TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Jaanika Raik 204388IAPM

Applying MCMC Methods in Stellar Spectroscopy to Derive Physical Parameters of Hotter (Early-Type) Stars

Master's thesis

Supervisor: Colin Folsom
PhD
Mihkel Kama
PhD

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jaanika Raik 204388IAPM

MCMC MEETODITE RAKENDAMINE TÄHESPEKTROSKOOPIAS KÕRGE TEMPERATUURIGA (VARAJAST TÜÜPI) TÄHTEDE FÜÜSILISTE PARAMEETRITE HINDAMISEKS

magistritöö

Juhendaja: Colin Folsom
PhD
Mihkel Kama
PhD

Tallinn 2023

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Jaanika Raik

08.05.2023

Abstract

Stellar spectra are useful to determine the physical parameters of stars, for example effective temperature. In this thesis MCMC methods are used to estimate stellar parameters.

The goal of the thesis is to assess the quality of MCMC methods in determining the physical parameters of early-type (hot) stars. The sample for this thesis consists of two stars: KELT-9 and HD 235349. The quality metrics are: reduced chi2, time consumption, proximity to literature values obtained in previous works, acceptance fraction, and autocorrelation time.

The main parts of the project are the Fortran-language Zeeman spectrum synthesis program code and the Python-language wrap-up program code that has an MCMC implementation. The main goal of this thesis is to compare the quality of this wrap-up compared to a Zeeman version that uses a chi2 minimization method instead of MCMC methods.

The optimal configuration for KELT-9 was achieved with the following set of hyperparameters: a wavelength range of 5100-5200 Å, a fixed continuum normalization, 50 walkers, 2000 steps, and an epsilon coefficient of 1.0. Similarly, for HD 235349, the ideal hyperparameters consisted of a wavelength range of 5000-5100 Å, a fixed continuum normalization, 100 walkers, 2000 steps, and an epsilon coefficient of 1.0. The use of continuum normalization as a free parameter was found to be unsatisfactory and yielded inconsistent results for these wavelength ranges. Additionally, Amdahl's law was employed to assess computation and real-time usage, and the results indicated that the project was effectively parallelized. This demonstrates the effectiveness of MCMC for analyzing metal lines and deriving stellar parameters.

This thesis is written in English and is 57 pages long, including 6 chapters, 21 figures and 18 tables.

Annotatsioon

MCMC meetodite rakendamine tähespektroskoopias kõrge temperatuuriga (varajast tüüpi) tähtede füüsiliste parameetrite hindamiseks

Tähtede spektrid on kasulikud tähtede füüsiliste parameetrite, näiteks efektiivtemperatuuri, määramiseks. Selles lõputöös kasutatakse MCMC meetodeid tähtede parameetrite hindamiseks.

Lõputöö eesmärk on hinnata MCMC meetodite kvaliteeti varajast-tüüpi (kõrge temperatuuriga) tähtede füüsikaliste parameetrite kindlakstegemisel. Selle lõputöö valim koosneb kahest tähest: KELT-9 ja HD 235349. Kvaliteedinäitajateks on: vähendatud hii-ruut, ajakulu, lähedus varasemates töödes saadud kirjanduslikele väärtustele, aktsepteerimisosakaal ja autokorrelatsiooni aeg.

Projekti kõige tähtsamad osad on Fortrani-keelne Zeemani spektrisünteesiprogrammi kood ja Pythoni-keelne ümbriskood, milles on implementeeritud MCMC. Selle lõputöö peamine eesmärk on võrrelda selle ümbriskoodi kvaliteeti võrreldes Zeemani versiooniga, mis kasutab MCMC meetodite asemel hii-ruut minimeerimismeetodit.

KELT-9 parim konfiguratsioon saavutati järgmise hüperparameetrite kombinatsiooniga: lainepikkuse vahemik 5100–5200 Å, fikseeritud kontiinumi normaliseerimine, 50 *walker*it, 2000 sammu ning epsiloni koefitsient 1,0. Sarnaselt olid parimad hüperparameetrid HD 235349 puhul lainepikkuse vahemik 5000–5100 Å, fikseeritud kontiinumi normaliseerimine, 100 *walker*it, 2000 sammu ja epsiloni koefitsient 1,0. Kontiinumi normaliseerimise kasutamine vaba parameetrina osutus nendes lainepikkuste jaoks probleemseks ja andis ebakõlalisi tulemusi. Lisaks kasutati Amdahli seadust teadusarvutuste paralleelsuse hindamiseks ning tulemused näitasid, et projekt oli edukalt paralleeliseeritud. See demonstreerib MCMC efektiivsust metalliliste spektrijoonte analüüsimisel ja tähtede parameetrite arvutamisel.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 57 leheküljel, 6 peatükki, 21 joonist, 18 tabelit.

List of abbreviations and terms

MCMC	Markov Chain Monte Carlo
vCPU	Virtual Central Processing Unit
Å	Ångström, wavelength unit (0.1 *nm*)
HD	Henry Draper star Catalogue
UHJ	Ultra-Hot Jupiter
T_{eff}	Effective temperature
log *g*	Logarithmic surface gravity
v sin *i*	Projected rotational velocity
V_r	Radial velocity
metal	Any chemical element except hydrogen and helium
metallicity	Chemical abundance of metal relative to the Solar abundance

Table of Contents

List of Figures

List of Tables

1 Introduction

Stellar physicists use spectroscopic observations, and fitting models to those observations, to evaluate stellar parameters, for example effective temperature. The observed spectrum consists of one-dimensional observations: wavelength on the x-axis and flux (intensity) on the y-axis. The goal is to match the observed points with a function that is as close to the empirical values as possible to get the most accurate estimates according to the model.

Usually, the spectral fitting was done using classical statistical methods [16], such as chi2 minimization. Markov Chain Monte Carlo (MCMC) methods are potentially useful because they eliminate the human factor: continuum fit is not done anymore by hand [7].

This thesis explores this automating alternative, testing and improving the wrap-up Python code [7] that performs an MCMC analysis using the Fortran-language spectrum synthesis code Zeeman [4]. This is tested on a sample of two early-type stars (hot stars) [5, 7].

1.1 Related work

There is a large body of literature on analysis of stellar spectra. Different previous works can be considered related to this thesis. This subsection provides a selection of repesentitive older works that are based on either classical statistics, newer MCMC-based projects or stellar spectrum synthesis program codes other than Zeeman.

1.1.1 Implementations based on chi-square method

Older works in this field have relied on the implementation of classical statistical methods. This is based on chi-square minimization and non-linear least squares methods. The goal is to minimize the chi square value with the least-squares method.

1.1.1.1 Analysis of ELODIE spectrograph observations

One of the older examples dates back to 1998 [16] and is based on the results of ELODIE spectrograph. The approach involves matching the spectrum of the star of interest against a collection of spectra taken with the same spectrograph, typically at a signal-to-noise ratio of 100. A library was constructed for this purpose, comprising spectra from 211 stars specifically chosen to cover the range of parameters observed in stars belonging to the halo, thick disk, and old thin disk of the Milky Way. These parameters include effective temperature (T_{eff}) within the range of 4000K to 6300K, surface gravity (log *g*) ranging from 0.6 to 4.7, and metallicity ([Fe/H]) ranging from -2.9 to 0.35. Since this work is old, the line data that is used there is also not up to current knowledge, therefore this work used observed templates rather than synthetic model spectra. However the precision of the study was limited by the ability to precisely characterize the template spectra.

1.1.1.2 Fingerprints of giant planets among Herbig stars

An important more recent example is the 2012 analysis [27] that was followed by the 2015 analysis [6], in which anomalies occurring in stars of medium mass called Herbig stars were studied. These are young stars that are still surrounded by a disk of material. A correlation between the presence of a cavity in the protoplanetary disk and the scarcity of refractory chemical elements (such as titanium and chromium) was presented.

In [27] Zeeman synthetic spectra was fit to the observed spectra. Wide spectral ranges with many metal lines were used. Effective temperature, surface gravity and distance, luminosity, mass, radius, age, pre-main sequence age and presence of a magnetic field were derived.

Zeeman-code based chi2 fitting routine that used Levenberg–Marquardt algorithm as a minimization method worked fairly well if the right wavelength region was chosen. The main drawbacks of the work were large uncertainties and not considering covariances between the different physical parameters of the stars.

1.1.2 Implementations based on MCMC

One more computationally advanced way to analyze stellar spectra is using MCMC to estimate the stellar parameters.

1.1.2.1 Analyzing WASP-33b

One example of MCMC-based analysis is [19] from year 2020. In that work, atmospheric composition of the giant planet WASP-33b was studied.

Ultra-Hot Jupiters (UHJs) are giant exoplanets that experience extreme heat due to the intense radiation from their host stars. As a result, they offer an excellent opportunity to study the chemistry and physics of planetary atmospheres in extreme conditions.

Using the CARMENES and HARPS-N spectrographs, the authors were able to observe four transits of the UHJ WASP-33b that orbits around the early-type host star WASP-33 (spectral class A5). After adjusting for the Rossiter-McLaughlin effect (refers to the radial velocity variations of a star caused by the planet's orbit) and centre-to-limb variation (refers to the change in the observed brightness of a star as the observer's line of sight moves from the center of the star's disk to its limb, or edge) , they could identify the Balmer Hα, Hβ, and Hγ transmission spectra of the planet's atmosphere. The combined Hα transmission spectrum revealed a significant absorption depth of 0.99±0.05%, indicating that the line probes neutral hydrogen atoms in the high-altitude thermosphere (the outermost layer of a star's atmosphere). While the detection of the Balmer lines was definitive, the strengths of the lines were impacted by the stellar pulsation. Modeling and correction of the spectral pulsation feature in the future will help to better constrain the line strength.

The PAWN model was used, assuming the atmosphere to be hydrodynamic and in local thermodynamic equilibrium, to fit the observed Balmer lines. The model fit provided a thermospheric temperature of $T = 12200^{+1300}_{-1000} K$ and a mass-loss rate $M = 10^{11.8^{+0.6}_{-0.5}} g\,s^{-1}$. The high mass-loss rate is in line with theoretical predictions for UHJs that orbit early type stars.

The Balmer lines had been detected in five UHJs to date (KELT-9b, KELT-20b/MASCARA-2b, WASP-12b, WASP-121b, and WASP-33b). Balmer absorption is likely a typical spectral feature in the transmission spectra of UHJ as their hot atmospheres are intensely irradiated by their host stars, which could produce a large number of hydrogen atoms in the excited state. However, for certain UHJs, such as those with low atmospheric scale heights or those with the Rossiter-McLaughlin effect detecting the Balmer features could be challenging. The authors conclude that extending observations to a larger sample of UHJs will enable systematic study of the Balmer lines and thermospheric conditions. This demonstrates the use of MCMC methods for spectroscopic analysis but this thesis focuses on deriving a wider range of parameters for stars rather than two parameters for planets.

1.1.2.2 Analyzing HD 235349

In 2021, a particular star called HD 235349 was studied [7]. This time, MCMC methods were introduced, which was beneficial for two reasons:

- It is convenient to describe the covariance of the studied parameters (effective temperature and logarithmic gravity) using MCMC.
- MCMC allowed ignoring insignificant nuisance parameters.

The solution was based on Zeeman code and the Python-language wrap-up code also used in this thesis.

At first an analysis of metal lines was done with a standard chi2 method. The errorbars were large and the covariance between the stellar parameters was poor. Then an MCMC-based Balmer line analysis was used, which produced better errorbars for effective temperature and surface gravity values. An advantage of the MCMC analysis here was that a polynomial approximation for the continuum was included. Then the polynomial coefficients could be treated as nuisance parameters and marginalized over, to get more precise values for effective temperature and surface gravity.

The continuum normalization was limited by Balmer lines that were too broad and caused uncertainty. In some stars there are not even good Balmer lines, for example Herbig stars, very hot stars or very cool stars, so this method is incomplete. Also, Balmer lines do not give information about chemical abundances. Metal lines are needed to estimate them.

The authors concluded that HD 235349 is not suitable for studying exoplanets, but as a rare binary star experiencing eclipses, it could instead help study high-temperature stars with chemical composition anomalies. In this thesis, the physical parameters of HD 235349 (including metallicity) will be estimated.

1.1.3 Spectral synthesis codes

There are multiple other stellar spectrum synthesis codes than Zeeman and its Python wrap-up code studied in this thesis. They uses different methods and have slightly different purposes. Here two alternative examples are presented.

1.1.3.1 FASMA and MOOG

One example of another stellar spectrum synthesis program is the FASMA code [14]. The main difference from Zeeman code is that FASMA is specialized in lower-temperature stars. Similarly to the wrap-up code in this thesis, FASMA is a Python wrap-up code around Fortran-based code MOOG [20].

MOOG has an ability to do on-line graphics meaning that the plotting commands are given within the FORTRAN code. The running options of MOOG include *abfind* that applies a method of adjusting species abundances to produce calculated equivalent widths that align with the observed values obtained from other software programs and *synth* that computes a set of trial synthetic spectra and matches them to an observed spectrum if the user asks for it. [20]

1.1.3.2 BACCHUS

Another stellar spectrum synthesis program code is BACCHUS (Brussels Automatic Code for Characterizing High accUracy Spectra) [17]. BACCHUS is a wrap-up around Fortran radiative transfer code Turbospectrum [29]. It is difficult to measure elements with weak and blended spectral features and they require specialized analysis methods to measure their chemical abundances precisely. This code has a unique feature: it uses four different methods to compare the observed and synthetic spectra within the chosen range and includes an abundance measurement for the following four methods:

- The “chi2” method: determines an abundance by minimizing the squared differences between synthetic and observed spectra

- The “syn” method: looks for the abundance that makes the difference between the synthetic and the observed points zero
- The “eqw” method: determines the abundance needed to match the equivalent widths of the synthetic spectra to the observations
- The “int” method: measures abundances by matching the line core in the synthetic and observed spectra

1.2 Contributions

In this thesis, the wrap-up Python code is tested for the first time in an automatized way. MCMC methods have become not that uncommon recent years, however few studies have used MCMC methods to derive stellar parameters from observed spectra, and almost all previous works were specialized on later-type stars unlike this thesis that is specialized on earlier type stars. MCMC was never used together with Zeeman on metal lines before [7] and, in this thesis, it gets a proper hyperparameter choice assessment for the first time. For the first time, metal lines are used instead of Balmer lines. Metal lines are beneficial because they provide diverse information about chemical abundances and precise rotational velocity values.

Also, the Zeeman code will be profiled for the first time, at least systematically. This is needed to assess the quality of past efforts to optimize and parallelize Zeeman.

As a side product, the structure of the project will get a proper documentation. Its folder structure will be described in Appendix 2.

1.3 Problem statement

The goal of this thesis is to run a set of experiments on the MCMC implementation and compare the results to the ones using the chi-square method. Additionally, the wrap-up Python code [7] will be improved and Zeeman code will be profiled.

In this thesis, the results of the the following hyperparameter combinations are compared that differ by the following aspects:

- Status of each parameter: free / fixed / unspecified
- Number of MCMC walkers
- Length of each chain
- Wavelength range
- Initial distributions of the parameters

I will compare both accuracy and efficiency of several combinations. Also, the MCMC results will be compared to results generated by the chi-square method.

The testing will be done using spectral observations of two different stars: KELT-9 and HD 235349. This initial data was collected before the thesis [5, 7].

1.4 Structure of the thesis

The thesis is structured as follows. Chapter 2 provides a brief overview of stellar spectroscopy, describes the Zeeman spectrum synthesis code, and presents the observed datasets used in this thesis. Chapter 3 gives an overview of the Markov Chain Monte Carlo class of Bayesian methods, and their *emcee* implementation in Python. Chapter 4 describes the process how the results are obtained. Chapter 5 analyzes the results. Chapter 6 concludes the thesis and proposes some future prospects.

2 Stellar physics and Zeeman stellar spectrum synthesis

Stars differ by many characteristics including age, mass and temperature. Astronomers have developed different methods to assess those parameters. The quantitative analysis of stellar spectra is one of the most powerful methods. This chapter introduces stellar spectroscopy and characteristics of stars and Zeeman code that will be given datasets to elaborate.

2.1 Stellar physics

2.1.1 Stellar characteristics

Stars are divided into spectral classes based on their temperature. The highest temperature stars are O-stars and the lowest are marked with an M-letter [10]. The table below summarizes the temperature ranges for the corresponding letters.

Table 1. Stellar classification and the corresponding ranges of effective temperature in Kelvins [11]

Spectral class	**Effective temperature**
O	28,000K to 50,000K
B	10,000K to 28,000K
A	7,500K to 10,000
F	6,000K to 7,500K
G	4,900K to 6,000K
K	3,500K to 4,900K
M	2,000K to 3,500K

Each spectral class is divided into 10 subclasses from hottest to coolest: from 0 to 9. Stars with smaller index are referred to as early-type stars and the ones with larger index as late-type stars. For example, A0-star is an early A-type star and A9-star is a late A-type star. The spectral class of the Sun is G2. [10]

Hertzsprung–Russell (H–R) diagram shows the temperature of the star on x-axis and its luminosity on y-axis.

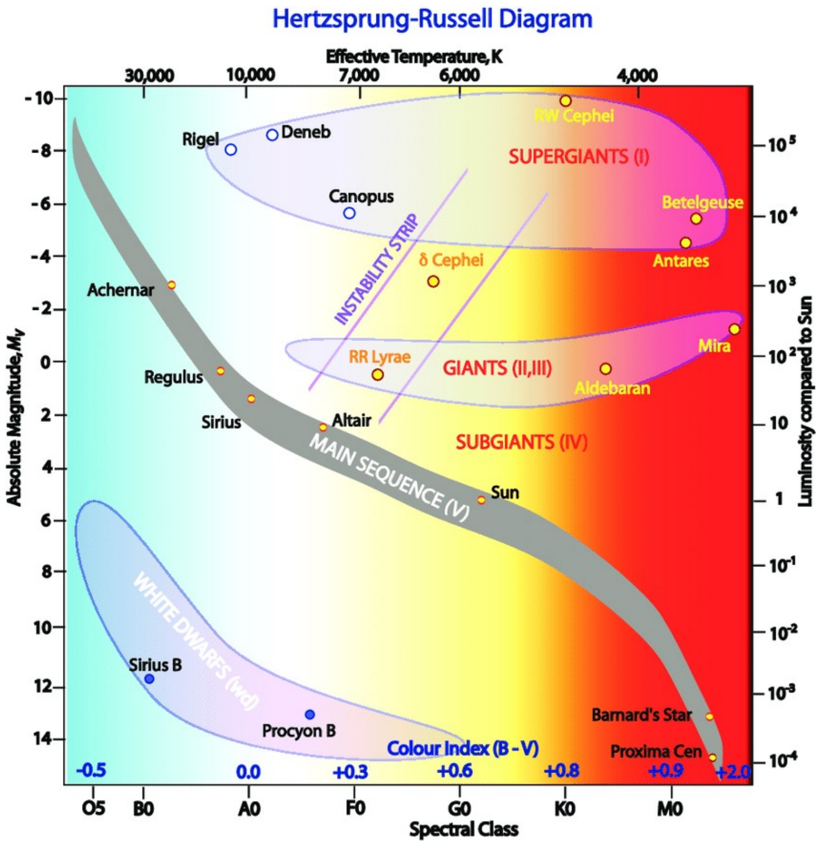


Figure 1: Hertzsprung–Russell diagram [9]

Most stars on the diagram are main sequence stars meaning that they are in the main phase of their lifetime.

The evolution of a star depends on its mass. Lower mass stars have longer lifespans. For example, on Figure 2, Sun has longer lifespan than more massive Sirius but shorter lifespan than less massive Barnard’s star.

2.1.2 Stellar spectroscopy

One of the main methods used in stellar physics is spectroscopy. The spectrum can tell a lot about the star's characteristics, for example its mass, temperature and its surroundings, for example planets and protoplanetary disk.

Spectral fitting is a process where the synthetic spectrum is matched to the observed (empirical) spectrum in order to determine model parameters. It helps to estimate multiple parameters, including effective temperature, radial velocity, projected rotational velocity, surface gravity and microturbulence.

There are different types of temperature measures. Effective temperature is the temperature the star would have if it were a perfect blackbody, based on the total luminosity it emits. This temperature is a crucial global characteristic representing the surface temperature of the star. It is calculated from the Stefan-Boltzmann law that states that the total radiation emitted by a blackbody is proportional to the fourth power of its temperature. [10]

the Stefan-Boltzmann law [10] can be stated as follows:

$$L = 4\pi R^2 \sigma T^4$$

where L is the total energy radiated per unit time by a spherical blackbody of radius R and temperature T, and $\sigma = 5.67 x 10^{-8} W/m^2 K^4$ is the Stefan-Boltzmann constant.

Abundance analysis is an important part of stellar spectroscopy. Every atomic and molecular species has unique wavelength lines that indicate its presence. In astronomy, all elements except hydrogen and helium are called metals. In this thesis, metallicity is defined as the relationship between metal abundance and hydrogen abundance.

There are two types of spectral lines: absorption lines and emission lines. Absorption lines appear as dark lines in the spectrum, caused by the absorption of specific wavelengths of light by atoms in the outer layers of the star's atmosphere. The wavelengths of the absorption lines correspond to the energy levels of the atoms and molecules present in the star, and can be used to identify the chemical elements present in the star's atmosphere. The strength of absorption lines can be used to infer the abundance of that element in the star. Emission lines, on the contrary, appear as bright lines in the spectrum, caused by the emission of specific wavelengths of light by excited atoms in the star's atmosphere. These emission lines also correspond to the energy levels of the species present, and can be used to determine the temperature and chemical composition of the emitting gas. [10]

Three different phenomena cause line broadening [10]:

- Natural broadening: Even if they are stationary and unconnected to other atoms, spectral lines cannot have an infinitely precise shape. This is due to Heisenberg's uncertainty principle, which implies that the more limited the time available for an energy measurement, the greater the inherent imprecision of the outcome.

- Doppler broadening: Doppler broadening occurs due to the thermal motion of atoms or molecules in a gas. As atoms move towards or away from an observer, the frequency of the radiation they emit or absorb is shifted, causing the spectral lines to appear broader. The degree of broadening depends on the temperature and velocity of the gas and can provide information on these properties. In addition to thermal Doppler broadening, there are other microscopic processes that can cause spectral line broadening, such as microturbulent broadening, which arises due to small-scale velocity fluctuations in the gas. On a larger scale, global Doppler broadening can occur due to the rotational motion of a star, causing the spectral lines to appear broader. Rotational broadening is an important effect in astronomy, as it can be used to determine the rotation rate of a star and other properties of its atmosphere.

- Pressure and collisional broadening: When an atom collides with a neutral atom or experiences a close encounter with the electric field of an ion, its orbitals can be disrupted. The outcome of such collisions is known as collisional broadening. On the other hand, the cumulative impact of the electric fields of a large number of closely spaced ions is referred to as pressure broadening.

Opacity is a measure that shows how much light is being absorbed or scattered. It can result in a decrease in the intensity of light emitted by the star at certain wavelengths, which produces dark lines in the spectrum. These spectral lines can reveal important information about the composition, temperature, and other properties of the star's atmosphere. [10]

2.1.2.1 Important spectral lines

Every atom and even molecule has unique absorption lines. This subsection introduces examples of fingerprints of specific chemical elements.

The best known lines in spectroscopy are Balmer lines: a series of spectral emission lines in the visible region of the hydrogen atom's spectrum. The Balmer series includes several lines, such as Hα (6562.81), Hβ (4861.34), Hγ (4340.48), Hδ (4101.75) and so on, which correspond to transitions between the excited states and the second energy level of the hydrogen atom. [10]

In this thesis, it is chosen to focus on metal lines, and the wavelength ranges are chosen in a region of the spectrum where A-type stars have a relatively high density of strong metal lines: 5000-5100 Å and 5100-5200 Å.

The following table presents the beginning of the wavelength range used in this thesis.

Table 2. Some examples of the spectroscopic lines

Line	**Wavelength (Å)**
Fe 2	5000.7304
Ca 2	5001.4790
Fe 1	5001.8630
Fe 2	5001.9529

2.2 Zeeman stellar spectrum synthesis

Zeeman is a Fortran-based stellar spectrum synthesis program. Current Zeeman program code (Zeeman2) is an updated version written in Fortran95 of an older Fortran77 program [4, 28]. The motivation of Zeeman was not to map magnetic field or abundances in a detailed way, but instead to get approximate models of both [4]. The code has a built-in fitting mechanism that is based on classical statistical methods: least-squares method and chi-square method [27].

Zeeman code is specializing mainly on earlier type stars. The stars should also be in the main sequence.

2.2.1 Precomputed datasets

Zeeman is using the results of another program named Atlas9 [22]. It is an atmosphere model library. For this thesis, the program is not run separately but is using already precomputed results that are stored in *.dat* files. Each file is specialized on a different atmosphere layer starting from the highest. The files are divided into five columns:

- column mass - the mass above the given atmosphere layer (g/cm^2)
- effective temperature (K)
- electron number density - number of electrons not attached to atoms (cm^{-3})
- ion number density - number of ionized atoms (cm^{-3})
- mass density (g/cm^3)

The thesis project also contains MARCS folder. It is similar to Atlas9 but is currently not used because it is designed for cooler stars. It will probably become important in the future when the project will be extended to a wider temperature range.

In addition to Atlas9, Zeeman is using data from VALD (The Vienna Atomic Line Data Base) [21]. The database contains useful data regarding atomic transitions that have a significant impact on absorption in the spectra of stars. In this thesis, the most important data about each absorption line includes: the element and its ionization status (for example “Fe 1” is neutral iron and “Fe 2” is ionized iron), the wavelength of the line in Å (1 Å = 0.1 nm), oscilator strength (unitless logarithmic quantity) and the excitation potentials (energies above the ground state) for the lower and upper levels of the transition (in eV). This data is necessary for calculating absorption in the modeled spectra and is located inside the file *vlines.dat* included in this project.

2.2.2 Parallelization

Spectral line synthesis is characterized as an “embarrassingly parallel” problem. The possibility to parallelize every specific wavelength makes the program very granular. The step size between two wavelengths is chosen to be 0.01 Å = 0.001 nm by default. [4]

The speedup of Zeeman code can be described by Amdahl’s law [4, 26]. It expresses a maximum speedup achievable by utilizing many processors in parallel. Maximum speedup S with P processors is equal to

$$S=\frac{1}{F+\frac{1-F}{P}}$$

where F is the sequential fraction of the calculation. The following figure illustrates how the maximum speedup of the program depends on the parallelizable fraction.

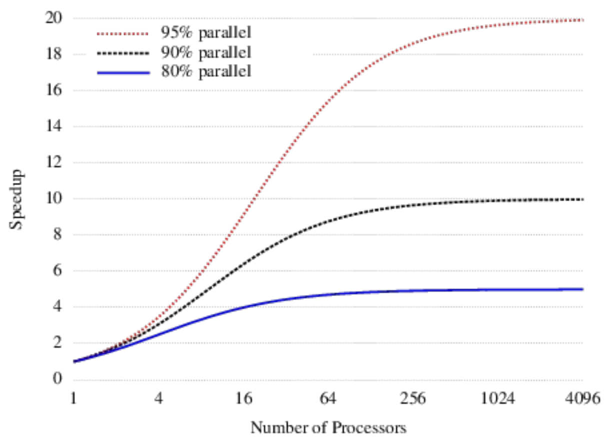


Figure 2: Amdahl’s Law [2]

2.2.3 Involved parameters

Zeeman estimates different parameters of the star using the given spectrum. This subsection gives a brief overview of them.

Table 3. Stellar parameters in Zeeman code

Short name	**Name / description**	**Unit**	**Included?**
Vr	Radial velocity	cm/s	yes
vsini	Projected rotational velocity	cm/s	yes
Vmic	Microturbulence	cm/s	yes
Vmac	Macroturbulence	cm/s	no
Teff	Effective temperature	K	yes
logg	Surface gravity	$\log_{10}$ cm/s^2	yes
metal	Metallicity	Logarithmic relative to Solar [X/H], where X is the scaling	yes

		factor for all metals	
contFlx	Additional continuum flux	Fraction of the stellar continuum flux	no
Bmono	Magnetic field strength for a uniform radial magnetic field	Gauss (1 G = 0.0001 T)	no
FFmono	Filling factor for uniform radial magnetic field	-	no
Bdip	Dipole magnetic field strength	Gauss	no
FFdip	Filling factor for dipole magnetic field strength	-	no
element	Atomic number of elements with specified (non-solar) abundances	-	no
abun	Abundance for that element	Logarithmic number density relative to H	no
contNorm	Continuum normalization	-	yes

The latter seven parameters are lists not single values and *element* is always fixed.

All the parameters can be set either as free, fixed or unspecified (using the default value 0.0 or using the Solar value (elements)). This thesis attempts to set as many parameters free as possible but if this will significantly decrease efficiency or accuracy, concessions will be made.

2.2.4 Structure and main functions

Zeeman code is located in multiple files and is based on subroutines. A brief overview of them is given below.

The function *zeemanu_*, calls readvald_ function that calculates the oscillator strength. *zeemanu_* then calls *dskint_* function that calculates opacity.

*dskint_*, in turn, calls *voigt_* function that is used to calculate line opacities and then it calls *linpro_* that calculates actual radiative transfer using those line opacities to produce spectra at one point on the stellar surface. *dskint_*, repeats this process for a range of positions on the stellar surface. After these are done, *dskint_* calculates Doppler shifts and adds the local spectra up to produce a ‘disk integrated’ spectrum.

If the code runs into a Balmer line then *voigt_* function calls *hline-extras_* to calculate hydrogen line opacity correctly.

2.2.5 Interaction with the wrap-up Python code

The interaction between the Zeeman code and the MCMC method occurs in the Python wrap-up code that is called *zemceeWrap03cont.py*.

The interaction with the Zeeman code occurs in *lnlike* function of the Python wrap-up code (see appendix for more details). This function takes the free parameters, keywords for the free parameters, as well as a dictionary for the fixed parameters, as inputs and passes all of them to Zeeman. Unspecified parameters are read from the standard Zeeman input files. Zeeman code is called there with *lmamp*. The inputs are passed by writing files for Zeeman, then the Zeeman executable is ran, and thereafter the resulting spectrum file is read back in.

Radial velocity value (V_r) is used to calculate the Doppler shift.

Next, the function *getZeemanSpec* is used to retrieve the resulting wavelengths of the Zeeman computation and their corresponding intensities.

After that, continuum normalization values are retrieved. The continuum normalization values are used to scale (re-normalize) the continuum level of the model. Then the resulting re-normalized model is interpolated onto the observed pixel wavelengths. They are used to interpolate over the observed values.

The function also calculates chi2 and reduced chi2 values. chi2 is calculated as

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

where O is the observed value and E is the estimated value according to the model.

Reduced chi2 [15] calculates chi2 per degrees of freedom. It helps to assess the accuracy of the results and should be relatively close to 1. If it is much smaller than 1, errorbars are overestimated or overfitting has taken place. On the other hand, much larger value than 1 suggests either underestimated errorbars or a low-quality fitting. It is calculated as follows:

$$\chi_v^2 = \frac{\chi^2}{v}$$

where v is the number of degrees of freedom. In this project, the degrees of freedom are the number of pixels in the observed spectrum minus the number of free parameters in the fit.

Finally, the function returns logarithmic likelihood *lnlike* $-0.5 * \chi^2$.

2.3 Datasets

The following thesis uses two datasets. I tested the code on the observations of two different early-type stars: KELT-9 and HD 235349.

The first dataset is taken from observations of the A0-type star KELT-9 [5]. The observations come from the Gemini Observatory GRACES spectrograph.

The second spectrum used in this thesis belongs to the B6-type star HD 235349. The data was collected in the Tõravere Observatory and published in [7]. This dataset has lower spectral resolution than the one of KELT-9: only 0.2 Å compared to the one of KELT-9 (0.0287 Å).

Both datasets consist of three columns:

- wavelength (Å)
- normalized flux (intensity) (unitless)
- uncertainties of normalized flux (unitless)

These particular datasets were chosen because the stars were already observed and had reasonable parameter estimates in previous published works. KELT-9 is interesting because it is orbited by the hottest discovered exoplanet KELT-9b. This planet evaporates sometimes and it can cause peculiarities in the chemical composition of KELT-9. [5] HD 235349 was a candidate of planet hosting in TESS mission and is also chemically peculiar. [9]

3 Markov Chain Monte Carlo methods

Markov Chain Monte Carlo (MCMC) [23] is a class of numerical methods based on Bayesian statistics. Classical statistics treat parameters as fixed values and observed data as random variables but in Bayesian statistics it is the other way around: the observed data is fixed and the goal is to estimate parameters. This chapter introduces MCMC methods, focusing on *emcee* implementation [3] in Python.

3.1 Bayes theorem

Bayes theorem helps calculating conditional probability of the event B if another event A is observed:

$$P(B|A)=\frac{P(A|B)P(B)}{P(A)} \quad .$$

The prior $P(B)$ is the initial probability assigned to a hypothesis or parameter before any data is observed. It represents the researcher's prior belief or knowledge about the hypothesis or parameter.

The likelihood $P(A|B)$ is the probability of observing the data given a specific hypothesis or parameter. It represents the strength of the evidence provided by the data.

The posterior $P(B|A)$ is the updated probability of the hypothesis or parameter after taking into account the observed data.

The marginal $P(A)$ is the probability of the data, obtained by integrating or summing over all possible values of the parameters or hypotheses. It represents the total probability of observing the data, regardless of the specific values of the parameters or hypotheses.

Bayes theorem can be extended [1] so that there are $m\geq 2$ possible events in a vector $B=(B_1, B_2, \ldots, B_m)$:

$$P(B_k|A)=\frac{P(A|B_k)P(B_k)}{P(A)}$$

where the marginal $P(A)$ is computed as follows

$$P(A)=\sum_{i=1}^{m} P(A|B_i)P(B_i) \quad .$$

In case of continuous sets there are probability density functions instead of the probabilities:

$$p(B_k|A)=\frac{p(A|B_k)p(B_k)}{p(A)} \quad .$$

The sum is replaced by the integral, so the marginal is computed as follows:

$$p(A)=\int_{i=1}^{m} p(A|B_i)p(B_i) \quad .$$

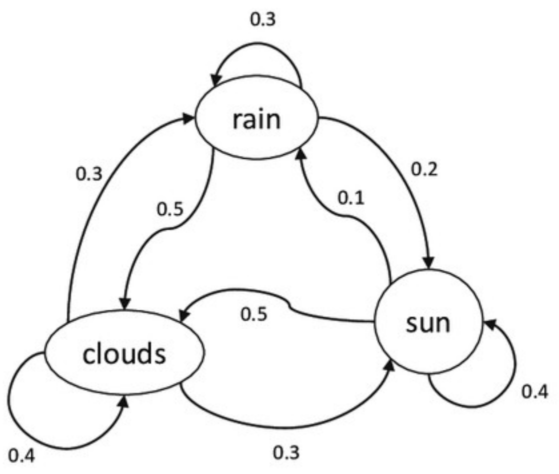
It is often expensive or intractable to compute the margin in multi-dimensional cases. This is where MCMC methods come to help.

3.2 MCMC methods in general

This class of methods combines two concepts: Markov chains and Monte Carlo methods. A brief overview of both is given below.

3.2.1 Markov chains

Markov chains result from stochastic (random) processes where the next step of the chain is dependent solely on the current step. They can be described by a transition graph or a transition matrix.



$$\begin{array}{c|ccc} & \text{clouds} & \text{rain} & \text{sun} \\ \hline \text{clouds} & 0.4 & 0.3 & 0.3 \\ \text{rain} & 0.5 & 0.3 & 0.2 \\ \text{sun} & 0.5 & 0.1 & 0.4 \end{array}$$

Figure 3: Example of a transition graph and matrix [13]

Convergence of a Markov chain refers to the tendency of the chain's distribution to approach a stable distribution over time that is called stationary distribution. Markov chain converges to a single stationary distribution if it is ergodic. Ergodicity of a Markov chain is the property that the long-term behavior of the chain is independent of its starting state. In other words, if the chain is ergodic, the distribution of the chain after a large number of steps will be the same regardless of the initial state. [23]

Markov chain is ergodic if the following criteria is satisfied [23, 24]:

- Irreducibility: it is possible to reach any state in the chain from any other state, directly or indirectly.
- Aperiodicity: it is possible to return to any state in the chain at any time. This means that the chain does not follow a predictable pattern, such as returning to a given state every other step.
- Positive recurrence: the expected number of steps to return to a state is finite. The chain will eventually return to a given state with probability 1.

3.2.2 Monte Carlo methods

Monte Carlo methods are a broad class of computational algorithms that use random sampling to solve problems that involve numerical integration, optimization, or simulation. The goal is to generate a large number of random samples to estimate the probability distribution of the given problem.

One common application of Monte Carlo methods is to estimate the value of complex integrals that cannot be solved analytically. In this case, random samples are drawn from the probability distribution of the integral, and the average of the function over these samples is used to approximate the integral value.

Importance sampling [25] is one of the Monte Carlo integration methods. The goal is to compute the expectation value of $f(x)$

$$E[f(x)] = \int p(x) f(x) dx$$

where $p(x)$ is a simple distribution (for example uniform or normal). We sample a large number of values from $x_i \sim p(x)$ and calculate the value of the function $f(x_i)$ using them as input. The average of the results is the approximation of the desired integral:

$$\int p(x) f(x) dx \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i), x_i \sim p(x) \quad .$$

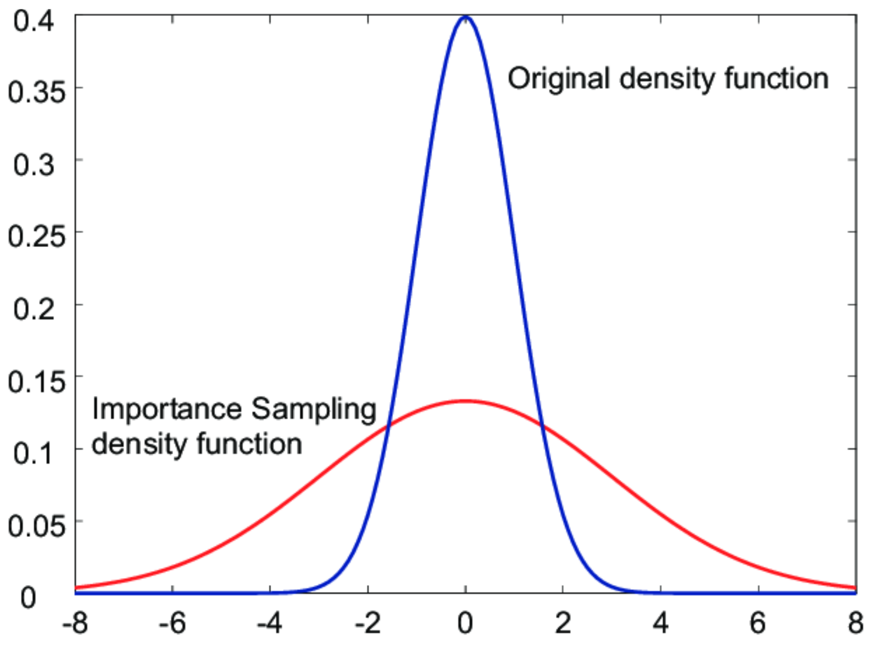


Figure 4: Importance sampling [18]

3.2.3 MCMC methods and their benefits

MCMC methods are useful because they eliminate the need to calculate the marginal in the Bayesian theorem that can be an intractable multidimensional integral. The goal is to estimate the parameters by comparing models to the data.

In MCMC, the input is the model parameter values at the initial time. During the algorithm's design phase, a proposal distribution is created to govern the transition from one state to another, serving as a guide for generating new candidate parameter values based on the current state. The output after many iterations is a sample of the posterior distribution.

Since the first step of the chain is chosen relatively arbitrarily, the beginning of the chain is usually discarded because otherwise the result would be biased. This beginning section is called a burn-in phase. [23]

The curse of dimensionality refers to the difficulties that arise when working with high-dimensional spaces, where the number of possible configurations increases exponentially with the dimensionality of the space. In such spaces, it becomes increasingly difficult to explore and sample from the distribution of interest, leading to inefficiencies and inaccuracies in Monte Carlo methods. [23]

MCMC methods are better at handling the curse of dimensionality than other Monte Carlo methods because they use Markov chains to explore the distribution of interest. The Markov chain allows the algorithm to focus on the regions of the distribution that are most relevant and to move efficiently between them, even in high-dimensional spaces. MCMC methods can also be designed to exploit the local structure of the distribution, which can be used to further improve their efficiency in high-dimensional spaces. [23]

3.3 Metropolis-Hastings algorithm

The most famous and the most commonly used MCMC method is Metropolis-Hastings algorithm [3]. The goal is to draw sample from the posterior distribution p. This is done with the help of proposal distribution Q, which is usually selected to be normal distribution.

First, the beginning value of the chain is initialized. Then a new value is accepted with probability, that is calculated using the following transition kernel:

$$min(1, \frac{p(Y|D)}{p(X(t)|D)} \frac{Q(X(t);Y)}{Q(Y;X(t))}) \quad .$$

There is a need to decide if the new value of the walker will be the same as the previous one or not. The decision is done with the help of the random value r, that is sampled from uniform distribution [0,1]. If $r \leq q$ then the process is repeated with the new value.

The following figure summarizes Metropolis-Hastings algorithm.

```
for t = 1, . . . , iterations do
   Draw a proposal Y ~ Q(Y ; X(t))
   q ← [p(Y ) Q(X(t); Y )]/[p(X(t)) Q(Y ;
X(t))] // This line is generally expensive
   r ← R ~ [0, 1]
   if r ≤ q then
     X(t + 1) ← Y
   else
     X(t + 1) ← X(t)
Return X
```

Figure 5. Metropolis-Hastings algorithm [3]

3.4 Ensemble of walkers and *emcee*

Standard Metropolis-Hastings algorithm has drawbacks. Since only one chain is being used, it can get stuck in local maxima and not get to explore other high probability regions. Also, the algorithm isn’t easy to run in parallel computations.

The solution is to run multiple chains. An ensemble of walkers is a system of multiple chains that can run in parallel.

3.4.1 Affine invariance

Foreman-Mackey et al. implemented a Python package *emcee* [3]. The most important property of the implementation is affine invariance, meaning that the performance does not depend on the aspect ratio in highly anisotropic distributions [8].

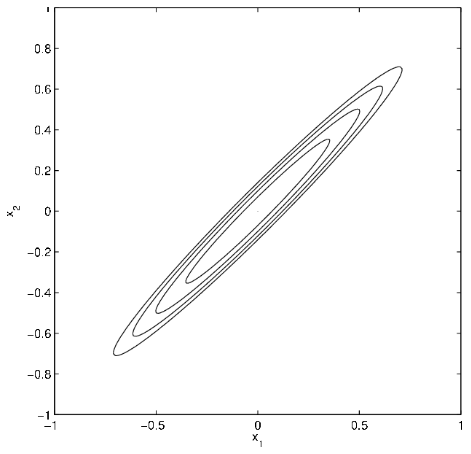


Figure 6: Example of a highly anisotropic distribution [8]

Affine transformation [8] is defined as an invertible mapping from R^n to R^n of the form $y=Ax+b$. If the probability density of X is $\pi(x)$, then probability density of $y=Ax+b$ is

$$\pi_{A,b}(y)=\pi_{A,b}(Ax+b) \propto \pi(x) \quad .$$

Affine invariance is a useful property of an object or a mathematical function that remains unchanged under affine transformations. An object or a function is affine-invariant if it retains the same shape, size, and orientation after an affine transformation. An affine-invariant algorithm operates on data without being affected by affine transformations, so it is more robust and reliable.

3.4.2 Stretch moves

The stretch move is similar to a regular step in Metropolis-Hastings algorithm but now the transition kernel also involves another walker. Next, two types of stretch moves, serial and parallel are introduced.

In case of serial stretch move [3], the walker at position X_k is updated with the help of another walker X_j that is randomly chosen from the set of all other walkers, also called *complementary ensemble* $S_{[k]}$. The new proposal position is calculated as follows:

$$X_k(t) \rightarrow Y = X_j + Z[X_k(t) - X_j]$$

where Z is a random variable that is sampled from a distribution $g(Z=z)$. If g satisfies

$$g(z^{-1})=z\,g(z)$$

then the previous proposal function is symmetric meaning that the probability of proposing a new state Y from the current state X_k is the same as the probability of proposing the current state X_k from the new state Y . The proposal is accepted with probability

$$min(1,Z^{N-1}\frac{p(Y)}{p(X_k(t))})$$

where N is the number of dimensions in the parameter space. The procedure is repeated for all the members in the ensemble.

```
for k = 1, . . . , K do
  Draw a walker $X_j$ at random from the
complementary ensemble $S_{[k]}(t)$
  z ← Z ~ g(z)
   $Y \leftarrow X_j + z[X_k(t) - X_j]$
   $q \leftarrow z^{N-1} p(Y)/p(X_k(t))$  //expensive
  r ← R ~ [0, 1]
  if r ≤ q then
      $X_k(t+1) \leftarrow Y$
else
      $X_k(t+1) \leftarrow X_k(t)$
  end if
end for
```

Figure 7. A single serial stretch move [3]

In case of parallel stretch move [3], the *complementary ensemble* $S_{[k]}$ is split into two subsets $S^{(0)}=X_k, \forall k=1,\ldots,K/2$ and $S^{(1)}=X_k, \forall k=K/2+1,\ldots,K$ to satisfy the detailed balance meaning that the chains will converge to the desired distribution [12]. Now, it is possible to simultaneously update every walker in $S^{(0)}$ using only positions of walkers in the other set $S^{(1)}$. Then, we can update $S^{(1)}$ with the new positions in $S^{(0)}$.

```
for i ∈ {0, 1} do
  for k = 1, . . . , K/2 do
    //the loop can now in parallel for all k
    Draw a walker $X_j$ at random from the
complementary ensemble $S_{[k]}(t)$
    z ← Z ~ g(z)
    $Y \leftarrow X_j + z[X_k(t) - X_j]$
    $q \leftarrow z^{(n-1)} p(Y)/p(X_k(t))$
    r ← R ~ [0, 1]
    if r ≤ q then
        $X_k(t+1/2) \leftarrow Y$
    else
        $X_k(t+1/2) \leftarrow X_k(t)$
    end if
  end for
  t ← t + 1/2
end for
```

Figure 8. Parallel stretch move update [3]

The authors note that the affine invariant ensemble with stretch moves outperforms significantly standard Metropolis-Hastings algorithm. It is faster, especially if the distribution is highly skewed. [3]

3.4.3 Startingpoints of the walkers

There are three ways to initialize walkers in *emcee* [3]:

- to initiate the process by either sampling from the prior distribution or distributing the starting points within a reasonable parameter space range
- to begin in a really narrow N-dimensional sphere in parameter space around one point which is guessed to be near the maximum probability point
- to start by sampling from the prior distribution and then proceeding with a "burn-in" phase. During this phase, the prior is gradually transformed into the posterior by raising the “temperature”, meaning that the step sizes get gradually faster.

Despite the first method being more objective, in practice, the second one turns out to be much more effective if there is any risk that walkers will get stuck in low probability modes of a multi-modal probability landscape. If the walkers were initialized in the small ball, they will expand out and fill the needed parts of parameter space. It takes only a few autocorrelation times. [3]

In stellar spectroscopy it is possible to make a good guess for the starting point of the walkers from the spectrum of the studied star. Each spectral class is characterized by typical absorption lines that can be seen on a spectrum. In this thesis, all the walkers start in a Gaussian probability distribution that is centered around the chosen value that is approximate typical value of the given spectral class.

3.5 MCMC diagnostics

There are two main ways to assess the quality of chains in *emcee*: autocorrelation time and acceptance fraction. They are described below.

3.5.1 Autocorrelation time

The autocorrelation time measures how many steps it takes to get independent samples [3].

The time series $X(t)$ can be described by autocovariance function

$$C_f(T) = \lim_{t \to \infty} cov[f(X(t+T)), f(X(t))]$$

where $f(\Theta)$ is the expectation value of a function of the model parameters Θ .

The formula above calculates the covariances between samples at a time lag T . The value $C_f(T) \to 0$ measures the needed number of samples to secure independence. The integrated autocorrelation time is the key metric for evaluating the effectiveness of the sampler:

$$\tau_f = \sum_{T=-\infty}^{\infty} \frac{C_f(T)}{C_f(0)} = 1 + 2\sum_{T=1}^{\infty} \frac{C_f(T)}{C_f(0)} \quad .$$

$C_f(T)$ for a Markov chain of M samples is estimated in practice as follows:

$$C_f(T) \approx \frac{1}{M-T} \sum_{m=1}^{M-T} [f(X(T+m)) - \langle f \rangle][f(X(m)) - \langle f \rangle] \quad .$$

The authors of [3] suggest to give walkers approximately 10 times more steps than the autocorrelation size is. Too many steps is an inefficient use of computer resources.

The main problem with autocorrelation is that it is difficult to estimate. If the chains are too short, the outcome can be wrong because *emcee* cannot "see" the right autocorrelation time. [3] To prevent this problem, *emcee* warns the user if the chain size is shorter than 50 times the integrated autocorrelation time for all free parameters and asks to use estimate with caution and run a longer chain.

3.5.2 Acceptance fraction

The acceptance fraction is the fraction of proposed values for the chain that are accepted. It should be optimal, not too low or too high (as a rule of thumb between 0.2 and 0.5). When the acceptance fraction is close to 0, it means that the majority of proposed moves are rejected, resulting in a chain with very few independent samples. This leads to poor representation of the target density and inefficient sampling. Similarly, when the acceptance fraction is close to 1, it indicates that almost all proposed moves are accepted, causing the chain to behave like a random walk with little regard for the target density. As a result, this also leads to poor representation of the target density and inadequate sampling. [3]

According to tests run by the authors of *emcee* the stretch scale parameter default value chosen by them $a=2$ is good in almost all situations, except complicated multimodal distributions [3].

4 Workflow

The practical contribution of this thesis consists of four parts with the main emphasis on the second one:

- improving and editing the code
- testing the performance of the code with different combinations of parameters
- determining the correlation between the stellar parameters
- running the pure version of Zeeman based on chi2 method
- profiling the code

The process is described below.

4.1 Improving and editing the project

While the main goal of the thesis was to test the code and compare different combinations of results, the code also got some improvements and additions. This subsection describes these.

A file called *requirements.txt* was created to include all Python modules that need to be installed when downloading the project for the first time.

I wrote a short *matplotlib* code *plotSpectra.py* to plot the three different spectra for comparison:

- the observed (empirical) spectrum
- the synthetic spectrum fitted by MCMC
- the synthetic spectrum fitted by MCMC after interpolation

The dataset of KELT-9 happened to have some unreliable values in the spectral region of blue light because the measuring instrument was not specialized on these wavelengths. Values smaller than zero were not taken into account.

The Python wrap-up code used to read some data in an inefficient way and had some duplications. Wavelength ranges that are used for the comparing model and the observation was modified to be read directly from *zmodel.dat* file.

The project was also designed to print out the percentage of the work that is completed to give the user some estimate how much time is left until the end.

4.2 Testing combinations of hyperparameters

It is possible to use different hyperparameter combinations: some hyperparameters have to be free and others have to be fixed. Also, it is possible to choose different number of walkers, chain size and burn-in size. There are also different wavelength ranges that can be used for fitting. Finally, the MCMC walkers can be given different initial values, described by the central values and standard deviations (epsilons) of the distributions of the free model parameters. This subsection describes all different experiments and their results.

In the beginning, first informal experiments were done by hand to get better overview of the program and convergence of the chains with the dataset of KELT-9. Then, a more systematic testing was done.

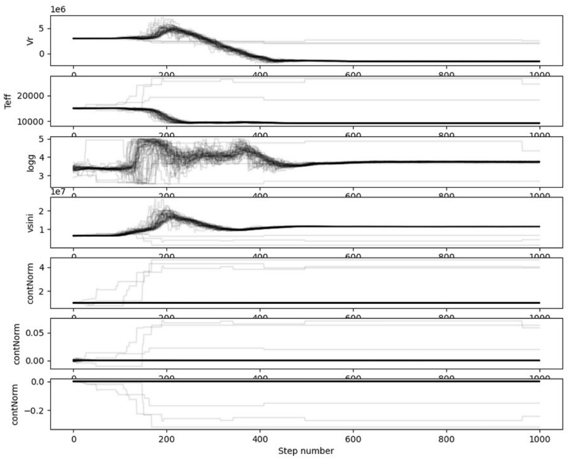


Figure 9: Resulting plot of walkers with 50 walkers and 1000 steps. Values for each model parameter are shown as a function of the step in the chain, with a line for each walker.

The initial values of the walkers were set by randomly sampling a Gaussian distribution in each free model parameter. The center of the starting points of the walkers were later chosen to be typical values of the given spectral class. Also, the standard deviation values were set (also referred as epsilons in *emcee* and this thesis). The initial values and epsilons tested are given in Table 4.

Table 4. Startingpoints and epsilons of the walkers

Parameter	**Value**	**Epsilons**
Teff	KELT-9: 10000.0 HD 235349: 15000.0	100.0
Vr	30.0e5	0.1e5
logg	3.4	0.1
vsini	65.0e5	1.0e5
Vmic	2e5	0.01e5
metal	0.0	0.1
contNorm	[1.0, 0.0, 0.0]	[0.01, 0.001, 0.001]

The burn-in phase was set to be half of the chain size, so the first part of the chains got discarded. This means that 500-step chains discard first 250 steps and 2000-step chains first 1000 steps.

Later, the testing system was automated using the *subprocess* and *argparse* modules in Python. s*ubprocess* allows calling other executable programs as though they were subroutines of the calling program and *argparse* allows to create Linux command line parameters. This implementation (*testCombinations.py*) was based on nested for-loops to reflect the multi-dimensional nature of the experiments. *testCombinations.py* calls the wrap-up code *zemceeWrap03cont.py*, the spectra plotting code *plotSpectra.py*, and *plotChain.py* that plots walkers and makes a corner plot, all with *subprocess* using the parameter combination generated by the nested for-loops.

The routine of experiments consists of two values for each hyperparameter, given in Table 5.

Table 5. Values in the set of experiments

Hyperparameter	**First value**	**Second value**
Number of walkers	50	100
Chain length	500	2000
Epsilons	1	5
Vmic	0 (fixed parameter)	1 (free parameter)
metal	0 (fixed parameter)	1 (free parameter)
contNorm	0 (fixed parameter)	1 (free parameter)

Epsilon values 1 and 5 mean that in the first version original epsilon values from Table 4 are used but in the second one, all of them are multiplied by 5. *ContNorm* is represented as a list, so all its elements get multiplied by 5.

To reduce the time consumption, the whole folder was zipped and transferred to a virtual machine with 48 vCPUs in Google Cloud.

Before the final experiment, a rough estimate was made of the time consumption. Some ensembles were run on the cloud and all of them were 50 steps long. It took approximately 10 minutes to run each such ensemble. So running two 500-step experiments and two 2000-step experiments should take approximately

$$10 \cdot 2 \cdot \left(\frac{500}{10} + \frac{2000}{10}\right) = 500\,min \quad .$$

If *contNorm*, *metal* and *vmic* can all be either fixed or free and there are two possible *epsilons*, the whole set should take approximately

$$2 \cdot 2 \cdot 2 \cdot 2 \cdot 500 = 8000\,min \quad .$$

To reduce the time consumption by a factor of four, *vmic* was set always as fixed and *metal* always as a free parameter.

Two wavelength ranges were chosen in this work 5000-5100 Å and 5000-5200 Å. This spectrum region has remarkable advantages: a high density of strong lines that makes estimating T_{eff} easier, and also a good signal-to-noise ratio. In lower wavelength regions the density of the lines is even higher but the lines bend too much together. On the other hand, in higher wavelength region the lines are located more sparsely.

The set of experiments was run in four parts:

- Wavelength range 5000-5100 Å, KELT-9
- Wavelength range 5100-5200 Å , KELT-9
- Wavelength range 5000-5100 Å, HD 235349
- Wavelength range 5100-5200 Å, HD 235349

Since the set takes days to run, Linux *screen* command was used to keep a session running in a hidden way. Four files were created after each experiment:

- Walkers
- Corner plot
- Spectra
- Text file

The following Table 6 summarizes the time consumption of all four sets:

Table 6. Time consumption of the sets of experiments. Real time refers to the duration between the beginning and end of a call, while user time pertains to the quantity of CPU time utilized by the user-mode code in a given process.

Set	Real time	User time
1	3649m31.153s	23430m0.565s
2	3482m19.318s	22760m26.036s
3	1957m15.235s	21173m10.107s
4	1885m1.152s	21549m52.303s

The set of tests retrieved 64 results, each one of them consisting of the four aforementioned parts (see appendix for more details). One example is provided below. It represents the analysis of the spectrum of KELT-9 in the wavelength range 5000-5100 Å, fixed continuum normalization parameters, larger epsilon vector, 50 walkers and 2000 steps.

One result image was plot of walkers (illustrated in Fig. 10). On this plot, positions of all walkers can be seen in all the steps.

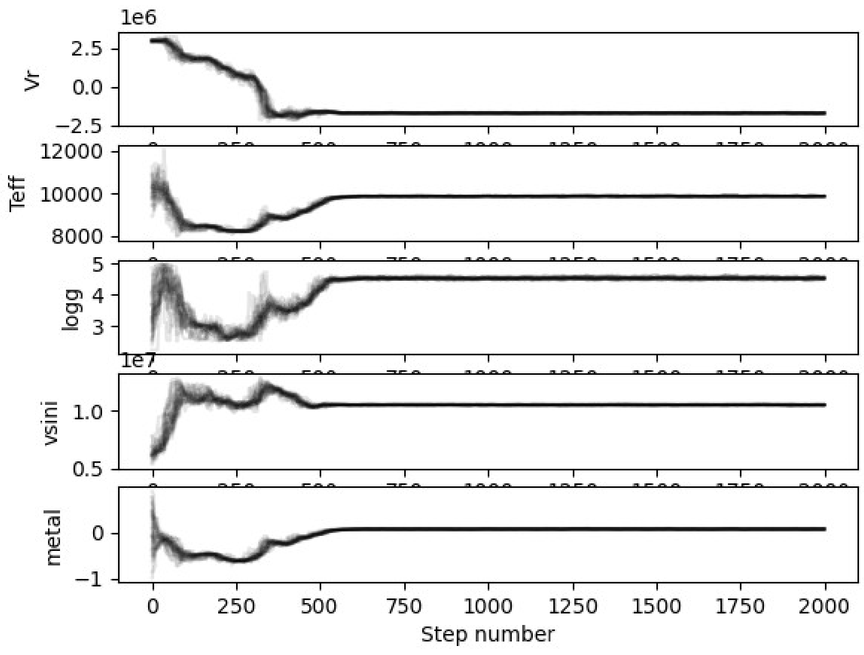


Figure 10: Plot of walkers

Another retrieved result was a corner plot (illustrated in Fig. 11). It consists of two parts:

- Histograms on the diagonal where each bar represents number of samples in the corresponding range of values. The burn-in phase is excluded from the samples.
- Two-dimensional probability distributions of all possible pairs of the parameters.

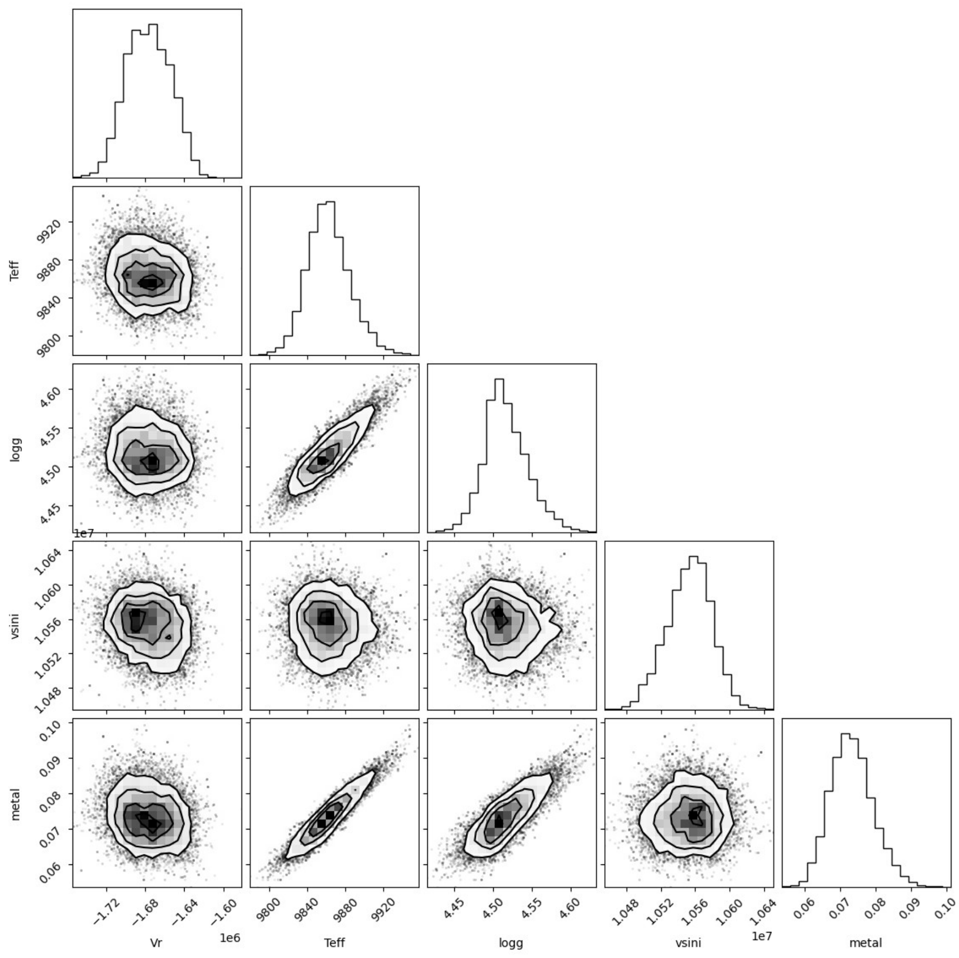


Figure 11: Corner plot

The spectrum plot (illustrated in Fig. 12) shows the observation, a model spectra at the final 'best' parameters and that model spectrum interpolated onto the observed wavelengths. In case of good result, they should be closely matching each other.

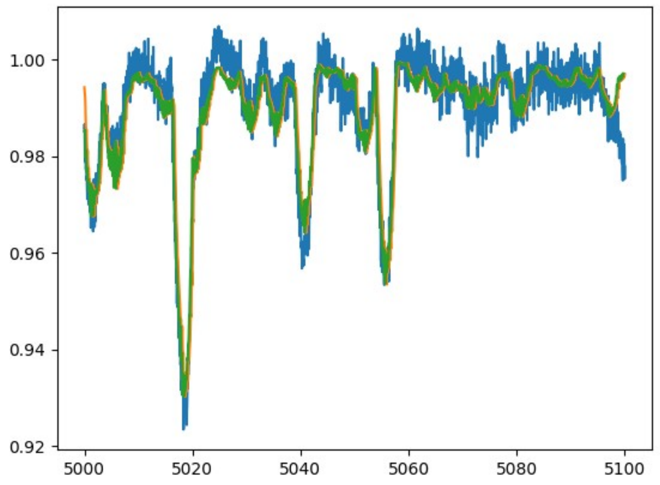


Figure 12: Spectrum plot

In addition to the three plots, a text file was produced. It consists of the following parts:

- chi2 and reduced chi2 value for a model with the final parameters
- Values of all the estimated parameters with error bars, based on the median (50th percentile), 16th and 84th percentiles of the distributions of parameters.
- Acceptance fraction and autocorrelation time estimate
- Real time of running the experiment in seconds

```
chi2 10306.74010862799
reduced chi2 3.222870578057533
Vr     -1.67688e+06 +23999.3 -22528.4
Teff     9860.75 +23.4454 -20.2498
logg     4.51339 +0.0304182 -0.0226443
vsini     1.05543e+07 +23919.0 -28353.1
metal     0.0729812 +0.00632129 -0.0056008
acceptance fraction 0.5005
autocorrelation time estimate [211.72777482
216.13258954 223.4097405   41.9663503
229.38502929]
real: 24171.165317058563
```

Figure 13. Text file

4.3 Determining the correlation between the parameters

Four correlation matrices were produced using *autocorr.py*, both visual and numerical. For both stars the versions with fixed and free continuum normalization were produced. They were calculated from *chain.dat* file after discarding the burn-in. One example is displayed below.

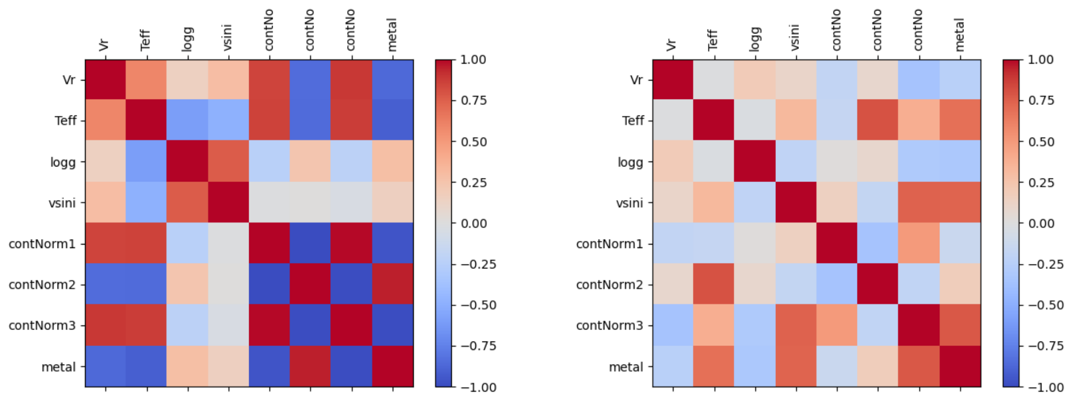


Figure 14: Correlation matrices between physical parameters for KELT-9 (left) and HD 235349 (right) if the continuum normalization is free

On this plot of the correlation matrix, red colors indicate strong positive correlation, blue colors show strong negative correlation and very light colors indicate weak or non-existent correlation.

4.4 Chi2 method based experiment

The pure Zeeman code without the wrap-up was run with both datasets, for a comparison with the results of MCMC experiments. Zeeman is fitting a synthetic spectrum to an observation using a Levenberg–Marquardt chi2 minimization routine, to determine stellar parameters. This was done using the same free parameters as in the MCMC experiments.

There were four experiments:

- Wavelength range 5000-5100 Å, KELT-9
- Wavelength range 5100-5200 Å, KELT-9

- Wavelength range 5000-5100 Å, HD 235349
- Wavelength range 5100-5200 Å, HD 235349

Each experiment retrieved results including the reduced chi2 value (see Table 7) and parameter estimates with confidence intervals.

Table 7. Reduced chi2 values of the four experiments based on the chi2 method

Star	**Wavelength range (Å)**	**Reduced chi2**
KELT-9	5000-5100	3.1025
KELT-9	5100-5200	2.9269
HD 235349	5000-5100	0.9343
HD 235349	5100-5200	0.5209

After each experiment, *plotSpectra.py* was run to produce the visuals. One example is provided below in Fig. 15.

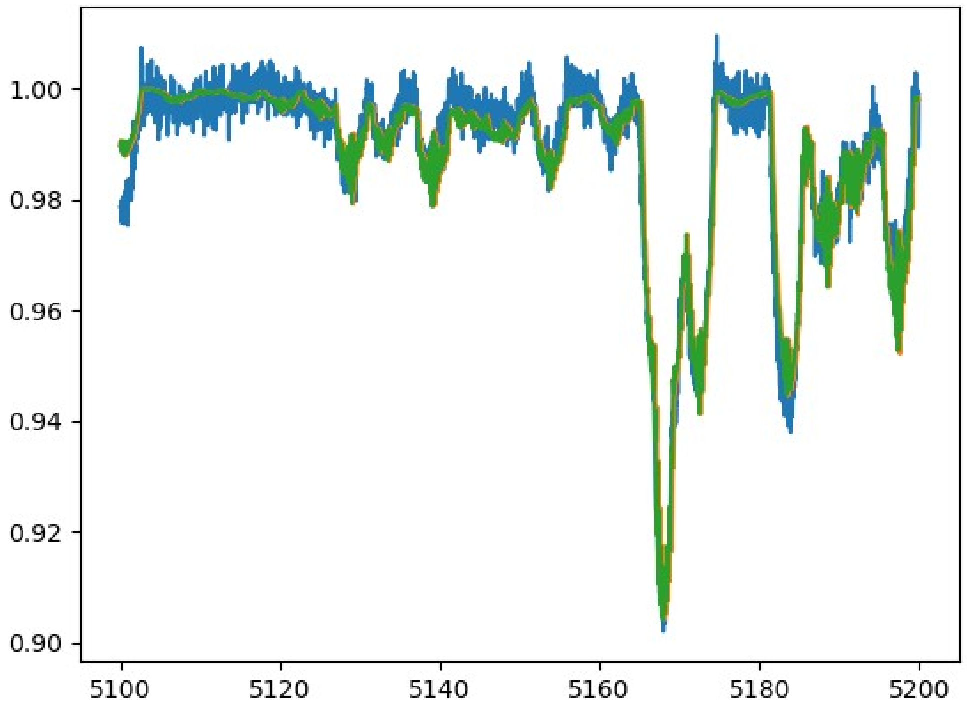


Figure 15: Spectrum of KELT-9 from 5100-5200 Å

4.5 Profiling

In addition to more general testing, the code also needed profiling. This is a process where the time consumption of all the subroutines is determined. It was done using Gprof profiling tool and the call graph was visualized using Gprof2dot.

First the table of the time consumption of the functions was generated using the following commands:

```
gfortran -o testProgram -pg -O3 lmau-zuc0.9.5.2-dil.f zuc-0.9.7.10-sub.f hline-extras-0.9.7.6n.f multi-magff-0.9.7-dil.f rewriterU0.9.7-dil.f

./testProgram

gprof testProgram
```

In the first command *-o* denotes file name of the output, *-pg* enables Gprof profiling and *-O3* turns on level 3 optimization.

The first result of the profiling was a flat profile that shows the total amount of time that the program spent executing every function. If transformed from text to table, the flat profile looks as follows in Table 8:

Table 8. Extract of flat profile of Zeeman code (full profile available in Appendix 4)

% time	**cumulative seconds**	**self seconds**	**calls**	**self ms/call**	**total ms/call**	**name**
75.86	0.22	0.22	8	27.50	27.50	linpro_
13.79	0.26	0.04	1	40.0	280.0	dskint_
3.45	0.27	0.01	4	2.50	2.50	spprof_
3.45	0.28	0.01	1	10.00	10.00	correctgf_
3.45	0.29	0.01	1	10.00	10.00	voigt_

The profiler also outputs explanations:

- % time - the percentage of the total running time of the program used by this function.

- Cumulative seconds - a running sum of the number of seconds accounted for by this function and those listed above it.
- Self seconds - the number of seconds accounted for by this function alone. This is the major sort for this listing.
- Calls - the number of times this function was invoked, if this function is profiled, else blank.
- Self ms/call - the average number of milliseconds spent in this function per call, if this function is profiled, else blank.
- Total ms/call - the average number of milliseconds spent in this function and its descendents per call, if this function is profiled, else blank.
- Name - the name of the function. This is the minor sort for this listing. The index shows the location of the function in the gprof listing. If the index is in parenthesis it shows where it would appear in the gprof listing if it were to be printed.

Another result of profiling was the graph profile that displays procedure subroutines using a call-tree format (Table 9). The function line, which corresponds to a procedure in the call-tree, is denoted by an index number enclosed in square brackets on the leftmost column. The lines above it are the parent lines, while the lines below it represent the descendant lines. The columns *self* and *children* are presented in seconds.

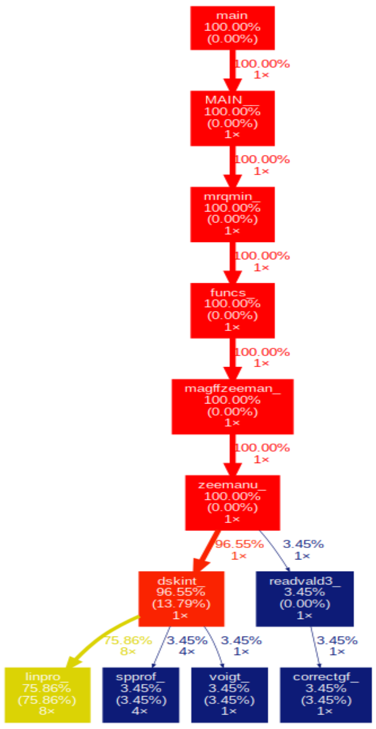
Table 9. Extract of graph profile of Zeeman code (full profile available in Appendix 4)

index	**% time**	**self**	**children**	**called**	**name**
[7]		0.04	0.24	1/1	zeemanu_ [5]
	96.6	0.04	0.24	1	dskint_ [7]
		0.22	0.00	8/8	linpro_ [8]
		0.01	0.00	4/4	spprof_ [9]
		0.01	0.00	1/1	voigt_ [12]
		0.00	0.00	8/8	magfld_ [18]

		0.00	0.00	8/8	abzsp_ [17]

In addition to general profiling, the call graph was also visualized using Gprof2dot with the following command:

gprof testProgram gmon.out | gprof2dot | dot -Tpng -o output.png



main 100.00% (0.00%)
100.00% 1×
MAIN__ 100.00% (0.00%) 1×
100.00% 1×
mrqmin_ 100.00% (0.00%) 1×
100.00% 1×
funcs_ 100.00% (0.00%) 1×
100.00% 1×
magffzeeman_ 100.00% (0.00%) 1×
100.00% 1×
zeemanu_ 100.00% (0.00%) 1×
96.55% 1×
3.45% 1×
dskint_ 96.55% (13.79%) 1×
readvald3_ 3.45% (0.00%) 1×
75.86% 8×
3.45% 4×
3.45% 1×
3.45% 1×
linpro_ 75.86% (75.86%) 8×
spprof_ 3.45% (3.45%) 4×
voigt_ 3.45% (3.45%) 1×
correctgf_ 3.45% (3.45%) 1×

Figure 16: Call graph of Zeeman code

The call graph (Fig. 16) shows how all the functions call each other and how big percentage of time is spent in each one of them.

5 Discussion

In the previous chapter, the process to retrieve results with different parameter combinations and profiling results was described. This chapter analyzes them using different quality metrics.

5.1 Reduced chi2 and time consumption as quality metrics

Reduced chi2 should be as close as possible to one to indicate a good fit. It would be user-friendly to consume as little time as possible. This subsection presents values of these two quality metrics for each experiment.

Table 10. Reduced chi2 values and time in seconds for KELT-9 if continuum normalization is fixed. Wavelength range (Å) and epsilon coefficient (rows) / number of walkers and their length (columns). Winning model with the best hyperparameters is marked with green.

	50, 500	**100, 500**	**50, 2000**	**100, 2000**
5000-5100, 1.0	3.2498 / 6022	4.3492 / 6015	3.2229 / 24396	3.2229 / 24292
5000-5100, 5.0	3.2417 / 5844	3.3140 / 5825	3.2229 / 24171	3.2229 / 24284
5100-5200, 1.0	3.7560 / 5734	2.7487 / 5914	2.7448 / 23481	2.7448 / 23494
5100-5200, 5.0	2.7860 / 5525	6.0297 / 5641	2.7449 / 23480	2.7448 / 23527

For KELT-9, a fixed continuum normalization produced a reliable and stable fit for 2000-step chains (Table 10). The table shows that 500-step chains produced more variation in chi2 values, and typically larger values, which makes them less reliable. 500-step experiments took on average 5815 seconds and 2000-step experiments 23891 seconds. It can be concluded, that it is worth to spend approximately four times more time to get better and more reliable fit.

Table 11. Reduced chi2 values for KELT-9 if continuum normalization is free. Wavelength range (Å) and epsilon coefficient / number of walkers and their length. Winning model with the best hyperparameters is marked with green.

	50, 500	**100, 500**	**50, 2000**	**100, 2000**

5000-5100, 1.0	18.5836 / 5753	13.6946 / 5624	17.2043 / 20209	14.7471 / 22168
5000-5100, 5.0	18.5882 / 5098	16.4935 / 5529	9.5754 / 21654	12.4732 / 21274
5100-5200, 1.0	28.2609 / 5381	24.5012 / 5737	18.7883 / 20518	17.5598 / 20078
5100-5200, 5.0	23.6422 / 4920	24.7596 / 5144	8.0799 / 21974	24.9132 / 19934

Setting continuum normalization free had negative impact on the results (Table 11). In none of the cases was the reduced chi2 close to one, the closest being 8.0799. Surprisingly, the experiments took slightly less time than the ones with fixed continuum normalization. An average time consumption of 500-step chains was 5398 seconds and the same metric for 2000-step chains was 20976.

Table 12. Reduced chi2 values and time in seconds for HD 235349 if continuum normalization is fixed. Wavelength range (Å) and epsilon coefficient / number of walkers and their length. Winning model with the best hyperparameters is marked with green.

	50, 500	**100, 500**	**50, 2000**	**100, 2000**
5000-5100, 1.0	0.8992 / 3182	0.8994 / 3112	0.8992 / 12728	0.8991 / 12623
5000-5100, 5.0	0.8992 / 3173	0.8991 / 3169	0.8991 / 12692	0.8991 / 12770
5100-5200, 1.0	0.5221 / 2933	0.5221 / 2886	0.5225 / 11664	0.5223 / 11938
5100-5200, 5.0	0.5228 / 3056	0.5226 / 2989	0.5223 / 12118	0.5222 / 12061

The dataset of HD 235349 produced very different results from KELT-9 (Table 12). If continuum normalization was fixed, all reduced chi2 values were less than one. As expected, low resolution reduced the time consumption: running 500-step chain took on average 3063 seconds and 2000-step took on average 12324.

Table 13. Reduced chi2 values for HD 235349 if continuum normalization is free. Wavelength range (Å) and epsilon coefficient / number of walkers and their length. Winning model with the best hyperparameters is marked with green.

	50, 500	**100, 500**	**50, 2000**	**100, 2000**
5000-5100, 1.0	1.6240 / 2968	1.4159 / 3027	0.7198 / 12050	0.7233 / 10149
5000-5100, 5.0	1.4560 / 2747	1.4336 / 2770	1.1672 / 9872	0.7228 / 10377
5100-5200, 1.0	0.7043 / 2894	0.7726 / 2824	0.4894 / 11566	0.4900 / 11354
5100-5200, 5.0	0.7450 / 2587	0.7966 / 2627	0.4905 / 9949	0.4897 / 9575

Setting continuum normalization free caused big differences between the two studied wavelength ranges (Table 13). The reduced chi2 values for 5100-5200 were always less than one, ranging from 0.4894 to 0.7966, but for 5000-5100 some of them were more than one, ranging from 0.7198 to 1.6240. The 500-step experiments took on average 2806 seconds to run and their 2000-step peers 10612 seconds.

The longer chains with free continuum sometimes reach a smaller reduced chi2 than their counterparts with a fixed continuum. But since the reduced chi2 is less than 1, that suggests these case may be over-fitting, and some of the variance of the reduced chi2 with the epsilon coefficient suggests the results with a free continuum may still be unstable.

Epsilon coefficient 5.0 made experiment take slightly less time than the default coefficient 1.0.

5.2 Proximity to literature values and uncertainties as quality metrics

In this subsection, the results of the experiments will be compared to the ones from previous literature based on [5] and [7]. The following tables compare these literature values with the results produced by the best models.

Table 14. Comparison of literature values of physical parameters of KELT-9 [5] and the values produced in this work of the winning model

Parameter	**Literature value**	**Literature error**	**Thesis value**	**Thesis error positive**	**Thesis error negative**
Teff	9495	104	9440	647	447
logg	4.17	0.17	4.60	0.37	0.90
vsini	114.9	3.4	112.19	1.93	1.73
Vr	-11.71	2.19	-13.21	3.93	1.95
metal	0.07	0.14	-0.09	0.27	0.13

It can be seen (Table 14) that the estimated parameter values are roughly consistent with the literature values. Only for surface gravity is the difference larger than the joint error bar, and even then the difference is less than two times the joint error. Problematically, effective temperature has 5.26 times larger errorbar from the MCMC analysis. In other words, this estimate is much more uncertain.

Table 15. Comparison of literature values of physical parameters of HD 235349 [7] and the values produced in this work

Parameter	**Literature value**	**Literature error**	**Thesis value**	**Thesis error positive**	**Thesis error negative**
Teff	14189	492	14233	268	326
logg	3.43	0.21	4.34	0.33	0.39
vsini	64.8	7.1	69.1	2.1	2.0
Vr	-0.14	8.58	0.16	0.13	0.13
metal	0.03	0.21	-0.15	0.05	0.06

The estimates for HD 235349 (Table 15) were quite consistent with [7] except for the surface gravity log *g*. Most errorbars were smaller than in the literature [7].

5.3 Autocorrelation time and acceptance fraction

The authors of *emcee* recommend to take into account acceptance fraction and the autocorrelation time estimate. They are analyzed below.

The lowest acceptance fraction was 0.2132 (KELT-9, 5000-5100, 5.0, 50, 500) and the highest 0.5410 (HD 235349, 5000-5100, 5.0, 100, 2000). This means that we can be satisfied with acceptance fraction of all experiments because it should be approximately from 0.2 to 0.5.

The autocorrelation time estimate is heavily influenced by the length of the chain. Below, the autocorrelation time estimate lists are presented for the winning models (marked with green in Tables 10-13). For each winning model autocorrelation times are also given for the 500-step analogue, and calculated the ratio between the 2000-step version and 500-step version is also calculated.

The parameters are presented in the following order: V_r, T_{eff}, log *g*, *v* sin *i*, *metal*. In case of free continuum normalization, the three parameters before the metal are *contnorm*.

The following list represents autocorrelation time for each of the free parameters of the winning model of KELT-9 with fixed continuum normalization (5100-5200, 1.0, 100 walkers, 2000 steps):

[173.35082135 33.35579734 121.98309726 134.39310998 50.0858676].

The following list is its 500-step alternative:

[68.5460505 36.81132621 43.1641339 36.09850946 38.49189672].

As it can be seen the in the longer chain version the estimates are the following times larger, the estimate cannot be trusted if the chain is only a factor of ten longer than the estimate itself:

[2.5267065, 0.9075969, 2.8274240, 3.7162809, 1.2989589].

Analogously, the following lists represent autocorrelation time for each of the free parameters of the winning model of KELT-9 with free continuum normalization (5100-5200, 5.0, 50 walkers, 2000 steps), its 500-step counterpart and the ratio between them:

[264.96362699 108.23807284 151.67406278 246.77036207 120.59512537 70.13791418 38.76992107 174.00586809]

[66.33547072 41.02577161 63.70516036 47.92537701 37.23602326 37.37296144 28.61336508 43.45130393]

[3.99671768 2.63564156 2.38499445 5.14610961 3.23344111 1.87644381 1.35503868 3.99600278]

The following lists represent autocorrelation time for each of the free parameters of the winning model of HD 235349 with fixed continuum normalization (5000-5100, 1.0, 100 walkers, 2000 steps), its 500-step counterpart and the ratio between them:

[171.68121849 65.53973087 51.87444833 73.82820409 65.07212845]

[65.2832794 30.03621738 41.52675992 56.75039525 27.15718095]

[2.62877391 2.1804173 1.24984705 1.30138732 2.39492474]

The following lists represent autocorrelation time for each of the free parameters of the winning model of HD 235349 with free continuum normalization (5000-5100, 1.0, 100 walkers, 2000 steps), its 500-step counterpart and the ratio between them:

[261.78635986 215.7502878 101.22448616 186.97573076 115.45336771 82.3972775 46.10974467 193.15826037]

[45.69057367 56.4375322 40.94100803 60.74025464 42.35353522 28.78095773 23.45365017 43.32959613]

[5.72782646 3.82491338 2.46829535 3.0789329 2.72760962 2.86402719 1.97221984 4.45371431]

If the length of the chain is increased by a factor of four, the autocorrelation time estimate increased usually two or three times. It can be implied that the autocorrelation times of too short chains are unreliable.

5.4 Time consumption and parallelization

As expected, MCMC methods were time consuming. Table 16 presents the speedup of the program computed as a relationship between the user and real time.

Table 16. Speedup computed as the relationship between real and user time (in seconds)

Set	**Real time**	**User time**	**Speedup**
1	1405801	218971	6.42
2	1365626	208939	6.54
3	1270390	117435	10.82
4	1292992	113101	11.43
Combined	5334809	658447	8.10

As mentioned in 2.2.2, Amdahl’s law connects the speedup $S=8.10$, number of processors (in this work equal to the number of vCPUs) $P=48$ and sequential fraction of the program F :

$$S = \frac{1}{F + \frac{1-F}{P}} \quad .$$

According to Amdahl’s law the sequential fraction of the program is $F = 0.105$, meaning that the parallel fraction is 0.895.

The efficiency of the project would benefit the most if *linpro_* subroutine would be optimized in Zeeman. According to the profiling call graph, it takes 75.89% of the time.

5.5 Comparison with chi-square method

This subsection compares the quality of the chi2-method based pure Zeeman code and the chi2-method based Python wrap-up code. The main quality metric is the reduced chi2 value.

Table 17. Reduced chi2 values of the four experiments. Comparison between average and best of MCMC experiments with 2000 steps and chi2-based experiments

Star	**Wavelength range**	**MCMC (average)**	**MCMC (best)**	**chi2**
KELT-9	5000-5100	8.3614	3.3140	3.1025
KELT-9	5100-5200	10.0401	2.7448	2.9269
HD 235349	5000-5100	0.8662	0.7233	0.9343
HD 235349	5100-5200	0.5061	0.5225	0.5209

The results suggest that chi2 method slightly outperforms the MCMC-based method if the wavelength range is chosen to be 5000-5100 but if the wavelength range is 5100-5200 it is the other way around. Paired sample T-test suggests that the difference in quality between the winning MCMC models and the chi2-based fittings are statistically insignificant if we use an alpha-value of 0.05 since the p-value computed by *pairedTTest.py* is much larger: 0.68.

As expected the reduced chi2 values for MCMC are similar to the ones of the chi2 minimization method. It confirms that MCMC also converges well. Sadly, MCMC methods are far more time consuming compared to chi2-based method that takes less than a minute to run.

5.6 Correlation between the stellar parameters

Correlation matrices for both studied stars were produced (both fixed (Fig. 17) and free (Fig. 14) continuum normalization. Some correlations between the parameters were similar for both stars, and others were different. In case of fixed continuum normalization, the most strongly correlated physical parameters for KELT-9 were Teff and *v* sin *i*, with correlation of -0.828046, and for HD 235349 T_{eff} and log *g* had correlation of 0.806103.

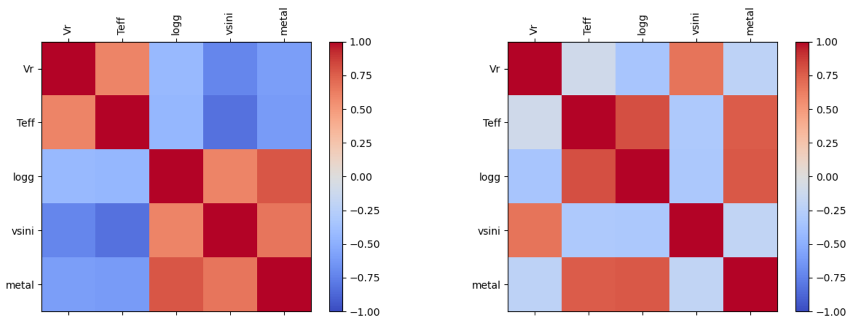


Figure 17: Correlation matrices between physical parameters for KELT-9 (left) and HD 235349 (right) if the continuum normalization is fixed

If continuum normalization was fixed, V_r and *v* sin *i* were relatively strongly correlated for both stars but in opposite directions: for KELT-9 the correlation was -0.724703, and for HD 235349 0.668147. Similarly, T_{eff} had opposite correlations with log *g* and metal: -0.451322 and -0.609801 for KELT-9 but for HD 235349 the same values were respectively 0.806103 and 0.760774.

On the other hand, some correlations were similar for both stars. The correlation between V_r and log *g* was -0.433320 for KELT-9 and -0.336929 for HD 235349. The correlation between log *g* and metal was even more similar: 0.785568 and 0.776555, respectively.

If continuum normalization was used as a free parameter, the three parts of continuum normalization were not strongly correlated with each other, nor with other parameters for HD 235349. It was very different for KELT-9, where the parts of continuum normalization were correlated with V_r and T_{eff} with an absolute value of around 0.85, and with each other with an absolute value of around 0.99.

5.7 Visual comparison of convergence

As described in the previous chapter, in addition to text files, also visual plots were produced.

It can be clearly seen (in Fig. 18) that 500-step experiments did not produce reliable results:

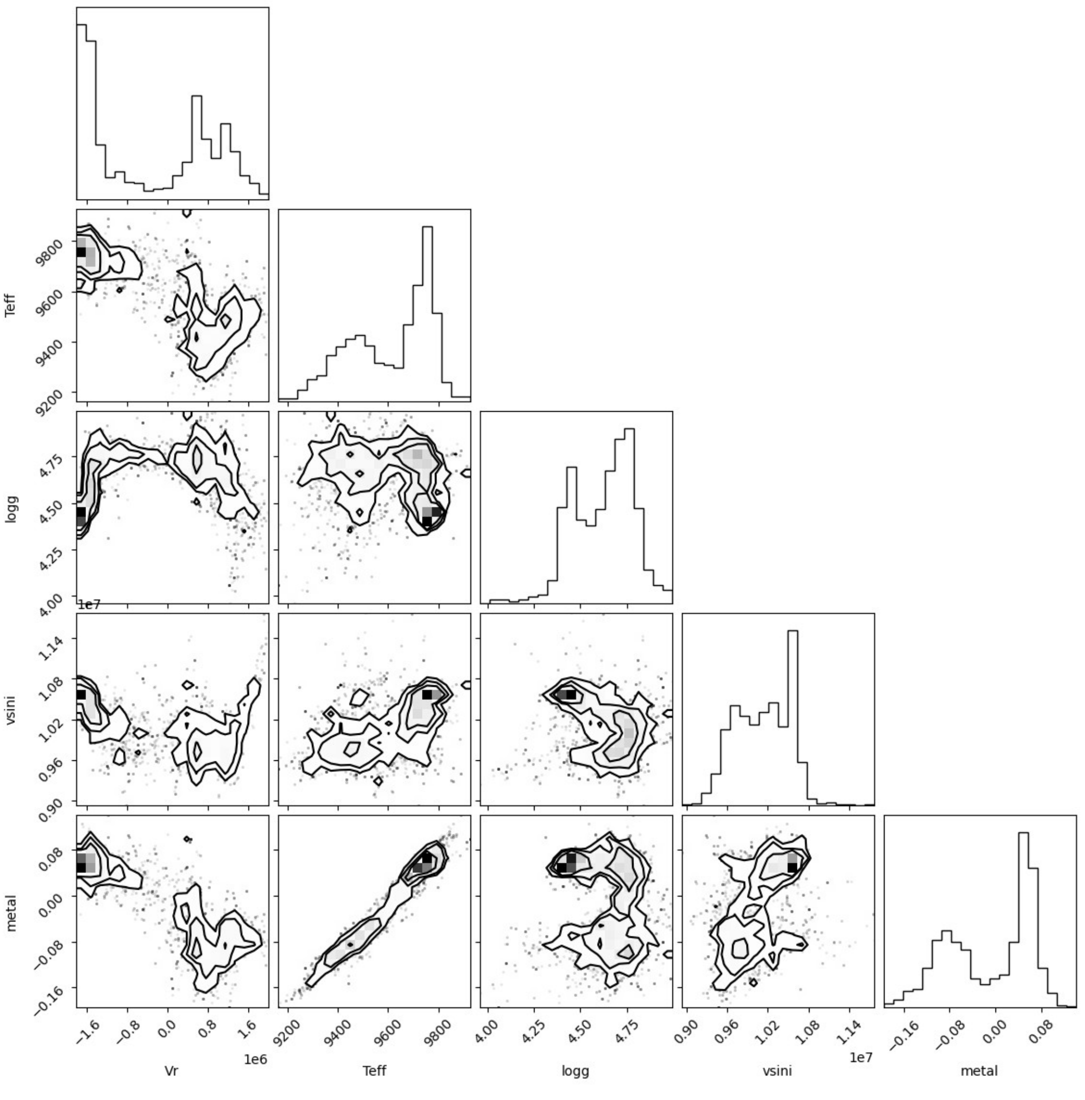


Figure 18: Corner plot of the experiment for KELT-9 with wavelength range 5000-5100 Å, fixed continuum normalization, 1.0 as epsilon coefficient,100 walkers and 500 steps

On the other hand, the 2000-step analogue of the previous example, presented in Fig. 19, converged well:

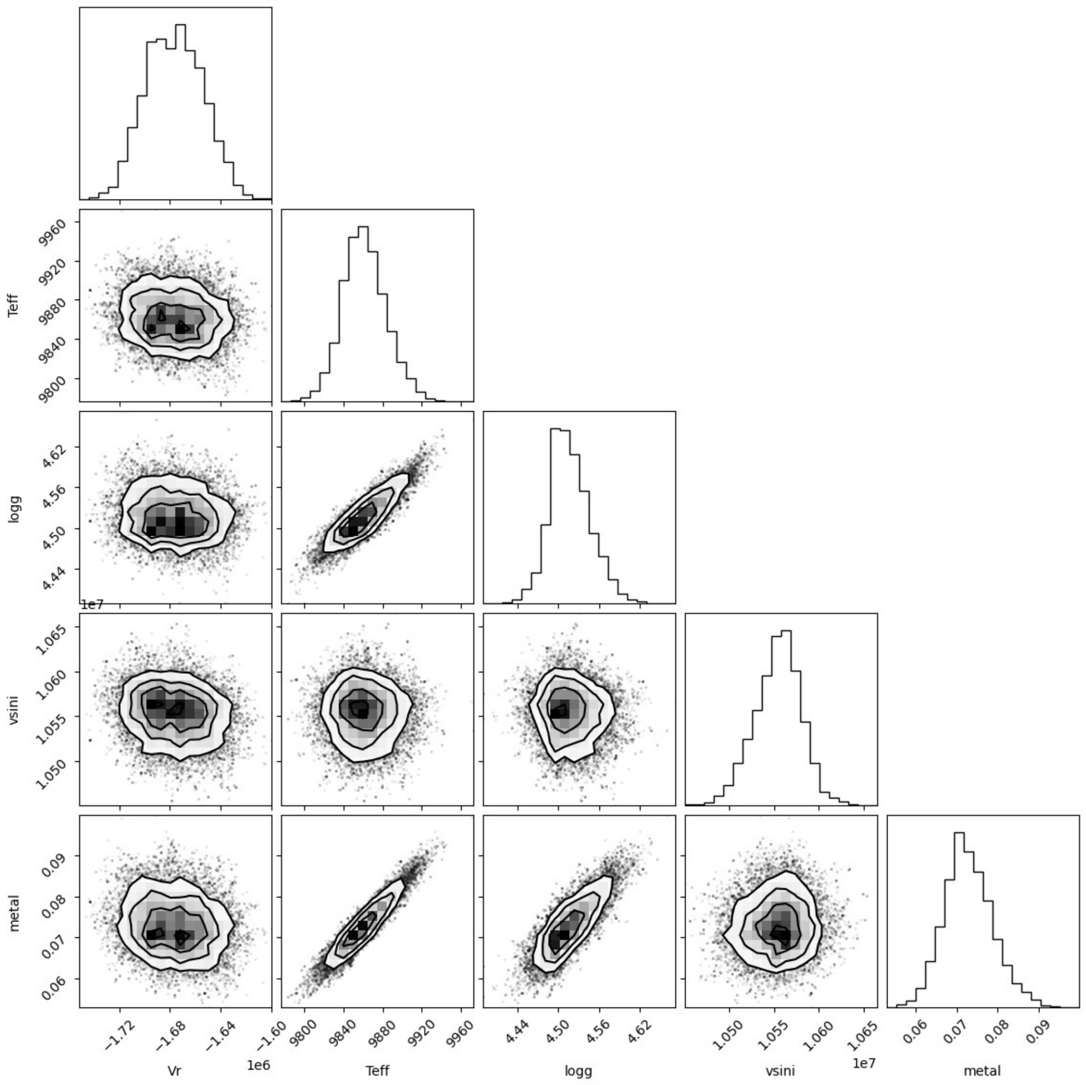


Figure 19: Corner plot of experiment with wavelength range 5000-5100 Å, fixed continuum normalization, 1.0 as epsilon coefficient, 100 walkers and 2000 steps

It can be also seen on the plot of walkers (Fig. 20) that it takes nearly 500 steps to converge and 500 steps is not long enough length for a chain.

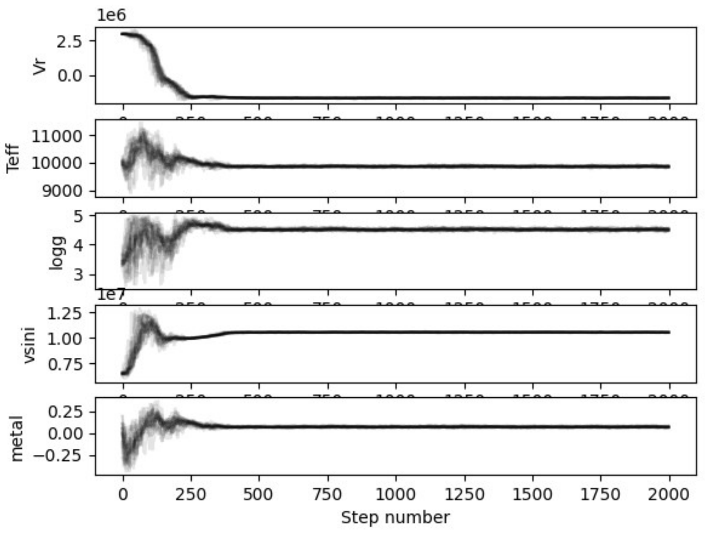


Figure 20: Walkers of experiment with wavelength range 5000-5100 Å, fixed continuum normalization, 1.0 as epsilon coefficient, 100 walkers and 2000 steps

It can be also seen in Fig. 21 that if the wavelength range is set to 5100-5200 Å, the quality of the result suffers because it is a multimodal distribution:

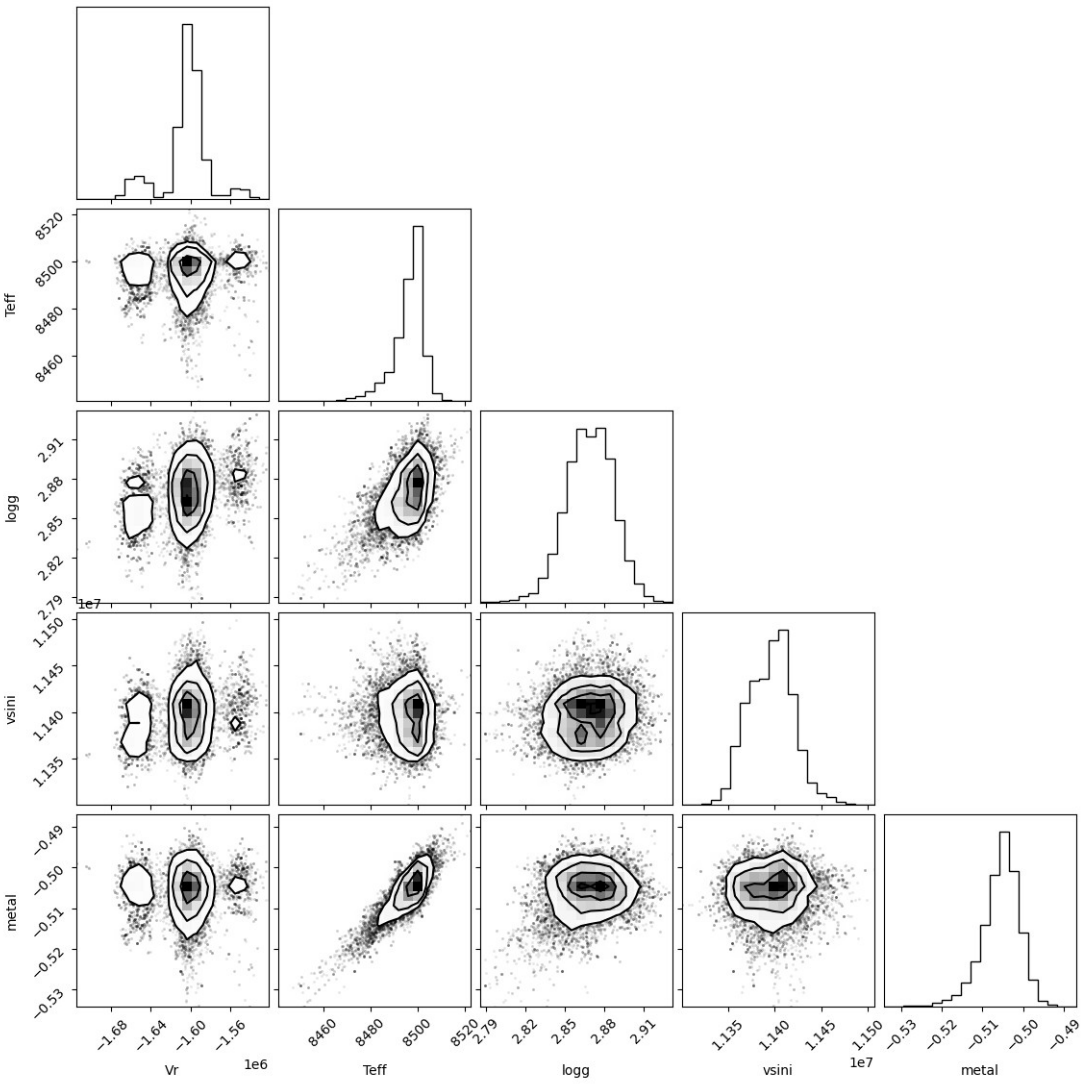


Figure 21: Corner plot of experiment with wavelength range 5100-5200 Å, fixed continuum normalization, 1.0 as epsilon coefficient,100 walkers and 2000 steps

While MCMC methods have some ability to handle multimodal distributions, this may slow convergence, and will likely lead to larger uncertainties than for the 5000-5100 Å case.

5.8 Comparison of the datasets

Above, multiple different quality metrics were used to analyze the results based on both datasets. This subsection summarizes the differences between the datasets based on the results and quality metrics.

On the one hand, the datasets of the two stars were quite different. The different instrument and lower signal-to-noise ratio for HD 235349 has probably caused the surprisingly small reduced chi2 value. If the fit to the observation is limited only by noise, as in this lower signal-to-noise case, then an overestimate in the errorbars will cause the reduced chi2 value to be small. In the observation of HD 235349 the scatter in values between adjacent pixels (in the continuum regions) is often less than the error bar, which supports the idea that these errors are overestimated.

On the other hand, the datasets had also significant similarities. Both of them produced results with similar error bars, as described in 5.2. Both of them gave physical parameter values that are consistent with result from the literature. In both cases, it was important to run long chains and keep the continuum normalization fixed.

6 Summary

In this thesis, integration of MCMC methods with the stellar spectrum synthesis code Zeeman was tested. The quality of MCMC methods was compared to results from chi2 minimization method. In addition to producing MCMC and chi2 results, the Zeeman code was profiled.

It can be concluded that the chains should be thousands of steps long to produce a reliable result. A high resolution and a high signal-to-noise ratio in the observed spectrum is also crucial to produce a good fit.

The best model for KELT-9 turned out to be the one with the following hyperparameters: wavelength range 5100-5200 Å, fixed continuum normalization, 50 walkers, 2000 steps and 1.0 epsilon coefficient; and for HD 235349 the following hyperparameters: wavelength range 5000-5100 Å, fixed continuum normalization, 100 walkers, 2000 steps and 1.0 epsilon coefficient. Continuum normalization did not perform well as a free parameter in this work and produced unstable results.

The results produced by chi2 minimization method and the ones produced by the corresponding winning models of MCMC were similar. According to the reduced chi2 values chi2 method slightly outperformed the MCMC-based method if the wavelength range wass chosen to be 5000-5100 Å but if the wavelength range was 5100-5200 Å it was the other way around.

The relationship between real and user time, and a computation based on Amdahl's law, shows that the project is well parallelized. According to the Gprof call graph the most time-consuming subroutine in Zeeman program code is *linpro_*, which evaluates the radiative transfer equation, and that would be useful to optimize in the future works.

In conclusion, the project fundamentally worked, but it needs sufficiently long chains to converge completely and reduce the error bars. In future projects, the program code could benefit from additional automation. For example, the experiment could be run in more stages. After the first stage, some parameters could become fixed based on their estimates from the first stage. Also, it would be good to experiment with some other wavelength ranges both with and without free continuum normalization.

This work taught me to connect knowledge from different branches of science. Stellar spectroscopy was a good example to get started with MCMC methods.

References

[1] Hu J, Qu X, 2020. Bayes' Theorem under Conditional Independence. *arXiv preprint* arXiv:2003.03970. 2020 Mar 9.

[2] Steinberg U, Kauer B, 2010. Towards a scalable multiprocessor userlevel environment. *In Workshop on Isolation and Integration for Dependable Systems*.

[3] Foreman-Mackey, D., Hogg, D.W., Lang, D. and Goodman, J., 2013. emcee: the MCMC hammer. *Publications of the Astronomical Society of the Pacific*, *125*(925), p.306.

[4] Wade, G.A., Bagnulo, S., Kochukhov, O., Landstreet, J.D., Piskunov, N. and Stift, M.J., 2001. LTE spectrum synthesis in magnetic stellar atmospheres-The interagreement of three independent polarised radiative transfer codes. *Astronomy & Astrophysics*, *374*(1), pp.265-279.

[5] Kama, M., Folsom, C.P., Jermyn, A.S. and Teske, J.K., 2023. KELT-9 and its ultra-hot Jupiter: Stellar parameters, composition, and planetary pollution. *Monthly Notices of the Royal Astronomical Society*, *518*(2), pp.3116-3122.

[6] Kama, M., Folsom, C.P. and Pinilla, P., 2015. Fingerprints of giant planets in the photospheres of Herbig stars. *Astronomy & Astrophysics*, *582*, p.L10.

[7] Folsom, C.P., Kama, M., Eenmäe, T., Kolka, I., Aret, A., Checha, V., Kasikov, A., Leedjärv, L. and Ramler, H., 2022. A rare phosphorus-rich star in an eclipsing binary from TESS. *Astronomy & Astrophysics*, *658*, p.A105.

[8] Jonathan Goodman. Jonathan Weare, 2010. Ensemble samplers with affine invariance. *Commun. Appl. Math. Comput*. Sci. 5 (1) 65 - 80. https://doi.org/10.2140/camcos.2010.5.65

[9] Althaus, L.G., Córsico, A.H., Isern, J. and García-Berro, E., 2010. Evolutionary and pulsational properties of white dwarf stars. *The Astronomy and Astrophysics Review*, *18*, pp.471-566.

[10] Carroll, B.W. , Ostlie, D.A., 2007. *An Introduction to Modern Astrophysics*. Pearson Addison-Wesley.

[11] Giridhar, S., 2010. Spectral Classification: Old and Contemporary. In *Principles and Perspectives in Cosmochemistry: Lecture Notes of the Kodai School on'Synthesis of Elements in Stars' held at Kodaikanal Observatory, India, April 29-May 13, 2008* (pp. 165-180). Springer Berlin Heidelberg.

[12] Cosma, I.A. and Asgharian, M., 2008. Principle of detailed balance and convergence assessment of Markov Chain Monte Carlo methods and simulated annealing. *arXiv preprint arXiv:0807.3151*.

[13] Seyr, H. and Muskulus, M., 2019. Decision support models for operations and maintenance for offshore wind farms: a review. *Applied Sciences*, *9*(2), p.278.

[14] Tsantaki, M., Andreasen, D.T., Teixeira, G.D.C., Sousa, S.G., Santos, N.C., Delgado-Mena, E. and Bruzual, G., 2018. Atmospheric stellar parameters for large surveys using FASMA, a new spectral synthesis package. *Monthly Notices of the Royal Astronomical Society*, *473*(4), pp.5066-5097.

[15] Andrae, R., Schulze-Hartung, T. and Melchior, P., 2010. Dos and don'ts of reduced chi-squared. *arXiv preprint arXiv:1012.3754*.

[16] Katz, D., Soubiran, C., Cayrel, R., Adda, M. and Cautain, R., 1998. On-line determination of stellar atmospheric parameters Teff, log g,[Fe/H] from ELODIE echelle spectra. I-The method. *arXiv preprint astro-ph/9806232*.

[17] Hayes, C.R., Masseron, T., Sobeck, J., García-Hernández, D.A., Prieto, C.A., Beaton, R.L., Cunha, K., Hasselquist, S., Holtzman, J.A., Jönsson, H. and Majewski, S.R., 2022. BACCHUS Analysis of Weak Lines in APOGEE Spectra (BAWLAS). *The Astrophysical Journal Supplement Series*, *262*(1), p.34.

[18] ter Maten, E.J.W., Doorn, T.S., Croon, J.A., Bargagli, A., Di Bucchianico, A. and Wittich, O., 2009. Importance sampling for high speed statistical Monte-Carlo simulations. *trials*, *10*(2), p.100.

[19] Yan, F., Wyttenbach, A., Casasayas-Barris, N., Reiners, A., Pallé, E., Henning, T., Mollière, P., Czesla, S., Nortmann, L., Molaverdikhani, K. and Chen, G., 2021. Detection of the hydrogen Balmer lines in the ultra-hot Jupiter WASP-33b. *Astronomy & Astrophysics*, *645*, p.A22.

[20] MOOG https://www.as.utexas.edu/~chris/moog.html

[21] VALD http://vald.oreme.org/~vald/php/vald.php

[22] Kurucz, R., 1993. CDROM Model Distribution. *Smithsonian Astrophys. Obs*.

[23] Andrieu, C., De Freitas, N., Doucet, A. and Jordan, M.I., 2003. An introduction to MCMC for machine learning. *Machine learning*, *50*, pp.5-43.

[24] Jespersen, N.S., 2010. An introduction to markov chain monte carlo. *Available at SSRN 1594971*.

[25] Tokdar, S.T. and Kass, R.E., 2010. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(1), pp.54-60.

[26] Amdahl, G.M., 1967. Validity of the single processor approach to achieving large scale computing capabilities, *AFIPS Conference Proceedings*, vol. 30, AFIPS Press, Reston, Va., 1967, pp. 483–485.

[27] Folsom, C.P., Bagnulo, S., Wade, G.A., Alecian, E., Landstreet, J.D., Marsden, S.C. and Waite, I.A., 2012. Chemical abundances of magnetic and non-magnetic Herbig Ae/Be stars. *Monthly Notices of the Royal Astronomical Society*, *422*(3), pp.2072-2101.

[28] Landstreet, J.D., 1988. The magnetic field and abundance distribution geometry of the peculiar A star 53 Camelopardalis. *The Astrophysical Journal*, *326*, pp.967-987.

[29] Alvarez, R. and Plez, B., 1997. Near-infrared narrow-band photometry of M-giant and Mira stars: models meet observations. *arXiv preprint astro-ph/9710157*.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Directory tree

Table 18. Directory structure of the project

File or directory	Description
data	Contains multiple files of which the most important are: *zmodel.dat* – input parameters *vlines.dat* – spectral lines Contains also *ATLAS9* and *MARCS* folders with precomputed results.
out	Directory that contains *out-atmosphere.krz* - copy that code uses, interpolated values *out-model.dat* - copy of *zmodel.dat* *outzfit-u.dat* - diagnostic log for potential problems
testedResults	Directory that contains the results of *testCombinations.py*
venv	Creates and manages virtual environments that isolate and manage dependencies for the project.
lmamp	Compiled executable file, generated by the compiler
multi-magff-0.9.7-dil.f	Should make multiple calls to Zeeman, merge the resuts, return them and provide a model spectrum, with different filling factors of different magnetic fields.
hline-extras-0.9.7.6n.f	Additional subroutines for ZEEMAN; includes features for improved calculation of Hydrogen lines
rewriterU0.9.7-dil.f	Generates the *out/out-model.dat*
zuc-0.9.7.10-sub.f	Subroutine version of Zeeman that can be ran by the fitting routine
zuc-0.9.7.10.f	Stand-alone version of Zeeman that does calculations but not fitting (currently not used)
lmau-zuc0.9.5.2-dil.f	Runs a Levenberg-Marquardt fitting routine,

File or directory	**Description**
	interpolation between model atmospheres
testProgram	The Gprof profiling routine created in the terminal.
output.png	The Gprof2dot call graph that is the result of profiling.
requirements.txt	A file that contains all the Python modules that have to be installed
zemceeWrap03cont.py	The wrap-up code around Zeeman code. Uses MCMC to fit the empirical observations in *observed.dat*.
plotChain.py	Creates two plots: a plot of walkers and the corner plot. Originally displayed them but was changed to save them to the directory *testedResults*.
plotSpectra.py	Plots the observed spectrum and the two spectra generated by the most recent *zemceeWrap03cont.py* run
testCombinations.py	Runs the set of experiments
plotCorrelation.py	Plots the visual correlation matrix and prints it out in the terminal
PairedTTest.py	Runs paired T-test to estimate the statistical significance (needed in 5.5)
observed.dat	Observations of HD 235349
observed_kelt9.dat	Observations of KELT-9
zmodel.dat	Input parameters
inlmam.dat	Contains the fittable parameters, overrides *zmodel.dat* if their content happens to be in conflict
plot1	Most recent synthetic spectrum from Zeeman
results.dat	Diagnostic output from the chi2 fitting routine, final parameters after minizing chi2
outSpeci.dat	The spectrum calculated with the median values in the chain
chain.dat	All positions of the walkers of the most recent run of *zemceeWrap03cont.py*
arraysizes.mod	Temporary file by compiler

File or directory	**Description**
savespec.mod	Temporary file by compiler
Makefile	Contains the compilation instructions
plotff1	The spectrum without interpolation
plotff1i	The spectrum after double shifting and interpolating with chi2
subprocess	Side product of profiling
gmon.out	Side product of profiling

Appendix 3 – Project code and results

The project with all the Python files is available at GitHub (some other files are not included because of copyright issues):

https://github.com/jaanikaraik/jaanika-raik-master-thesis

The results of the experiments are located in the folder *testedResults*. They are distributed between four folders:

- *KELT9* – all MCMC experiments with the KELT-9 dataset
- *HD235349* – all MCMC experiments with the Hd 235349 dataset
- *chi2experiments* – all four chi2 experiments
- *correlationMatrices* – all four visual correlation matrices and their numberical values in a text file

Appendix 4 – Full profiling results

```
Flat profile
Each sample counts as 0.01 seconds.
 %   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
75.86      0.22     0.22        8    27.50    27.50  linpro_
13.79      0.26     0.04        1    40.00   280.00  dskint_
 3.45      0.27     0.01        4     2.50     2.50  spprof_
 3.45      0.28     0.01        1    10.00    10.00  correctgf_
 3.45      0.29     0.01        1    10.00    10.00  voigt_
 0.00      0.29     0.00     9728     0.00     0.00  ltelc_
 0.00      0.29     0.00      192     0.00     0.00  kappac_
 0.00      0.29     0.00      192     0.00     0.00  stancilh2p_
 0.00      0.29     0.00        9     0.00     0.00  parse_quantum3_
 0.00      0.29     0.00        8     0.00     0.00  abzsp_
 0.00      0.29     0.00        8     0.00     0.00  magfld_
 0.00      0.29     0.00        1     0.00   290.00  MAIN__
 0.00      0.29     0.00        1     0.00     0.00  compon_
 0.00      0.29     0.00        1     0.00     0.00  correctvald_
 0.00      0.29     0.00        1     0.00     0.00  covsrt_
 0.00      0.29     0.00        1     0.00   290.00  funcs_
 0.00      0.29     0.00        1     0.00     0.00  gaussj_
 0.00      0.29     0.00        1     0.00     0.00
magffinterppassed_
 0.00      0.29     0.00        1     0.00   290.00  magffzeeman_
 0.00      0.29     0.00        1     0.00     0.00  modelatmo_
 0.00      0.29     0.00        1     0.00   290.00  mrqmin_
 0.00      0.29     0.00        1     0.00    10.00  readvald3_
 0.00      0.29     0.00        1     0.00     0.00  rewriteru_
 0.00      0.29     0.00        1     0.00   290.00  zeemanu_

%         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
seconds   for by this function and those listed above it.
```

```
self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
          listing.

calls      the number of times this function was invoked, if
          this function is profiled, else blank.

self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
          else blank.

total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
          function is profiled, else blank.

name       the name of the function.  This is the minor sort
          for this listing. The index shows the location of
          the function in the gprof listing. If the index is
          in parenthesis it shows where it would appear in
          the gprof listing if it were to be printed.


Copyright (C) 2012-2022 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.


                    Call graph (explanation follows)


granularity: each sample hit covers 4 byte(s) for 3.45% of 0.29
seconds

index % time    self  children    called     name
               0.00    0.29       1/1           main [6]
```

```
[1]    100.0    0.00    0.29       1         MAIN__ [1]
               0.00    0.29       1/1           mrqmin_ [4]
-----------------------------------------------
               0.00    0.29       1/1           mrqmin_ [4]
[2]    100.0    0.00    0.29       1         funcs_ [2]
               0.00    0.29       1/1           magffzeeman_ [3]
               0.00    0.00       1/1           modelatmo_ [24]
               0.00    0.00       1/1           rewriteru_ [25]
-----------------------------------------------
               0.00    0.29       1/1           funcs_ [2]
[3]    100.0    0.00    0.29       1         magffzeeman_ [3]
               0.00    0.29       1/1           zeemanu_ [5]
               0.00    0.00       1/1           magffinterppassed_
[23]
-----------------------------------------------
               0.00    0.29       1/1           MAIN__ [1]
[4]    100.0    0.00    0.29       1         mrqmin_ [4]
               0.00    0.29       1/1           funcs_ [2]
               0.00    0.00       1/1           gaussj_ [22]
               0.00    0.00       1/1           covsrt_ [21]
-----------------------------------------------
               0.00    0.29       1/1           magffzeeman_ [3]
[5]    100.0    0.00    0.29       1         zeemanu_ [5]
               0.04    0.24       1/1           dskint_ [7]
               0.00    0.01       1/1           readvald3_ [11]
               0.00    0.00    9728/9728        ltelc_ [13]
               0.00    0.00     192/192         kappac_ [14]
               0.00    0.00       1/1           compon_ [19]
-----------------------------------------------
                                                <spontaneous>
[6]    100.0    0.00    0.29                 main [6]
               0.00    0.29       1/1           MAIN__ [1]
-----------------------------------------------
               0.04    0.24       1/1           zeemanu_ [5]
[7]     96.6    0.04    0.24       1         dskint_ [7]
               0.22    0.00       8/8           linpro_ [8]
               0.01    0.00       4/4           spprof_ [9]
               0.01    0.00       1/1           voigt_ [12]
               0.00    0.00       8/8           magfld_ [18]
               0.00    0.00       8/8           abzsp_ [17]
-----------------------------------------------
               0.22    0.00       8/8           dskint_ [7]
[8]     75.9    0.22    0.00       8         linpro_ [8]
-----------------------------------------------
               0.01    0.00       4/4           dskint_ [7]
[9]      3.4    0.01    0.00       4         spprof_ [9]
-----------------------------------------------
               0.01    0.00       1/1           readvald3_ [11]
[10]     3.4    0.01    0.00       1         correctgf_ [10]
-----------------------------------------------
               0.00    0.01       1/1           zeemanu_ [5]
[11]     3.4    0.00    0.01       1         readvald3_ [11]
               0.01    0.00       1/1           correctgf_ [10]
               0.00    0.00       9/9           parse_quantum3_ [16]
               0.00    0.00       1/1           correctvald_ [20]
```

```
-----------------------------------------------
               0.01    0.00       1/1           dskint_ [7]
[12]     3.4    0.01    0.00       1         voigt_ [12]
-----------------------------------------------
               0.00    0.00    9728/9728        zeemanu_ [5]
[13]     0.0    0.00    0.00    9728         ltelc_ [13]
-----------------------------------------------
               0.00    0.00     192/192         zeemanu_ [5]
[14]     0.0    0.00    0.00     192         kappac_ [14]
               0.00    0.00     192/192         stancilh2p_ [15]
-----------------------------------------------
               0.00    0.00     192/192         kappac_ [14]
[15]     0.0    0.00    0.00     192         stancilh2p_ [15]
-----------------------------------------------
               0.00    0.00       9/9           readvald3_ [11]
[16]     0.0    0.00    0.00       9         parse_quantum3_ [16]
-----------------------------------------------
               0.00    0.00       8/8           dskint_ [7]
```

```
[17]     0.0    0.00    0.00       8         abzsp_ [17]
-----------------------------------------------
               0.00    0.00       8/8           dskint_ [7]
[18]     0.0    0.00    0.00       8         magfld_ [18]
-----------------------------------------------
               0.00    0.00       1/1           zeemanu_ [5]
[19]     0.0    0.00    0.00       1         compon_ [19]
-----------------------------------------------
               0.00    0.00       1/1           readvald3_ [11]
[20]     0.0    0.00    0.00       1         correctvald_ [20]
-----------------------------------------------
               0.00    0.00       1/1           mrqmin_ [4]
[21]     0.0    0.00    0.00       1         covsrt_ [21]
-----------------------------------------------
               0.00    0.00       1/1           mrqmin_ [4]
[22]     0.0    0.00    0.00       1         gaussj_ [22]
-----------------------------------------------
               0.00    0.00       1/1           magffzeeman_ [3]
```

```
[23]     0.0    0.00    0.00       1         magffinterppassed_ [23]
-----------------------------------------------
               0.00    0.00       1/1           funcs_ [2]
[24]     0.0    0.00    0.00       1         modelatmo_ [24]
-----------------------------------------------
               0.00    0.00       1/1           funcs_ [2]
[25]     0.0    0.00    0.00       1         rewriteru_ [25]
-----------------------------------------------

This table describes the call tree of the program, and was sorted by
the total amount of time spent in each function and its children.

Each entry in this table consists of several lines.  The line with the
index number at the left hand margin lists the current function.
The lines above it list the functions that called this function,
and the lines below it list the functions this one called.
This line lists:
    index      A unique number given to each element of the table.
```

```
                Index numbers are sorted numerically.
                The index number is printed next to every function name
so
                it is easier to look up where the function is in the
table.

    % time      This is the percentage of the `total' time that was
spent
                in this function and its children.  Note that due to
                different viewpoints, functions excluded by options,
etc,
                these numbers will NOT add up to 100%.

    self        This is the total amount of time spent in this
function.

    children    This is the total amount of time propagated into this
                function by its children.

    called      This is the number of times the function was called.
                If the function called itself recursively, the number
                only includes non-recursive calls, and is followed by
                a `+' and the number of recursive calls.
```

name — The name of the current function. The index number is printed after it. If the function is a member of a cycle, the cycle number is printed between the function's name and the index number.

For the function's parents, the fields have the following meanings:

self — This is the amount of time that was propagated directly from the function into this parent.

children — This is the amount of time that was propagated from the function's children into this parent.

called — This is the number of times this parent called the function `/' the total number of times the function was called. Recursive calls to the function are not

```
              included in the number after the `/'.

    name      This is the name of the parent.  The parent's index
              number is printed after it.  If the parent is a
              member of a cycle, the cycle number is printed between
              the name and the index number.
```

If the parents of the function cannot be determined, the word `<spontaneous>' is printed in the `name' field, and all the other fields are blank.

For the function's children, the fields have the following meanings:

```
    self      This is the amount of time that was propagated directly
              from the child into the function.

    children  This is the amount of time that was propagated from the
              child's children to the function.
    called      This is the number of times the function called
                this child `/' the total number of times the child
                was called.  Recursive calls by the child are not
                listed in the number after the `/'.

    name        This is the name of the child.  The child's index
                number is printed after it.  If the child is a
                member of a cycle, the cycle number is printed
                between the name and the index number.

If there are any cycles (circles) in the call graph, there is an
entry for the cycle-as-a-whole.  This entry shows who called the
cycle (as parents) and the members of the cycle (as children.)
The `+' recursive calls entry shows the number of function calls that
were internal to the cycle, and the calls entry for each member shows,
for that member, how many times it was called from other members of
the cycle.
```

```
Index by function name

  [1] MAIN__                 [22] gaussj_                 [4] mrqmin_
 [17] abzsp_                 [14] kappac_                [16] 
parse_quantum3_
 [19] compon_                 [8] linpro_                [11] 
readvald3_
 [10] correctgf_             [13] ltelc_                 [25] 
rewriteru_
 [20] correctvald_           [23] magffinterppassed_      [9] spprof_
 [21] covsrt_                 [3] magffzeeman_           [15] 
stancilh2p_
  [7] dskint_                [18] magfld_                [12] voigt_
  [2] funcs_                 [24] modelatmo_              [5] zeemanu_
```