden. Ebenso ergab die Auswertung wenige Galaxien, die signifikante Calcium-K Linien zeigten, weshalb keine statistische Analyse dieser wenigen Daten sinnvoll war.

Um die resultierenden Balmer Dekremente zu verifizieren, wurden die Messdaten anschließend mit dem Laborwert von $\mathrm{H}_\alpha/\mathrm{H}_\beta = 2.859$ verglichen und zeigten gute Übereinstimmung mit diesem. Abweichungen werden später als Extinktion und Rötung aufgegriffen. Es war ebenso möglich, die Orientierung der Galaxie zur Erde mit dem gemessenen Balmer Dekrement zu korrelieren.

3 Introduction

It was the year 964AD, when Persian astronomer Abd al-Rahman al-Sufi made an entry in his "Book of Fixed Stars"[Sūf14] of a *"nebulous smear"* in the Andromeda constellation. He, as well as many other astronomers throughout the centuries, were not aware of the scientific breakthrough he just achieved and referred to it simply as the Andromeda Nebula, one of many inside the Milky Way Galaxy. It would take until the early 20th century, when astronomers found indisputable evidence that it is an entirely own system outside of the Milky Way, another galaxy, itself. This debate was settled by Edwin Hubble, who found Cepheid variable stars inside of this nebula, of which he could calculate the distance by using a formula developed by Henrietta Swan Leavitt - her so called period-luminosity-relation[Hub29][LP12]. As a women, she often got overlooked in her discoveries, but her law ultimately made it possible for Edwin Hubble to achieve this groundbreaking discovery, that made him renowned.

A hundred years later with huge advancements in telescopes and technology and two Hubble Deep Fields, extragalactic astrophysicists study *photometric* and *spectroscopic* measurements of tens of thousands of catalogued galaxies.

The analysis of those spectra can be arbitrarily complicated and very time consuming, depending on the amount of data one likes to extract from them. To retrieve statistically relevant measurements, also a large number of samples has to be analyzed. Thankfully, astronomical catalogs are becoming larger with every year, with some already containing reduced data from objects like redshift or a set of pre-identified emission lines. This however makes manual surveying and analysis extremely time- and cost intensive. Thus, a more future-oriented and time-efficient way of approaching this task is needed. Possible methods include developing an algorithm or using neural networks or computational intelligence. Especially the latter has gained a lot of popularity and has been researched heavily in recent years through all fields of science. It has also found its way into astrophysical research, where it is already used for classification and some analysis in big data sets.[ZBB21] This thesis will explore the development of an algorithm for spectral analysis of galaxy data. For the initial design, 1d galaxy spectra from the MUSE Ultra Wide Survey are used, with close attention afterwards to it being able, to analyze data from other catalogs as well. It should be able to find certain lines and fit them according to a continuum at the given position, while requiring no manual interaction. Furthermore, it needs a way to determine the strength of a set of given emission or absorption lines in order to get an estimate on the dust ratio inside of the galaxy's halo. For these measurements, the program should calculate the equivalent width and line flux of a set of predefined spectral lines.

4 Theoretical Background

The purpose of this thesis shall be the development of an algorithm that automatically analyses given parts of a spectrum such that the resulting data can be used in scientific research. This requires not only expertise in programming and computational science, but also in the (astro)-physical processes that govern the analysed spectral features.

The following chapter will provide the basis for both of the above mentioned fields by introducing astrophysical concepts and concepts of data science and data analysis that have been used throughout this thesis, both for designing the algorithm and analysing its resulting data.

4.1 Data Analysis

Thanks to vast investments in astrophysical research, today, everywhere on and above earth, telescopes produce immense amounts of scientific data every second. While the mountain of available data grows, new techniques for analysis of same data have to be researched as well. For a long time, the field of data analysis in astrophysical research has been dominated first by manual analysis and later by hand-written algorithms. This is still the case for most scientific analyses, although machine learning algorithms begin to take over certain tasks.

Algorithms are mostly used when there is a definite order of tasks to be addressed. This includes spectral analysis, reducing observational data and many more. With these kind of tasks, that can be divided into subroutines, algorithms have proven to being superior over every other approach.

However, this reaches its limit, when it is not possible to divide the task into a sequence of subroutines or the task is too complicated to be solved with a simple program. A very popular example for this is the classification of pictures. Humans are very well capable of distinguishing, say, an open star cluster from a galaxy, but an algorithm cannot, just because there is no true programmable way of telling it, what it should base its decision upon[1]. In this case, there is just no unique solution and thus no straight-forward algorithm to determine what is on the picture. As this is an optimization problem rather than a problem of automatization, computational intelligence gets involved[ZBB21]. This approach does not rely on a set algorithm, but rather on *learning*-based algorithms. These are *taught* by using so called *training data* that are used within the algorithm and produces a certain end

---

[1] There are actually also cases, where scientist are uncertain, whether an object is a star cluster or a galaxy.

result, say deciding that a picture shows a star cluster. Afterwards this result is compared with what is actually on the picture, leading to inbuilt parameters being continuously changed in order to optimize the result and get closer to the optimum result. After many iterations of this routine, the accuracy of such a program can be remarkably close to 100%.

Although this seems promising, the reasons why this thesis will not use the latter approach are as follows: the task of spectral analysis is one of automatization, rather than one of optimization. There is a unique strategy of estimating line flux and equivalent widths. Also an important part of writing a Bachelor's thesis should be gaining knowledge in the field the thesis is written in. This aspect is lost, when creating a neural network, because there is no saying how this computational intelligence works and also little to no way of implementing physical processes into it. This is an important issue, since detailed analysis of certain features relies heavily on underlying physics and expert knowledge to be executed properly. Therefore, a better way to implement this, is writing a hand-crafted universal algorithm for detailed spectral analysis of galaxy spectra. Nevertheless, computational intelligence might also be suitable to analyse spectra one day.

Another important decision to make is the programming language to write the algorithm in. For this purpose, PYTHON[VD09] has been chosen, because it is one of the most popular languages in astrophysical research. It is very easy to learn for beginners, since it assigns variable types automatically and can be adapted to problems of varying difficulty levels, because of many existing libraries with prewritten routines. However, for projects with many of calculations or solving of differential equations, C++ might be a better alternative, but for this project PYTHON shall suffice.

4.2 Spectral Analysis

When probing an object in space with spectral data, there are a lot of processes along the line of sight that can produce certain spectral features. These are transitions of atoms or molecules from one energetic state to another[Bar17]. They involve electronic, vibrational and rotational states, whose energy is not arbitrary, but quantised according to quantum mechanics. When a transition occurs from an initial state A to a final state B, an atom or molecule can either emit or absorb electromagnetic radiation in form of photons with energy corresponding to the energy difference of both states. Whenever the initial state has a higher energy than the final state, a photon is either emitted via spontaneous or stimulated emission. This can be observed when a large amount of excited gas (meaning excited atoms or

molecules) is present, like in the accretion disk of a quasar. When the light from a quasar passes through colder gas, absorption lines also become visible in the quasar spectrum. These lines come from absorption of photons by atoms or molecules moving to a higher final energy state, becoming excited.
The number of photons or flux gathered over a wavelength range makes up a spectrum, which is the basis of nearly every astrophysical research, since it is unfortunately not possible to visit these objects directly. Stunningly though, based on these spectra, a lot of properties of the intervening gas and the background source can already be researched today! They can often be directly derived from line intensities, line widths, line ratios and most importantly what lines are present in the first place.

For the purposes of this thesis and to produce an algorithm that can deal with a lot of different spectra and tasks, such an algorithm should be able to analyse both absorption and emission lines. Therefore, to introduce spectral features that can later be added upon, two emission lines and two prominent absorption lines present in the used galaxy spectra are taken into first consideration: Calcium K & H and both Balmer Hα and Hβ lines.
Some galaxy spectra show a very distinct Calcium absorption in form of the CaII K & H doublet at $\lambda\lambda 3934.78\text{Å}, 3969.60\text{Å}$ rest-frame wavelengths. Studies have indicated that strong CaII absorption can be observed in the circum-galactic medium of galaxies and correlates to high stellar mass and high star formation rates[ZM13]. Estimating equivalent widths of such absorption lines might yield data for further scientific analyses and will therefore be included into the analysis routine. While most absorption lines have their origin in the colder gas in and around the galaxy, emission lines come from another source. The continuum light coming from a galaxy consists of the superposed blackbody radiation of all of its stars, usually dominated by the brightest ones. This means that emission lines on top of a galaxies continuum are due to processes emitting photons for example in stellar chromospheres. Very popular, because very distinct and strong are the Balmer emission lines in the optical range. These transitions arise from recombination events of hydrogen atoms from a higher energy level into the second lowest level $n_{\text{B}} = 2$, and are named subsequently with Greek letters starting from α where $n_{\text{A}} = 3$ over β with $n_{\text{A}} = 4$ and further. Bright stars with surface temperatures around 10000K show intense Hα emission lines at $\lambda 6564.614\text{Å}$[Bar17]. Also planetary nebulae as a site of late stars, show strong Hα emission lines[Bar17]. However, in later type stars, where temperatures don't allow for a large number of excited hydrogen atoms, absorption processes and thus Balmer absorption features are more dominant in stellar photo-

spheres. Inbetween this range, a feature consisting of an emission line inside of an absorption region for both Balmer lines, while more distinct for Hβ at λ4862.721, can be observed. Balmer lines have proven to being very important in studying effects like star formation in a galaxy. Since these lines are strongest coming from hot, massive and therefore short lived stars, a high intensity or line flux in these lines is therefore a clue to a large amount of young hot stars and a high star formation rate. Also, their ratio can be used to estimate the amount of dust attenuation (see 4.4).

4.2.1 Line Shapes and Broadening

Since transitions, like the ones mentioned before, arise at specific wavelengths corresponding to the energy differences between both energy levels, one could assume, that they would result in a spectral feature at exactly that wavelength or frequency, respectively. However, this proves to not be the case because there are certain processes that *broaden* these spectral lines.

Starting with a simple *line shape function* $\psi(\nu)$, such that:

$$\int_{\text{Line}} \psi(\nu)\mathrm{d}\nu = 1,$$

it can be observed that $\psi(\nu)$ does not simply follow a delta distribution $\delta(\nu - \nu_0)$, but has to be more complex. This is in the first place due to the atomic energy states having a finite lifetime. The Heisenberg uncertainty relation states, that energy and time can not both be measured to arbitrary accuracy, but that there will always be an intrinsic uncertainty ΔE in the final energy state E_b[Hei27]. This results in so called **natural broadening** of a spectral line. That is why atoms will absorb or emit photons of wavelengths around the central wavelength with $\Delta\lambda = hc/\Delta E$ or frequency around ν_0 respectively, creating a so called **Lorentzian** line shape[2]:

$$\psi'(\nu) = 1/\pi \cdot \frac{\gamma_k}{(\gamma_k^2 + \Delta\nu^2)} \text{ with } \gamma_k = 1/4\pi \sum_{k>i} A_i \text{ for } \mathrm{V} = 0.$$

This natural broadening process is inherent to every atomic transition. However, it is "inconsequential in comparison"[Fu06] to other broadening processes triggered by collision and motion of atoms and molecules.

---

[2] taken from Lecture 2 Interstellar Absorption Lines: Line Radiative Transfer 1. Atomic absorption lines 2. Application of radiative transfer to absorption emission 3. Line broadening curve of growth 4. Optical/UV line formation 5. Optical/UV line observations by **Al Glassgold**, found at `https://w.astro.berkeley.edu/~ay216/06/NOTES/`; References: [Spi04] [DS03]

Collision of molecules and atoms can lead to excitation or deexcitation of those collision partners, which - under high pressure - happens on shorter timescales than the intrinsic lifetime of such an energy state[Nov73]. This again leads to an additional broadening process. However, the necessary particle densities for this to be significant are usually only reached for example in stellar atmospheres, which is why these broadening processes are dominating in stellar lines. **Collisional broadening** also results in **Lorentzian** line shapes.

The above mentioned processes are all assuming that the absorbing material is at rest with respect to its motion, which in this case also relates to a low temperature. When this is not the case and atoms or molecules are in motion with respect to another, the line broadening is dominated by **Doppler broadening**. This motion can either be turbulent or thermic, where the latter is due to a high temperature of the system.

When the absorbing gas particle is moving with respect to the observer with a velocity V along the line of sight, then one can introduce the following notations and equations[3]:

ν_0	rest frame frequency
$\nu_0' = \nu_0 \cdot (1 - \mathrm{V}/c)$	Doppler-shifted frequency
$\Delta\nu = \nu - \nu_0$	

Thus the absorption line profile $\psi(\nu)$ depends on the difference between frequency and Doppler shifted frequency $\nu - \nu_0'$:

$$\begin{aligned}\psi'(\nu - \nu_0') &= \psi'(\nu - \nu_0[1 - \mathrm{V}/c]) \\ &= \psi'(\Delta\nu + \nu_0 \mathrm{V}/c)\end{aligned}$$

The complete line profile function is therefore a convolution of the probability of such a velocity V occurring ($P(\mathrm{V})$) with the profile it generates ($\psi'(\Delta\nu + \nu_0\mathrm{V}/c)$), integrated over all velocities.

$$\psi(\nu) = \int_{-\infty}^{\infty} P(\mathrm{V})\psi'(\Delta\nu + \nu_0\mathrm{V}/c)\mathrm{dV}$$

---

[3] see footnote 2

Gas particles and atoms within a gas with a certain temperature obey the Maxwell velocity distribution:

$$P(\mathrm{V}) = A \cdot e^{-\frac{\mathrm{V}^2}{b^2}} \text{ with } A = \frac{1}{\sqrt{\pi} b} \tag{1}$$

Here b is the width of this broadened profile, which can be expressed as[4]:

$$b^2 = b^2_{\text{thermic}} + b^2_{\text{turbulence}} = \frac{2kT}{m} + b^2_{\text{turbulence}}.$$

The parameter b therefore has components not only from thermal, but also from turbulent motions of entire regions of the gas, representing a velocity dispersion $\sigma^2_{\text{turb.}} = b^2_{\text{turb.}}/2$. Both of these parameters are described by distributions that follow a Gaussian distribution, which is why the ultimate result is also a **Gaussian** line profile. Also it can be deduced, that the amount of Doppler broadening is heavily dependent on the gas temperature. That is why the temperature can also be calculated from the line width itself.

As discussed earlier, the impact of natural broadening, especially collisional broadening, grows with particle densities, in this case, column density. Therefore, natural and Doppler broadening are both observed in regions with very high column densities and high optical depths. In this case they yield a so called **Voigt profile**:

$$\psi(\nu) = \frac{1}{\sqrt{\pi^3} b} \int_{-\infty}^{\infty} e^{-(\mathrm{V}/b)^2} \frac{\gamma_k}{\gamma_k^2 + (\Delta\nu + \Delta\nu_D \mathrm{V}/b)^2} \mathrm{dV}$$

with the Doppler width $\Delta\nu_D = (\nu_0/c)b = b/\lambda_0$; coming back to the arbitrarily complexity of spectral analysis. This profile is most dominant when probing optically thick ($\tau \gg 1$) gas, resulting in saturated, meaning very strong, absorption lines, because of dominating natural broadening. As will be discussed later, when photons travel through a material, the resulting intensity is given by:

$$I = I_0 e^{-\tau}$$

with the optical thickness τ being a measure of how much energy or intensity is taken away by absorption processes inside of the material. For optically thin gas, which is relevant for CaII in these spectra, the natural broadening can be neglected compared

---

[4] taken from **M. Pettini**: Physical Cosmology - Lecture 10: "Absorption Line Formation and the Curve of Growth"

to Doppler broadening, reducing the final line profile to that of a **Gaussian**:

$$\psi(\lambda) = A \cdot \exp\left[\frac{-(\lambda - \lambda_0)^2}{2\sigma^2}\right] + C \tag{2}$$

with a scaling factor A, the difference to central wavelength $\Delta\lambda_{\text{obs}} = \lambda - \lambda_0$ and the variance $\sigma^2 = b^2/2$. Aside from theoretical studies, observations give the convolution of the intrinsic spectral line profile with the *instrumental profile*[CS04]. This is defined by the instrument and is fortunately well approximated by a Gaussian for the MUSE instrument and as such, the observed FWHM $\Delta\lambda_{\text{obs}}$ can be written as:

$$\Delta\lambda_{\text{observed}} = \sqrt{\Delta\lambda^2_{\text{Intrinsic}} + \Delta\lambda^2_{\text{Instrument}}}.$$

This simplifies the analysis a lot, but further iterations of the algorithm could possibly also account for other instrumental profiles when implementing other catalogues.

4.2.2 Measuring Equivalent Widths and Line Flux

Approximating an emission or absorption line with such a profile can help tremendously with estimating above mentioned properties, especially when working with poorly resolved lines, since the resolution is increased artificially; more on the topic of resolution and its definition will be discussed later in section 5. One problem with low resolution spectra, especially in this project, was that certain regions might not be properly resolved to show possible overlapping absorption features. An example for this would be the Hϵ line at λ3970.07Å, which is lying right inside of the CaH absorption region at λ3969.60Å and therefore both lines will be falsely measured as one. This was ultimately also the reason why only CaK was analysed in the following and fitting two Gaussians will be included in later versions of this algorithm.

Another issue is that emission lines like Hα or Hβ cannot be distinguished from underlying absorption processes coming from metals or Balmer absorption itself. Though being important for accurate measurements, accounting for this absorption when measuring line flux cannot be done without a high enough resolution power. According to [CS04], there are two cases to be considered here for the full width at half maximum $\Delta\lambda$:

- $\Delta\lambda_{\text{obs}} \gg \Delta\lambda_{\text{Instr.}}$, in which case the line is resolved. The spectral line thus contains all information about the intrinsic profile and can be fitted accordingly.

- $\Delta\lambda_{\text{obs}} \approx \Delta\lambda_{\text{Instr.}}$, in which case the line is poorly resolved and other methods have to be applied in order to estimate the *strength* of the spectral line.

There clearly are spectrographs working in the first regime, like the Ultraviolet and Visual Echelle Spectrograph(UVES), which has a resolution power of $R \sim 80000 - 110000$[ESO21] depending on slit width and wavelength range. The BACHES spectrograph at the University of Potsdam also has a quite high mean resolution power of $R \sim 18000$[Gmb21] comparable even to that of the Cosmic Origin Spectrograph COS on the Hubble Space Telescope[Sod21]. The still to be introduced MUSE spectrograph operates at much lower resolution powers, since it is not only designed to take spectra, but also pictures at the same time. Thankfully, the MUSE instrument has just high enough resolution power to show absorption regions underlying the Balmer emission lines, which enables measuring line fluxes with correction for said regions. However, for estimating the strength of all spectral features, absorption and emission alike, it is interesting to know, what solutions can be found to still measure certain line parameters in the second case *independent of resolution*, when other methods are not possible. This is where measuring equivalent widths (W_λ) has been introduced[CS04].

This technique is independent of instrumental resolution and has been used already in the past to investigate absorption features.

$$W_\lambda = \int_{\text{Line}} (1 - F_\lambda/F_0)\,d\lambda. \tag{3}$$

The idea is to normalize the continuum F_0 to 1 for absorption lines, such that if whenever $F_\lambda \neq F_0$, $W_\lambda \neq 0$, which is an indication for a spectral line. Therefore W_λ represents the area of the line when the adjacent continuum is normalized to 1. Since red-shifted spectra are observed, the rest-frame equivalent width is:

$$W_{\text{rest}} = W_\lambda/(1+z).$$

This value can then be used for example to compute column densities and analyse the amount of absorbing gas. This is implemented into the algorithm by normalizing the flux density according to 3, fitting a model to the line and integrating over the model.

This requires some more information and thoughts. As a simple trick to make computing the above integral easier and faster, the assumed model (2) and equivalent width has been changed as follows:

$$W_\lambda = \int_{\text{Line}} \psi'(\lambda)\mathrm{d}\lambda = \int_{\text{Line}} \frac{\psi(\lambda)}{\int \psi(\lambda)\mathrm{d}\lambda}\mathrm{d}\lambda \tag{4}$$

$$= A\int_{\text{Line}} \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left[\frac{-(\lambda-\lambda_0)^2}{(2\sigma^2)}\right]\mathrm{d}\lambda = A \tag{5}$$

By normalizing the area under the model-gaussian to one, the integral in (4) essentially reduces to 1 thus the equivalent width is then just the scaling factor A of the model curve. As said, this method is usually used with absorption lines, but has been used for emission lines as well (see [Bac+15] or [GBW11]). As part of this thesis it will therefore be examined, whether the above mentioned formula (3) can be also used for emission lines or if changes have to be made. Almost as important as the actual value is an estimate for the uncertainty. As this procedure includes a substantial amount of model fitting, a correct uncertainty calculation is quite complicated and will be discussed further in sections 4.3 and 7 together with other factors.

As said before, another way of estimating the strength of a given spectral line is to determine its line flux. One could therefore simply determine the maximum flux of the respective line, which might yield sufficient results for very narrow, highly resolved lines, where the maximum flux density falls perfectly onto a line pixel. That this is not acceptable for this project, should be certain after the above explanations. Also this method does not account for the width of the line, which is most certainly broadened.

A way to account for line widths is to compute the sum over the continuum-subtracted flux $(f_i - c_i)$ of all pixels that make up the line multiplied by the pixel width (Δx):

$$F_\lambda = \sum_{i=1}^{N}(f_i - c_i) \cdot \Delta x. \tag{6}$$

That way it gives an estimate as to how much flux or in this case energy in the form of light is added on top of the continuum by certain emission processes. The same holds for absorption features. However, here this value indicates how much flux or energy is taken away from the continuum by an absorption process. This works best when the lines are very well resolved. Also it can be used without adjustments for both absorption and emission lines, which is the reason why it has been used with this algorithm as well. Since measurements can never be done without uncertainty estimates, this has to be taken into account when computing and later discussing line fluxes. The impact of individual flux uncertainties to their sum can be computed with propagation of uncertainty.

Let $h(x_1,x_2,...,x_N)$ be a function with x_i $i \in \{1,...,N\}$ having some uncertainty δx_i. Then:

$$\delta h(x_1,x_2,...,x_N)^2 = \sum_{i=1}^{N} (\partial_{x_i} f(x_1,x_2,...,x_N) \cdot \delta x_i)^2. \tag{7}$$

Therefore with $h = F_\lambda(f_i,c_i)$:

$$\delta F_\lambda^2 = \sum_{i=1}^{N} \sqrt{(\partial_{f_i} F_\lambda \cdot \delta f_i)^2 + (\partial_{c_i} F_\lambda \cdot \delta c_i)^2}^{\,2} = \Delta x^2 \cdot \sum_{i=1}^{N} (\delta f_i)^2 + (\delta c_i)^2. \tag{8}$$

Thus the squared uncertainty in estimated Line Flux can be computed by summing the squared uncertainty in flux and the one in continuum over all pixels that make up the line and multiplying by the pixel width squared.

4.3 Fitting a Model to Data

Fitting a model to a data set can be a powerful tool for confirming a prognosis or validating a technique that has been used. Here, it will be used to fit the shape of the lines, that are to be examined, fit the continuum of a spectrum and in the end approve the below theory of a constant flux ratio of both Balmer lines. Choosing the model always has to be justified by some physical law, which has been done above (see section 4.2.1). That is also why a Gaussian will be used for fitting the lines and why a linear function will be used in order to validate the Balmer decrement (see section 4.4). The only step where this is not the case, is, when matching the continuum with polynomials (see section 6.2). There are two main fitting techniques that will be used in this thesis. One only accounts for Gaussian uncertainties in one dimension and will be used for the analysis part, while the other accounts for uncertainties in two dimensions and therefore will be used for displaying the resulting data.

When the data set only has uncertainties in one dimension or has negligible uncertainties in the other dimension, the function **`curve_fit`** introduced by the **`scipy.optimize`**[Vir+20] package will be used. This routine operates with *mini-*

mized square fitting[HBL10], which intends to minimize the value:

$$\chi^2 = \sum_{k=1}^{N} \frac{[y_k - f(x_k)]^2}{\sigma_{y_k}^2}. \tag{9}$$

Here χ^2 represents the sum of all squared distances $[y_k - f(x_k)]^2$, where y_k is the real or measured data point and $f(x_k)$ is the modelled data point, divided by the squared uncertainty, $\sigma_{y_k}^2$, of the measured data point; uncertainty, not error[5]. One immediately recognises that for *optimal data*, where, on average, every measured value has an uncertainty or standard deviation of 1σ, χ^2 is at its lowest with $\chi^2 = N$. For normalization, χ^2 is often divided by the number of degrees of freedom $(N - M)$, since these are the parameters that can influence the result, such that:

$$\chi^2_{\text{reduced}} \overset{!}{\geq} \frac{N}{N - M},$$

with the number of data points N and the number of fit parameters M. That means, in order to minimize χ^2, the number of fit parameters M should be as small as possible, which results in:

$$\lim_{M \to 0} \chi^2_{\text{reduced}} = 1. \tag{10}$$

The value of χ^2 therefore is a representation of how good the fit models the data (see section 6.2 for further discussion on that subject), where both $\chi^2 \gg 1$ and $\chi^2 \ll 1$ are clues to a model that does not approximate the data well; all under the assumption of correct data in the first place. In the later discussion, when displaying relations, there are other models necessary to deal with uncertainty in both dimensions(see figure 8). For this, orthogonal distance regression (ODR[BR90]) has been used.

---

[5] as a small, but important footnote from a paper Gábor Worseck told me about: Data has "uncertainties, not errors; if they were *errors*, we would have corrected them!"[HBL10]

4.4 Extinction and Dust Attenuation

A very important step during the development of every scientific program is a final verification, done by applying the resulting measurements to a physical law or prognosis. For the purposes of this thesis, both Balmer emission line measurements will be compared and it will be examined whether they are in agreement with the following extinction and dust attenuation model.

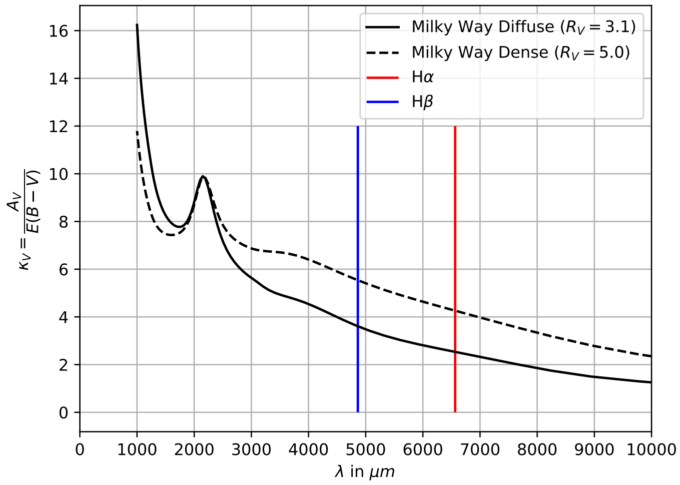

Figure 1: This plot shows two extinction curves and has been created with data from [CCM89] for both diffuse and dense ISM environments. The locations of both Balmer Hα(red) and Hβ(blue) are marked for better visualization.

Galaxies consist of stars, molecular clouds, gas and dust, all of which have a significant impact on taken spectra as has been described earlier. This is surprising when taken into account that dust only makes up around 0.2% of total baryonic mass of the galaxy [Dra+08].

Dust grains have an influence on photons of all wavelengths, leading to different apparent magnitudes than calculated from absolute magnitude and distance. They can absorb energy in from of photons and re-emit this energy in the infrared, effectively deleting flux from spectra in certain ranges (*reddening*). Also dust grains might scatter photons out of the line of sight (*extinction*). When analyzing spectra in the visible range (which is the case here), one can ignore the re-emission in the infrared, while also ignoring the fact that photons might be scattered into the line of sight from other sources. The reduction in observed flux can then be expressed as:

$$F(\lambda) = F_{\text{int}}(\lambda)e^{-\tau} = F_{\text{int}}(\lambda)10^{-0.4 \cdot A_\lambda}, \tag{11}$$

with the extinction factor A_λ, the observed flux $F(\lambda)$, the intrinsic flux $F_{\text{int}}(\lambda)$ and the optical depth τ[6]. That means the observed flux is reduced by a factor

---

[6] lecture script of "Grundkurs Astrophysik 2021" chapter 09-2 by Prof. Dr. Lutz Wisotzki

corresponding to the extinction $A(\lambda)$. This value can be expressed as a product of color excess ($E(B-V)$) and an extinction coefficient κ, where the prior is the above mentioned discrepancy in apparent magnitudes in B and V range:

$$A(\lambda) = E(B-V)\kappa(\lambda).$$

Since all of these quantities are dependent on wavelength, it makes sense to plot them against λ to retrieve an estimate for $\kappa(\lambda)$ as seen in figure 1. The resulting plot is called an *extinction curve*. It describes how the coefficient κ changes with wavelength. At first sight, it is obvious that the extinction coefficient heavily decreases from smaller/bluer wavelengths towards larger/redder wavelengths. There are, however, some peculiarities, like the so called 2200Å bump or another smaller bump at $\sim$ 4000Å, which are both still subject in today's research. In this plot, it can be seen, that the influence of dust grains is strongest in the UV and optical wavelengths, while diminishing towards higher (redder) wavelengths. The color excess $E(B-V)$ is usually calculated by using observed and calculated B and V magnitudes and their differences, called color excess', respectively[Bar17].:

$$E(B-V) = (B-V) - (B-V)_0. \tag{12}$$

With a way to calculate magnitudes from flux densities, the color excess could be directly derived from those flux densities in the respective wavelength regions or, in this case, line fluxes from very distinct spectral lines in these regions. One could derive the ratio of two lines in both the blue and red wavelength range respectively in order to estimate a numerical value for $E(B-V)$. Commonly used are the two Balmer lines Hα and Hβ, highlighted in the above displayed plot(1).
With using a so called 'maggy'[7][GBW11] with magnitude m_0 and flux density $f_0 = 1$, that sets the zeropoint magnitude scale, one can achieve the desired equation:

$$m_{\mathrm{B/V}} = m_0 - 2.5\log_{10}\left(\frac{f_{\mathrm{B/V}}}{f_0}\right). \tag{13}$$

When plugged into equation 12, and assuming $f_\mathrm{B}(f_\mathrm{V})$ to be $F_{\mathrm{H}\alpha}(F_{\mathrm{H}_\beta})$, respectively, the result states in agreement with Miller & Matthews[MM72]:

$$E(\beta-\alpha) = -2.5\log_{10}\left[\frac{F_{\mathrm{int}}(\mathrm{H}\alpha)/F_{\mathrm{int}}(\mathrm{H}\beta)}{F(\mathrm{H}\alpha)/F(\mathrm{H}\beta)}\right], \tag{14}$$

---

[7] As described in the SDSS documentation on measuring magnitudes and fluxes at `https://www.sdss.org/dr13/algorithms/magnitudes/#:~:text=A%20\T1\textquotedblleftmaggy\T1\textquotedblright%20is%20the%20flux,%5D%20\T1\textendash%202.5%20log10%20f%20.`

where $F_{\text{int}}(\text{H}\alpha)/F_{\text{int}}(\text{H}\beta)$ is a well known ratio called the *Balmer Decrement*, which can be derived theoretically. For the purposes of this thesis, a Balmer Decrement of

$$\frac{F_{\text{int}}(\text{H}\alpha)}{F_{\text{int}}(\text{H}\beta)} = 2.859 \tag{15}$$

will be used, which has been calculated for Case B recombinations in a gaseous nebula environment of $T = 10^4\text{K}$ and $N = 10^2\text{cm}^{-3}$ in Table I of [Bro71]. For Case B recombinations, it is assumed that the region from where photons are emitted is optically thick to ionizing radiation, meaning radiation coming from all transitions except those of the Balmer lines is absorbed again. Any amount of dust in the line of sight of an object will lead to a deviation from this Balmer Decrement towards higher values, which can then be picked for further analysis. One reason for a deviation towards higher values could be the orientation of galaxies towards the viewer, since when taking spectra from an *edge-on* galaxy, the galaxy's disk, which contains the most dust, is directly in the line of sight. This will be one part of the evaluation in the end, while also examining, whether equivalent width measurements can be used instead of line fluxes.

5 Galaxy Spectra and MUSE

The program developed in the course of this thesis is designed to handle data from a number of different catalogues, but for calibration purposes, the MUSE-Wide Survey, released on 23rd of May 2017[Her+17a], has been used. The Multi Unit Spectroscopic Explorer (MUSE for short) is an array of instruments constructed with the Leibniz Institute for Astrophysics Potsdam (AIP). MUSE is an integral-field spectrograph, meaning it scans the night sky taking pictures and spectra at the same time.

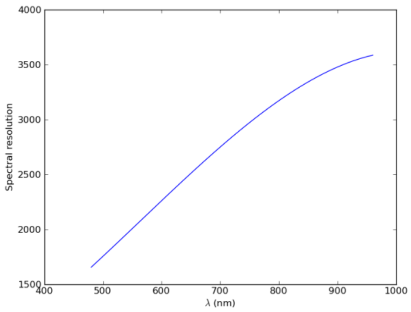

Figure 2: The above plot shows the resolution power of the MUSE instrument across its covered wavelength range. It can be seen that R increases rapidly towards higher wavelengths.

It currently operates at the Very Large Telescope at the Paranal Observatory in northern Chile in the visible rest frame wavelength range from 4650 Å to 9350 Å with a resolution power ranging from $R = 1770$ (480 nm) to $R = 3590$ (930 nm)[8], as seen on the right in Figure (2). The resolving power or spectral resolution $R = \lambda/\Delta\lambda$ is a measure for the smallest distinguishable distance in wavelengths $\Delta\lambda$ at a given wavelength λ.

The above mentioned MUSE-Wide Survey data set consists of 1D spectra and data cubes of 831 emission line galaxies and a master file containing useful information about each galaxy, like its redshift. Such immense amounts of gathered astronomical data are routinely saved in the form of fits (Flexible Image Transport System) files. Arranged as tables they are usually easy and fast to read in, when one knows how to address certain rows and columns in them; this will be discussed later.

Based on this structure, the algorithm has been designed. Other data sets with different structure have then been used to test the ability to also work from different sources, which, in the end, changed the form of the algorithm slightly.

---

[8] `https://www.eso.org/sci/facilities/paranal/instruments/muse/overview.html`

6 Designing the Algorithm

Most tools for spectral analysis still require some sort of human interaction, which does leave room for human error[9], while also lowering time efficiency tremendously. The first step to writing a program or an algorithm should always be making a schematic or a flow chart. That way one can think about what it should do and in what order beforehand. This is to be done very carefully and thoroughly, since it makes writing the algorithm more time efficient, but should take a finite time, as there will be problems that will not occur until the program is being written and tested.

The algorithm's main task will be to import given galaxy spectra, calculate the strength of both Balmer Hα and Hβ lines and both CaK and CaH lines, and give an estimate on the extinction in the galaxy's line of sight based on the Balmer Decrement. After examining the data and experimenting with them, the main task could be split up into three subroutines:

- Target selection
- Continuum Fitting
- Spectral Analysis

Where the data are first read in, then a continuum is fit onto the spectrum and afterwards the spectral lines are analyzed.

6.1 Target Selection

First, there has to be a filter routine that decides which spectra can be used for further analysis. There are a lot of parameters this filtering could be based upon. As said before, the MUSE instrument works in the wavelength range of 4750 Å to 9350 Å. In order to analyze both red-shifted Balmer and Calcium lines, they have to lie in this interval. With that, the redshift range can be calculated by finding the minimum redshift where CaK(λ3933.66Å) and CaH(λ3968.47Å) are still in the spectral range and the maximum redshift where Hα(λ6564.614Å) is still present:

$$\begin{aligned} 3934.78\text{Å}\cdot(1+z) &> 4750\text{Å} \qquad & 6564.614\text{Å}\cdot(1+z) &< 9350\text{Å} \\ z &> 0.20 & z &< 0.42. \end{aligned} \tag{16}$$

The filter parameter is therefore set to only account for galaxies with a redshift between $z = 0.20$ and $z = 0.42$. These two values could be imported by **global**

---

[9] yes, error!

variables, after which the master file could be skimmed through, saving only the IDs of those galaxy spectra in the respective redshift range. After this very important first step, the spectra will be imported into the analysis routine. To import the data properly, one has to first know, how the data are structured. As mentioned before, the MUSE data consists of 831 1d spectra decoded in fits files and one master file. These 1d spectra files contain the flux data with a separate flux uncertainty array and the respective wavelengths in air and vacuum frame. The master file contains, among other data, the ID, right ascension, declination and redshift for every galaxy. For the purposes of this thesis, only the ID and redshift are taken from the pre-existing catalog. The redshift is needed to filter the galaxies as described earlier, while the ID is the connection between the master file and the 1d spectra. After filtering, the algorithm needs a way to find the suitable galaxy spectra with the ID given. Rather than reading all spectra at once and then proceeding with the analysis, it would be more efficient to draw a loop over the whole analysis and run it on every galaxy individually. That way in the end, one can run the algorithm on multiple CPUs simultaneously, reducing the processing time tremendously.

The next step will be estimating the strength of some chosen spectral lines by equivalent width and line flux estimation. The procedure will be similar for every line, regardless of it being an absorption or emission feature except when determining the line flux, which has a different meaning for absorption lines. There are special line-finding algorithms that can detect spectral features, but because in this thesis the focus should lie on just the two Balmer Hα and Hβ and the two CaK and CaH lines, such routines will not be used. Instead, **global variables** are prepared for the fixed vacuum wavelengths of all lines. The algorithm then performs a fit for a certain range around this specified wavelength. This range could be a fixed parameter depending only on the redshift and hence the actual position of the line of interest in the data. This approach will be used for both Calcium lines as well, but might have to be changed, since these lines are part of one feature that takes the shape of a doublet.

6.2 Continuum Fitting

The basis for the further spectral analysis will be a continuum fit. Thus, the focus needs to lie on getting a very good approximation for every individual spectrum. In order to estimate the *goodness* of a fit, one can use the χ^2 (9) value. Unfortunately, this does not give an estimate on how good the fit is, but how good it fits the data, with no way of knowing whether the data is actually correct or not. Continuum fits can often be computed theoretically because they are closely tied together with

the physical background. For individual stars, for instance, one can, in a first-order approximation, assume a black body curve with the star's effective surface temperature as its continuum. This is suitable and sufficient for most hot stars with a small amount of absorption features, while exceptions from that are usually solvable by using templates either computed theoretically or created from observed data.
Since a galaxy is an accumulation of many stars of different stellar populations with different magnitudes, temperatures and metallicities, in theory a model curve for every galaxy given its different stellar populations and composition could be created. This has been done already and is being done to this day under the name of SED[Bae19], but because of its complexity it is not suited for a Bachelor thesis and thus will be not considered here. Instead, one part of this thesis will be an approach in approximating the continuum with polynomials. It will be examined whether this leads to acceptable results one can afterwards work with and what problems will occur on the way to this result.

6.3 Spectral Analysis

With a basic continuum fit, one can now start with iterating over all features and run a routine computing the strength of respective features by estimating equivalent widths and line fluxes. While the line flux is rather easy to compute, the basis for determining equivalent widths will be yet another fitting procedure, now approximating the shape of the line. When fitting a model to data, it is always crucial to analyze the data beforehand. Obviously, there is no reason to fit a Gaussian curve to a data set without expecting a Gaussian shape in the first place, but as mentioned in section 4, there is reason to believe that the shape could follow a Gaussian profile. Therefore, for this project and for the corresponding lines, a Gaussian profile is used, but for further development of this project, other line forms could be used, too. As there are a lot of different line finding algorithms, there are also a lot of different line fitting algorithms.

To keep things on an appropriate level, the generic data fitting procedure introduced to PYTHON by the **`scipy.optimize`** package will be used for this step. The included function **`curve_fit`**[Vir+20] assumes negligible uncertainties in one dimension of 2 dimensional data, which fits nicely to this data since it is safe to assume there are negligible uncertainties in the wavelength range, leaving only those in the flux value. In the ESO MUSE user manual it is written that "[t]he final accuracy that can be obtained depends in many other factors, but values around 0.1 Å or better have been reported."[10] This also follows a certain rule of thumb for many instruments of roughly 1/10 pixel size in wavelength accuracy. Applying this function to all four spectral lines produces fit parameters which can later be used to determine the equivalent width with a respective uncertainty.

---

[10] `https://www.eso.org/sci/facilities/paranal/instruments/muse/doc/ESO-261650_MUSE_User_Manual.pdf`

7 Applying the Algorithm

Although the design seemed comparably simple to execute, over the course of writing the program, a lot of unexpected problems occurred that resulted in a more complex code. These problems, along with their solution, will be outlined in the following. To describe the finished program in detail, a flow chart has been created to show the different steps of the program, which can be seen in the Appendix (Figure 16). For the purpose of description it can be divided into four main parts:

- Data Import and Target Preselection
- Continuum Fitting
- Spectral Analysis
- Data output.

The code will be displayed partially and simplyfied for the purposes of description in the following chapter. The whole program can be seen in the appendix.

7.1 Data Import and Target Preselection

The algorithm was supposed to start with reading in the above mentioned master file containing all 831 galaxy spectra. However, when applying the algorithm to a different catalog it was made obvious that unfortunately and incomprehensibly, there is still no true universal standard to providing big data sets in catalogs. During implementation of other data sets, already a change in nomenclature of column names, or the differences in the way the individual spectra are saved, made it impossible to blindly run the algorithm on multiple data sets without first checking, how the new data are provided. This is why new data sets first have to be restructured and standardized to work with any software package, which is actually the only step, where manual interaction is indispensable when implementing a new catalog. The restructuring has been done by saving all spectra within a specified folder. This is part of the preprocessing and requires no manual interaction later on. Also every galaxy should have a redshift value assigned to it, which will be saved together with the ID in a **Master File** in the same folder. The algorithm checks whether such a file is present or not and requires additional redshift information in the latter case. After preparing and importing the data set, the algorithm proceeds to its first main task, which is filtering the galaxies according to their redshift. The above calculated redshift range (16) is saved by two global variables, `Z_MIN` and `Z_MAX`, that are given their respective values. Since this part is rather self explanatory, it will not be included as a code snippet in the text. The resulting array contains only the IDs of those galaxies with respective redshift values. This routine is essentially only important to get an estimate on the amount of galaxies that are suitable for analysis before running the main algorithm and to implement the location where the data should be imported from.

As established earlier in section 6.1, the following main analysis part of the program is enveloped by a **`for-loop`** starting with saving the ID and redshift of a galaxy from the master file in two separate variables **`galaxy_id`** and **`galaxy_redshift`**. It then proceeds with the actual filter routine by continuing with the current galaxy, in case the redshift of the current galaxy is in the specified range or skipping to the next one when this condition is not met. Also the counter is incremented once a sufficient galaxy is found to keep track of the number of galaxies that already have been analyzed.

```
 1  MUSE_Data = fits.open(MASTER_FILE)
 2  TABLE = MUSE_Data[1].data
 3
 4  for galaxy in TABLE:
 5      galaxy_id, galaxy_ra, galaxy_dec, galaxy_redshift,
 6      galaxy_redshift_unc = galaxy[0:5]
 7      spectrum_file = 'spectrum_%s.fits' % galaxy_id
 8      filename = join(base_dir, spectrum_file)
 9      spectrum = fits.open(filename)
10
11      WAVELENGTH = pyasl.airtovac2(spectrum[1].data['WAVE_AIR'])
12      FLUX = spectrum[1].data['FLUX']
13      FLUX_UNC = spectrum[1].data['FLUXERR']
14      LINES = [[HALPHA, 0, 'Halpha'], [HBETA, 0, 'Hbeta'],
15               [CAK, 1, 'CaK'], [CAH, 1, 'CaH']]
16
17      if galaxy_redshift > Z_MIN and galaxy_redshift < Z_MAX:
18          GALAXY_COUNT +=1
19          #Continue with current galaxy
20      else:
21          continue
22          #skip to the next galaxy
```

The data in most catalogues are usually saved in FITS[11] files[Pen+10] by using so called record arrays. This methods makes it possible to assign a certain description or name to the columns, which later can be used conveniently to access the column, where for example the wavelength data are stored (see line 11). Unfortunately, this

---

[11] Flexible Image Transport System

also introduces the problem of a universal description norm. When introducing a different data set to this algorithm, it was observed that the keyword for column description changed from 'WAVE_AIR' to 'WAVE', which is again why restructuring is necessary. Also, since the telescopes with which the data were observed are terrestrial, the wavelengths are usually given in air frame, which is converted into vacuum frame wavelengths by a routine from the pyasl[Cze+19][12] package in the code. There is also the alternative of using the vacuum wavelengths directly from the data set, but to account for other sets, where only the air wavelength might be given, this is converted to vacuum wavelengths in the script. The following `LINES` array introduces all features and spectral lines that are to be examined. The way it is implemented will later be discussed.
The next step after preselection will be fitting a continuum.

7.2 Continuum Fitting

As described in the previous section, this routine will try to approximate the continuum of a given galaxy spectrum by using a polynomial. The general shape of the spectra leaves only a polynomial approach, since there is no explicit (simple-to-implement) formula for the continuum of a galaxy. Just by looking at some examples like the on seen in figure 3, it is obvious, that the given polynomial must be of high order. A polynomial of lower order might not approximate the spectrum successfully, while a polynomial of even higher order is not well constrained and also runs the risk of fitting local features of the spectrum rather than the continuum flux itself. This becomes evident when examining the environment around the Hβ emission line, which lies in a very large absorption region. This region is not part of the natural continuum of the galaxy, but would be falsely taken into it by a polynomial with a much larger degree. To prevent this, after examination, a polynomial of order 10 has been chosen. The following picture presents the result after estimating the continuum without any further modifications to the data and the one with improvements that are described below.

---

[12] Documentation found under `https://github.com/sczesla/PyAstronomy`

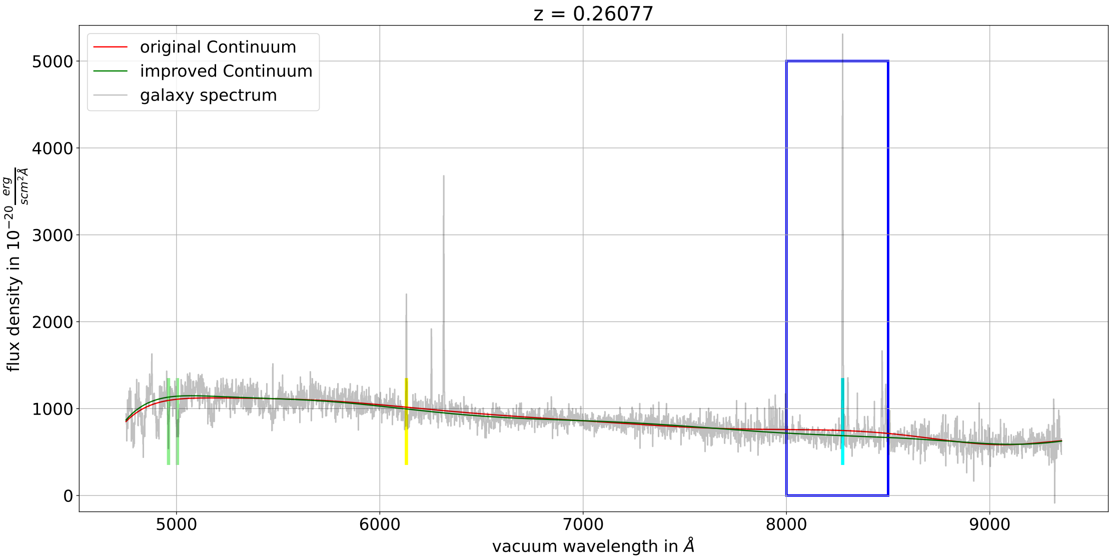

Figure 3: A typical galaxy spectrum created from the MUSE Wide Survey is displayed. The examined spectral features are marked as follows: CaH & CaK in green, Hβ in yellow, Hα in blue. The preliminary and improved continuum fits are shown as red and green lines respectively, while a next plot is zooming in on the Hα line.

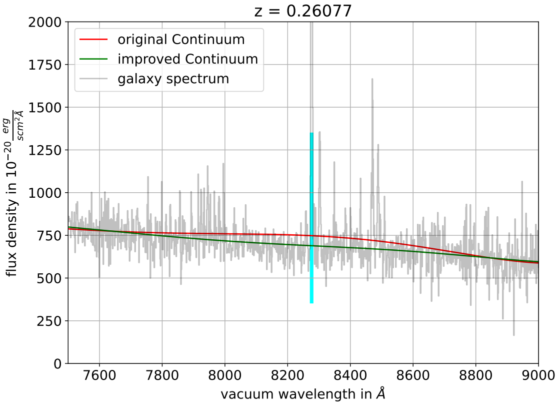

Figure 4: A zoom in on the Hα line demonstrates the significant deviation of the original continuum fit from the improved one resulting from the large spectral line itself.

At first look, the red curve is already a very nice approximation for the galaxy continuum, but at second glance (as seen on the right in figure 4), it is clearly visible, that very strong emission lines like Hα and Hβ and also both CaH and CaK absorption lines in the spectrum influence the red polynomial fit heavily. Because the fitting procedure tries to minimize the squared distances of data points to the fit, χ^2 is being overestimated for outliers like the Hα emission line. The algorithm needs to have a way of dealing with these outliers properly.
Since these features are not part of the continuum and should therefore not be fitted with this approximation in the first place, a method could be found to estimate the continuum in absence of those intense emission and absorption lines.

One way of doing so could be to take the average of the spectrum and remove every data point above and below a certain threshold around the average. This method, however, cannot be used for spectra like the one shown in Fig. 3, since

taking one average value over the whole spectrum simply does not suffice. To solve this problem, one could divide the spectrum into smaller parts and compute the average in these parts, which would work, but makes the whole procedure rather messy. Considering that, apart from some intervals, the original fit was already approximating the continuum well, it could be used in the above mentioned methods in place of the average value. That way, there is already a rough estimation of the continuum around which data points above a certain threshold are cut out.
The following code snippet shows the function `_continuumFit`, a full routine, that takes both the wavelength and flux arrays for one galaxy spectrum and a number of iterations as input and calculates an improved continuum fit of arbitrary degree.

```
 1  def _continuumFit(wavelength_Array, flux_Array, iterations, degree)
 2      temp_Wavelength = numpy.copy(wavelength_Array)
 3      temp_Flux = numpy.copy(flux_Array)
 4
 5      #Fit a preliminary continuum with a loose polynomial of 10th degree
 6      fit_Coefficients = numpy.polyfit(temp_Wavelength, temp_Flux, degree)
 7      polynomial_Fit = numpy.poly1d(fit_Coefficients)
 8      temp_Continuum = polynomial_Fit(temp_Wavelength)
 9      original_Continuum = temp_Continuum
10
11      #Improve the preliminary continuum fit
12      for i in range(iterations):
13          SIGMA = numpy.std(temp_Flux-temp_Continuum)
14          temp_Flux = ma.masked_where(abs((temp_Flux-temp_Continuum)) > 3*SIGMA,
15                      temp_Flux)
16          fit_Coefficients = numpy.ma.polyfit(temp_Wavelength, temp_Flux, degree)
17          polynomial_Fit = numpy.poly1d(fit_Coefficients)
18          temp_Continuum = polynomial_Fit(temp_Wavelength)
19
20      Continuum = temp_Continuum
21      return Continuum, SIGMA
```

The foundation of said continuum is a first approximation by a 10th order polynomial (lines 6 - 9). Like described earlier, this alone would lead to false results, but it is a very good basis for the improved continuum. The following `for-loop` in lines 12 to 18 improves upon this preliminary fit. It starts by calculating the standard deviation `SIGMA` of the flux in accordance to the continuum. Of course, the goal of this function is to minimize this value throughout the spectrum and therefore it is repeated a number of times according to the value of `iterations`. After calculating the deviation of flux and continuum fit, the flux array is masked according to where the difference between it and the baseline exceeds a given threshold, here set to be 3·`SIGMA`. The resulting masked array consists of the original flux data, but wherever the flux value was too high or too low according to the threshold, this value is masked. The reason, why masked arrays are being used instead of deleting entries, is that masked arrays contain their length. This greatly improves the handling in the following script.

After coming up with this routine, researching lead to an approach already known as sigma clipping, a technique that cuts all data points out of a set that are over or under a given threshold defined in units of standard deviation. Hence, there is already such a function, called **`sigma_clip`**, implemented in the **`Astropy.stats`**[Cze+19] library. Although it works fine with the algorithm, since the main goal of this thesis is to learn about spectral analysis, it posed a very nice challenge to not use predefined subroutines, but developing them as needed. That way one has control about the function itself and knows exactly what it does.

Following masking the respective values, another polynomial fit is created based on the new flux array, overwriting the original continuum. These steps are then repeated, decreasing **`SIGMA`** further. By plotting the value of σ to the number of iterations, one can clearly see that the value of sigma decreases rapidly for the first iterations and then converges after a number of steps. When choosing the number of iterations given to the function, it has to be accounted for the fact that the number of steps until **`SIGMA`** converges is not constant for every galaxy [see Figure 5]. Therefore, after examining a number of different galaxies, a value of 20 has been chosen to also account for possible outliers.

After 20 iterations of adjusting the continuum, a final fit is returned together with the latest value for **`SIGMA`**. Incidentally, this value for **`SIGMA`** is already a measure for the uncertainty of the approximated continuum, which is needed for estimating the uncertainty in line flux (8).

Although designed with great care, throughout the program this continuum fit plays only a subordinate role, since for every line, another continuum fit has to be made in the region of the respective feature, but it is a great basis for said line continuum approximations.

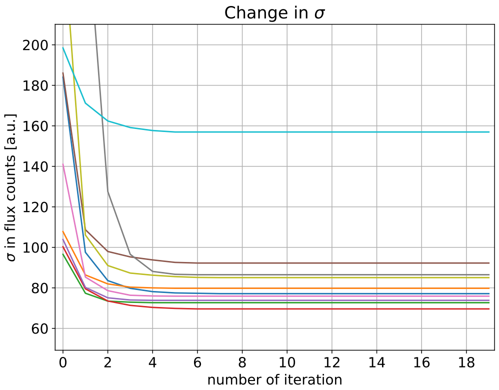

Figure 5: Change in σ over all iterations for different galaxy spectra. Already after a small number of iterations, the value in sigma approaches a constant value.

Before starting with the spectral analysis part however, a last quantity is estimated, that corresponds to the *quality* of the spectrum. The so called Signal To Noise Ratio (SNR) is computed in five regions of the spectrum by dividing the mean value by the standard deviation of the flux in this region. The average is then taken as the **`SNR`** of the respective spectrum, which will be used later in estimating the uncertainty in equivalent width.

7.3 Spectral Analysis

Now that preparations are finished, another **`for-loop`** over every entry in the above mentioned **`LINES`** array carries out the spectral analysis. This arrangement makes it very convenient and also user-friendly to add new features to be analysed just by adding a new entry to this array. The following code snippets show the implementation of the **`LINES`** array followed by the main parts of this analysis routine with some minor or less relevant parts left in for demonstration purposes.

```
LINES = [[HALPHA, 0, 'Halpha'], [HBETA, 0, 'Hbeta'],
        [CaK, 1, 'CaK'], [CaH, 1, 'CaH']]
#[redshifted wavelength of the line, 0 for emission line 1 for absorption, name]
```

```
stepWidth = _determineStepWidth(wavelength_Array)
for feature in LINES:
    #feature = [wavelength of  line in Angstroem, line type, name]
    if(numpy.min(wavelength_Array) > feature[0] or
       numpy.max(wavelength_Array) < feature[0]):
        continue
        #Feature not in spectrum, skip to the next feature

    line_Type = feature[1] #1 for absorption, 0 for emission
    line_Wavelength = _determineNearest(wavelength_Array, feature[0])
    line_Position = int(numpy.argwhere(line_Wavelength == wavelength_Array))

    #Find line region
    regionStart, regionEnd =
        _determineLineRegion(wavelength_Array, flux_Array, line_Position,
                             Continuum, line_Type)

    #Fit local continuum around spectral line
    line_Continuum, line_Continuum_unc =
        _determineLineContinuum(wavelength_Array, flux_Array, flux_unc_Array,
                                Continuum, Continuum_unc, line_Region)

   #Normalize for equivalent width estimation
   flux_normalized_Array = 1 - flux_Array/line_Continuum
```

This first part of the analysis section starts with a small but incredibly important subroutine called **`_determineStepWidth`**. It only computes the pixel width of the spectrum, but since this algorithm is supposed to be universal and not built only for the MUSE instrument, it is necessary to make the pixel size a variable to be determined from the data set as well.

Afterwards a **`for-loop`** is drawn over all features and spectral lines that are to be analyzed. This loop includes a lot of *Fail Saves* to account for damaged or degenerate data, that have been implemented in order to give the algorithm a way to deal with these types of data sets without it crashing or needing human interaction. The list of possible fail saves is most certainly going to grow with every new catalog imported, but the current one should suffice for the purposes here. First on the list and an example for such a *Fail Save* are three lines of code in lines 4 to 6, that check, whether the feature is actually in the spectrum and not cut out because of some reason. Although the preselection routine should have sorted this problem out, there were still some spectra that did not cover the whole range over which they were taken, which will probably be an issue with other catalogs as well. This has been resolved with these 3 lines of code. In order to analyze any spectral feature, the algorithm has to know where the line is in the spectrum and in the data, whether it is an emission or absorption line and in what region the line sits. This is done by two small subroutines namely **`_determineNearest`** in line 10, which finds the wavelength in the spectrum that corresponds best to the predetermined wavelength and **`_determineLineRegion`** in line 15, which determines the region, where the line resides. Normally one could just fit the line with a Gaussian without having to know about the region where it lies, but for the following continuum fit this was necessary. This routine increments over all pixels of the line starting from its center to higher and lower wavelengths as long as the slope between neighbouring points is below or above zero for emission and absorption respectively. The region between these two points is then the region where the line resides.

In section 4.2, it was already mentioned that there can be an absorption region underlying both Balmer emission lines. This also had an influence in the prior continuum fitting procedure for the whole spectrum. Because these regions are not part of the continuum, but have to be accounted for when normalizing the flux onto the continuum, another fit has to be done. The procedure will be largely based on the existing method described in section 7.2, again using a hand-written sigma clipping routine to cut out the actual line before fitting. This is also why there has to be an estimation of where the line sits on top of or below the continuum in order to specify the interval, where this has to be cut out to perform a proper approximation for the underlying absorption region. As discussed earlier in section 4.2.1, this absorption region should again be of Gaussian shape, but to keep things simple and because it did not have a great impact on the results, a polynomial of second degree was sufficient as well.

7.3.1 Equivalent Width Estimation

With these steps done, the equivalent width can be estimated. Following the steps described in section 4.2 using equation (3), the normalized flux is fit by another Gaussian of which the scaling factor corresponds to the equivalent width of the line respectively. Another way of doing so would be to simply sum the normalized flux over all pixels multiplying by the pixel width, comparable to the line flux estimation. An important part that has to be accounted for is the estimation of respective uncertainty for the equivalent width values. When using fit parameters (in this case the scaling factors a and expected values λ_0) as measurements, it cannot be computed with standard uncertainty propagation, but has to be determined from the fit itself. Thankfully, the used **`curve_fit`** routine has an inbuilt feature giving out a so called covariance matrix, with the diagonal entries providing the variance of fit parameters. The uncertainty can then be computed to be the square root of those entries.

However this step poses yet another challenge for an own implementation because when there exists no way of acquiring the uncertainty of one parameter or measurement, it is often possible to artificially generate/simulate it. This has been done with the following subroutine called **`_equivalentWidthSim`**. This approach is related to the generic approach known as Monte Carlo.

```python
def _equivalentWidthSim(flux_Array, flux_unc_Array, wavelength_Array,
                        deviation, stepWidth, iterations, line,
                        line_Region, fit_Region, redshift):
    for i in range(iterations):
        flux_distribution_Array
            = flux_Array
            + numpy.random.normal(0, (1/deviation), size=len(flux_Array))

        fit_Parameters, fit_Covariance =
            curve_fit(_gauss, wavelength_Array, flux_distribution_Array)

        #Equivalent Width by fitting
        EW_Area = fit_Parameters[0] / (1+galaxy_redshift)
```

This routine adds Gaussian deviations onto the flux values by adding a randomized normal distribution on top of the original flux array (see line 2-4). This essentially probes the way how small Gaussian deviations affect the resulting fit parameters and thus the resulting equivalent widths. This step is repeated 200 times (see figure 6 with 20 iterations) with different deviations for every iteration and every pixel. The spectrum is then fit by a Gaussian with variable parameters. From these fit parameters, the scaling factor and the maximum position are saved. The scaling factor is the equivalent width estimate, while the maximum position will be used to calculate the redshift. After the specified number of iterations, both resulting values and uncertainties are then taken according to following description. It will later be discussed, whether this leads to scientifically relevant results and the uncertainties will be compared with those provided by the **`curve_fit`** function.

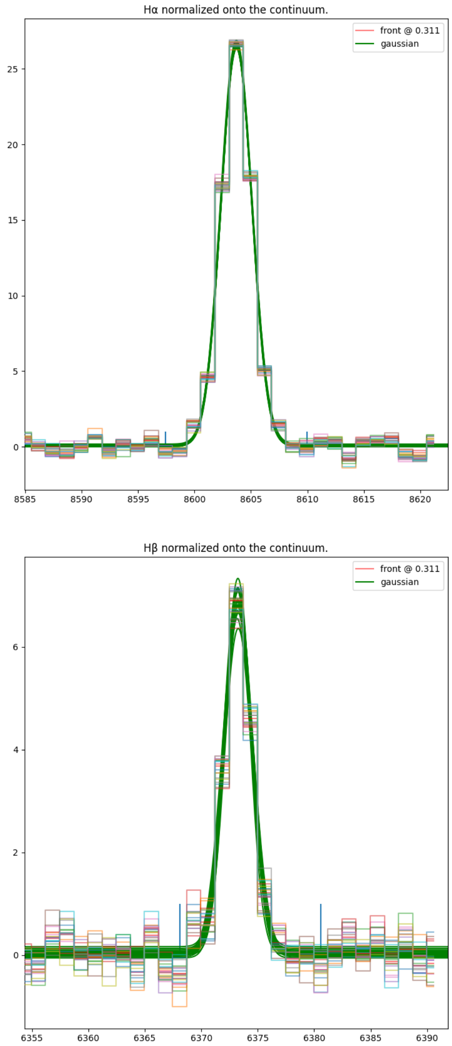

Figure 6: Simulation routine with 20 iterations.

Because the resulting fit parameters usually scatter around a certain value with some outliers, where fitting was unsuccessful, the median of all 200 measurements and not the average is afterwards taken to be the final result. Hence, the parameter uncertainty is the absolute median standard deviation of all measurements. The two figures on the left (fig. 6) are used for visualization of 20 iterations. It can be seen that, since the ratio of Hα to Hβ is always bigger or equal to 2.859 (see section 4.4), small deviations have a larger impact on the individual pixels and thus on the equivalent width and relative equivalent width uncertainty of the Hβ line. That is also why in Hβ it is often larger than in Hα (this will also be discussed later). It also has to be mentioned that this method introduces a deviation on top of the already present uncertainty in flux that is given by the instrument.

Although the resulting values of this method will be discussed later, at this point there is another result to be mentioned. It was initially introduced in order to estimate uncertainties, but as a very convenient side effect, this routine can sometimes predict, whether spectral features are actually present and distinguishable from noise or not. When working with very low Signal to Noise ratios, which corresponds to very noisy spectra, the Gaussian deviations get very large, because they scale with 1/`SNR`. The results are a large number of different approximations for the same feature, when this routine is run multiple times. The resulting equivalent width uncertainty, because of it being the median absolute standard deviation of the fitted parameters, will be comparably large. Therefore, when this uncertainty exceeds 1/2 of the equivalent width value, it is safe to say that this measurement was flawed because the spectral line is indistinguishable from the background noise. Lines, identified in this way, are saved as undetected (see table 1). On the other hand, when working with very prominent features like Hα for example, these Gaussian deviations, while still scaling with the SNR of the whole spectrum, are comparably smaller to the flux values of each line pixels.

7.3.2 Line Flux Estimation

Using most of the previous steps, calculating the line flux can now easily done by just one function that gets all the earlier estimated continuum fits as input. Although it is not complicated, it will be displayed in the following for sake of completion.

```
def _estimateLineFlux
    (flux_Array, flux_unc_Array, line_Continuum, line_Continuum_unc,
     Continuum, Continuum_unc, stepWidth, line):

    if(line == 'Hbeta'):
        flux_region = flux_Array-line_Continuum
        continuum_unc = line_Continuum_unc
    else:
        flux_region = flux_Array - Continuum
        continuum_unc = Continuum_unc

    line_flux = numpy.sum(flux_region*stepWidth)
    line_flux_unc = stepWidth *
                    numpy.sqrt(numpy.sum((flux_unc_Array)^2+(continuum_unc)^2))

    return line_flux, line_flux_unc
```

The above listed code calculates the line flux accordingly to equation (6) and uncertainty, respectively. The procedure is slightly different for the Hβ line because of the underlying absorption region. It did not have a significant impact on the Hα line, which is why the previous continuum was taken for calculation without approximating another line continuum.

7.4 Data Output

The resulting data is afterwards saved in a human readable ASCII format text file and has the following structure:

$$\text{ID} \mid \text{SNR} \mid \text{H}\alpha \mid \text{H}\beta \mid \text{CaK}$$

With every feature subdivided into:

Quantity	Short Description	Unit
EW	Equivalent Width of the line*	Å
EW Uncertainty	Equivalent Width Uncertainty of the line*	Å
Line Flux	Line Flux of the line*	10^{-20}erg/s/cm^2
Line Flux Uncertainty	Line Flux Uncertainty of the line*	10^{-20}erg/s/cm^2
Line Continuuum	Average of continuum around the line	10^{-20}erg/s/cm^2/Å
Flux Density Uncertainty	Average of the Flux Density Uncertainty around the line	10^{-20}erg/s/cm^2/Å

Table 1: Structure of every galaxy entry of the resulting data set. *Whenever the line is undetected, the respective value is set to zero

The code, as well as a read me file with instructions on how to operate it, will be provided along with some mock spectra for demonstration porposes.

8 Result

As indicated before, the resulting data from the program has to ultimately be verified whether it can actually be used for further scientific research. The gathered equivalent widths and line fluxes will therefore be displayed and their relevance will be examined using the described extinction and dust attenuation model. Afterwards, the calculated redshift values will be shown and compared to those given by the data.

The program, as well as some example spectra, a read-me file and instructions on how to operate it will be provided in my directory[13] and to anyone interested.

8.1 Line Flux Measurements

In order to prove the program's ability to also be used with other data sets, aside from the MUSE Ultra Wide Survey and to increase the sample size for better interpretation, another data set from the MUSE Instrument has been introduced. The MUSE Hubble Deep Field South is yet another set of galaxy spectra with different redshifts. Unfortunately as seen below, only nine of the 585 galaxy entries were in the respective redshift range, with seven of them showing significant Hα and Hβ features. The MUSE Ultra Wide Survey data, however, provided 92 galaxies out of 381 in the respective redshift range, with 70 showing both Balmer features according to the condition introduced in section 7.3.1.
Displayed below are the distributions of line flux measurements for both Balmer lines from both data sets.

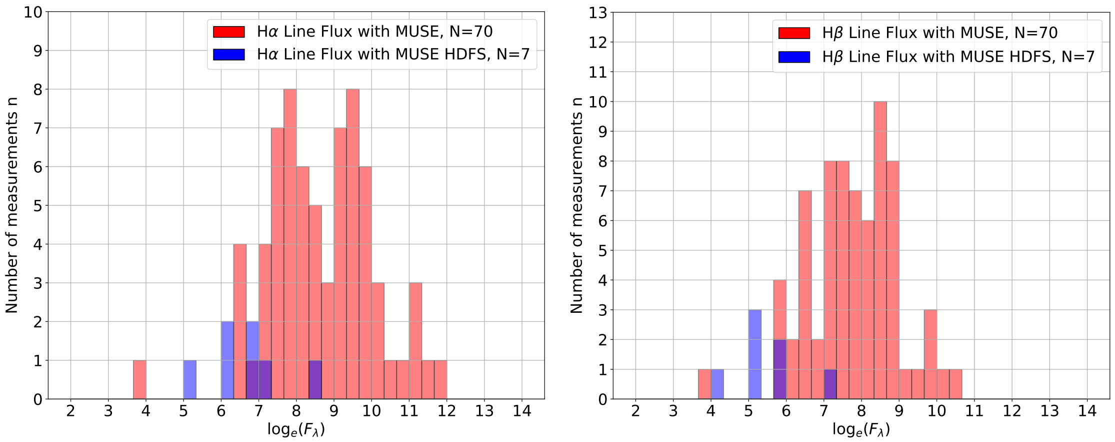

Figure 7: The above plots show the distribution of measured line fluxes for both Hα(left) and Hβ(right) for the above mentioned MUSE Ultra Wide Survey(red) and an additional catalog called the MUSE Hubble Deep Field South[Bac+15](blue).

---

[13] /home/pollux/mitzerott/2022/BachelorThesis/

It is clearly visible that introducing the second data did not increase the sample size reasonably, but in the end confirmed that the program can indeed be used on other data sets. The measured line fluxes show a distribution in logarithmic line fluxes with maxima between 8 and 10 for Hα and 7 and 9 for Hβ respectively.
In order to verify the line flux measurements, the above mentioned extinction and dust attenuation model will be applied to the data. This is being done by plotting the measured line fluxes of Hα against that of Hβ. According to the model introduced in section 4.4, the data should follow a linear fit with slope $F_{H\alpha}/F_{H\beta} = 2.859$ in a laboratory frame without interstellar dust. With dust present, the data points should lie slightly above this curve.

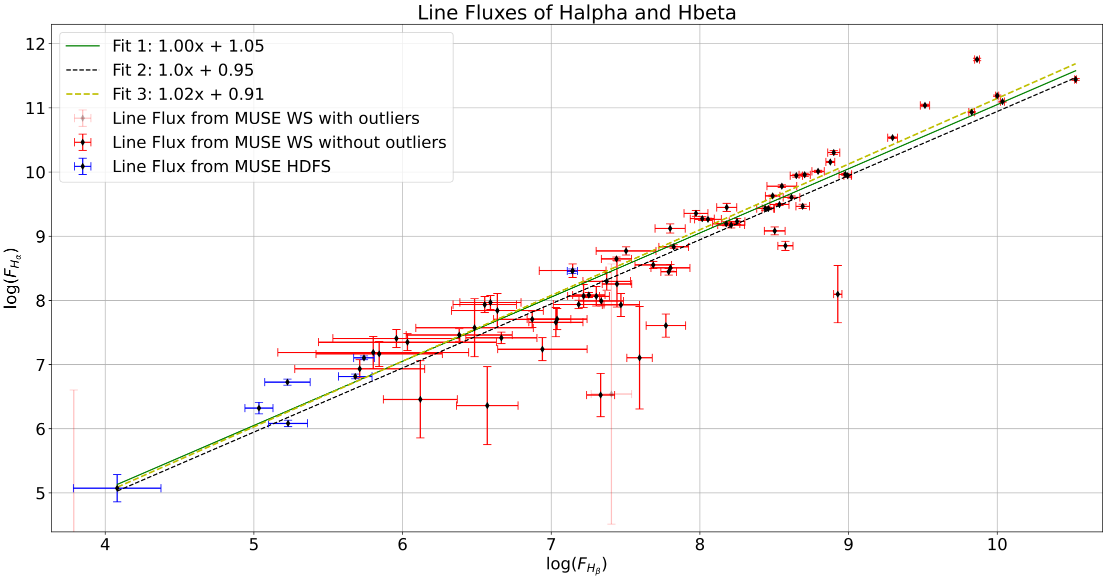

Figure 8: The logarithmic line fluxes of Hα vs. Hβ are shown. Color coding applies as usual. The data is approximated by two separate fits, with one approximating the data without accounting for uncertainty(black), while the other one accounts for uncertainty in both dimensions(yellow) as mentioned in section 4.3. An additional curve shows the laboratory model(green). The relative uncertainty is displayed by the red and blue error bars respectively, while outliers with unreasonably high uncertainty are greyed out and do not contribute to the approximations.

At first glance, it is clearly visible, that most of the data points lie within a certain interval of the green line and are approximated nicely by both fits. Since the data are plotted logarithmically, one would expect a model curve to have a slope of 1 and an interception of $\log(2.859) = 1.05$, which the yellow and black fits are close to:

$$\mathrm{F(H\alpha)} = 2.869 \cdot \mathrm{F(H\beta)} \Rightarrow \log(\mathrm{F(H\alpha)}) = 1 \cdot \log(\mathrm{F(H\beta)}) + \log(2.859).$$

There are, however, some drastic outliers, where $\mathrm{F_{H\alpha}} < \mathrm{F_{H\beta}}$, which cannot be physically correct and therefore do not contribute to the fits. This could be explained from errors in the line flux estimation routine, for example a wrong or flawed line continuum fitting procedure, which did not account correctly for the underlying Hβ absorption. It can also be observed, that for most of the data points, the relative uncertainty of line flux in Hβ is significantly bigger than that in Hα. This has been discussed earlier in theory in section 7.3.1, but has now been verified. The reason, why relative uncertainties have been used, is that otherwise in this logarithmic plot, the data points would have asymmetric uncertainties, which would pose an unnecessary challenge for the fitting procedure. However, data points at higher line fluxes tend to show little to none uncertainties. This needs to be attended and will be one part of the improvements, future versions of the program have to include. It can be observed from the data, that higher line flux values tend to be in close connection with higher Signal to Noise ratios, therefore lower overall uncertainty in flux, which could lead to small uncertainty in line flux. They could however also just seem small due to the logarithmic plot and them being relative and not absolute uncertainties. However, when picking an example in the top right corner of the plot, the relative uncertainties in line flux are 0.031 in Hα and 0.016 in Hβ.

My supervisor Martin Wendt also ran the finalized tool on an unpublished catalog of galaxy spectra to evaluate its performance on another dataset the code was not trained and designed for, specifically. The plot below shows the results for Hα and Hβ fluxes:

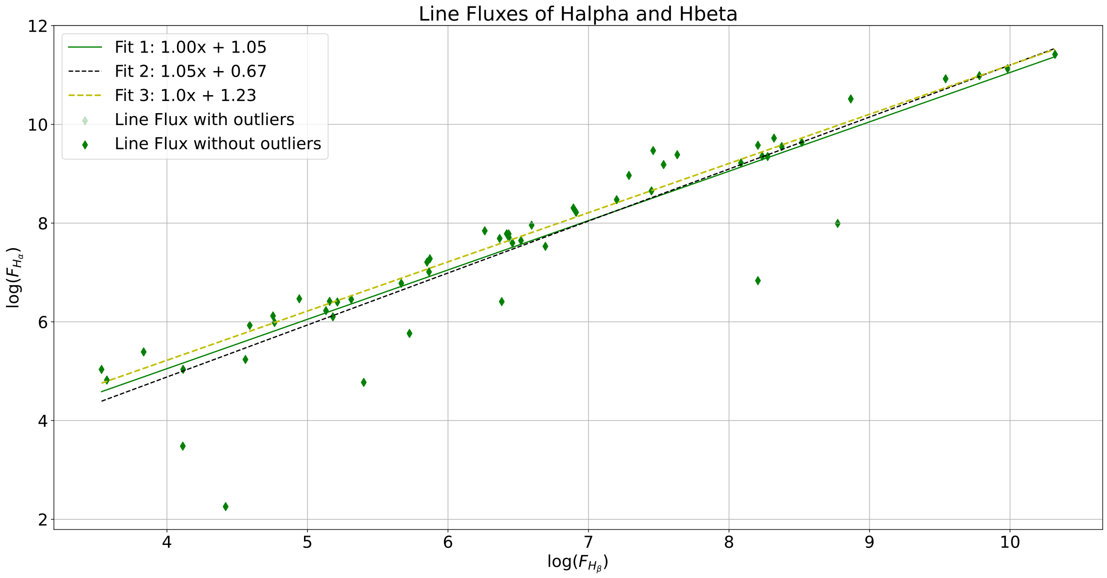

Figure 9: The above plot shows the results of a yet unpublished revision of a dataset from the MUSE instrument[Bac+17]. Credit belongs to Roland Bacon (contributing author of [Bac+17], [Bac+15] and [Her+17b]), from whom permission has been granted to use these datapoints in this Bachelor thesis. Fitting procedure applies as described in the above plots.

As can be seen, this plot continues the trend, visible in the above listed plots and proves the availability of the code to also be used on other data sets and, most importantly, produce scientifically relevant data concerning ratios of line flux measurements.

8.2 Equivalent Width Measurements

As mentioned in section 4.4, in the following, it will also be examined, whether the equivalent width measurements can also be used to create a relation similar to the one, shown in figure 8. As mentioned, this would be *independent of resolution* and would therefore allow usage of data, otherwise not well enough resolved for line flux estimation.
The Equivalent Width distribution is shown below in figure 10.

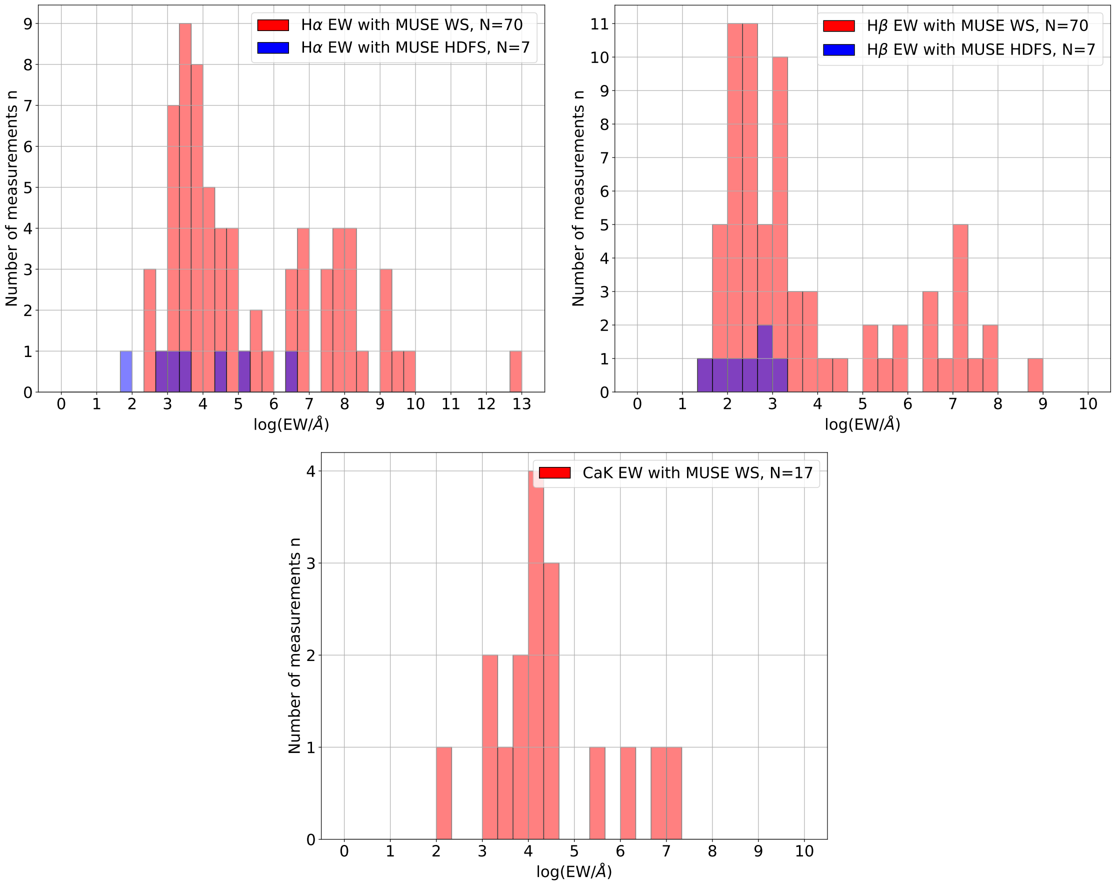

Figure 10: The above plots show the distribution of measured equivalent widths for both Hα(left) and Hβ(right) and CaK(bottom) for the above mentioned MUSE Ultra Wide Survey(red) and an additional catalog called the MUSE Hubble Deep Field South[Bac+15](blue). It needs to be mentioned, that the MUSE Hubble Deep Field South did not contain any spectra in the used redshift range showing sufficient CaK absorption features

At first glance, it can be seen that both logarithmic Balmer line equivalent width distributions have their maximum between 3 and 4 for Hα and 2 and 3 for Hβ. The sample size, however, is limited still, which makes statistical analysis of the resulting data difficult and prone to error. It can be also seen, that despite clumping around mentioned ranges, there are outliers in both distributions, that cannot simply be explained by statistical deviations. Although possible, most of these outliers, especially the one with a logarithmic EW of close to 13 can be traced back to cases where the underlying continuum is close to zero. Thus, when normalizing, one cannot simply follow the equation (3). These cases were observed often and were first solved by not normalizing at all, but estimating the equivalent width from the initial flux densities. Also, with a continuum flux close to zero over the whole spectrum, absorption features cannot be detected, which explains the very sparse amount of CaK measurements in figure 10. It could also be looked into further whether adjusting the set accuracy limit for equivalent width measurements referred to in section 7.3.1 might lead to more usable measurements. As of now it seems that only the very strong CaK features were detected by the program, so the distribution in figure 10 is somewhat biased.

This, however, also introduces the problem that measuring throughout the data set is not uniform anymore, thus another method has to be found. Research by other groups lead to a paper with the same problem that estimated equivalent widths by integrating over the continuum subtracted flux rather than the continuum normalized flux[GBW11]. In the following, resulting plots of both methods will be displayed and discussed: Method 1, which applies the formula known for absorption lines(3), is displayed above in figure 11. It can be observed, that the number of outliers has increased significantly compared to the line flux measurements. This is largely due to the fact mentioned before, that the continuum around the line was close to zero and therefore the equivalent width was measured differently. To account for this, the second method calculates equivalent widths by integrating over the continuum-subtracted flux density. The result can be seen below:

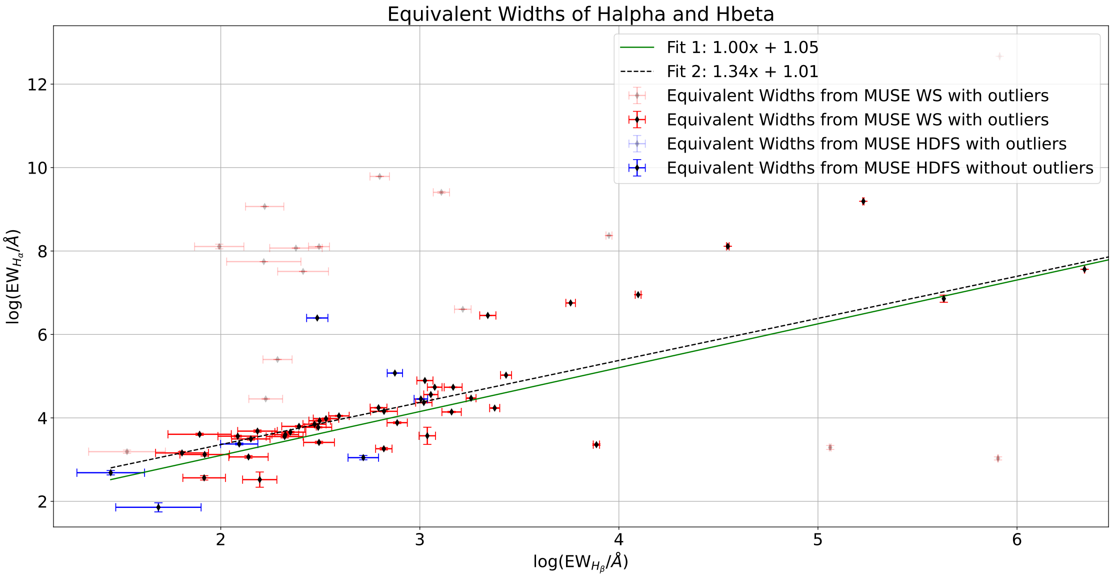

Figure 11: The logarithmic equivalent widths of Hα vs. Hβ are shown. Color coding applies as usual. The data is approximated by one separate fit, which approximates the data without accounting for uncertainty(black); this will be discussed later. An additional curve shows the laboratory model(green). The relative uncertainty is displayed by the red and blue error bars respectively, while outliers with unreasonable uncertainty are greyed out and do not contribute to the approximations.

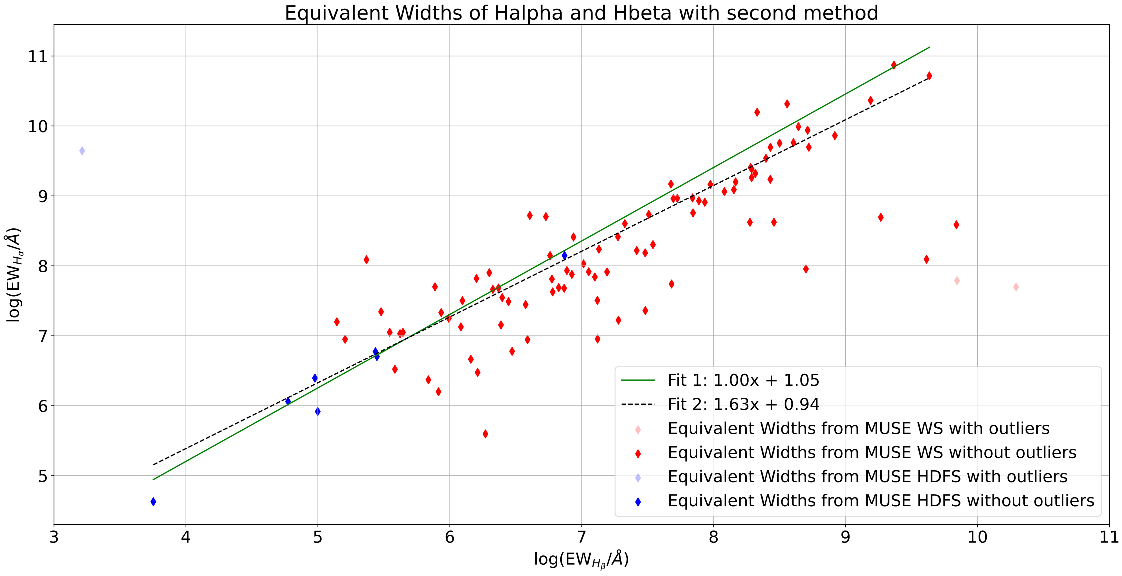

Figure 12: see figure 11. With using the second method, the equivalent width measurements are displayed without uncertainties.

It is immediately obvious that the number of outliers is significantly reduced compared to the other method. The data points are plotted without uncertainties because for estimating those, the **`_equivalentWidthSim`** routine has to be adapted. This is because the deviation gets added on top of the flux density and scales with Signal to Noise ratio, but not with flux densities. Therefore, it is comparably smaller for method 2, since normalization is done by subtraction rather than division. Thus, the impact of the deviations on the fitting procedure becomes irrelevant resulting in a very small overall uncertainty. Still, the linear approximation(black) has an interception of log(2.559), comparable to that of the laboratory frame with log(2.859), which is slightly worse than that of method 1. Also it can be seen, that fits for both methods have a bias of log(3.82Å) and log(5.10Å) respectively. Further research lead back to the above mentioned SDSS paper[GBW11], in which the authors had to introduce a correction factor of $R = 4$Å in order to correct for the "offset of the SDSS' galaxies' EW Balmer decrements to the measured Hα/Hβ ratios"[GBW11]. Regarding the overall uncertainties in equivalent width estimation, it needs to be said that the fit uncertainties given by the `curve_fit` routine are often more reasonable and also scale appropriately with the measurement. However, as these values are calculated also using the predefined fit parameters, significant outliers are present here as well in abundance. That means that there has to be another way of estimating those uncertainties in further iterations of the algorithm and in order to do proper statistical analysis with the equivalent width data.

8.3 Balmer Decrements

As described earlier, a Balmer Decrement higher than 2.859 suggests dust to be present along the line of sight. This could be due to the galaxy's orientation being "edge-on". That means, the line of sight goes through the galactic disk, which contains most of the dust.

Therefore, one could look for galaxies with a Balmer decrement higher than and close to 2.859 in figure 8 and use the 3d data cubes from the MUSE Wide Survey to generate a velocity map for these objects. With a script, one could analyze the shift in velocity space of a very prominent spectral line in every pixel of the object's datacube and calculate it's rest frame velocity. This has been done for 6 hand-picked targets, three of them on or close to the line corresponding to a Balmer decrement of 2.859 and three of them with a Balmer decrement higher than that. The target's line fluxes in Hα and Hβ are displayed along with a value for their Balmer decrement and calculated reddening using equation 14:

object ID	F_{H_α} in Flux Units	F_{H_β} in Flux Units	Balmer decrement	E(β-α)
124002008	39385.72	5797.67	6.79	0.94
105002016	127223.10	19239.85	6.62	0.91
104002019	31068.48	4095.97	7.59	1.06
109005032	21232.16	7929.65	2.68	-0.07
101001006	14844.91	5520.58	2.69	-0.07
118001006	21113.08	6031.02	3.50	0.22

Table 2: This table shows a selection of galaxies from the the MUSE-Wide Survey[Her+17a]. Objects were selected to have either a Balmer decrement close to 2.859 or one considerably larger than that. Also using the above mentioned equation, the color excess has been computed.

As said before, a high Balmer decrement could correspond to an *edge-on* oriented galaxy, while a Balmer decrement close to 2.859 could correspond to one in *face-on* orientation. In order to visualize the data presented above better and to verify this theory, velocity maps for each of the six objects have been created and are displayed below:

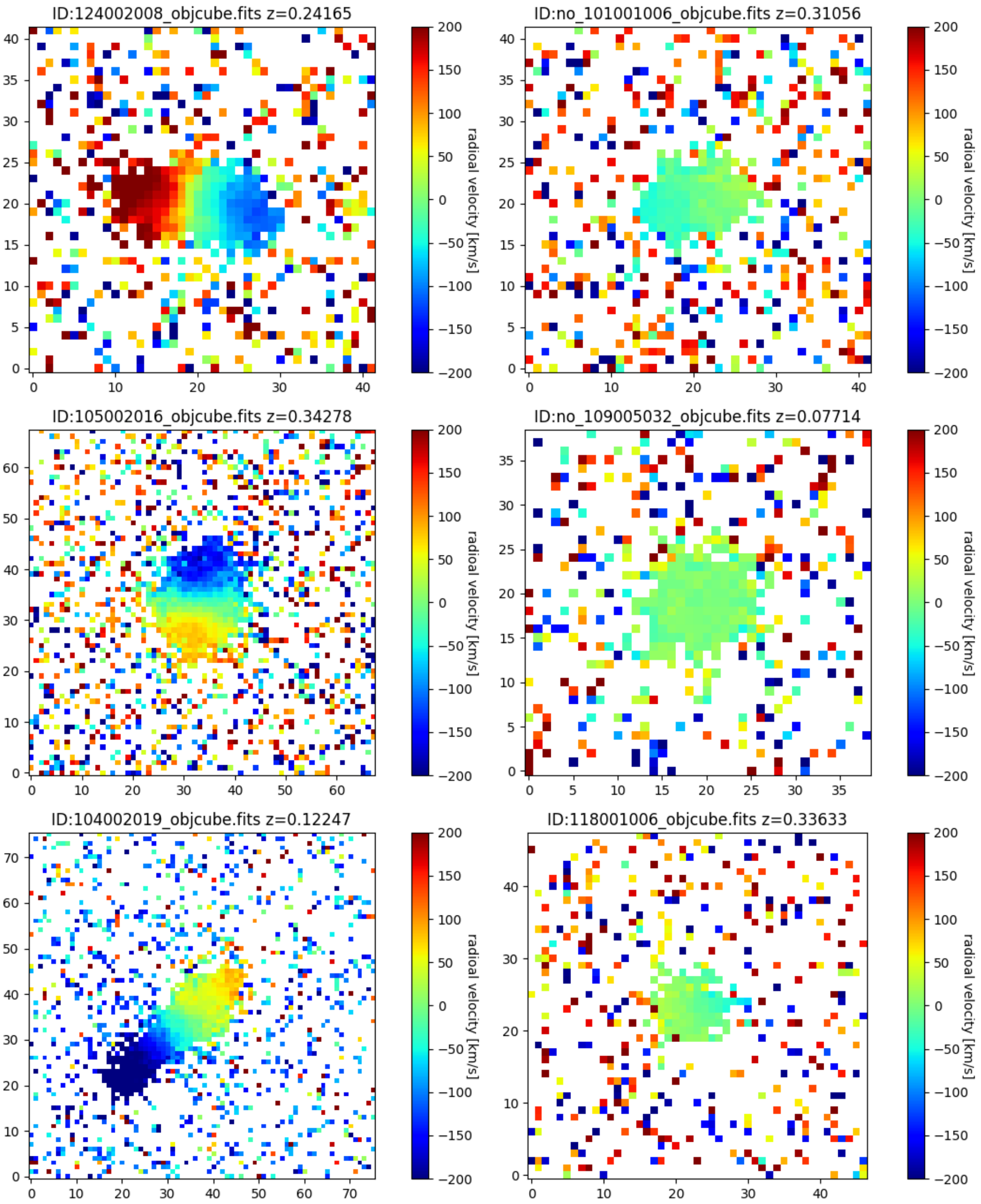

Figure 13: The above plot shows six velocity maps generated from the 3d data cubes for the 6 objects presented in table 2. The three plots on the left show a large rest-frame velocity gradient, which depicts an edge-on orientation, whereas the three plots on the right show a rest-frame velocity gradient close to zero, which is an indicator for a face-on orientation.

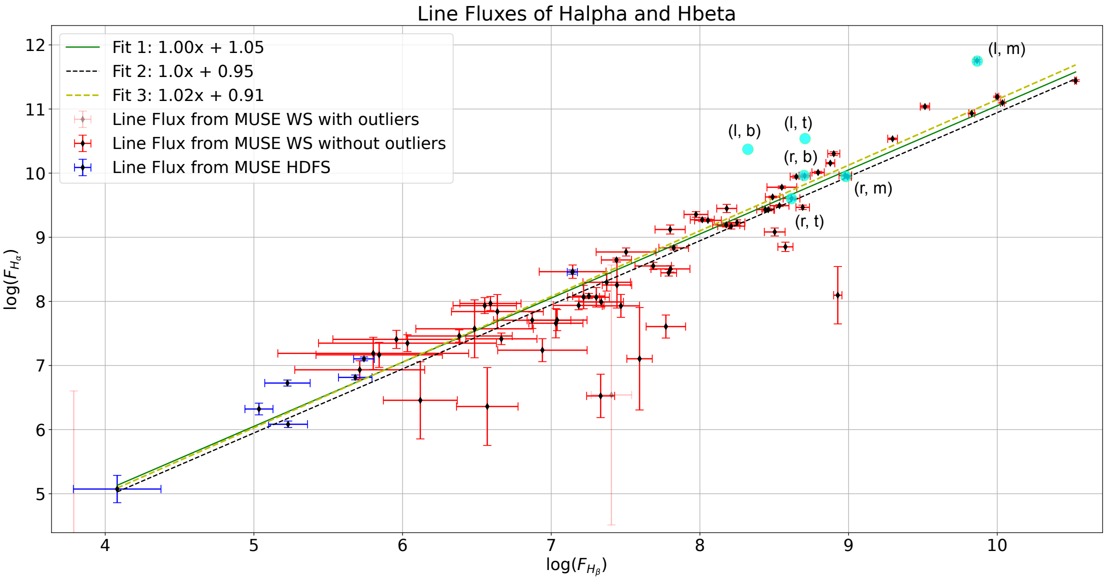

Figure 14: Some of the above measurements displayed in the above plot. The letters refer to the position of the respective object in the figure displayed above(13); (l, b) refers to left, bottom, etc. Unfortunately two objects are missing in this plot, because to the program, their features were not distinguishable from the background noise. There location is nevertheless indicated on the above plot, because there is reason to believe, that the algorithm was wrong in its decision regarding those two objects.

As seen above, a high Balmer decrement corresponds to a high color excess, which in return depicts, that the line of sight crosses dust in the disk of the galaxy. That way it has been possible to validate the line flux measurements once again.

8.4 Redshift Measurements

At last, the estimated redshift measurements can be compared to those given by the data set. The redshift has been calculated for every feature by analysing the expected value of the Gaussian fit. Displayed in the below plot is the deviation in normalized redshift measurement for the three measured features. For better visualization, the deviation in radial velocity has been calculated from the redshift measurement and plotted instead:

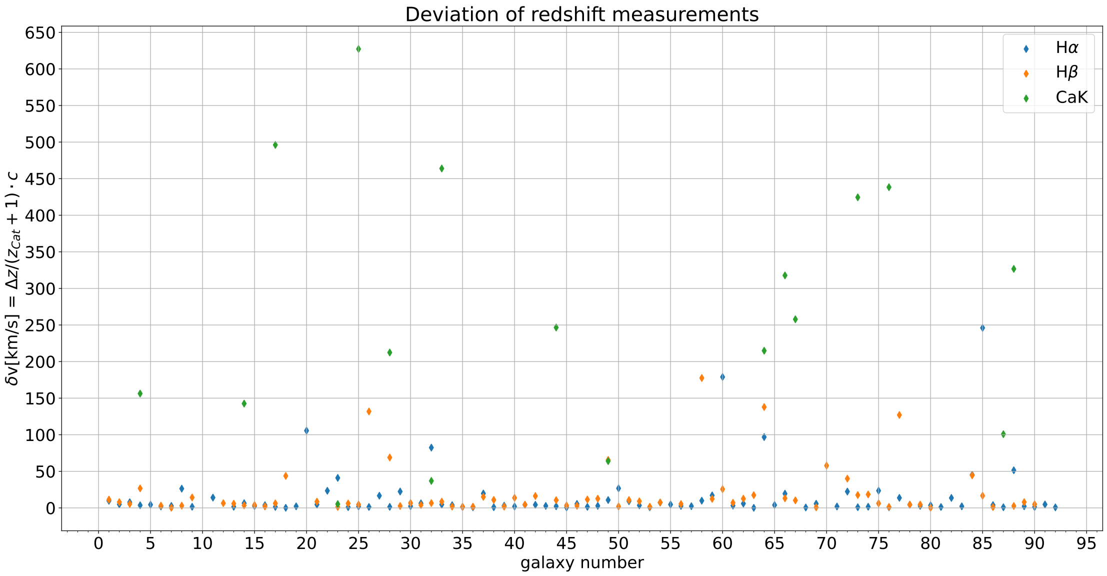

Figure 15: The above plot shows the deviation in velocity calculated from redshift measurements in comparison to the one given by the data set for all three features. The value of Δz corresponds to $|z_{\text{Cat.}} - z_{\text{meas.}}|/(1 + z_{\text{meas.}})$. It can be seen, that the deviation usually is below 50 km/s for most measurements in Hα and Hβ, whereas the CaK measurements are sparse and unevenly distributed.

Concluding this, the measured redshift values are remarkably close to the ones, given by the catalog. The still present deviation could be explained by the transition from air to vacuum wavelengths in the script or by the fact that the correct maximum position of a spectral line might not lie exactly on one pixel, but between two neighbouring pixels and therefore was not being resolved properly. This transition however was observed to not introduce an uncertainty, besides one that may arise numerically inside of the pyasl routine. The second reason however is, given the pixel width of the MUSE spectra of 1.25 Å plausible to account for this uncertainty. Nevertheless, as an example, for the object with ID 105002016 in table 2, an average deviation of $\Delta z = 1.036 \cdot 10^{-5}$ has been calculated for a catalog redshift value of $z = 0.34278$. That, assuming small and non relativistic velocities, corresponds to a radial velocity of:

$$v = z \cdot c = 0.34278 \cdot c = 102761.5\ \frac{\text{km}}{\text{s}}$$

$$\delta v = \frac{\Delta z}{1 + z_{\text{Cat.}}} \cdot c = 1.49917 \cdot 10^{-5} \cdot c = 4.5 \frac{\text{km}}{\text{s}},$$

with the speed of light c. It needs to be said that this is not the uncertainty in velocity, but rather the deviation from the one given by the catalogue. This comparably small velocity deviation also validates the redshift measurements.

8.5 Conclusion

For a final verdict, the algorithm works as intended with some points still to be improved upon. It was possible to measure line fluxes, equivalent widths and redshifts for a handful of selected galaxies. Validating the measured data lead to even more possible future tasks for the algorithm and other interesting physics. Just by analysing line flux or equivalent width measurements from a galaxy spectrum, it was possible to calculate the radial velocity with respect to the Earth and, even more astounding, correctly predict the orientation of said galaxy. All this without even looking at it, with a few lines of code. This also shows again, how powerful algorithms and, more general, automated routines, are for the analysis of big data sets. Writing the algorithm and going through the 381 MUSE Wide survey spectra would have probably taken a similar amount of time, however, now the next data set will be analyzed in a fraction of that time. Runtime-analysis actually suggested a time around one and a half hour for the whole algorithm to analyze the suitable galaxy spectra from the MUSE-Wide Survey. That makes on average roughly one minute per spectrum with 200 iterations in the equivalent width estimation.

9 Outlook

As with every algorithm, code or program, there is always room for improvement. Some routines could be made more efficient, while more advanced physical models could be used for analysation. The list of possible improvements includes, but is not limited to:

- Improving the equivalent width uncertainty estimation.
- SED[14]-fitting of the galaxy continuum instead of using polynomials.
- The Hbeta Absorption region could be fitted by a Gaussian instead of a polynomial of 2nd degree.
- Accounting for other surveys using other instruments having other instrumental profiles.
- Implementing the possibility of fitting Voigt profiles and double-Gaussian profiles to certain lines and doublets.

Further iterations of this algorithm could include some or all of the above mentioned upgrades and are certainly going to include a lot more *Fail Saves* for other data sets.

At last, all there is left to do is come up with a short, yet recognizable name for this program that has little to nothing to do with what it is actually about, but sounds great nevertheless. Something like ANGEL[15]. This will be subject for further development and discussion.

---

[14] as seen in [Bae19]

[15] Automatic aNalysis of Galaxy spectra measuring Equivalent widths and Line fluxes

10 Acknowledgements

Writing a Bachelor thesis is the first major step, physics students have to undergo on their long way to becoming a researcher. Although being an interesting and fun experience, a level of stress and pressure, never before endured, is therefore predetermined. Lucky so, that there are people, that help ease certain problems and give support on the way.

At first I would like to thank Dr. Martin Wendt. His experience and seemingly infinite patience allowed me to understand certain physical processes and techniques, even if it took more than one explanation; I wish him all the best for his future work as student advisor for the Master in Astrophysics. He, together with Prof. Dr. Philipp Richter as my advisors, were always available for questions and got me introduced into a lot of different, highly interesting, astrophysical research in their weekly research group meetings. Also I would like to thank them for revising this thesis and the code several times.

Choosing a topic based on my own interest in programming and data analysis, was a luxury and a gesture, I am very grateful for.

In addition I would like to thank Dr. Gabor Worseck for daily talks on questions of mine, his work on quasars and what is going wrong in the (astrophysics-)world. Thanks to him, I was also able to see beyond my own research and learn interesting and new things every day.

Finally, I would like to thank my family who, despite being quite a long drive away, always supported me, had my back in difficult situations and was always curious about my work.

11 Appendix: The "ANGEL.py" Code

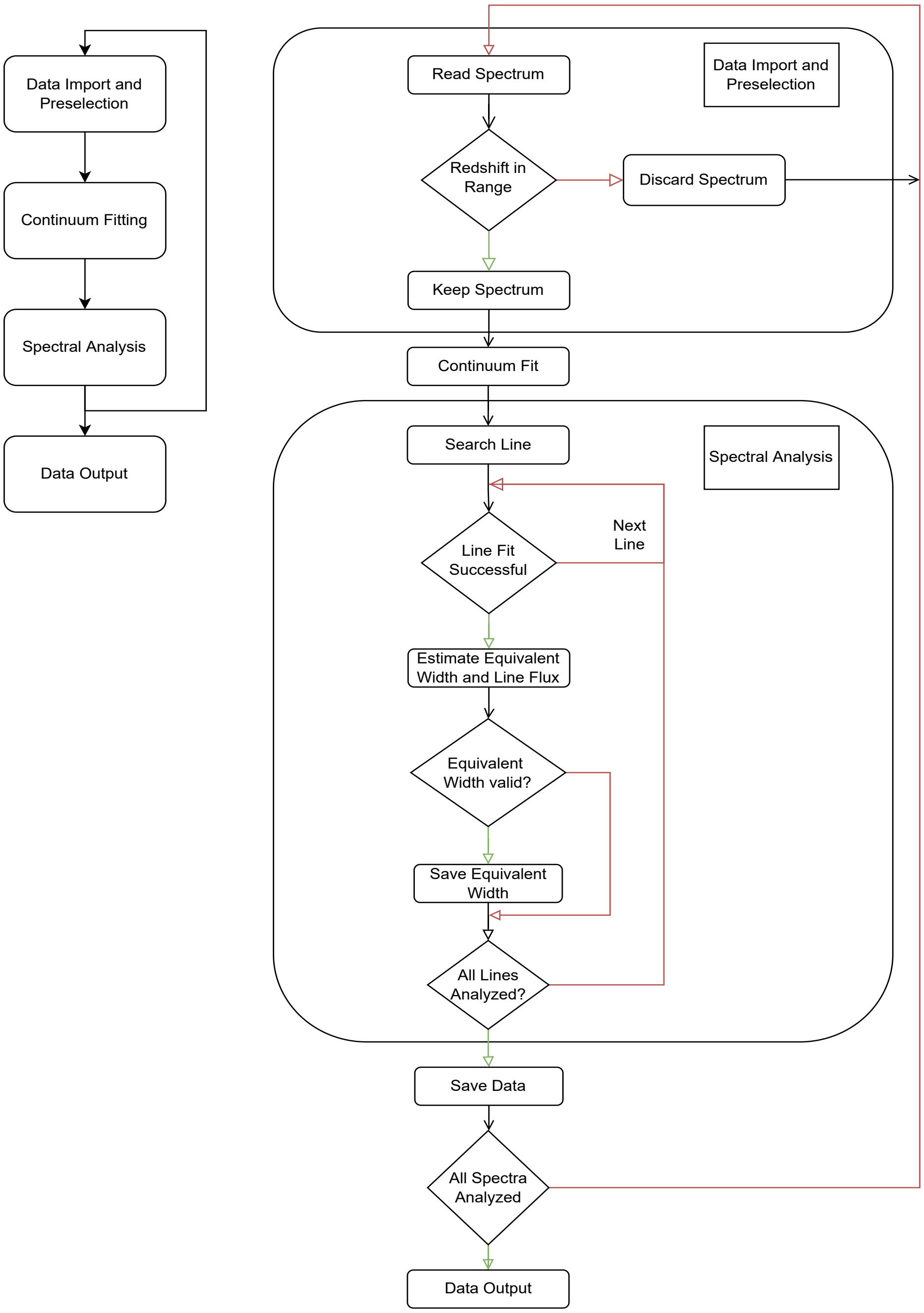

Figure 16: The above flow chart describes the function of the program. As per usual convention, when there is a conditional, the red arrow symbolizes the condition to not be met while for the green arrow the condition is met.

Listing 1: ANGEL.py

```python
import sys
import matplotlib.pyplot as plt
from math import *
from os.path import join
import os
from astropy.io import fits
import IPython
import numpy as np
import numpy.ma as ma
from numpy.core.numeric import argwhere
from scipy.optimize import curve_fit
import scipy.constants as const
from scipy import stats
from PyAstronomy import pyasl

########################################################################################################################
########################################### Definitions of useful functions ############################################
########################################################################################################################

def truth(Input):
    while(Input != 'y' and Input != 'n'):
        Input = input('again please! y or n: ')
    if(Input == 'y'): return True
    if(Input == 'n'): return False

########################################################################################################################
#Look for the actual pixel of the line in the spectrum!
def _determineNearest(wavelength_Array, Position):
    Pos = 0
    Difference = Position - wavelength_Array[0]
    for i in range(len(wavelength_Array)):
        if(abs(Position - wavelength_Array[i]) < Difference):
            Difference = Position - wavelength_Array[i]
            Pos = wavelength_Array[i]
    return Pos

########################################################################################################################
#Determine line region for fitting!
def _determineLineRegion(wavelength_Array, flux_Array, Position, Continuum, lineType):
    temp_Wavelength = np.copy(wavelength_Array)
    temp_Flux = np.copy(flux_Array)
    turningPointRight = Position
    turningPointLeft = Position
    Range = 50

    #increment the counter as long as the slope between neighbouring points to the left is positive
    #(negative) for emission(absorption) lines
    while(turningPointLeft > Position-Range):
        if(lineType == 0):
            if(Slope(temp_Wavelength, temp_Flux, turningPointLeft-1, turningPointLeft) >= 0):
                turningPointLeft -= 1
            else:
                if(Slope(temp_Wavelength, temp_Flux, turningPointLeft-2, turningPointLeft) > 0):
                    turningPointLeft -= 1
                else: break
        if(lineType == 1):
            if(Slope(temp_Wavelength, temp_Flux, turningPointLeft-1, turningPointLeft) <= 0):
                turningPointLeft -= 1
            else:
                if(Slope(temp_Wavelength, temp_Flux, turningPointLeft-2, turningPointLeft+1) < 0):
                    turningPointLeft -= 1
                else: break

    #increment the counter as long as the slope between neighbouring points to the right is negative
    #(positive) for emission(absorption) lines
    while(turningPointRight < Position + Range):
        if(lineType == 0):
            if(Slope(temp_Wavelength, temp_Flux, turningPointRight+1, turningPointRight) <= 0):
                turningPointRight += 1
            else:
                if(Slope(temp_Wavelength, temp_Flux, turningPointRight+2, turningPointRight) < 0):
                    turningPointRight += 1
                else: break
        if(lineType == 1):
            if(Slope(temp_Wavelength, temp_Flux, turningPointRight+1, turningPointRight) >= 0):
                turningPointRight += 1
            else:
                if(Slope(temp_Wavelength, temp_Flux, turningPointRight+2, turningPointRight-1) > 0):
                    turningPointRight += 1
                else: break

    #Fail save to ensure, that the region has been determined in the right order, in case not, switch it!
    if(turningPointLeft > turningPointRight):
        Marker = turningPointRight
        turningPointRight = turningPointLeft
        turningPointLeft = Marker

    return turningPointLeft, turningPointRight

####################################################################################################
def gauss(x, a, sigma, x0, c):
    return a/(np.sqrt(2*np.pi*sigma**2))*np.exp(-(x-x0)**2/(2*sigma**2))+c

####################################################################################################
def Slope(x_Array, y_Array, Point1, Point2):
    return (y_Array[Point2]-y_Array[Point1])/(x_Array[Point2]-x_Array[Point1])

####################################################################################################
def _estimateEquivalentWidth(flux_Array, flux_unc_Array, Start, End, stepWidth):
    #sum over all flux pixel that make up the line multiplied by the pixel size.
    equivalentWidth = sum(flux_Array*stepWidth)
    equivalentWidthUnc = np.sqrt(sum((flux_unc_Array*stepWidth)**2))
    return abs(equivalentWidth), equivalentWidthUnc

####################################################################################################
def _estimateLineFlux(flux_Array, flux_unc_Array, line_Continuum, line_Continuum_unc, Continuum,
                      Continuum_unc, stepWidth, line):
    if(line == 'H\u03B2'):
        flux_region = flux_Array-line_Continuum
        continuum_unc = line_Continuum_unc
    else:
        flux_region = flux_Array-Continuum
        continuum_unc = Continuum_unc


    line_flux = np.sum(flux_region*stepWidth)

    line_flux_unc = stepWidth * np.sqrt(np.sum((flux_unc_Array)**2+(continuum_unc)**2))

    return [abs(line_flux), line_flux_unc]

####################################################################################################
#determine wavelength interval between datapoints. Pixel width: (MUSE: 1.25 A)
def _determineStepWidth(array):
    stepWidth = 0
    for i in range(len(array)-1):
        stepWidth += array[i+1]-array[i]
    stepWidth /= len(array)-1
    return stepWidth

####################################################################################################
def _continuumFit(wavelength_Array, flux_Array, iterations, degree):
    temp_Wavelength = np.copy(wavelength_Array)
    temp_Flux = np.copy(flux_Array)

    #Fit a preliminary continuum with a loose polynomial of 10th degree!
    fit_Coefficients = np.polyfit(temp_Wavelength, temp_Flux, degree)
    polynomial_Fit = np.poly1d(fit_Coefficients)
    temp_Continuum = polynomial_Fit(temp_Wavelength)
    original_Continuum = temp_Continuum

    #Improve the preliminary continuum fit!
    for i in range(iterations):
        SIGMA = np.std(temp_Flux-temp_Continuum)
        temp_Flux = ma.masked_where(abs((temp_Flux-temp_Continuum)) > 3*SIGMA, temp_Flux)
        fit_Coefficients = np.ma.polyfit(temp_Wavelength, temp_Flux, degree)
        polynomial_Fit = np.poly1d(fit_Coefficients)
        temp_Continuum = polynomial_Fit(temp_Wavelength)

    Continuum = temp_Continuum

    return Continuum, SIGMA

########################################################################################
#determine continuum around given feature
def _determineLineContinuum(wavelength_Array, flux_Array, flux_unc_Array, continuum,
                            continuum_unc, interval):
    temp_Wavelength = np.copy(wavelength_Array)
    temp_Flux = np.copy(flux_Array)
    temp_Flux_unc = np.copy(flux_unc_Array)

    #Fit line continuum with a loose polynomial of 2nd degree
    Degree = 2
    iterations = 200

    temp_Flux = ma.masked_where((temp_Wavelength > interval[0]) & (temp_Wavelength < interval[1]),
                                temp_Flux)
    temp_Flux_unc = ma.masked_where((temp_Wavelength > interval[0]) & (temp_Wavelength < interval[1]),
                                    temp_Flux_unc)

    temp_Continuum = continuum
    original_Continuum = temp_Continuum

    for i in range(iterations):
        SIGMA = np.std(temp_Flux-temp_Continuum)
        temp_Flux = ma.masked_where(abs((temp_Flux-temp_Continuum)) > 3*SIGMA, temp_Flux)
        coef = np.ma.polyfit(temp_Wavelength, temp_Flux, Degree)
        polyFit = np.poly1d(coef)
        temp_Continuum = polyFit(temp_Wavelength)

    #Fail save to detect when the line continuum could not be estimated.
    if(len(temp_Continuum) == 0):
        return continuum, continuum_unc
    else:
        return temp_Continuum, SIGMA

########################################################################################
def SignalToNoise(wavelength_Array, flux_Array, redshift, start, end):
    Interval = np.argwhere((wavelength_Array >= ((1+redshift)*start)) &
                           (wavelength_Array<=((1+redshift)*end)))
    return (np.average(flux_Array[Interval])/np.std(flux_Array[Interval]))

########################################################################################
#calculate the EW and EW uncertainty. Similar to a MonteCarlo simulation approach.
def _equivalentWidthSim(flux_Array, flux_unc_Array, wavelength_Array_Part, Deviation,
                        stepWidth, iterations, line, line_Region, fit_Region, redshift):

    FitRegionStart = fit_Region[0]
    FitRegionEnd = fit_Region[1]
    lineRegionStart = line_Region[0]
    lineRegionEnd = line_Region[1]

    #predefined parameters for the fitting routine
    Gauss_Parameters_min = [-np.inf, 0, wavelength_Array[lineRegionStart], -1]
    Gauss_Parameters_max = [np.inf, 5000, wavelength_Array[lineRegionEnd], 1]
    Linspace = np.linspace(wavelength_Array[FitRegionStart], wavelength_Array[FitRegionEnd], 1000)

    EW_Area = np.zeros(iterations+1)

    #First fit to model
    try:
        popt2, pcov2 = curve_fit(gauss, wavelength_Array_Part, flux_Array, sigma = flux_unc_Array,
                                 bounds = ([Gauss_Parameters_min[0], Gauss_Parameters_min[1],
                                            Gauss_Parameters_min[2], Gauss_Parameters_min[3]],
                                           [Gauss_Parameters_max[0], Gauss_Parameters_max[1],
                                            Gauss_Parameters_max[2], Gauss_Parameters_max[3]]))
        Successful = True
        Fit_Error_before = np.sqrt(np.diag(pcov2)[0])
    except:
        Successful = False
        Fit_Error_before = 0
        popt2 = np.zeros(4)
        pcov2 = np.zeros(4)
        print("Could not fit Gauss to normalized Flux!")

    if(PlotFitting == True and Successful == True):
        ax2.step(wavelength_Array_Part,flux_Array,'r-',label='front @ %.3f' % galaxy_redshift,zorder=1000,
                 where='mid', alpha = 0.5)
        ax2.plot(Linspace, gauss(Linspace, *popt2), label = 'preliminary gaussian fit', color = 'green')

    Fit_Parameters = np.zeros((iterations,popt2.size))
    Area = abs(popt2[0])
    EW_Area[0] = Area/(1+redshift)

    #Fehler vor Iteration vs Nach Iteration vergl.
    for i in range(iterations):
        flux_distribution_Array = flux_Array + np.random.normal(0, (1/Deviation), size=len(flux_Array))

        #Fit again with distribution
        try:
            popt2, pcov2 = curve_fit(gauss, wavelength_Array_Part, flux_distribution_Array,
                                     bounds = ([Gauss_Parameters_min[0], Gauss_Parameters_min[1],
                                                Gauss_Parameters_min[2], Gauss_Parameters_min[3]],
                                               [Gauss_Parameters_max[0], Gauss_Parameters_max[1],
                                                Gauss_Parameters_max[2], Gauss_Parameters_max[3]]))
            Successful = True
        except:
            Successful = False
            popt2 = np.zeros(4)
            print("Could not fit Gauss to deviated Flux!")

        if(Successful == True):
            #popt2 = [Area, Sigma, Maximum Position, Offset]

            ###but by assuming a normalized gauss curve, one can just take popt[0]=a as the area
            EW_Area[i+1] = abs(popt2[0])/(1+redshift)

            Fit_Parameters[i] = popt2

    #[EW_Value, MonteCarlo Uncertainty, Fit Uncertainty]
    if(Successful == True and np.median(EW_Area)/2 >
                              stats.median_abs_deviation(EW_Area, scale = 1.)*Deviation):
        EquivalentWidthArea = [np.median(EW_Area), stats.median_abs_deviation(EW_Area, scale = 1.)*Deviation
                               , Fit_Error_before]
        Redshift = np.median(Fit_Parameters[:,2])/line - 1
    else:
        EquivalentWidthArea = np.array([0, 0, 0])
        Redshift = 0

    return EquivalentWidthArea, Redshift

##########################################################################################################
###################################### Initialising of Main Program ######################################
##########################################################################################################

scriptname = sys.argv[0]

#Sets the target folder, where the spectra are saved together with the Master File
base_dir = sys.argv[1]

#Determines when to start with the PlotSpectrum of foreground galaxies #10 means end at 10/47 #-10 means
#start at 10/47
LIMIT = int(sys.argv[2])

#List of most prominent absorption features, further iterations of this program might analyze as well!
diblist=[4430.0, 4765.0, 4880.0, 5705.1, \
  5780.5, \
  5797.1, \
  6203.6 , \
  6283.8 , \
  6613.6 , \
  7664.91, \
  7698.97, \
  8621.1 ]


#Important constant parameters and wavelengths!
HALPHA = 6564.614   #HALPHA in vacuum
HBETA = 4862.721    #Hbeta in vacuum
CAK = 3934.78       #CaK in vacuum
CAH = 3969.60       #CaH in vacuum
Z_MIN = 0           #set to zero, because Hbeta is still visible at z=0
Z_MAX = 0.42        #no Halpha Line after

#PlotSpectrum = True shows the galaxy spectrum!
PlotSpectrum = False

#CalculateEquivalentWidths = True fits and calculates equivalent widths!
#Method 1 normalizes the continuum by 1-F/C with C being the continuum, while Method 2 normalizes by F-C.
#Method 1 is better suited for absorption lines, while Method 2 is better suited for emission lines.
CalculateEquivalentWidths = False
#Method1 = 'Continuum_Normalization'
#Method2 = 'Continuum_Subtraction'

#PlotFitting = True displays the fitting procedure!
PlotFitting = False

#PrintResults = True prints the results onto the command promt, while
#PrintResultsToFile = True prints them to the designated file!
PrintResults = False
PrintResultsToFile = False

######################################## User Interaction
print('y or n?')
PlotSpectrum = truth(input('Do you want to see the whole spectrum? '))
CalculateEquivalentWidths = truth(input('Do you want to calculate equivalent widths? '))
if(CalculateEquivalentWidths == True):
    Method = int(input('Method 1 normalizes the continuum by 1-F/C with C being the continuum, while
                       Method 2 normalizes by F-C. Method 1 is better suited for absorption lines,
                       while Method 2 is better suited for emission lines.' + '\n' + '1 or 2? '))
PlotFitting = truth(input('Do you want to see the fitting procedure? '))

PrintResults = truth(input('Do you want to print the results to the command prompt? '))
PrintResultsToFile = truth(input('Do you want to print the results to a file? '))
if(PrintResultsToFile == True): Filename = input('What should the file be named? ')
######################################## End of user interaction

#Which catalog is to be used and how the spectra are imported, using a Master File or loose spectra.
if('MasterFile.fits' in os.listdir(base_dir)):
    MasterFile = True
    MUSE_Data = fits.open(join(base_dir, 'MasterFile.fits'))
    table=MUSE_Data[1].data

    galaxy_number = 0
    #Get an estimate on the number of suitable galaxies for more efficient array creation
    for g in table:
        g_redshift = g[3]

        if(g_redshift > Z_MIN and g_redshift < Z_MAX):
            galaxy_number += 1

else:
    MasterFile = False
    table = os.listdir(base_dir)
    galaxy_number = len(table)
    #Redshift values have to be implemented manually when no Master File is present!
    Redshifts = [0.2249, 0.3179, 0.3649, 0.2297, 0.4200, 0.3210, 0.2836, 0.2946, 0.3178]

#Array Creation for Output
EW_Array = np.zeros((galaxy_number, 3, 3))
Line_Flux_Array = np.zeros((galaxy_number, 3, 2))
Line_Continuum_Array = np.zeros((galaxy_number, 3))
Flux_Unc_Array = np.zeros((galaxy_number, 3))
Redshift_Calculated = np.zeros((galaxy_number, 4))
SN = np.zeros((galaxy_number))
ID = np.zeros((galaxy_number))

```

```
if(PrintResultsToFile == True):
    with open(Filename, 'w') as DataFile:
        # Resulting File should be structured like a table for easy reading in.
        print('ID', 'SNR', 'EW_HALPHA','EW_HALPHA_ERR','LINEFLUX_HALPHA', 'LINEFLUX_HALPHA_ERR',
              'CONTINUUM_HALPHA', 'FLUX_HALPHA_ERR', 'EW_HBETA', 'EW_HBETA_ERR', 'LINEFLUX_HBETA',
              'LINEFLUX_HBETA_ERR', 'CONTINUUM_HBETA', 'FLUX_HBETA_ERR', 'EW_CAK', 'EW_CAK_ERR',
              'LINEFLUX_CAK', 'LINEFLUX_CAK_ERR', 'CONTINUUM_CAK', 'FLUX_CAK_ERR', 'z_Katalog',
              'z_Ha', 'z_Hb', 'z_CaK', file = DataFile)

##############################################################################################################
########################################### Start of Main Routine ############################################
##############################################################################################################
galaxy_count = 0

for galaxy in table:
    if(MasterFile == True):
        galaxy_id, galaxy_ra, galaxy_dec, galaxy_redshift, galaxy_redshift_error = galaxy[0:5]
        spec_file='spectrum_%s.fits' % galaxy_id
    else:
        spec_file = galaxy
        galaxy_id = 0
        galaxy_redshift = Redshifts[galaxy_count]

    filename=join(base_dir, spec_file)
    sp1 = fits.open(filename)

    #Here one has to account for the possible change in columnnames when using another data set
    if(MasterFile == True):
        WAVELENGTH = pyasl.airtovac2(sp1[1].data['WAVE_AIR'])
        FLUX = sp1[1].data['FLUX']
        FLUX_UNC = sp1[1].data['FLUXERR']
    else:
        WAVELENGTH = pyasl.airtovac2(sp1[1].data['WAVELENGTH'])
        FLUX = sp1[1].data['FLUX']
        FLUX_UNC = sp1[1].data['DFLUX']

    #Position of lines in red-shifted spectrum from 4750A to 9350A
    CaH  = (galaxy_redshift + 1) * CAH
    CaK  = (galaxy_redshift + 1) * CAK
    LineHalpha  = (galaxy_redshift + 1) * HALPHA
    LineHbeta  = (galaxy_redshift + 1) * HBETA

    #Which features should be analyzed
    LINES = [[LineHalpha, 0, 'Halpha'], [LineHbeta, 0, 'Hbeta'], [CaK, 1, 'CaK']]
    #[redshifte vaccum wavelength of the line, 0 for emmission 1 for absorption, name]

    if galaxy_redshift > Z_MIN and galaxy_redshift < Z_MAX:
        galaxy_count += 1
        ID[galaxy_count-1] = galaxy_id
    else:
        continue

    #start at LIMIT\NumberOfGalaxies
    if LIMIT >= 0 and galaxy_count > abs(LIMIT): continue

    #end after LIMIT\NumberOfGalaxies
    if LIMIT <= 0 and galaxy_count < abs(LIMIT): continue

    print(bcolors.HEADER + "Currently working on (%2s/%2d)" % (galaxy_count, galaxy_number) + bcolors.ENDC)
    try:
        print(galaxy_id)
    except:
        print('NoID')

    #Calculate a continuum fit for the spectrum
    CONTINUUM, CONTINUUM_UNC = _continuumFit(WAVELENGTH, FLUX, iterations = 20, degree = 10)

    #Calculate the Signal to Noise Ratio in different intervals
    SNRInterval = [5050, 5100, 5150, 6000, 6100]
    for Interval in SNRInterval:
        SN[galaxy_count-1] += SignalToNoise(WAVELENGTH, FLUX, galaxy_redshift, Interval, Interval+50)
    SN[galaxy_count-1]/=len(SNRInterval)

    #Plot galaxy spectrum for visualization
    if(PlotSpectrum == True):
        top     = np.median(FLUX)+500
        bottom  = np.median(FLUX)-500

        plt.figure(figsize=(15,5))
        plt.xlabel(r'$\lambda$' + ' in ' + r'$\AA$')
        plt.ylabel('flux in ' + r'$10^{-20}erg/s/cm^2/\AA$')
        plt.ylim((np.min(FLUX)-100, np.max(FLUX)+100))
        plt.title('z = ' + str(np.round(galaxy_redshift,3)))
        plt.step(WAVELENGTH,FLUX,'r-',label='front @ %.3f' % galaxy_redshift,zorder=1000,where='mid',
                 alpha = 0.5)
        plt.step(WAVELENGTH, FLUX_UNC+FLUX, 'b-', label ='front @ %.3f' % galaxy_redshift, zorder=1000,
                 where='mid', alpha = 0.25)
        plt.plot(WAVELENGTH, CONTINUUM, label = 'Improved Continuum', color = 'black')
        plt.vlines(CaH, top,bottom, color='lightgreen', zorder=0, linewidth=4)
        plt.vlines(CaK, top,bottom, color='lightgreen', zorder=0, linewidth=4)
        plt.vlines(LineHalpha, top,bottom, color='cyan', zorder=0, linewidth=4)
        plt.vlines(LineHbeta, top,bottom, color='yellow', zorder=0, linewidth=4)

        for Interval in SNRInterval:
            plt.vlines((1+galaxy_redshift)*Interval, top,bottom, color='black', zorder=0, linewidth=2)
            plt.vlines((1+galaxy_redshift)*(Interval+50), top,bottom, color='black', zorder=0, linewidth=2)

        for position in diblist:
            pos_obs = position * (galaxy_redshift+1)
            plt.vlines(pos_obs, top,bottom, color='lightgrey', zorder=0, linewidth=4)
            plt.text(pos_obs, top, "%.0f" % position, fontsize=12)
        plt.legend()
        plt.show()

    #Calculate Equivalent Width
    if(CalculateEquivalentWidths == True):
        stepWidth = _determineStepWidth(WAVELENGTH)

        for feature in LINES:
            #Fail save to detect degenerate or broken data that does not span the whole wavelength range
            if(np.min(WAVELENGTH) > feature[0] or np.max(WAVELENGTH) < feature[0]):
                #Feature not in spectrum, so skip to the next one!
                continue

            print(bcolors.HEADER + "Doing " + str(feature[2]) + bcolors.ENDC)

            line_Index = LINES.index(feature)
            line_Type = feature[1] #1 for absorption, 0 for emission

            #search for the feature wavelength in the wavelength array
            line_Wavelength = _determineNearest(WAVELENGTH, feature[0])
            line_Position = int(np.argwhere(WAVELENGTH == line_Wavelength))

            #If necessary, correct the line position
            if(line_Type == 0):
                if(np.max(FLUX[line_Position-5:line_Position+5]) != FLUX[line_Position]):
                    line_Position = int(np.argwhere(FLUX == np.max(FLUX[line_Position-5:line_Position+5])))
            if(line_Type == 1):
                if(np.min(FLUX[line_Position-5:line_Position+5]) != FLUX[line_Position]):
                    line_Position = int(np.argwhere(FLUX == np.min(FLUX[line_Position-5:line_Position+5])))

            #Determine the region where the line resides.
            regionStart, regionEnd = _determineLineRegion(WAVELENGTH, FLUX, line_Position, CONTINUUM,
                                                          line_Type)

            #Determine the fitting region!
            #Is the fitting window valid? If not, change it. Mostly changed, when redshifts are close to
            #the CaK cutoff.
            if(int((abs(regionEnd-regionStart))*1.5) < regionStart):
                Offset = int((abs(regionEnd-regionStart))*1.5)
                fitRegionStart = line_Position - Offset
            else:
                Offset = line_Position
                fitRegionStart = 0

            fitRegionEnd = line_Position + Offset

            #Wavelength and Flux inside of the interval
            Wave_Part = WAVELENGTH[fitRegionStart:fitRegionEnd]
            Flux_Part = FLUX[fitRegionStart:fitRegionEnd]
            Flux_Unc_Part = FLUX_UNC[fitRegionStart:fitRegionEnd]
            Continuum_Part = CONTINUUM[fitRegionStart:fitRegionEnd]

            #Fit local continuum around Peak
            #for emission lines only, for absorption lines use the prior estimated continuum of the spectrum
            if(line_Type == 0):
                LINECONTINUUM, LINECONTINUUM_UNC =
                    _determineLineContinuum(Wave_Part, Flux_Part, Flux_Unc_Part, Continuum_Part,
                                             CONTINUUM_UNC, [WAVELENGTH[regionStart], WAVELENGTH[regionEnd]])
            elif(line_Type == 1):
                LINECONTINUUM, LINECONTINUUM_UNC = Continuum_Part, CONTINUUM_UNC

            #Normalize for equivalent width estimation
            if(np.min(LINECONTINUUM) > 1/SN[galaxy_count-1] and LINECONTINUUM.all() != 0):
                flux_normalized_Array = 1 - Flux_Part/LINECONTINUUM
                flux_normalized_unc = np.sqrt(((-1/LINECONTINUUM * Flux_Unc_Part)**2)+
                    (Flux_Part/LINECONTINUUM**2 * LINECONTINUUM_UNC)**2)
            else:
                flux_normalized_Array = Flux_Part
                flux_normalized_unc = Flux_Unc_Part
                print("Changed Continuum")

            #Display the fitting routine for EW estimation
            if(PlotFitting == True):
                fig, (ax1, ax2) = plt.subplots(1,2)
                ax1.axis(xmin = WAVELENGTH[fitRegionStart], xmax = WAVELENGTH[fitRegionEnd])
                ax1.step(WAVELENGTH, FLUX,'r-', label='front @ %.3f' % galaxy_redshift, zorder=1000,
                         where='mid', alpha = 0.5)
                ax1.plot(WAVELENGTH, CONTINUUM, label = 'Galaxy Continuum', color = 'black')
                ax1.set_xlabel(r'$\lambda$' + ' in ' + r'$\AA$')
                ax1.set_ylabel('flux in ' + r'$10^{-20}erg/s/cm^2/\AA$')

                ax1.set_title(str(feature[2]))
                ax1.legend()
                ax1.vlines(WAVELENGTH[regionEnd], 0, 5000)
                ax1.vlines(WAVELENGTH[regionStart], 0, 5000)

                ax2.axis(xmin = WAVELENGTH[fitRegionStart], xmax = WAVELENGTH[fitRegionEnd])
                ax2.set_title(str(feature[2] + " normalized onto the continuum."))
                ax2.set_xlabel(r'$\lambda$' + ' in ' + r'$\AA$')
                ax2.set_ylabel('normalized flux')
                ax2.vlines(WAVELENGTH[regionEnd], 0, 1)
                ax2.vlines(WAVELENGTH[regionStart], 0, 1)

            #Begin of Variation Routine similar to MonteCarlo
            if(Method == 1):
                EW_Area, Redshift =
                    _equivalentWidthSim(flux_normalized_Array, flux_normalized_unc, WAVELENGTH,
                                        Wave_Part, SN[galaxy_count-1], stepWidth, 200,
                                        float(feature[0])/(galaxy_redshift + 1),
                                        ([regionStart, regionEnd]),
                                        ([fitRegionStart, fitRegionEnd]), galaxy_redshift)
            elif(Method == 2):
                EW_Area, Redshift =
                    _equivalentWidthSim(Flux_Part-LINECONTINUUM,
                                        np.sqrt(Flux_Unc_Part**2 + LINECONTINUUM_UNC**2),
                                        WAVELENGTH, Wave_Part, SN[galaxy_count-1], stepWidth,
                                        200, float(feature[0])/(galaxy_redshift + 1),
                                        ([regionStart, regionEnd]),
                                        ([fitRegionStart, fitRegionEnd]), galaxy_redshift)

            Redshift_Calculated[galaxy_count-1, 0] = galaxy_redshift
            EW_Array[galaxy_count-1, line_Index] = EW_Area
            Redshift_Calculated[galaxy_count-1, line_Index+1] = Redshift
            Line_Continuum_Array[galaxy_count-1, line_Index] = np.average(LINECONTINUUM)
            Flux_Unc_Array[galaxy_count-1, line_Index] = np.average(Flux_Unc_Part)
            Line_Flux_Array[galaxy_count-1, line_Index] =
                _estimateLineFlux(Flux_Part, Flux_Unc_Part, LINECONTINUUM, LINECONTINUUM_UNC,
                                  Continuum_Part, CONTINUUM_UNC, stepWidth, feature[2])

            if(PlotFitting == True):
                ax2.legend()
                plt.show()

    #Print results onto the command prompt
    if(PrintResults == True):
        print('EWs: ' + '\t' + str(np.round(EW_Array[galaxy_count-1,0,0], 2)) + '\n' + '\t' +
              str(np.round(EW_Array[galaxy_count-1,1,0], 2)) + '\n' + '\t' +
              str(np.round(EW_Array[galaxy_count-1,2,0], 2)))
        print('Line Fluxes: ' + '\t' + str(np.round(Line_Flux_Array[galaxy_count-1,0,0], 2))
             + '\n' + '\t' + '\t' + str(np.round(Line_Flux_Array[galaxy_count-1,1,0], 2))
             + '\n' + '\t' + '\t' + str(np.round(Line_Flux_Array[galaxy_count-1,2,0], 2)))


    #Print results to the designated file
    if(PrintResultsToFile == True):
        with open(Filename, 'a') as DataFile:
            print(ID[galaxy_count-1], SN[galaxy_count-1],
            EW_Array[galaxy_count-1, 0, 0], EW_Array[galaxy_count-1, 0, 1],
            Line_Flux_Array[galaxy_count-1, 0, 0], Line_Flux_Array[galaxy_count-1, 0, 1],
            Line_Continuum_Array[galaxy_count-1, 0], Flux_Unc_Array[galaxy_count-1, 0],
            EW_Array[galaxy_count-1, 1, 0], EW_Array[galaxy_count-1, 1, 1],
            Line_Flux_Array[galaxy_count-1, 1, 0], Line_Flux_Array[galaxy_count-1, 1, 1],
            Line_Continuum_Array[galaxy_count-1, 1], Flux_Unc_Array[galaxy_count-1, 1],
            EW_Array[galaxy_count-1, 2, 0], EW_Array[galaxy_count-1, 2, 1],
            Line_Flux_Array[galaxy_count-1, 2, 0], Line_Flux_Array[galaxy_count-1, 2, 1],
            Line_Continuum_Array[galaxy_count-1, 2], Flux_Unc_Array[galaxy_count-1, 2],
            Redshift_Calculated[galaxy_count-1, 0], Redshift_Calculated[galaxy_count-1, 1],
            Redshift_Calculated[galaxy_count-1, 2], Redshift_Calculated[galaxy_count-1, 3],
            file = DataFile)

    #After every spectrum, stop the analysis by typing 'Stop', otherwise type 'Continue' or bypass
    #with the second line
    Continue = input("Continue or Stop? ")
    #Continue = "Continue"
    if(Continue == "Stop"):
        break
```

References

[Bac+15] Bacon, R. et al. "The MUSE 3D view of the Hubble Deep Field South". In: *A&A* 575 (2015), A75. DOI: `10.1051/0004-6361/201425419`. URL: `https://doi.org/10.1051/0004-6361/201425419`.

[Bac+17] Roland Bacon et al. "The MUSE Hubble Ultra Deep Field Survey". In: *Astronomy Astrophysics* 608 (Nov. 2017), A1. ISSN: 1432-0746. DOI: `10.1051/0004-6361/201730833`. URL: `http://dx.doi.org/10.1051/0004-6361/201730833`.

[Bae19] Maarten Baes. "Panchromatic SED fitting codes and modelling techniques". In: *Proceedings of the International Astronomical Union* 15.S341 (Nov. 2019), pp. 26–34. ISSN: 1743-9221. DOI: `10.1017/s1743921319003016`. URL: `http://dx.doi.org/10.1017/S1743921319003016`.

[Bar17] C. Barbieri. *Fundamentals of astronomy.* CRC Press, 2017. Chap. 5.1.

[BR90] P. T. Boggs and J. E. Rogers. *"Orthogonal Distance Regression" in "Statistical analysis of measurement error models and applications: proceedings of the AMS-IMS-SIAM joint summer research conference held June 10-16, 1989,".* 1990.

[Bro71] M. Brocklehurst. "Calculations of level populations for the low levels of hydrogenic ions in gaseous nebulae." In: 153 (Jan. 1971), p. 471. DOI: `10.1093/mnras/153.4.471`.

[CCM89] Jason A. Cardelli, Geoffrey C. Clayton, and John S. Mathis. "The Relationship between Infrared, Optical, and Ultraviolet Extinction". In: 345 (Oct. 1989), p. 245. DOI: `10.1086/167900`.

[CS04] F. Combes and M. Seymour. *Galaxies and Cosmology.* Springer, 2004, pp. 290–292.

[Cze+19] Stefan Czesla et al. *PyA: Python astronomy-related packages.* June 2019. ascl: `1906.010`.

[Dra+08] B. Draine et al. "Dust Masses, PAH Abundances, and Starlight Intensities in the SINGS Galaxy Sample". In: *The Astrophysical Journal* 663 (Dec. 2008), p. 866. DOI: `10.1086/518306`.

[DS03] Michael A. Dopita and Ralph S. Sutherland. "Astrophysics of the Diffuse Universe". In: *Astronomy and Astrophysics Library* (2003). DOI: `10.1007/978-3-662-05866-4`.

[ESO21] ESO. *UVES instrument description.* 2021. URL: `https://www.eso.org/sci/facilities/paranal/instruments/uves/inst.html` (visited on 12/16/2021).

[Fu06] With Qiang Fu. “4 - Radiative Transfer11We thank Qiang Fu for his guidance in preparing this chapter.” In: *Atmospheric Science (Second Edition)*. Ed. by John M. Wallace and Peter V. Hobbs. Second Edition. San Diego: Academic Press, 2006, pp. 113–152. ISBN: 978-0-12-732951-2. DOI: `https://doi.org/10.1016/B978-0-12-732951-2.50009-0`. URL: `https://www.sciencedirect.com/science/article/pii/B9780127329512500090`.

[GBW11] Brent Groves, Jarle Brinchmann, and Carl Jakob Walcher. “The Balmer decrement of Sloan Digital Sky Survey galaxies”. In: *Monthly Notices of the Royal Astronomical Society* 419.2 (Nov. 2011), pp. 1402–1412. ISSN: 0035-8711. DOI: `10.1111/j.1365-2966.2011.19796.x`. URL: `http://dx.doi.org/10.1111/j.1365-2966.2011.19796.x`.

[Gmb21] Baader Planetarium GmbH. *BACHES Echelle Spektrograf.* 2021. URL: `https://www.baader-planetarium.com/de/spektroskopie/baches-echelle-spektrograf/baches-echelle-spektrograf.html` (visited on 12/16/2021).

[HBL10] David W. Hogg, Jo Bovy, and Dustin Lang. *Data analysis recipes: Fitting a model to data.* 2010. arXiv: `1008.4686 [astro-ph.IM]`.

[Hei27] W. Heisenberg. “Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik”. In: *Zeitschrift für Physik* 43.3 (1927), pp. 172–198. ISSN: 0044-3328. DOI: `10.1007/BF01397280`.

[Her+17a] Edmund Christian Herenz et al. “The MUSE-Wide survey: A first catalogue of 831 emission line galaxies”. In: *Astronomy Astrophysics* 606 (Sept. 2017), A12. ISSN: 1432-0746. DOI: `10.1051/0004-6361/201731055`. URL: `http://dx.doi.org/10.1051/0004-6361/201731055`.

[Her+17b] Edmund Christian Herenz et al. “The MUSE-Wide survey: A first catalogue of 831 emission line galaxies”. In: *Astronomy Astrophysics* 606 (Sept. 2017), A12. ISSN: 1432-0746. DOI: `10.1051/0004-6361/201731055`. URL: `http://dx.doi.org/10.1051/0004-6361/201731055`.

[Hub29] E. P. Hubble. “A spiral nebula as a stellar system, Messier 31.” In: 69 (Mar. 1929), pp. 103–158. DOI: `10.1086/143167`.

[LP12] Henrietta S. Leavitt and Edward C. Pickering. "Periods of 25 Variable Stars in the Small Magellanic Cloud." In: *Harvard College Observatory Circular* 173 (Mar. 1912), pp. 1–3.

[MM72] Joseph S. Miller and William G. Mathews. "The Recombination Spectrum of the Planetary Nebula NGC 7027." In: 172 (Mar. 1972), p. 593. DOI: `10.1086/151378`.

[Nov73] Eva Novotny. *Introduction to stellar atmospheres and interiors.* Oxford University Press, 1973.

[Pen+10] W. D. Pence et al. "Definition of the Flexible Image Transport System (FITS), version 3.0". In: 524, A42 (Dec. 2010), A42. DOI: `10.1051/0004-6361/201015362`.

[Sod21] et al. Soderblom D. *COS Data Handbook, Version 5.0.* 2021.

[Spi04] Lyman Spitzer. *Physical processes in the interstellar medium.* Wiley-VCH Verlag, 2004.

[Sūf14] 903-986 Sūfī Abd al-Ramān ibn Umar. *Book of Fixed Stars, Suwar al-kawākib.* 1 Mharram 820 H [18 Februrary 14]. URL: `https://www.loc.gov/item/2008401028/`.

[VD09] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: `10.1038/s41592-019-0686-2`.

[ZBB21] I. Zelinka, M. Brescia, and D. Baron. *Intelligent astrophysics.* Springer, Dec. 2021.

[ZM13] Guangtun Zhu and Brice Ménard. "CALCIUM H & K INDUCED BY GALAXY HALOS". In: *The Astrophysical Journal* 773.1 (July 2013), p. 16. DOI: `10.1088/0004-637x/773/1/16`. URL: `https://doi.org/10.1088/0004-637x/773/1/16`.