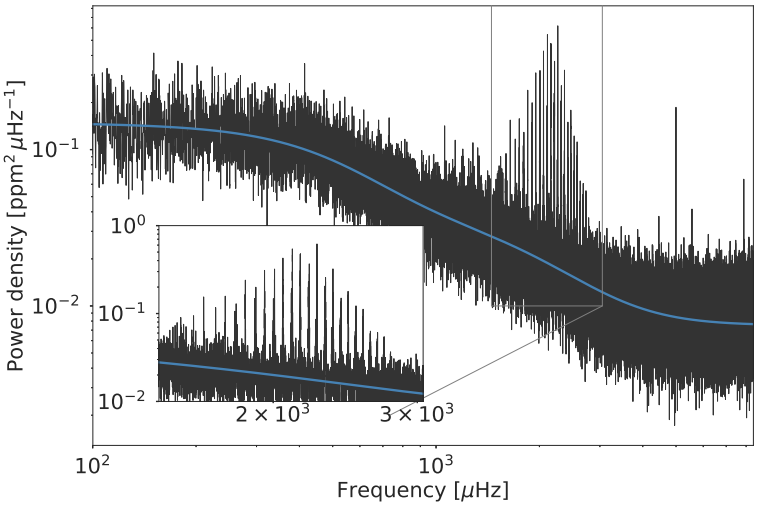
Asteroseismic analysis and age estimation of selected tess stars



Simon Banerjee

Master's Thesis in Astronomy

September 2019

Supervisor: Hans Kjeldsen

Co-Supervisor: Rasmus Handberg

Co-Supervisor: Mikkel Nørup Lund

Department of Physics and Astronomy

Aarhus University

Colophon

Asteroseismic analysis and age estimation of selected TESS stars

— Asteroseismisk analyse og aldersbestemmelse af udvalgte TESS stjerner

Cover image: Power spectrum for 16 Cyg A with a zoom-in indicating the oscillation envelope.

Master's thesis by Simon Banerjee. Written under supervision by Prof. Hans Kjeldsen, Department of Physics and Astronomy, Aarhus University.

Typeset by the author with LaTeX and the memoir document class, using Linux Libertine and Linux Biolinum 11.0pt.

Printed at Aarhus University

Abstract (English)

In this thesis I have performed asteroseismic analysis of 5 selected stars from the TESS (The Transiting Exoplanet Survey Satellite) mission. Asteroseismology is the study of stars through observing their oscillations. These oscillations provide an unique window to the hidden interiors of the star. The purpose of this thesis is to construct algorithms to calculate fundamental asteroseismic parameters, which are used to estimate the mass, radius and age of each star. In this project I estimate the age using two different methods.
The first method is testing two new scaling relations and the second method is creating theoretical stellar evolution models. The scaling relations are calibrated to work for main sequence stars and giants. However, in this thesis I have worked with 3 subgiants, a giant and a giant/red clump star. The scaling relation for main sequence stars gave some results with small deviations from the literature for two of the subgiants. This indicates that the scaling relation may extend further into the subgiant regime. The same can be said with the scaling relation for giants, which produced a result with minor deviation from the reference value for a subgiant and the giant. This also indicates that the lower boundary for the scaling relation may extend into the subgiant regime.
The theoretical stellar models provided results for the age of each star that were in correlation with the scaling relations, however, with some deviations. The deviations come from the mass used in the stellar models. The mass is a mean of the estimated masses using the scaling relations, and if the mean mass is higher or lower than the reference value for the mass, it will have an impact on the estimated age from the models.
In the case for the giant/red clump star, none of the scaling relations or the stellar models seemed to match the reference value and was a factor 2 too low from my estimations in both cases (in the case it is a red giant and the case where it is a red clump star). This led to creating a stellar model using the mass, metallicity and radius from the paper used as reference in the case it is a red clump star. This gave an age a factor 2 above the estimated age in the reference paper, which indicates that the results obtained in my thesis is closer to the true age of the star.

Resumé (Dansk)

I denne afhandling har jeg udført en asteroseismisk analyse af 5 udvalgte stjerner fra TESS (The Transiting Exoplanet Survey Satellite) missionen. Asteroseismologi er studiet af stjerners oscillationer. Disse oscillationer giver et unikt vindue til stjernens skjulte interør. Formålet med denne afhandling er at konstruere algoritmer, der kan udregne fundamentale asteroseismiske parametre, som kan bruges til at estimere massen, radius og alderen for hver stjerne. I dette projekt estimerer jeg alderen på to forskellige måder.
Den første metode er afprøvningen af to nye skaleringsforhold, og den anden metode er ved at lave teoretiske stjernemodeller. Skaleringsforholdene er kalibrerede til at fungere for hovedseriestjerner og giganter. I dette speciale har jeg arbejdet med 3 subgiganter, en gigant og en gigant/rød klumpstjerne. Skaleringsforholdene for hovedseriestjernerne gav resultater med små afvigelser fra litteraturværdien for to af subgiganterne. Dette indikerer, at skaleringsforholdet måske også dækker over en del af subgigant området. Det samme gælder skaleringsforholdet for giganter, der også gav resultater med små afvigelser fra referenceværdien for en subgigant og giganten. Dette indikerer at den nedre grænse for skaleringsforholdet måske også dækker over en del af subgigantområdet.
De teoretiske stjernemodeller gav en alder, der var i overensstemmelse med skaleringsforholdene, dog med minimale afvigelser. Afvigelserne kommer fra den anvendte masse i stjernemodellerne. Massen anvendt i stjernemodellerne er et gennemsnit af de masser, der er blevet estimeret på baggrund af skaleringsforholdene, og hvis massen er højere eller lavere end referenceværdien, så vil dette have en indvirkning på de estimerede aldre fra modellerne. I tilfældet, at det er en gigant/rød klumpstjerne, passede hverken skaleringsforholdene eller stjernemodellerne til referenceværdien og var en faktor 2 for lav i forhold til mine estimeringer i begge tilfælde (i tilfældet at det var en rød gigant og i tilfældet at det var en rød klumpstjerne). Dette førte til at lave en stjernemodel ved at bruge massen, metalliciteten og radius fra den artikel, der er brugt som reference i tilfælde af, at det er en rød klumpstjerne. Dette gav en alder, der var en faktor 2 over den estimerede alder i artiklen, hvilket indikerer, at resultaterne opnået i min afhandling er tættere på stjernens sande alder.

Acknowledgements

This thesis is the product of my research carried out over the last year corresponding to 60 ects points. The hand in of this thesis also means that I have finished my master's degree in astronomy meaning that my time as a master student at Aarhus University has come to an end. In this thesis I have used data from the TESS satellite to perform asteroseismic analysis on some of the targets that displayed solar-like oscillations.
There are several people who have helped me during this process, and they all deserved my sincerest gratitude, because without them this thesis would never have been possible.
First of all I want to thank my supervisor Prof. Hans Kjeldsen for his fantastic supervision. Hans always provided great input and helped me when I experienced problems in both coding or writing. I have enjoyed all of our small meetings and his excitement and drive for science inspired me during this whole process.
A big thank you is also deserved for my two co-supervisors Rasmus Handberg and Mikkel Nørup Lund whose door was always open for discussion and who helped me when I had problems with my code.
A special thanks to Earl Bellinger for his help with MESA and GYRE making it possible to create these stellar evolutionary models, and his time to explain some of the physical elements in asteroseismology and of course for reading some of the chapters of my thesis and giving me corrections.
I would like to thank my friends at the office Janne Højmark Mønster, Rasmus Lind Bjerre and Dorte Thrige Plauborg for always being ready to discuss problems and and motivating me during the past year. I also want to thank my good friend Matias Wallenius for reading my thesis and giving me corrections, and just for being helpful whenever I needed it. Finally, I want to thank my family and friends for their support and understanding especially in the last part of the process. A special thanks is also needed

for my rock and fantastic girlfriend Linnea who have been understanding, supporting and loving me through this last year.

Contents

Chapter 1

Introduction

The points of light in the sky are not constant but rather variable, meaning they dim and brighten over time. Some of these variations are periodic, which mean that the dim and brighten occur with some regularity. This observation may have been known since the Egyptians who wrote calenders, in which they recorded the 2.85 day period of the so-called "Demon Star", Algol (Jetsu and Porceddu 2015). This indicates that astronomy is one of the most ancient and fascinating sciences we have.
However, the problem with astronomy is that it is also one the most constrained sciences. Astronomy is an observational science, meaning that we can not recreate the physics happening inside stars in the laboratories. This forced astronomers to learn how to dissect the light we received from these stars.
Stars are one of the most fundamental pieces in astrophysics. This is because stars produce all the elements that other astronomical objects are comprised of. They also produce most of the light the astronomers can observe and are therefore very important for our ultimate search for life outside our own planet. Learning about the properties of stars can tell us something about the environment of a potential orbiting planet. These stellar properties are also the cornerstone in determining basic properties such as mass, density, radius etc. of new found exoplanets.
Even though we can see the light from these stars, the light is so far away making it impossible to resolve their surface details. Furthermore, some of the most interesting properties of the star are hidden in the core, a

place we still can not directly observe. Sir Artur Eddington put it very beautifully:

"At first sight it would seem that the deep interior of the sun and stars is less accessible to scientific investigation than any other region of the universe. Our telescopes may probe farther and farther into the depths of space; but how can we ever obtain certain knowledge of that which is hidden behind substantial barriers? What appliance can pierce through the outer layers of a star and test the conditions within?" Eddington 1920.

The answer to the question asked by Eddington is asteroseismology, which is the study of the oscillations of stars. This can determine basic properties of a distant star and give an insight of the internal mechanisms occurring there. These unique environments are powerful laboratories for many parts of physics such as hydrodynamics, nuclear fusion and neutrino physics. As mentioned, it is not possible to resolve the surface of the stars, meaning that the information about the stellar interior comes from the emitted light. From the light it is possible to extract information about the oscillation frequencies of the star. These frequencies depend on the temperature, density, composition and other properties of the stellar interior. This means that these frequencies can infer physical properties of the star.
The development of this field is due to the success and achievements of helioseismology, which is the study of the solar interior, by observing the frequencies of the Sun. These observations made it possible to predict the inner structure of the Sun in higher detail than before. The quality of the solar data is better than we can ever achieve for any other stars, which means that most of our knowledge of other stars is based on our knowledge of the Sun. This success of helioseismology led to the idea of applying similar methods on other stars. This started the search for solar-like oscillations in the nearest solar-like stars through ground based observations. The field has been developing rapidly since the first detection almost 2 decades ago. These days we receive excellent data from space missions such as CoRoT and TESS. However, these days asteroseismology is not limited to solar-like stars. Almost every star is believed to pulsate at some point of its evolution, and therefore asteroseismology is performed on a large variety of stars.

This Thesis

In this thesis I present a detailed asteroseismic analysis of 5 stars. The analysis is based on photometric data from the TESS mission. In this thesis I will:

- write a code that can take raw light curves and estimate stellar parameters such as mass, radius and age based on asteroseismic parameters.
- analyze different types of stars that are in different stages of their evolution.
- test two new scaling relations to estimate the age of the stars.
- create theoretical stellar evolution models to estimate the age of the stars and compare these with the results from the analysis.

Outline

This thesis is structured as follows:

Chapter 2 gives an introduction to asteroseismology and what solar-like oscillation are. In this chapter I present some of the asteroseismic parameters used in the thesis. I explain how the different oscillation modes behave and are identified for solar-like oscillations. Lastly I introduce one of the methods to estimate the age in the thesis.

Chapter 3 gives an introduction to the TESS mission and its predecessor Kepler. Here I present the difference between the two missions relative to observational methods and spectral ranges.

Chapter 4 gives an introduction to the code MESA used to create the stellar evolution models and the code GYRE used to calculate the frequencies of the star.

Chapter 5 I introduce the theory and derive the exponents used to calculate the power spectrum for the stars, and show the comparison to other common used methods.

Chapter 6 I present the methods for preparation of the light curves and estimation of the asteroseismic parameters used to estimate the stellar properties.

Chapter 7 presents and discusses the main results of this thesis, which are the asteroseismic parameters estimated for each star. Furthermore I present and discuss the estimated mass, radius and age for each star obtained from both scaling relations and theoretical stellar evolution models.

Chapter 8 includes the discussion and conclusion. In this chapter I discuss about possible improvements of the code and the challenges faced in the process. Finally, I compare the results obtained in this thesis to results from other papers.

Chapter 2

Solar-Like Oscillations

In this chapter some of the theory used to make the asteroseismic analysis will be presented. I will describe how stellar oscillations occur and how to use these to tell something about the stars. Lastly, I will introduce some of the methods used to estimate these stellar parameters of the stars considered in this thesis.

Stellar Oscillations

Stars are spheres of hot gas and plasma held together by gravity in hydrostatic equilibrium. If a star is moved away from its equilibrium by some mechanism, it will start to oscillate. However, due to internal friction in the star within the gas, these oscillations will be intrinsically dampened, so if a star is to keep oscillating, a driving mechanism must be present to keep exciting the oscillations.

The modes oscillating in the star are characterized by the restoring force which mainly comes in two variants. When the pressure gradient is the restoring force, the mode is referred to as a p-mode. A p-mode thus constitutes a standing wave in the star. When buoyancy is the restoring force, the mode is refereed to as a g-mode, or gravity mode. These two kinds of modes are easily distinguishable in most stars, but in some evolved stars these modes start to couple and gives rise to what is known as mixed modes. These types of modes behave like g-modes in the interior and p-modes near the surface. Not so important, but there are also other types of modes, such as Rossby waves. The modes of oscillation depend on the structure of

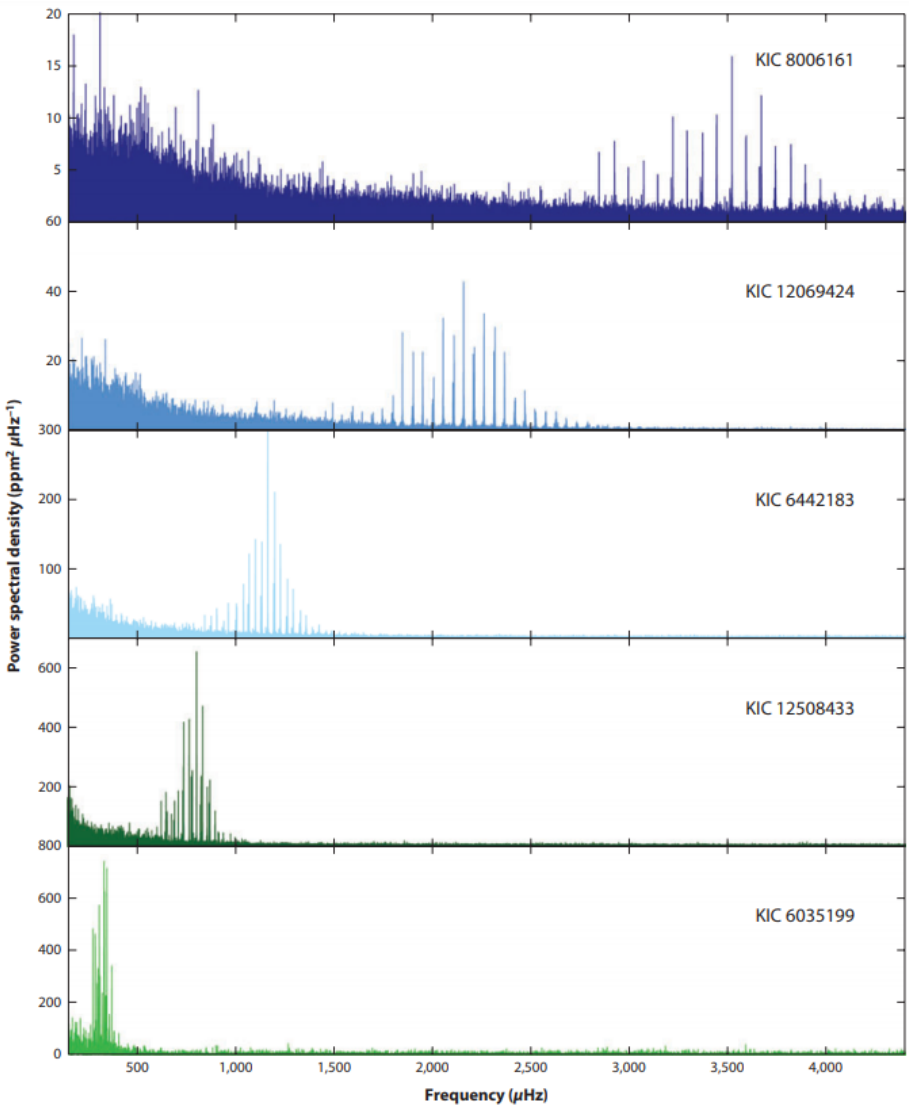


Figure 2.1: Solar-like oscillations for 5 different $\sim 1M_{\odot}$ stars observed with Kepler. The stars are arranged after their $\nu_{\max}$ value (from highest to lowest). The two first stars, KIC 8006161 and KIC 12069424 (16 Cyg A), are main sequence stars. The third and fourth star, KIC 6442183 (HD 183159) and KIC 12508433, are subgiants. The bottom star KIC 6035199 is located in the base of the RGB (Red Giant Branch). Figure is adapted from Chaplin and Miglio 2013.

the star. This is due to the fact that only certain cavities can hold certain frequencies. In musical instruments, a large cavity results in low frequency oscillations and vice versa for a small cavity. It is the same with stars, where giants oscillate in low-frequency high-amplitude mode, while stars like main-sequence stars oscillate in high-frequency low-amplitude modes (see Fig. 2.1).

As seen on Fig. 2.2, the Hertzsprung-Russell diagram (HR-diagram), one can find different types of oscillations, where the types are determined by the stellar structure, which again determines which sort of excitation mechanisms drive the oscillations. Oscillations caused by g-modes can for example only propagate in radiative zones. In many parts of the HR-diagram the excitations mechanism works as a form heat engine, where it converts thermal energy into mechanical energy. In many stars this specific mechanism is refered to as the kappa-mechanism due to major effect from opacity (κ). In this mechanism opacity drives the oscillations by increasing in the ionization zone when the temperature and pressure increases. This, however, makes a higher absorption of energy from the stellar interior. This causes heating and expansion of the outer layer, which lowers the opacity causing a lower absorption energy. The layers will then again contract and the process will repeat itself. This type of mechanism can drive both p- and g-modes, however, these modes are intrinsically unstable due to the fact that the oscillation will continue to rise until some sort of limiting factor sets in. This type of driving mechanism is responsible for the classical p-mode oscillations in many stars such δ-Scuti stars, RR Lyrae stars, β-Cep stars and Cepheids as seen in Fig. 2.2. Another important type of oscillations is the solar-like p-mode oscillations, named because the it is the same oscillations dominating the solar spectrum. The solar-like oscillations are intrinsically stable and stochastically excited from the turbulent convection in the outer parts of the star. These types of oscillations are expected to happen in stars that have a convective envelope including subgiants (β-hydri, Bedding et al. 2007a and ν-Indi, Bedding et al. 2006a). The oscillations in stars give rise to changes in pressure and temperature along with velocities at the surface. These variations then causes variations in the wavelength and intensity of emitted light from the stellar surface and it is these variations in intensity and radial velocity that allows us to study the oscillations of stars. In this thesis I will focus on the solar-like oscillations.

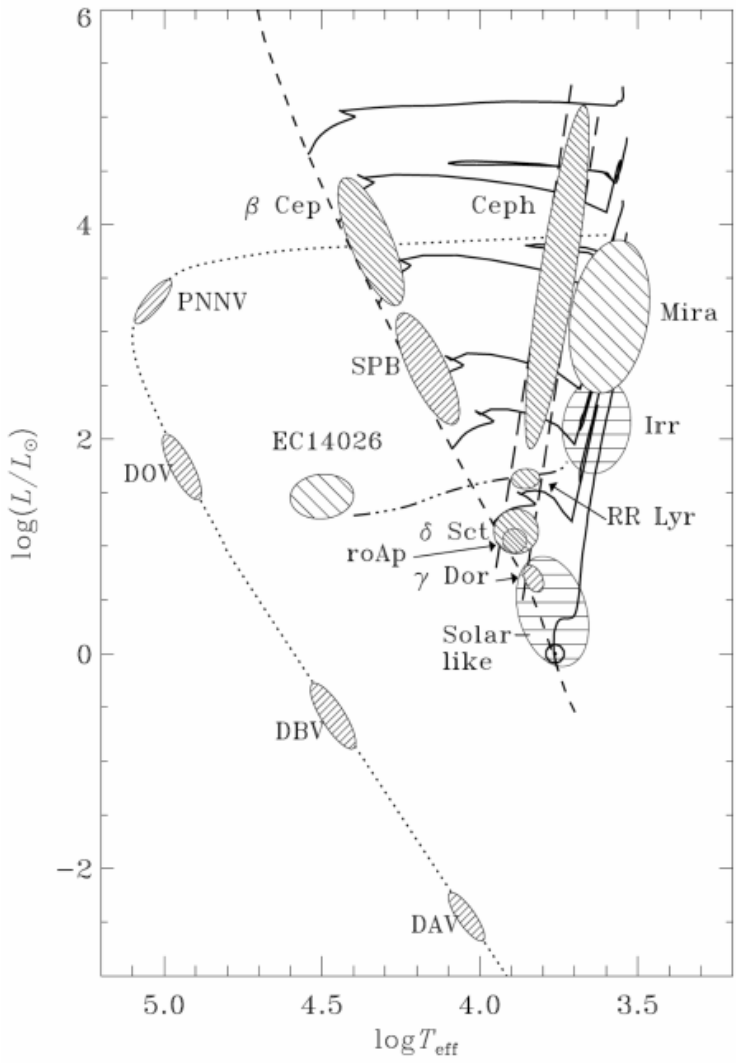


Figure 2.2: A Hertzsprung-Russell diagram showing different classes of pulsating stars where asteroseismology is possible. In the figure three different patches are seen each indicating a different type of oscillation. Patches with horizontal stripes (≡) indicate solar-like oscillations, backward diagonal stripes (\\\\) indicate the driving of heat-engine p-modes, forward diagonal stripes (////) indicate the driving of heat-engine g-modes. Some of the acronyms used are: rapidly oscillating Ap; Slowly Pulsating B (SPB); subdwarf B variables (sdBV); the DBV and DAV stars are variable DB (helium-rich), DA (hydrogen-rich) white dwarfs, red giants (RG), andsemiregular variables (SR). The parallel long-dashed lines indicate the Classical instability strip. Figure is adapted from Aerts et al. 2010.

Solar-Like Oscillations

Solar-like oscillations are standing waves known as p-modes (the pressure gradient is the restoring force) caused by stochastic excitation which is driven by turbulent convection in the outer layers (Christensen-Dalsgaard 2004). The amplitudes are determined by the balance between energy input from the convection (which depends on the frequency) and the dampening rate. As mentioned earlier, these small amplitude oscillations of a spherical symmetric star can be seen as small perturbations around the equilibrium of the star. These oscillations can be described using a basis of spherical harmonics functions $Y_l^m(\theta, \phi)$ and in spherical coordinates (r,θ,ϕ), where r is the distance to the center, θ is the co-latitude and ϕ is the longitude. Solving these equations of motion of the displacement of the gas gives the three following equations

$$\xi_r(r, \theta, \phi, t) = a(r) Y_l^m(\theta, \phi) \mathrm{e}^{-i 2\pi \nu_{\mathrm{nlm}} t}, \tag{2.1}$$

$$\xi_\theta(r, \theta, \phi, t) = b(r) \frac{\partial Y_l^m(\theta, \phi)}{\partial \theta} \mathrm{e}^{-i 2\pi \nu_{\mathrm{nlm}} t}, \tag{2.2}$$

$$\xi_\phi(r, \theta, \phi, t) = \frac{b(r)}{\sin\theta} \frac{\partial Y_l^m(\theta, \phi)}{\partial \phi} \mathrm{e}^{-i 2\pi \nu_{\mathrm{nlm}} t}, \tag{2.3}$$

where $a(r)$ and $b(r)$ are amplitudes and ξ_r, ξ_θ and ξ_ϕ are displacements, ν_{nlm} is the oscillation frequency of the mode characterized by the numbers n, l and m which defines the surface structure through the mode of the the spherical harmonic function Y_l^m.

The degree of n is related to the number of radial modes and is described as the overtone or radial order, l is the degree of mode and specifies the number of surface modes, m is the azimuthal order of of the node, where |m| indicates the number of l nodal lines in the longitudinal direction. It follows therefore that a given node has $2l$+1 m-components in the range of $-l$ to $+l$, and named as zonal (m=0), tesseral ($0 < |\mathrm{m}| < l$) and sectoral (m $\pm$ l) (Aerts et al. 2010, and Lund 2015).
The equations displayed in Eq. 2.1 - Eq. 2.3 are not directly used in my code. However, they are used to describe the different ways stars can oscillate, which is shown in Fig. 2.3, where different examples are given for stars with different modes on the surface. A radial component (which is not shown in Figure 2.3) is denoted by l=0 and gives the expansion and contraction of the whole star. Non-radial modes are given by l>0 and are

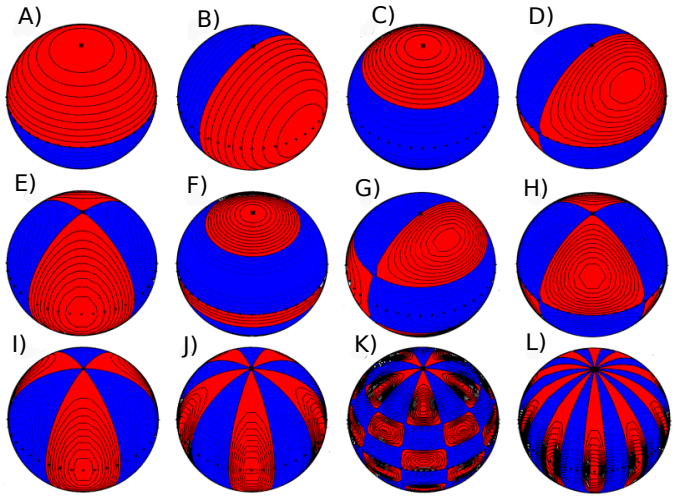


Figure 2.3: Illustration of the stellar surface profile for various modes given by the real part of the spherical harmonics Y_l^m. The red regions are areas at which the star is expanding, while the blue regions are areas where the star is contracting. The equator of the star is shown by "++++". The following cases are illustrated: A) l= 1, m = 0; B) l= 1, m = 1; C) l= 2, m = 0; D) l= 2, m = 1; E) l= 2, m = 2; F) l= 3, m = 0; G) l= 3, m = 1; H) l= 3, m = 2; I) l= 3, m = 3; J) l= 5, m = 5; K) l= 10, m = 5; L) l= 10, m = 10. The figure is adapted from Lund 2015 which is a modified version of Christensen-Dalsgaard 1997.

named the following way in asteroseismology: dipole (l= 1), quadrupole (l= 2), octupole (l= 3), hexadecapole (l= 4), and dotriacontapole modes (l= 5). Higher degree of modes are generally not observed in stars due to partial cancellation (see Figure 2.4) (Lund 2015), but would be observed in the observations of the surface of the Sun, which is done using helioseismic observations.

The Power Spectrum of Solar-Like Oscillations

If a star is spherical symmetric, then the frequencies only depend on n and l(Tassoul 1980), given by the asymptotic relation:

$$\nu_{nl} \approx \Delta\nu \left(n + \frac{l}{2} + \varepsilon \right) - l(l+1)D_0 \tag{2.5}$$

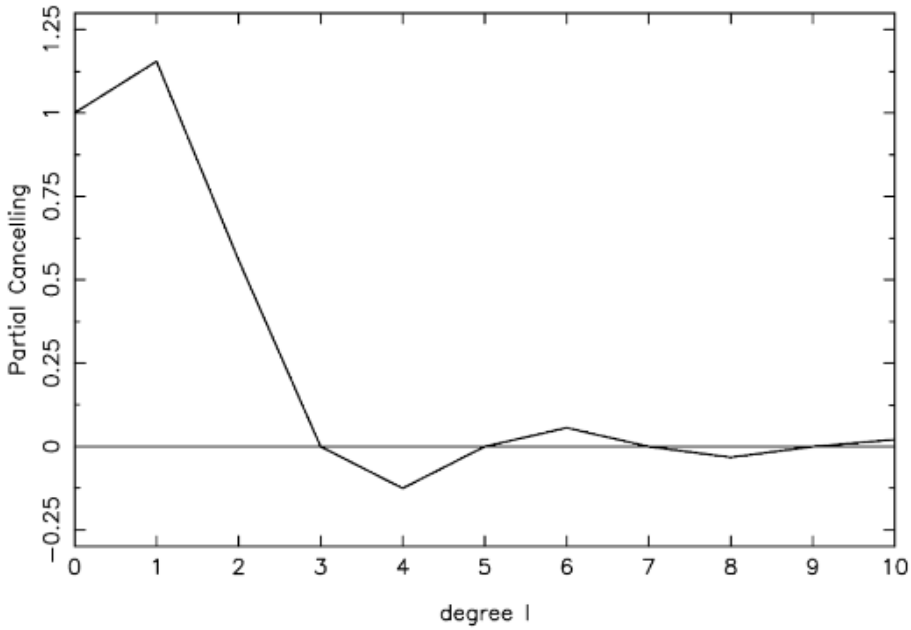


Figure 2.4: A visibility plot of different modes as function of l. The figure is adapted from Aerts et al. 2010 .

where $\Delta\nu$ is the large frequency separation, which is the inverse of the sound travel time for a sound wave from the surface of the star to the core and back again (Aerts et al. 2010), given by

$$\Delta\nu = \frac{1}{2}\left[\int_0^R c(r)^{-1} dr\right], \tag{2.6}$$

where $c(r)$ is the sound speed and R is the stellar radius. The parameter D_0 is sensitive to the speed near the core of the star, where ε is more sensitive to the surface layers of the star. The parameter D_0 is used to express a quantity called the small frequency separation indicating the distance between two peaks with specific l-number, for example $\delta\nu_{02}$ indicates the separation between l=0 and l=2 in the power spectrum. In Fig. 2.5 one can see a small section of the solar spectrum marked with some the frequency separations. If Eq. 2.5 is assumed to be correct, one can find these

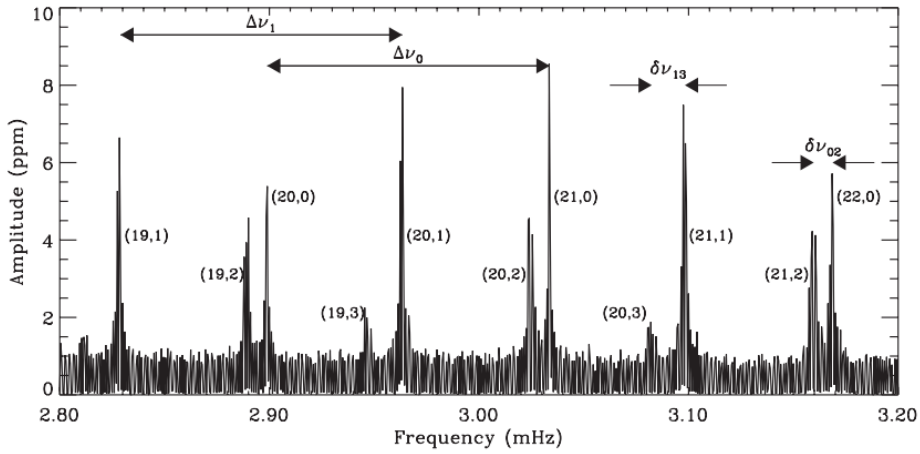


Figure 2.5: A section of the power spectrum of the Sun marked with several frequency separations. Figure is adapted from Bedding and Kjeldsen 2003.

frequency separations using D_0 the following way

$$D_0 = \frac{1}{6}\delta\nu_{02} = \frac{1}{2}\delta\nu_{01} = \frac{1}{10}\delta\nu_{13}. \tag{2.7}$$

Due to the regularities in the power spectrum one often plots a power spectrum in a so-called echelle-diagram as shown in Fig. 2.6.

The term "solar-like oscillations" are these days also being applied to stars not very similar to the Sun (see Fig. 2.2). When a star evolves away from the main sequence, its core will contract, whereas the outer layers will start to expand due to the lack of energy production in the core of the star. This will cause a decrease in the effective temperature. This will cause the p-modes to change towards lower frequencies, where it the same time will cause the g-modes located in the inner part of the star to move towards higher frequencies. Eventually, the frequencies of these two kinds of modes will coincide. In Fig. 2.7 one can see the p-modes of a solar-like star as a function of time. Here one can see that some of the l=1 lines shifts in what is known as an avoided crossing. This phenomenon occurs when the frequency of the p-mode are shifted due to the g-mode rising in frequency. Sometimes when the g-mode crosses the p-mode, the p-mode can actually take on the character of a g-mode in what is called a mixed mode.

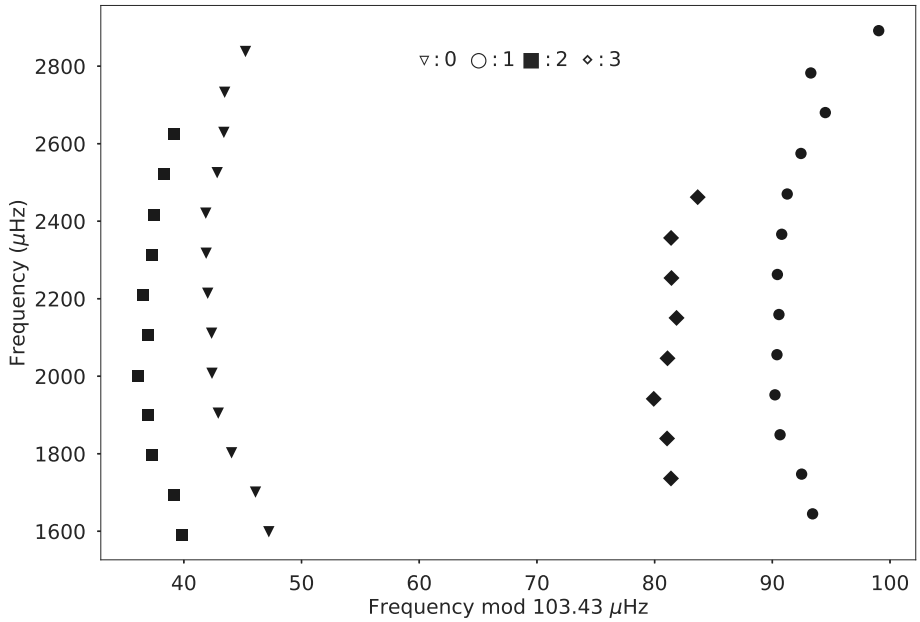


Figure 2.6: Echelle-diagram for the 16 Cyg A showing the modes for l=0, l=1, l=2 and l=3.

Avoided Crossings

Radial modes in stars are pure acoustic modes with pressure as the restoring force (p-modes). Non-radial modes in stars have mixed modes where buoyancy is the restoring force in the deep interior (g-modes) of the star and pressure is the restoring force in the outer layers of the star (Hekker and Christensen-Dalsgaard 2017). In some types of stars, these p- and g-modes get very close to each other and couple. So instead of propagating through each other we have this attempt at avoiding crossing of frequencies, which means we have an unseen g-mode shifting the p-mode from its expected location. This so-called mode bumbing is a great tool to find these g-modes. However, the method can only be used if both the p- and g- modes have the same properties and frequencies nearing each other. Normally, g-modes have much lower frequencies than p-modes, but that changes when a star evolves and starts leaving the main sequence (Hekker and Christensen-Dalsgaard 2017).

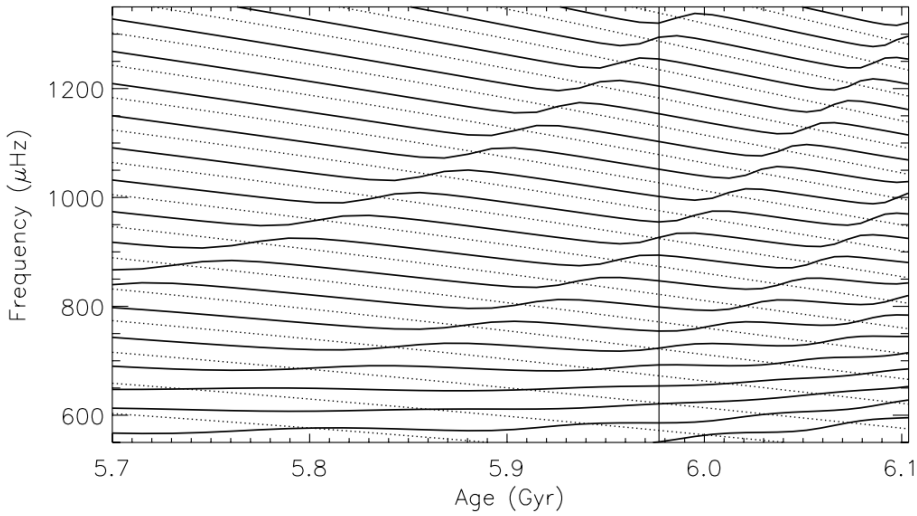


Figure 2.7: The p-mode oscillations as a function of the stellar age of the star KIC 11026764. The dashed lines indicate l=0 and the solid lines indicate l=1 modes. Figure is adapted from Metcalfe et al. 2010.

Scaling Relations

A solar scaling relation is an equation to estimate some unknown properties of star from observations by scaling from known properties of the Sun (Kjeldsen and Bedding 1995; Stello et al. 2008). The scaling relations have the form (Bellinger 2019a)

$$\frac{Y}{Y_\odot} \simeq \prod_i \left(\frac{X_i}{X_{\odot,i}}\right)^{P_i}, \tag{2.8}$$

where Y is the stellar parameter we try to estimate (e.g. mass, radius or age), whereas $Y_\odot$ is the corresponding stellar parameter for the Sun (e.g. solar mass, solar radius or solar age). The quantity X is more a vector X which contains measurable properties of the star, such as luminosity or effective temperature. The vector $\mathrm{X}_{\odot,i}$ then contains the corresponding solar properties. The quantity P is also more a vector, P, of exponents. The exponents will change for the fraction $\left(\frac{X_i}{X_{\odot,i}}\right)$ depending on which property

of the star we are estimating plus which measurable properties we are using. A simple scaling relation can be derived from the Stefan-Boltzmann law to estimate stellar radii. It is given as

$$\frac{R}{R_\odot} \simeq \left(\frac{T_{\rm eff}}{T_{\rm eff,\odot}}\right)^{-2} \left(\frac{L}{L_\odot}\right)^{\frac{1}{2}}, \tag{2.9}$$

where R is the radius, $T_{\rm eff}$ is the effective temperature and L is the luminosity.
In the field of asteroseismology the relations are called seismic scaling relations. They are used to estimate the unknown masses and radii of stars by scaling asteroseismic observations with their helioseismic counterparts. The properties used in these scaling relations include the frequency spacing between radial mode oscillations of consecutive radial order, also called the large frequency separation, $\Delta\nu$, and the frequency of maximum oscillation power, $\nu_{\rm max}$.
It is known from several testing and stellar models that there is a correlation between the mean stellar density, ρ and the large frequency separation (Kjeldsen and Bedding 1995) given as

$$\Delta\nu \propto \sqrt{\rho} \tag{2.10}$$

$$\frac{\rho}{\rho_\odot} \cong \left(\frac{\Delta\nu}{\Delta\nu_\odot}\right)^2. \tag{2.11}$$

There is also a relation between the large frequency separation and the mass and radius of the star given as

$$\Delta\nu \propto \left(\frac{M}{R^3}\right)^{\frac{1}{2}}. \tag{2.12}$$

Just as with the large frequency separation ($\Delta\nu$), there is also a relation between the frequency of maximum oscillations ($\nu_{\rm max}$) and the temperature (Kjeldsen and Bedding 1995) of the star given as

$$\nu_{\rm max} \propto g\, T_{\rm eff}^{-\frac{1}{2}} \tag{2.13}$$

which can be written into

$$\nu_{\max} \propto \left(\frac{M}{R^2 \sqrt{T}} \right). \tag{2.14}$$

Using Eq. 2.12 and Eq. 2.14, we can rewrite the scaling relations to estimate the mass M and the radius R of the star via

$$\frac{M}{M_\odot} \simeq \left(\frac{\nu_{\max}}{\nu_{\max,\odot}} \right)^3 \left(\frac{\Delta\nu}{\Delta\nu_\odot} \right)^{-4} \left(\frac{T_{\text{eff}}}{T_{\text{eff},\odot}} \right)^{\frac{3}{2}} \tag{2.15}$$

$$\frac{R}{R_\odot} \simeq \left(\frac{\nu_{\max}}{\nu_{\max,\odot}} \right) \left(\frac{\Delta\nu}{\Delta\nu_\odot} \right)^{-2} \left(\frac{T_{\text{eff}}}{T_{\text{eff},\odot}} \right)^{\frac{1}{2}} \tag{2.16}$$

where $\nu_{\max,\odot} = 3090 \pm 30\ \mu$Hz, $\Delta\nu_\odot = 135.1 \pm 0.1\ \mu$Hz (Huber et al. 2011) and $T_{\text{eff},\odot} = 5772.0 \pm 0.8$ K (Pra et al. 2016). These scaling relations are extremely useful due to the fact that the asteroseismic data can be obtained with very high precision. For a well-observed solar-like star, the relative error for $\Delta\nu$ and $\nu_{\max}$ are around 0.1% and 1% respectively (Bellinger et al. 2019).

Scaling Relation For Stellar Age

In the more recent years there have been a push in improving the determination of the stellar properties, and especially the stellar age. The stellar age is a useful property when looking at the formation of the Galaxy or trying to understand stellar and exoplanetary formation. Unlike for mass and radius, there is no scaling relation for stellar age. However, there are several methods that have been developed to match asteroseismic observations with models of stellar evolution.

Scaling relations are excellent resources due to the fact that they can be used immediately on observations without the need of any theoretical models. In Bellinger 2019a a new scaling relation is proposed. The purpose of this scaling relation is to estimate the stellar age of main-sequence stars and improve the estimate of the radius and mass of the star.

The way of doing so is to look at the abundance of hydrogen in the core, which is also an approximate indicator for the age of the star. It is well

	K	ν_{max} α	$\Delta\nu$ β	$\delta\nu$ γ	T_{eff} δ	[Fe/H] ε
Classic	M	3	−4		1.5	
New	M	0.975	−1.435		1.216	0.270
Classic	R	1	−2		0.5	
New	R	0.305	−1.129		0.312	0.100
Seismic	R	0.883	−1.859			
New	age	−6.556	9.059	−1.292	−4.245	−0.426

Table 2.1: Classical and new MCMC-fitted exponents for scaling relations of main sequence stars.

known that this abundance is well correlated with the frequency spacings between the radial and quadrapole oscillation modes (Christensen-Dalsgaard 1984; Aerts et al. 2010; Basu and Chaplin 2017). This spacing is the asteroseismic parameter denoted as $\delta\nu$ and called the small frequency seperation. Using $\delta\nu$, the metallicity and the other asteroseismic parameters mentioned above, a good estimate of the stellar age can be made. The exponents in this scaling relation were estimated by calibrating 80 solar type stars whose age and other stellar parameters have been determined through detailed models and simulations, Bellinger 2019a. The scaling relations shown in Eq. 2.15 and Eq. 2.16 can be written in a more general way as follows for an arbitrary quantity K (and corresponding solar quantity $K_\odot$) as follows:

$$\frac{K}{K_\odot} \simeq \left(\frac{\nu_{max}}{\nu_{max,\odot}}\right)^{\alpha} \left(\frac{\Delta\nu}{\Delta\nu_\odot}\right)^{\beta} \left(\frac{\delta\nu}{\delta\nu_\odot}\right)^{\gamma} \left(\frac{T_{eff}}{T_{eff,\odot}}\right)^{\delta} \exp\left([Fe/H]\right)^{\varepsilon} \tag{2.17}$$

.

For suitable powers $P = [\alpha, \beta, \gamma, \delta, \varepsilon]$. The values for the solar data ($\delta\nu_\odot$ and solar age ($\tau_\odot$)) are given by $\delta\nu_\odot = 8.957 \pm 0.059$ μHz (Davies et al. 2014) and $\tau_\odot = 4.569 \pm 0.006$ Gyr (Bonanno and Fröhlich 2017). Using the MCMC method (Markov chain Monte Carlo) the following exponents for Eq. 2.17 can be seen in table Table 2.1.

A similar study has been made for red giant branch stars (Bellinger 2019b). The reason for two different scaling relations is because that it is two different asteroseismic parameters that are used to give an estimate

of the age of the star. On the main sequence, the age is linked to the amount of hydrogen in the core of the star, which is probed by the small frequency separation. In a star on the red giant branch, however, the core is emptied for hydrogen and no fusion is taking place, which is why the small frequency separation is not good for measuring the age of red giants. Besides the purely acoustic oscillations observed in solar-like stars, red giants also display mixed modes, i.e., non-radial acoustic oscillations, which are coupled with gravity-mode oscillations that propagate through the core. Gravity modes are nearly equally spaced in period (frequency clumps together), and this clumping of frequencies indicates how far in the evolution the star is (for example if it is burning helium in the core or burning hydrogen in the shell, Bedding et al. 2011). The modifed scaling relation for red giant branch stars is:

$$\frac{K}{K_\odot} \simeq \left(\frac{\nu_{\max}}{\nu_{\max,\odot}}\right)^{\alpha} \left(\frac{\Delta\nu}{\Delta\nu_\odot}\right)^{\beta} \left(\frac{\Delta\Pi}{\Delta\Pi_{\mathrm{ref}}}\right)^{\gamma} \left(\frac{T_{\mathrm{eff}}}{T_{\mathrm{eff},\odot}}\right)^{\delta} \exp\left([\mathrm{Fe/H}]\right)^{\epsilon}, \tag{2.18}$$

where K is the quantity wanted (e.g the stellar mass, stellar radius or age), $\nu_{\max}$ is the frequency of maximum power, $\Delta\nu$ is the large frequency separation and $\Delta\Pi$ is the period spacing. Due to the fact that the paper is yet not published, I present here the results for the different variable combinations for age, mass, and radius respectively in Table 2.2, Table 2.3 and Table 2.4. For more understanding of how the combinations have been estimated, I refer to Bellinger 2019b. The main results from this paper that I use in this thesis are the variable combinations where I have every parameter estimated (either using my own code or by finding values in the literature) or the combination where the period spacing ($\Delta\Pi$) is unknown (variable combination 0 and 3 in Table 2.2, Table 2.3 and Table 2.4). The value of $\Delta\Pi$ is only available for one of the stars.

	$\Delta\nu$	$\nu_{\max}$	$\Delta\Pi$	T_{eff}	[Fe/H]	
Combo	α	β	γ	δ	ϵ	$\Delta\Pi_{\mathrm{ref}}$
0	−10.99	14.58	0.068 29	−5.129	0.7266	0.0004487
3	−8.65	11.68	–	−10.35	0.2914	–

Table 2.2: Calibrated exponents for the red giant scaling age relation. The rows are grouped after number of variables. When the exponent for $\Delta\Pi$ (γ) is included in the variable set, the value for $\Delta\Pi_{\mathrm{ref}}$ is included as well.

	$\Delta\nu$	$\nu_{\max}$	$\Delta\Pi$	$T_{\rm eff}$	[Fe/H]	
Combo	α	β	γ	δ	ϵ	$\Delta\Pi_{\rm ref}$
0	3.211	−4.233	−0.022 75	0.9885	−0.077 23	75.19
3	3.176	−4.195	–	1.076	−0.0704	–

Table 2.3: Calibrated Exponents for the Red Giant Scaling Mass Relation. The rows are grouped after number of variables. When the exponent for $\Delta\Pi$ (γ) is included in the variable set, the value for $\Delta\Pi_{\rm ref}$ is included as well.

	$\Delta\nu$	$\nu_{\max}$	$\Delta\Pi$	$T_{\rm eff}$	[Fe/H]	
Combo	α	β	γ	δ	ϵ	$\Delta\Pi_{\rm ref}$
0	1.086	−2.099	−0.004 593	0.3539	−0.027 01	75.19
3	1.079	−2.091	–	0.3709	−0.025 65	–

Table 2.4: Calibrated Exponents for the Red Giant Scaling Radius Relation. The rows are grouped after number of variables. When the exponent for $\Delta\Pi$ (γ) is included in the variable set, the value for $\Delta\Pi_{\rm ref}$ is included as well.

Chapter 3

Instruments

In this chapter the instruments used to take the measurements will be discussed. I first describe the Kepler mission and then its successor, the TESS mission. I describe the different observational techniques the two missions used and show the different bands they used and what effect they had on their observations.

The Kepler Mission

The NASA Kepler satellite was launched in March 2009, and the primary goal was to discover habitable exoplanets around distant solar-like stars (Borucki et al. 2008). Besides discovering multiple new exoplanets, the satellite also provided excellent data on stellar oscillations through the light curves of the stars.
Kepler was continuously observing approximately 150.000 stars in a field of ~105 deg^2 in the Cygnus-Lyra region. The satellite was carrying a 95 cm telescope with 42 charged coupled device (CCD) detectors. The spectral range at which Kepler was observing can be seen in Fig. 3.1. The Kepler satellite was designed to detect exoplanets orbiting stars using the transit method, where a small drop in the light curve of the star will occur when a planet passes in front of the star. Using the transit method, one can regain information about the radius and mass of the planet based on the radius and mass of the star. A way to determine the radius and mass of the host star is through the use of asteroseismology.
Kepler has two different observational modes:

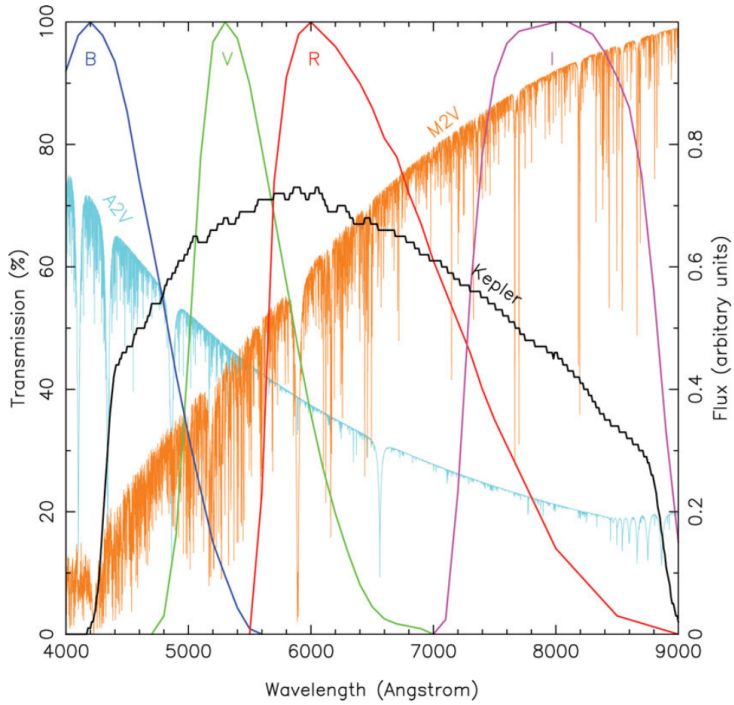


Figure 3.1: The figure illustrates the spectral sensibility of the Kepler satellite as a function of wavelength. The figure is adapted from Borucki et al. 2008.

- Short cadence, where measurements are taken every 58.9 s.
- Long cadence, where measurement are taken every 29.4 h.

Out of the around 150.000 stars, only ~ 4000 stars were being observed using the short cadence mode, the rest was being observed using the long cadence mode. When doing asteroseismology for solar-like stars primarily short cadence data are used. This is due to the fact that the time resolution is not high enough when using long cadence data.
The Kepler satellite is orbiting the Sun and therefore it needed to turn every third month so that the solar panels on the satellite are always pointed towards the Sun. These periods between these turns are called quarters, and each quarter is then divided into 3 parts. This means that the satellite transmitted data down to Earth once a month.

The raw pixel data received from the Kepler satellite then went though several processes before they are available for the Kepler Asteroseismic Science Consortium (KASC) members on the KASOC (Kepler Asteroseismic Science Operations Center) webpage.
Some of the steps in the process is described here, and a full description is given in Jenkins et al. 2010. The data went through a pipeline (Jenkins et al. 2010) which calibrated the data. The pipeline consists of the following steps:

- The bias level is subtracted which corrects for the artificial voltage on the CCD's making sure that only positive charges are detected. Because Kepler does not have a shutter, the CCD will be irradiated during the transmission. By subtracting the bias level, the pixel-to-pixel variations will be accounted for, which is caused by the constant irradiation and the transmission noise. Without a shutter a bias picture is not possible, which is why the bias level is estimated using some of the pixels at the edge of the CCD, some of which are not used for observations.

- The observed photoelectrons are converted to analog-to-digital units (ADU), and non-linear effects are taken into account during the conversion.

- A flat field correction takes the sensitivity variations in the different pixels in the CCD into account. The flat field used taken before the satellite was launched, since no flat fields are taken when in space.

The next step in the process is to do a photometric analysis of the calibrated pixels. Here aperture photometry is used, where an aperture is placed around the star. The flux of the star is then estimated based on the pixels located inside the aperture, where the background is estimated by collecting apertures across the focal plane from where a measure of the background is estimated. Each timeserie is scanned for cosmic radiation using a median filter, where those pixels that deviate significantly from the median filter are removed. That is the end of the calibration and correction of the raw flux values.
In July 2012 Kepler lost one of the four reaction wheels or gyroscopes, and in May 2013 a second reaction wheel failed. It is these reaction wheels that kept Kepler pointed precisely on its target field, and without these Kepler would no longer be able to keep orientation. This was the end of the main

Kepler mission. However, this was not the end of the Kepler satellite. The so-called K2 mission started in 2014 and used the radiation pressure from the Sun and the remaining two reaction wheels to stabilize the telescope, so that the search for new exoplanets could continue (Howell et al. 2014).

The TESS Mission

The Transiting Exoplanet Survey Satellite (TESS) was launched in April 2018, and its primary goal is to search for planets transiting bright and nearby stars (Ricker et al. 2016). During the two years TESS is set to observe, it will perform an all sky survey using four wide field CCD cameras to monitor at least 200.000 main sequence dwarf stars. The observing time of each star is depended on where on the sky the star is located (the ecliptic latitude of the star). The stars with the longest observation intervals will be the stars located near the ecliptic poles. Just like Kepler, TESS has two observational modes. The brightness of preselected targets will be recorded every 2 min, and full frame images will be recorded every 30 min (Ricker et al. 2016). Overall, the stars observed by TESS will be 10-100 times brighter than the stars observed by its predecessor Kepler.
TESS will hopefully find more than a thousand planets smaller than Neptune, and with any luck some of which are comparable with the Earth. These types are of interest due to the fact that there are no such examples in our own Solar System. The Kepler mission revolutionized the field of exoplanet science by revealing that such types of planets do exist and are abundant (Borucki et al. 2011; Fressin et al. 2013), and that they seem to have different types of compositions (Marcy et al. 2014) and interesting orbital configurations (Lissauer et al. 2011). The problem, however, was that most of the Kepler stars were too faint to perform detailed follow-up observations. That is why TESS will search for these types of planets around the nearest stars as these are bright enough to perform follow-up spectroscopy to measure the planetary masses and detailed analysis of the atmospheric compositions. Due to the fact that the brightest stars are nearly evenly distributed in the sky, an all sky survey was chosen for the mission.

Regarding to the periods (the time it takes a planet to orbit its host star), TESS would ideally detect planets with a wide range of periods, with $P_{\min}$ near the Roche limit (the minimum distance a planet can have

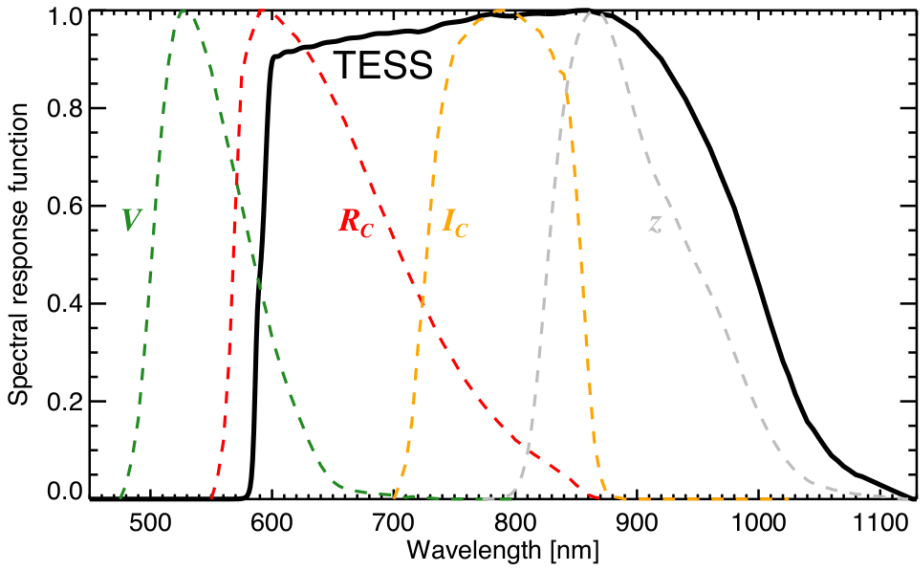


Figure 3.2: The figure illustrates the spectral sensibility of the TESS satellite as a function of wavelength. The figure is adapted from Ricker et al. 2016.

from the host star before its gravitational force can no longer withstand the tidal forces from the host star, and the planet starts to disintegrate) of a few hours, and $P_{\max}$ of a year or longer. However, the choice of $P_{\max}$ also determines the duration of the mission and thereby also the cost of the mission. Kepler detected planets with a period of 10 days, showing that many discoveries can still be made with a short $P_{\max}$. Problem with choosing a $P_{\max}$ of 10 days is that it rules out the detection of habitable-zone planets around Sun-like stars. However, by increasing $P_{\max}$ to around 40 days for a portion of the sky, it should be possible to gain access to habitable-zone planets around M stars (Ricker et al. 2016). This led to the choice of a general $P_{\max}$ of 10 days and 40 days for as large an area as possible around the ecliptic poles.

TESS will mainly observe main-sequence stars of the type F5 to M5 (stars of spectral classes FGKM). This is because evolved stars and early type dwarfs are large, making it difficult to detect small planets. Another reason for avoiding stars with spectral class earlier than F5 is that these types of stars rotate very rapidly, which broadens their spectral lines, preventing

precise radial velocity monitoring (Ricker et al. 2016). The most attractive targets TESS will observe are the M dwarfs, as three quarters of the stars in the solar neighborhood are of the type M0-M5 dwarfs. These types are relatively unexplored for transiting planets due to the fact that they constituted only a small fraction of the stars in the Kepler target list. Furthermore, the transit signal of smaller transiting planets is easier to detect for M dwarfs. The reason that M5 dwarfs are the cut-off is due to the fact that stars with spectral types later than M5 are relatively faint.
The bandpass of TESS is set to 600-1000 nm (Fig. 3.2), which means that compared to Kepler (400-900 nm), the bandpass of TESS is more in the redder area. The enhanced sensitivity to the redder wavelengths is because small planets are more easily detected around small stars as these are cool and red.
The main payload on board TESS is the four identical cameras. Each camera consists of a lens assembly with seven optical elements (with an entrance pupil of 10.5 cm), and a detector assembly with four CCDs. These four cameras are then all mounted to a single plate (see Fig. 3.3). Just like Kepler, TESS is equipped with four reaction wheels that are used to achieve the fine pointing.

As mentioned previously, TESS is set to observe for two years using these four cameras. The four cameras act as 1×4 array, which combined provides a FOV (field of view) of $24^\circ \times 96^\circ$ or 2300 deg^2 (see Fig. 3.4). The northern and southern ecliptic hemisphere are each divided into 13 overlapping regions of $24^\circ \times 96^\circ$. Each of these regions are observed for a total number of 27.4 days. It takes one year to observe an entire hemisphere and two years to perform the all-sky survey. As mentioned earlier, different stars will be monitored for a different amount of time depending on their position in the sky (see Fig. 3.4). Approximately 30.000 deg^2 are observed for at least 27 days, close to the ecliptic poles around 2800 deg^2 are observed for more than 80 days, and surrounding the ecliptic poles approximately 900 square degrees are observed for more than 300 days.
When the data from TESS have been transmitted down to Earth, it will be processed with a data reduction pipe line. The software for this pipeline was developed for the Kepler mission (Jenkins et al. 2010). The data processing includes pixel-level calibration, background subtraction, aperture photometry, identification and removal of systematic errors, and the search for the transits with a wavelet-domain matched filter.
TESS is also observing the photometric variations due to stellar oscillations,

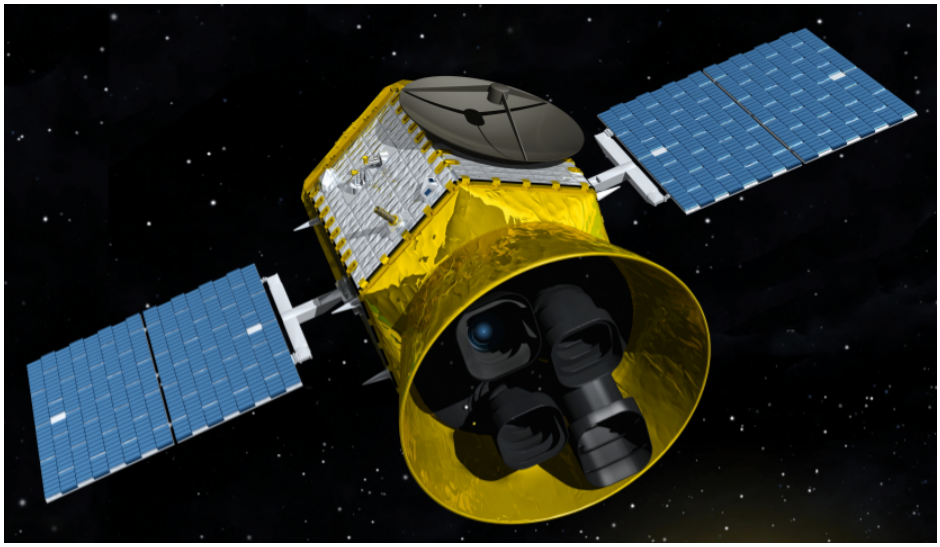


Figure 3.3: Artist's conception of the TESS satellite and its payload. The figure is adapted from Ricker et al. 2016.

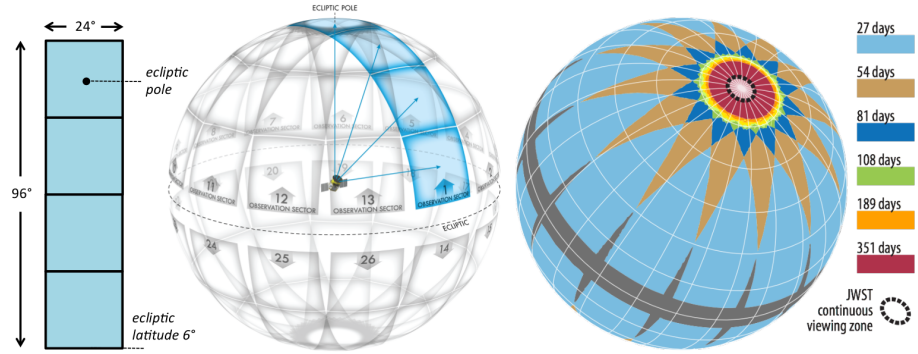


Figure 3.4: Left: The combined field of view for the TESS cameras. Middle: The two hemispheres divided into 13 observation sectors (26 total). Right: Duration of the observation for each sphere, where overlaps between the sectors are taken into account. The dashed black circle at the ecliptic pole indicates the region where the James Webb Space Telescope will be able to observe at any time. The figure is adapted from Ricker et al. 2016.

meaning that TESS is providing excellent data to perform asteroseismology. Based on the experience with Kepler, TESS is expected to detect p-mode oscillations in nearly 6000 stars brighter than V=7.5, including a) the majority of all stars brighter than V=4.5, b) about 75 stars smaller than the Sun, c) about 2000 upper-main-sequence and subgiant stars, and d) virtually all the giant stars (Ricker et al. 2016).

Chapter 4

Models

In this chapter I will give an introduction to the two programs used to create the stellar evolution models and the calculation of frequencies used as a comparison to the observational data.

MESA

Modules for Experiments in Stellar Astrophysics or MESA is a suite of open source libraries that can be used to compute different applications in stellar astrophysics (Paxton et al. 2010). For further documentation go to `http://mesa.sourceforge.net/`. MESA consists of several modules that works together to create these computational models of stellar astrophysics. Each module is constructed as a separate `Fortran 95` library. Several detailed examples has verified the capabilities of MESA. These examples include computing evolutionary tracks for a $1M_{\odot}$ from pre-main sequence to a white dwarf, evolutionary tracks for a massive star from pre-main sequence to the onset of core collaps and so on. In this project stellar models (example can be seen in Fig. 4.1) are made to test the estimated values found from the asteroseismic analysis. The input parameters for the MESA run for each star is the mass estimated using the scaling relations and the metallicity taken from the litterature which are converted to initial values for Y and Z (helium abundance and heavy elements abundance) which are given in terms of mass fraction 1=X+Y+Z. The conversion from [Fe/H] (metallicity) to initial Y and Z depends on which definition for [Fe/H] is used, here I use the one from Grevesse and Sauval 1998 with an

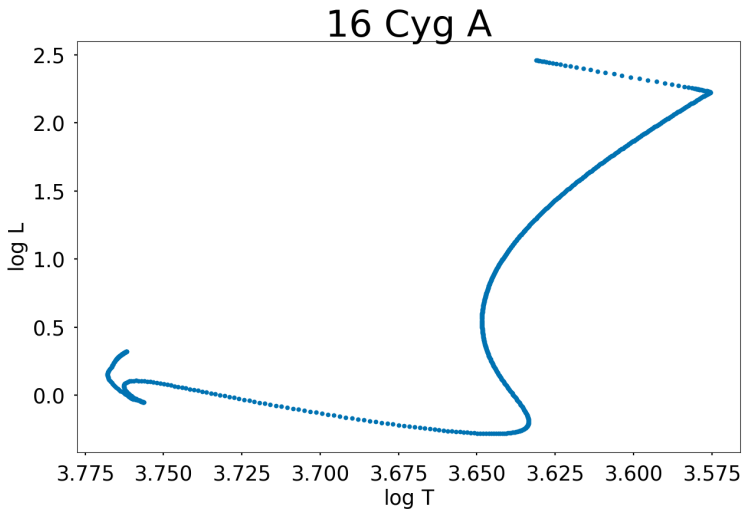


Figure 4.1: Stellar evolution model for 16 Cyg A from pre-main sequence to a pre-determined age using MESA. Mass for the star is 1.063 $M_{\odot}$, α=1.845, Y_{initial}= 0.27454206 and Z_{initial} = 0.0201729.

initial $Y_0 = 0.273$ and $Z_0 = 0.019$ which are the protosolar helium and heavy element abundance from solar calibration. The stopping criteria is set at a certain age to be certain to hit the correct model with the right temperature, right radius and thereby also the right $\Delta\nu$.

MESA Output Files

MESA produces three different types of output files which can be used for different purposes. The first output file is the history file (Fig. 4.2) which contains the history of the entire MESA run. The header of the file contains information about the initial values of the run such as initial mass and Z. Hereafter one line corresponds to one model. The first column indicates the model_number. The rest of the columns contain the calculated values for the properties such as age, mass, temperature, luminosity and so on.

The next output file is the profiles (Fig. 4.3). Unlike the history file, the profiles only contain informations about one model. The first lines con-

1	2	3	4
version_number	initial_mass	initial_z	burn_min1
8845	1.4099999999999999E+000	1.3152900000000000E-002	5.0000000000000000E+001

1	2	3
model_number	star_age	star_mass
1	1.0000000000000001E-005	1.4099999999999999E+000
2	2.2000000000000003E-005	1.4099999999999999E+000
3	3.6400000000000004E-005	1.4099999999999999E+000
4	5.3680000000000001E-005	1.4099999999999999E+000
5	7.4416000000000000E-005	1.4099999999999999E+000
6	9.9299199999999986E-005	1.4099999999999999E+000
7	1.2915903999999998E-004	1.4099999999999999E+000
8	1.6499084799999998E-004	1.4099999999999999E+000
9	2.0798901759999997E-004	1.4099999999999999E+000
10	2.5958682111999992E-004	1.4099999999999999E+000
11	3.2150418534399988E-004	1.4099999999999999E+000
12	3.9580502241279986E-004	1.4099999999999999E+000
13	4.8496602689535978E-004	1.4099999999999999E+000
14	5.9195923227443165E-004	1.4099999999999999E+000
15	7.2035107872931792E-004	1.4099999999999999E+000
16	8.7442129447518155E-004	1.4099999999999999E+000
17	1.0593055533702177E-003	1.4099999999999999E+000
18	1.2811666640442612E-003	1.4099999999999999E+000
19	1.5473999968531134E-003	1.4099999999999999E+000
20	1.8668799962237360E-003	1.4099999999999999E+000
21	2.2502559954684829E-003	1.4099999999999999E+000
22	2.7103071945621796E-003	1.4099999999999999E+000
23	3.2623686334746159E-003	1.4099999999999999E+000
24	3.9248423601695387E-003	1.4099999999999999E+000
25	4.7198108322034464E-003	1.4099999999999999E+000
26	5.6737729986441349E-003	1.4099999999999999E+000
27	6.8185275983729619E-003	1.4099999999999999E+000

Figure 4.2: Picture of the history file and some of the parameters calculated during the MESA run.

tain information about the age, temperature, and radius radius of the star in that stage of the evolution. The rest of the file shows the properties for each point or zone in the model. Because it is computationally heavy to write out these profiles, it is good to evaluate if all zones are necessary in the output file. For this project it was sufficient to only look at the surface.

The third output file is the profiles.index. Depending on how precise you have set your profile interval, MESA does not save a profile for every model. The profile number and model number are therefore not necessarily the same. The profiles.index file tells the user which model_number corresponds to which profile.

7	8	9
Teff	photosphere_L	photosphere_r
4.8300609701170688E+003	1.8141925933694377E+001	6.0910940514696899E+000
7	8	9
eps_grav	log_abs_eps_grav_dm_div_L	signed_log_eps_grav
-4.8537098487032887E-005	-1.8010276763255746E+001	0.0000000000000000E+000
1.0737936160276004E-001	9.3837433281353425E-002	0.0000000000000000E+000

Figure 4.3: Picture of one of the profiles for a random MESA run with some of the parameters for the star in that specific stage of its evolution.

1684 models.	lines hold model number, priority, and profile number.	
1	2	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	7
8	1	8
9	1	9
10	1	10
11	1	11
12	1	12
13	1	13
14	1	14
15	1	15
16	1	16
17	1	17
18	1	18
19	1	19
20	1	20

Figure 4.4: Picture of profiles.index for a random MESA run. In this run 1684 models were made, and in this case each model number corresponds to the same profile number. This is not necessarily the same for every run since it depends on the profile interval sat in the inlist_project file.

Besides these three output files, another output file is needed. This is the output file used to calculate the frequencies in another program called GYRE. This is done by implementing `write_pulse_data_with_profile = .true.` in my inlist_project (Fig. 4.5) file where all the other controls for the MESA run are listed. This produces an extra file called profile.GYRE for each corresponding profile.

GYRE

In order to interpret the asteroseismic obersavation of recent missions such as MOST (Walker et al. 2003; Matthews 2007), CoRoT (Michel et al. 2008), Kepler (Borucki et al. 2008) and TESS (Ricker et al. 2016), a stellar oscillation code which calculates the eigenfrequency spectrum from an input stellar model is needed. MESA has an asteroseismology module which is based on the Aarhus Pulsation code (`ADIPLS`; Christensen-Dalsgaard 2008), however, when various physical processes such as non-adiabaticity, rotation and magnetic field fields were taken into account, there were complications. This led to a new oscillation code, `GYRE` (Townsend and Teitler 2013). `GYRE` is written in `FORTRAN 98` and is based on Magnus Multiple Shooting (MMS) scheme to solve linearized pulsation equations. A typical `GYRE` run involves these step:

Stellar model: `GYRE` needs to read a stellar model as input file. This model can either be read from a file or built analytically. `GYRE` supports three classes of stellar models. The first model is a evolutionary model generated by a stellar evolution code (MESA), which is the one I am using. The second class is the polytropic models based on the Lane-Emden equation, and the last class is the analytical models which are based on explicit expression for the structure coefficients.

Grid calculation: When the file has been read, `GYRE` constructs a calculation grid. The grids are used for multiple shooting and eigenfunction reconstruction. The type of grid depends on the type of input model.

Eigenfrequency search: `GYRE` searches for eigenfrequencies within a user-specified frequency interval. This is done by evaluating the discriminant at different points distributed within the interval. This gives some initial guesses for the roots.

```
&controls
   ! defining interval for history file
    history_interval = 1

  ! defining interval for profile data
    profile_interval = 1
    max_num_profile_zones = 2

    max_num_profile_models = 10000


  ! starting specifications
    initial_mass = 1.41 ! in Msun units

  ! mixing length for The Sun
    mixing_length_alpha = 1.845

  ! Stop when the star has this age (age of Sun)
    max_age = 3.0d9

  ! Initial Z and Y for the Sun
    initial_z = 0.0131529
    initial_y = 0.264714

  ! control to print to GYRE
    write_pulse_info_with_profile = .true.
    pulse_info_format = 'GYRE'
    add_atmosphere_to_pulse_info = .true.
```

Figure 4.5: Picture of the inlist_project file containing the controls used to create the stellar evolution models. In this case the initial mass is set to 1.41 $M_{\odot}$, the mixing length $\alpha = 1.845$, $Y_{\text{initial}} = 0.264714$ and $Z_{\text{initial}} = 0.0131529$. Here the MESA run is set to stop at an age of 3.0 Gyr.

Eigenfunctions: Based on the initial guesses of the roots, the corresponding eigenfunctions are constructed.

This is just the basic steps in `GYRE`, for a more detailed description of the code the reader is referred to Townsend and Teitler 2013.

GYRE Output File

The output file (Fig. 4.6) created by `GYRE` contains several columns, where the first column indicates the type of mode going from l=0 to l=3. One of the columns gives the order n for each mode and one of the columns gives the frequency for the specific l mode with that specific order n.

2	3	4	5
n_pg	n_p	n_g	Re(freq)
5	5	0	0.1009016991457415E+003
6	6	0	0.1174733763594341E+003
7	7	0	0.1330912797671028E+003
8	8	0	0.1490508399575293E+003
9	9	0	0.1647757576156039E+003
10	10	0	0.1802302569609735E+003
11	11	0	0.1961369248902859E+003
12	12	0	0.2122090587432907E+003
13	13	0	0.2282271849896463E+003
14	14	0	0.2444017078757237E+003
15	15	0	0.2607514395980526E+003
16	16	0	0.2771577500664245E+003
17	17	0	0.2936066673621650E+003
18	18	0	0.3101035436729781E+003
19	19	0	0.3265965812827021E+003
20	20	0	0.3430323027410304E+003
21	21	0	0.3592771491952060E+003
-110	4	114	0.1013427599445767E+003
-109	4	113	0.1022188766707364E+003
-108	4	112	0.1031100978058877E+003
-107	4	111	0.1040209875893515E+003
-106	4	110	0.1049495160044915E+003
-105	4	109	0.1058904673275439E+003
-104	4	108	0.1068382946182472E+003
-103	5	108	0.1077797034865409E+003
-102	5	107	0.1086310420335216E+003

Figure 4.6: The GYRE output file contains a total of seven columns, which are not all displayed here. Here I have displayed columns 2-5. The first column is the l modes, second column is the n-order for p- and g-modes when they are coupled, third and fourth column are the n-order for the p- and g-modes alone, column 5 and 6 are the real and imaginary part of the frequency respectively, column 7 is the inertia of each frequency.

Chapter 5

Time Series Analysis

When doing asteroseismology, the data set one is working with is often a so-called time series. A time series is a set of measurements x = $(x_1, x_2, \ldots, x_N)$ taken at times t = $(t_1, t_2, \ldots, t_N)$. Often each measurement is labelled with a statistical weight w = $(w_1, w_2, \ldots, w_N)$ indicating the quality of each data point, which are estimated using the relative error of each data point. Usually the weight is defined $w_i = \sigma_i^{-2}$, where σ_i is the error on the measurement. In this chapter I follow the approach from Handberg 2013.

The Weighted Least Squares Method

When looking at oscillations in stars it is favorable to use Fourier transformation to transform the measurements from the time domain to the frequency domain. This is the first step when doing analysis of this kind, which in the discrete case is defined the following way:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i k n}{N}}, \qquad x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{-2\pi i k n}{N}} \tag{5.1}$$

where X_k is the Fourier transform of x_n. The power of the standard Fourier transform can then be calculated very fast using the Fast Fourier Transform (FFT) algorithm. Normally when searching for a signal with the form

$$A \sin(2\pi\nu t + \delta) = \alpha \sin(2\pi\nu t) + \beta \cos(2\pi\nu t) \tag{5.2}$$

one constructs the power spectrum $P_k = |X_k|^2$ to search for significant peaks.
One can immediately see that neither the time nor the weights are included in the expressions above, nor is there any choice of frequency for where the power spectrum is calculated. This is because it is assumed that the measurements are equally spaced in time, meaning that there are no gaps in the data, and that each measurement have equal statistical weight. Therefore when doing time series analysis on asteroseismic data, another method is needed, one that takes all the information into account.
One way to construct a power spectrum that includes all information into account is to perform what is called a least-squares fit to the time series with harmonic functions (like in Eq. 5.2), and for a given frequency ν, determine the amplitude A and phase ϕ, or equivalently the α and β coefficients. The residuals of the weighted least squares fit is given by

$$R(\nu) = \sum_{i=1}^{N} w_i \left(x_i - [\alpha \cdot \sin(2\pi\nu t_i) + \beta \cdot \cos(2\pi\nu t_i)]\right)^2 \tag{5.3}$$

and we need to find the value for α and β that minimizes Eq. 5.3. This means finding the α and β that satisfies the following equation.

$$\frac{\partial R}{\partial \alpha} = \frac{\partial R}{\partial \beta} = 0. \tag{5.4}$$

Before performing the calculation we switch to angular frequency $\omega = 2\pi\nu$ to simplify the notation. The two differentiation are given as

$$\begin{aligned}
\frac{\partial R}{\partial \alpha} &= \sum_{i=1}^{N} w_i \cdot 2\left[x_i - \alpha \sin(\omega t_i) - \beta \cos(\omega t_i)\right] \cdot \left(-\sin(\omega t_i)\right) \\
&= -2 \sum_{i=1}^{N} \left[w_i x_i \sin(\omega t_i) - \alpha w_i \sin^2(\omega t_i) - \beta w_i \sin(\omega t_i)\cos(\omega t_i)\right] \\
&= -2 \left[\sum_{i=1}^{N} w_i x_i \sin(\omega t_i) - \alpha \sum_{i=1}^{N} w_i \sin^2(\omega t_i) - \beta \sum_{i=1}^{N} w_i \sin(\omega t_i)\cos(\omega t_i)\right]
\end{aligned} \tag{5.5}$$

$$\begin{aligned}\frac{\partial R}{\partial \beta} &= \sum_{i=1}^{N} w_i \cdot 2\left[x_i - \alpha \sin(\omega t_i) - \beta \cos(\omega t_i)\right] \cdot \left(-\cos(\omega t_i)\right) \\ &= -2 \sum_{i=1}^{N} \left[w_i x_i \cos(\omega t_i) - \alpha w_i \sin(\omega t_i)\cos(\omega t_i) - \beta w_i \cos^2(\omega t_i)\right] \\ &= -2 \left[\sum_{i=1}^{N} w_i x_i \cos(\omega t_i) - \alpha \sum_{i=1}^{N} w_i \sin(\omega t_i)\cos(\omega t_i) - \beta \sum_{i=1}^{N} w_i \cos^2(\omega t_i)\right].\end{aligned} \tag{5.6}$$

This means that we can now introduce the following notation from Frandsen et al. 1995:

$$s = \sum_{i=1}^{N} w_i \cdot x_i \cdot \sin(\omega t_i) \tag{5.7}$$

$$c = \sum_{i=1}^{N} w_i \cdot x_i \cdot \cos(\omega t_i) \tag{5.8}$$

$$ss = \sum_{i=1}^{N} w_i \cdot \sin^2(\omega t_i) \tag{5.9}$$

$$cc = \sum_{i=1}^{N} w_i \cdot \cos^2(\omega t_i) \tag{5.10}$$

$$sc = \sum_{i=1}^{N} w_i \cdot \sin(\omega t_i) \cdot \cos(\omega t_i). \tag{5.11}$$

Using these definitions and setting the differentials (Eq. 5.5 and Eq. 5.6) equal to zero we obtain the following equations

$$s - \alpha \cdot ss - \beta \cdot sc = 0 \tag{5.12}$$

$$c - \alpha \cdot sc - \beta \cdot cc = 0 \tag{5.13}$$

These two equations can be written as a linear matrix of the form Ax=b:

$$\begin{bmatrix} ss & sc \\ sc & cc \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} s \\ c \end{bmatrix} \tag{5.14}$$

and can be solved using Cramer's rule, which is done by taking the determinant of the A matrix

$$\det(\mathrm{A}) = \begin{vmatrix} ss & sc \\ sc & cc \end{vmatrix} = ss \cdot cc - sc^2 \tag{5.15}$$

and the determinant of the result of the matrices when substituting the b-vector into the columns of A:

$$\det(\mathrm{B}_\alpha) = \begin{vmatrix} s & sc \\ c & cc \end{vmatrix} = s \cdot cc - c \cdot sc, \qquad \det(\mathrm{B}_\beta) = \begin{vmatrix} ss & s \\ sc & c \end{vmatrix} = ss \cdot c - sc \cdot s. \tag{5.16}$$

Cramer's rule then says that the solutions to these two equations are

$$\alpha(\nu) = \frac{\det(\mathrm{B}_\alpha)}{\det(\mathrm{A})} = \frac{s \cdot cc - c \cdot sc}{ss \cdot cc - sc^2} \tag{5.17}$$

$$\beta(\nu) = \frac{\det(\mathrm{B}_\beta)}{\det(\mathrm{A})} = \frac{c \cdot ss - s \cdot sc}{ss \cdot cc - sc^2}. \tag{5.18}$$

The power (which is the amplitude squared) can then be derived from the relation between the amplitude, α and β which can be derived from Eq. 5.2

$$P_{\mathrm{WLS}}(\nu) = \alpha(\nu)^2 + \beta(\nu)^2. \tag{5.19}$$

By calculating the $\alpha(\nu)$ and $\beta(\nu)$ for a range of frequencies ν, one can construct a power spectrum with no constraints on frequency-range and resolution (other than the ones in the time series itself).

Equivalence with The Fourier Transform

Now we have shown the results for uneven spaced data, but for this to be useful we need to show that it gives the same result using the traditional way of calculating the power spectrum (Handberg 2013). In this section we look at the result using an equal sampling $t_n = \frac{n}{N}T$, where T is the length of the timeseries. The timeeseries is evaluated in the fundamental frequencies

$$\omega_k = \frac{2\pi}{T}k \qquad , \qquad k = -N/2, \ldots, N/2. \tag{5.20}$$

To lighten the notation, we introduce the following parameter Θ which is the argument given in the sine and cosine functions

$$\Theta_n = \omega_k t_n = \frac{2\pi k n}{N}. \tag{5.21}$$

By putting $w_i = 1$ the expressions in Eq. 5.7 - Eq. 5.11 can be written as

$$s = \sum_{n=0}^{N-1} x_n \cdot \sin(\Theta_n) \tag{5.22}$$

$$c = \sum_{n=0}^{N-1} x_n \cdot \cos(\Theta_n) \tag{5.23}$$

$$ss = \sum_{n=0}^{N-1} \sin^2(\Theta_n) = N/2 \tag{5.24}$$

$$cc = \sum_{n=0}^{N-1} \cos^2(\Theta_n) = N/2 \tag{5.25}$$

$$sc = \sum_{n=0}^{N-1} \sin(\Theta_n) \cdot \cos(\Theta_n) = 0. \tag{5.26}$$

Inserting these into the expressions for α (Eq. 5.17) and β (Eq. 5.18) we obtain the following expressions

$$\alpha = \frac{s}{ss} = \frac{2}{N} \sum_{n=0}^{N-1} x_n \sin \Theta_n \quad , \quad \beta = \frac{c}{cc} = \frac{2}{N} \sum_{n=0}^{N-1} x_n \cos \Theta_n. \tag{5.27}$$

Now we can calculate the power spectrum the traditional way by taking the absolute square of the Fourier transform which is given as

$$\begin{aligned} P_k = |X_k|^2 &= \left| \sum_{n=0}^{N-1} x_n \mathrm{e}^{\frac{-2\pi i k n}{N}} \right|^2 \\ &= \left| \sum_{n=0}^{N-1} x_n \left[\cos\Theta_n - i \sin\Theta_n \right] \right|^2 \\ &= \left\{ \sum_{n=0}^{N-1} x_n \cos\Theta_n \right\}^2 + \left\{ \sum_{n=0}^{N-1} x_n \sin\Theta_n \right\}^2 \\ &= \frac{N^2}{4} [\beta^2 + \alpha^2] \\ &= \frac{N^2}{4} P_{\mathrm{WLS}}. \end{aligned} \tag{5.28}$$

From this it is clear that the result obtained using Least Square spectrum and the traditional Fourier transform is equivalent, except for a normalization constant of $\frac{N^2}{4}$ (Handberg 2013). Using this method gives some great advantages over the FFT when analyzing asteroseismic data. This algorithm is for example not restricted to timeseries which are equidistant in time, but are capable of handling timerseries with gaps. Using this method it is also possible to over-sample the spectrum to get the maximum frequency resolution, this can be seen in Fig. 5.1. This problem can however be solved by filling the data with zeros before performing FFT. Using the Least Square method gives an extra advantage which can not be achieved using FFT. This is the ability to incorporate statistical weights in the calculation of the spectrum. The effect of statistical weights has proven to b enormous when analyzing asteroseismic data. The purpose of the weights is to down-weight bad data points and give larger value to good data points. However, doing asteroseismic analysis of timeseries which have measurements that are equidistant in time where no statical weights are used, the FFT method is an extremely used tool, mostly due to the fact that it is way superior in terms of computational speed.

Lomb-Scargle Periodiogram

The expression for the power spectrum defined in section 5.1 is invariant under time shifts. This means that we can freely add any constant to t_i

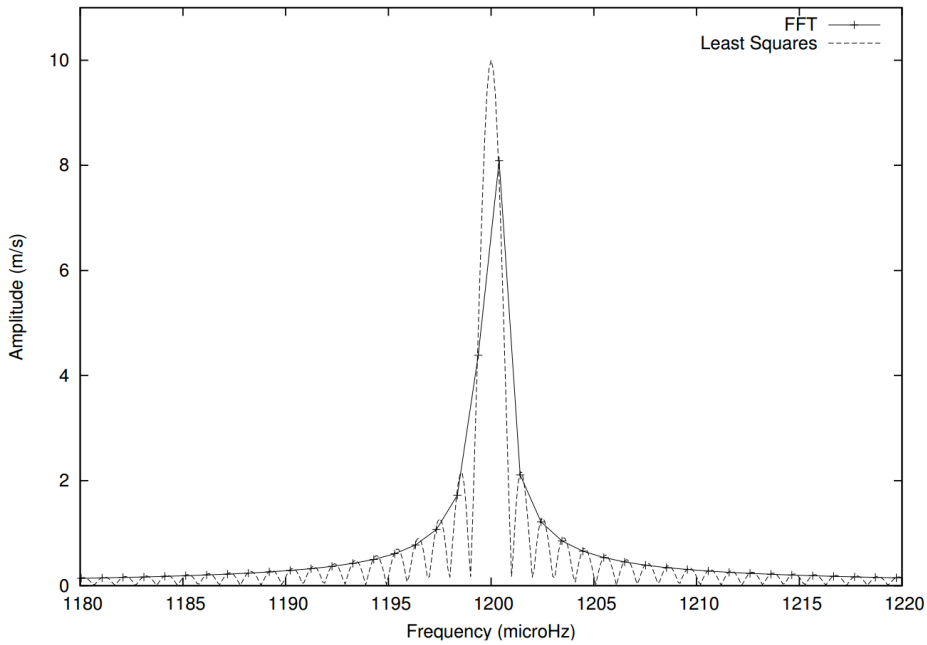


Figure 5.1: A comparison of using the FFT vs Least Square method. The figure shows the importance of oversampling. Figure is adapted from Handberg 2013.

and the power spectrum will stay the same. We are therefore allowed to introduce a time shift parameter, τ, which will shift the times ($t_i \rightarrow t_i - \tau$) in such a way that sc will be zero:

$$0 = \sum_{i=1}^{N} w_i \sin(2\pi\nu(t_i - \tau)) \cos(2\pi\nu(t_i - \tau)) \tag{5.29}$$

$$= 2 \sum_{i=1}^{N} w_i \sin(4\pi\nu(t_i - \tau)) \tag{5.30}$$

$$= 2 \sum_{i=1}^{N} w_i \{\sin(4\pi\nu t_i) \cos(4\pi\nu\tau) - \cos(4\pi\nu t_i) \sin(4\pi\nu\tau)\} \tag{5.31}$$

$$= 2\cos(4\pi\nu\tau) \sum_{i=1}^{N} w_i \sin(4\pi\nu t_i) - 2\sin(4\pi\nu\tau) \sum_{i=1}^{N} w_i \cos(4\pi\nu t_i). \tag{5.32}$$

Rearranging Eq. 5.32 gives the following expression

$$\frac{\sin(4\pi\nu\tau)}{\cos(4\pi\nu\tau)} = \frac{\sum_{i=1}^{N} w_i \sin(4\pi\nu t_i)}{\sum_{i=1}^{N} w_i \cos(4\pi\nu t_i)}. \tag{5.33}$$

Isolating τ this gives

$$\tau = \frac{1}{4\pi\nu} \arctan\left(\frac{\sum_{i=1}^{N} w_i \sin(4\pi\nu t_i)}{\sum_{i=1}^{N} w_i \cos(4\pi\nu t_i)}\right). \tag{5.34}$$

In this case, this reduces Eq. 5.19 to what is known as the Lomb-Scargle Periodiogram (Lomb 1976, Scargle 1989),

$$P_{\rm LS} \equiv \frac{1}{2\sigma^2}\left\{\frac{\left[\sum_{i=1}^{N} x_i \cos(2\pi\nu(t_i-\tau))\right]^2}{\sum_{i=1}^{N}\cos^2(2\pi\nu(t_i-\tau))} + \frac{\left[\sum_{i=1}^{N} x_i \sin(2\pi\nu(t_i-\tau))\right]^2}{\sum_{i=1}^{N}\sin^2(2\pi\nu(t_i-\tau))}\right\}, \tag{5.35}$$

where the parameter σ^2 is given as

$$\sigma^2 \equiv \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \langle x\rangle)^2. \tag{5.36}$$

This is one of the most common ways to calculate the power spectrum for unevenly sampled timeseries. This is mostly due to the fast computational implementation by Press and Rybicki 1989 which implements FFT to significantly speed up the calculation. This is done by using extirpolation (reverse interpolation) where one replaces a function value at an arbitrary point by several function values on a regular mesh, doing this in such a way that sums over the mesh are an accurate approximation to sums over the original arbitrary point (Press and Rybicki 1989). This is why one forces ones time sampling onto a regular grid. However despite implementing FFT in the calculation, the algorithm is still no way equivalent to the conventional FFT analysis.

Cross- and Auto-correlation

Most of the signals in asteroseismology are harmonic (repeating) signals (at least those in this thesis) where the Fourier Transform described above

are used as the main tool for data analysing. However, when working with non-harmonic signals (non-sinussoidal signals) the cross- and auto-correlation are useful tools.
The cross-correlation measures the similarity of two data series as a function of a displacement of one of the series relative to the other. The auto-correlation is then defined as the cross-correlation to the function with itself.
The auto-correlation is very useful to find periodic signals in the time series, that are not nessecarily harmonic signals. One can for example use the auto-correlation to detect exoplanet transits, which are periodic signals, but not very sinusoidal. The auto-correlation can also be used to search for repeating patterns in the power spectra. If the autocorrelation is zero, it means that there is no correlation between the two signals, while an autocorrelation larger than zero means that there is some sort of correlation between the two functions. The best correlation will be when there is no shift (shift = 0) which is in the start where we are matching the signal to itself without any displacement. The autocorrelation is calculated through

$$\text{autocorrelation}(k) = \frac{\sum_{i=1}^{N-k}(x_i - \bar{x}_{1\to N-k})(x_{i+k} - \bar{x}_{1+k\to N})}{N-k}. \tag{5.37}$$

The autocorrelation is the sum of the data set minus its average multiplied itself shifted some amount k minus the reduced average of the reduced data set, where N is the length of the power spectrum in number of data points.

Chapter 6

Data Preparation

Preparing Data For Asteroseismic Analysis

When doing any sort of measurement there will always be some sort of noise on top of the signal we are interested in. The noise can occur from different sources such as the instruments, the spacecraft or even the stars themselves. Doing asteroseismology, the signal we are interested in is often buried deep under the noise, meaning that a preprocessing of the raw data from the telescope is needed. The purpose of this preprocessing is to filter out any noise coming from the instruments or stars themselves that are of no interest to our research.

Another obstacle when filtering the noise is the presence of one or several extra-solar planets. The challenge with the planets is to remove the planetary signal (or planetary noise) in a way that you preserve and highlight the asteroseismic signal. If the removal of the planetary signal is to be done in such way that it is ideal, one would do a full transit-model including all the physical effects that are known to originate from the planets and that could effect the asteroseismic signal (e.g. primary and secondary eclipses, phase variations, ellipsoidal variations, transit timing variations (TTVs) and more).

This model would then be subtracted from the time series and then one could begin the asteroseismic analysis. It could be argued that this process should be iterated until an optimal fit for both problems were achieved. This is however not necessary when working with solar-like stars because the stellar oscillations constitute minor pertubations to the light curve.

However, in cases where other types of pulsating stars are involved, these types of iterations are necessary. I have managed to remove the planetary signal using the same method as in Handberg and Lund 2014 which is explained in Section 6.2.

Removing Noise and Exoplanet Transits

When studying stars which have exoplanets around them, the signal from the transits only reveal the relative size between the planet and the star, so to acquire the actual size of the planet an independent study of the host star is needed. Furthermore, it is of great importance to know the fundamental parameters of the host star (age, temperature etc.) to study the planetary system. Such important information can be obtained using asteroseismic analysis on the signal from the light curve of the host star (Handberg and Lund 2014).

When using any sort of space based instruments, a variety of strange effects and peculiarities can complicate the analysis. These effects can be things such as outliers in the time series originating from cosmic events, also called Argabrightening effects, to changes in the pointing of the telescope (which was for example the problem with Kepler) Handberg and Lund 2014.

A filter is created to remove "planetary noise", due to the fact that when observing stars with one or several planetary companions, it is not only the "noise" from the previous mentioned effects, but also the noise from the planetary companions that needs to be accounted for. The filter creates a time series data set without any transits (both known and unknown), long-term trends and instrumental effects.

The first thing the filter inspects is the long-term trends caused by both instrumental drifts and stellar activity. This is done by applying a moving median filter to the light curve, where the median of the measurements is calculated in a given time interval. The moving median is given as

$$x_{\text{long}} = \text{movingmedian}(x, \tau_{\text{long}}), \tag{6.1}$$

where τ_{long} is the time scale over which the median of the data points are found. The default values for τ_{long} depends on whether one is using short cadence or long cadence. Here, however, a τ_{long} of 2 days is used. The value of τ_{long} will be scaled up for stars which have previously been confirmed to have low frequencies (red giants). This is done due to the fact that the filter will remove any periodicities below τ_{long}, so if a star is known to have

oscillations at low frequencies, τ_{long} will be scaled up.
A moving median is chosen over a moving average because a moving median is more robust against outliers. This means that the filter is much less influenced by a planetary companion and will therefore follow the overall trend of the light curve much better.
Next step is to remove any sharp features left in the light curve that are not accounted for by τ_{long}. This is done by running another moving median filter with a short time scale through the data points after subtracting x_{long}. This constructs a new light curve as follows

$$x_{\text{short}} = \text{movingmedian}(x - x_{\text{long}}, \tau_{\text{short}}) + x_{\text{long}}, \tag{6.2}$$

where τ_{short} is the time scale chosen for the short filter. The default values for τ_{short} is also depending whether short cadence or long cadence is used. In this case a τ_{short} of 2 hours is used.
To detect any sort of sharp features, a diagnostic signal is constructed containing both filters

$$w = \frac{x_{\text{long}}}{x_{\text{short}}} - 1. \tag{6.3}$$

This makes sure that when the two filters follow each other, the diagnostic signal will be close to zero. However, if any sharp features are present, the short filter will deviate from the long filter and the diagnostic signal will deviate from zero (see Fig. 6.1).

The next step in the process is to create a final filter to apply on the light curve returning the final light curve without any "noise" and sharp features. To do this, the relative standard deviation of the signal, $\sigma_w/\langle\sigma_w\rangle$, is needed. This is calculated using the moving median absolute deviation (MAD).

$$\text{MAD} = \text{median}(|X_i - \text{median}(X)|) \tag{6.4}$$

The purpose of the standard deviation is to weigh the short filter higher when the short filter deviates a lot from the long filter. This means that the final filter can be seen as a weighted mean between the short and long filter:

$$\text{filter} = c \cdot x_{\text{short}} + (1 - c) \cdot x_{\text{long}}, \tag{6.5}$$

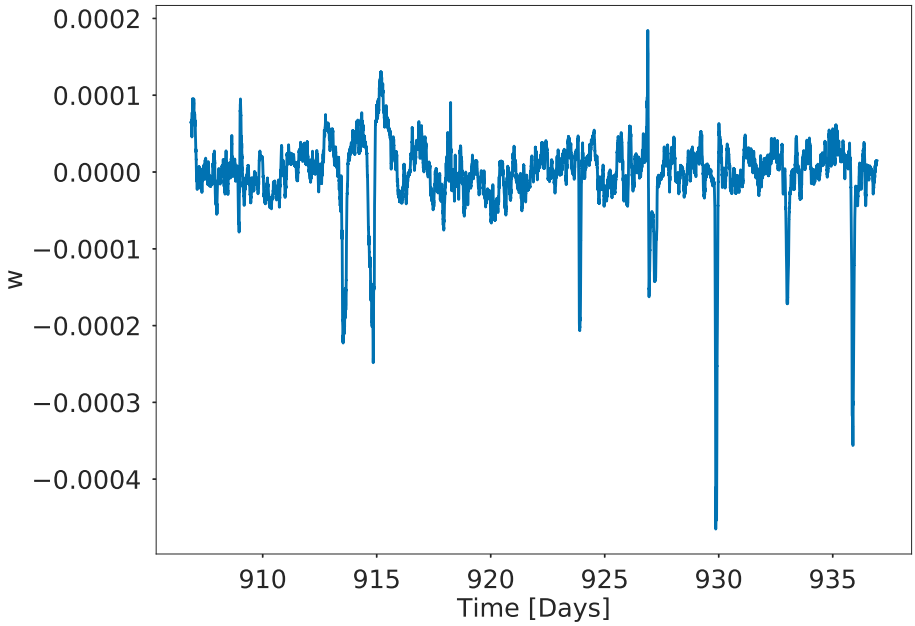


Figure 6.1: The figure shows the diagnostic signal for one of the time series for 16 Cyg A. The sharp features indicates where the short and long filter deviates from each other.

where c is the turnover function between the two filters, which returns values between 0 or 1, depending on $\sigma_w/\langle\sigma_w\rangle$. Any function with these characteristics can be used, however in this case a cumulative normal distribution function, $\Phi(x;\mu,\sigma)$, has been chosen

$$c = \Phi(\sigma_w/\langle\sigma_w\rangle;\mu_{\rm to},\sigma_{\rm to}), \tag{6.6}$$

where $\mu_{\rm to} = 5$ and $\sigma_{\rm to} = 1$ is here used as default values (same as in Handberg and Lund 2014). The value of $\mu_{\rm to}$ is when the two filters are weighted equally (c=0.5) and $\sigma_{\rm to}$ indicate the smoothness of the transition between the two filters.
Once the final filter is made using Eq. 6.5, the original light curve is divided with the filter function providing a new light curve, $x_{\rm filt}$, which has the relative flux in units of parts-per-millions

$$x_{\rm filt} = 10^6\left(\frac{x}{\rm filter} - 1\right) \tag{6.7}$$

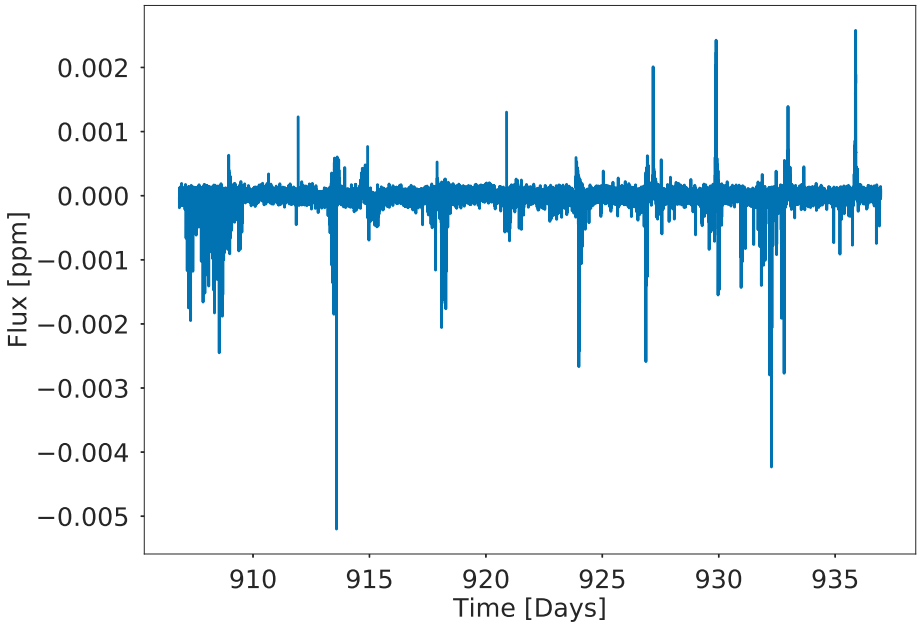


Figure 6.2: The figure shows the final filter for one of the time series for 16 Cyg A.

which can be seen in Fig. 6.2.

Lastly a sigma clipping is performed to remove the last outliers. The sigma clipping is performed with a standard deviation of $\sigma = 4$. The final light curve can be seen in Figure 6.3.

Background Noise

After removing the planetary noise we look at any excess of power in the power spectrum which consists of peaks with any periodic structure. Different analysis teams have done this in different ways (Karoff 2008; Huber et al. 2009; Hekker et al. 2010; Mathur et al. 2010; Lund et al. 2012; Handberg et al. 2017).
In Fig. 6.4 one can see the power spectrum of the Sun where some of the features contributing to the noise are indicated. These features are typically also present in solar-like stars, however at much lower resolution. From the figure it can be seen that the noise is not white (which means

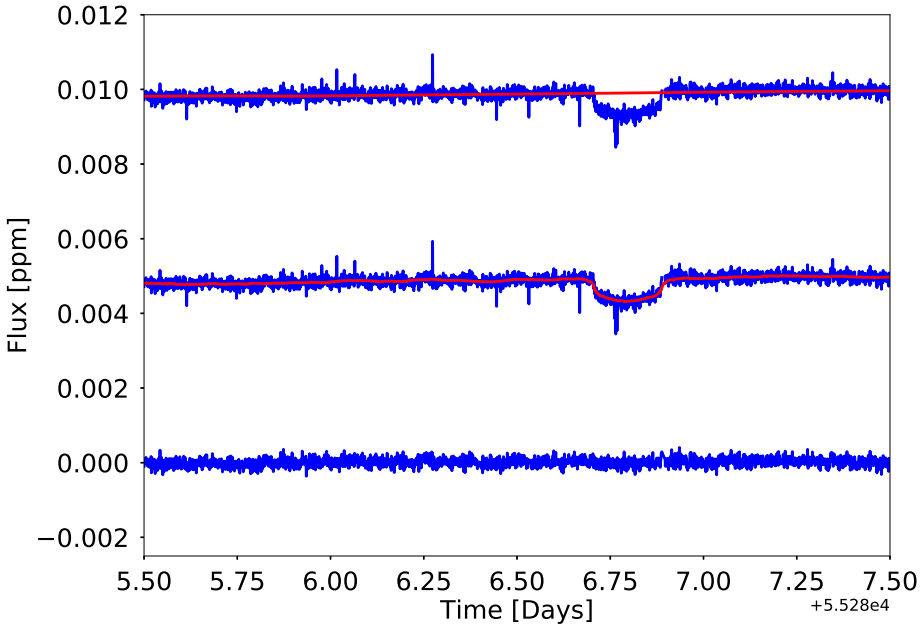


Figure 6.3: The figure shows data for a Kepler-37 where there is a clear exoplanet transit at 6.7d. The top curve shows the raw (normalized) data where no trends and outliers have been corrected for. The curve also include the long time scale median filter (the red line) which are looking at any long trends in the light curve. The middle curve is also the raw light curve however with the short time scale median filter which is used to model sharp features (e.g. transits). The curve at the bottom shows the corrected time series after transits and outliers have been removed.

it is equal at all frequencies), but is from the turbulent processes that are taking place on the surface of the star. It is features such as rotation, granulation and faculae etc. that contribute to the noise. The faculae are bright spots on the stellar surface often close to star spots. The faculae are due to changes in the opacity caused by the magnetic field, i.e. we see into the granulation cells rather than seeing the surface of the granulation cells. As the temperature is higher inside the granulation cells than on the surface, the faculae will appear brighter than their surroundings, Keller et al. 2004

On top of the noise from the turbulent processes there is also the photon-

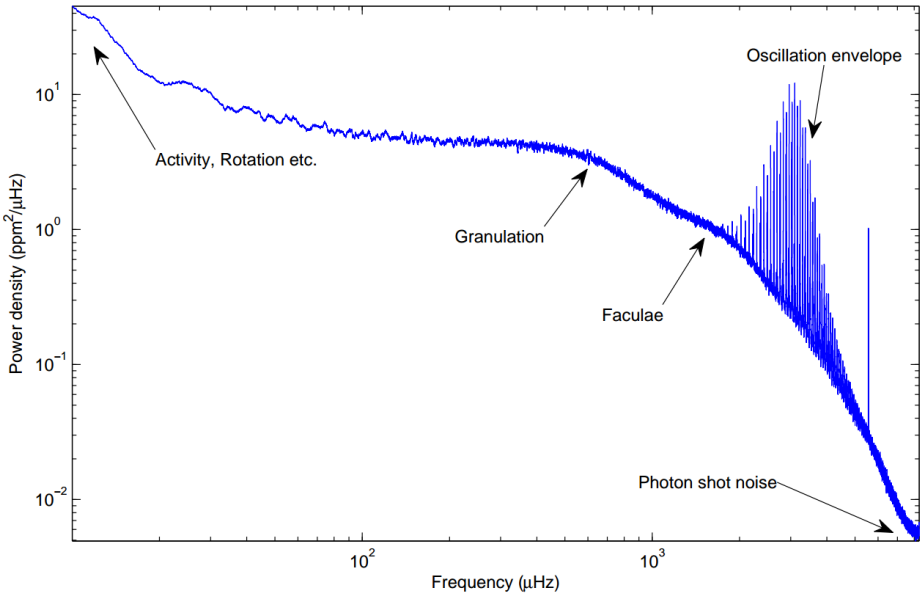


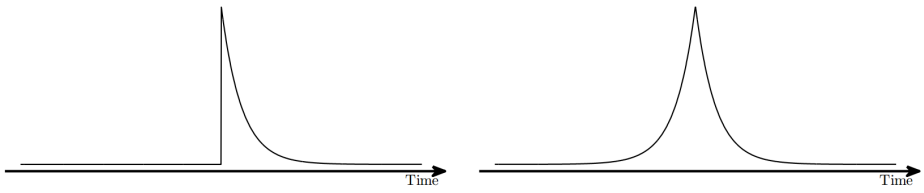
Figure 6.4: A power spectrum of the Sun consisting of 12 years of data taken from the green channel of the VIRGO SPM instrument on SoHO spacecraft. The power spectrum is smoothed with a 2 μHz boxcar filter. The figure is adapted from Handberg 2013.

noise (Poisson distributed noise that arises from the limited number of photons coming from the star), instrumental noise and then there are the oscillations themselves.

The first step in asteroseismic analysis is therefore to estimate this background noise on top of the oscillations. This sounds trivial, however, it is actually a major difficulty in these types of analysis due to the fact that there are no universal description of convection and how it affects the power spectrum noise of a star.
Here I use a sum of so-called Harvey models (Harvey 1985) plus a white noise contribution to describe the noise as a function of frequency given as

$$N(\nu) = \sum_{k=1}^{N_{\mathrm{Harvey}}} \frac{4\sigma_k^2 \tau_k}{1 + (2\pi\nu\tau_k)^{\alpha_k}} + K \tag{6.8}$$

where σ_k is the amplitude of the signal, τ is the characteristic time-scale, α_k



(a) Original Harvey model **(b)** More realistic model

Figure 6.5: Models illustrating the brightness of individual granules or faculae corresponding to Eq. 6.8 and Eq. 6.11. Figures are adapted from Handberg 2013.

is the degree of memory in the system and K is the white noise contribution. Using scaling relations, a relation for σ_k and τ_k can be found in terms of stellar properties, Karoff 2008:

$$\sigma \propto \frac{L^{0.5}}{M\ T_{\rm eff}^{0.5}} \tag{6.9}$$

and

$$\tau \propto \frac{L}{M\ T_{\rm eff}^{3.5}} \tag{6.10}$$

where L is the luminosity, M is the mass and $T_{\rm eff}$ is the effective temperature of the star. In the original model, Harvey proposed a $\alpha_k = 2$ which corresponds to the situation where the brightness of the granules rise instantly and then exponentially fall off afterwards. However, it is often seen that using $\alpha_k = 2$ does not fit the data very well. This is often overcome by letting α_k be a free parameter when doing the fitting of the background. Instead of letting α_k be free, I have chosen to use a modified version of Eq. 6.8, which is primarily motivated by fitting the Solar power spectrum Karoff 2008 and Handberg et al. 2017:

$$N(\nu) = \sum_{k=1}^{N_{\rm Harvey}} \frac{4\sigma_k^2 \tau_k}{1 + (2\pi\nu\tau_k)^2 + (2\pi\nu\tau_k)^4} + K. \tag{6.11}$$

This is a more realistic model where the brightness is rising gradually instead of instantaneous and falling of gradually afterwards (Figure 6.5b). An example of using a Harvey profile to estimate the background can be seen in Fig. 6.6

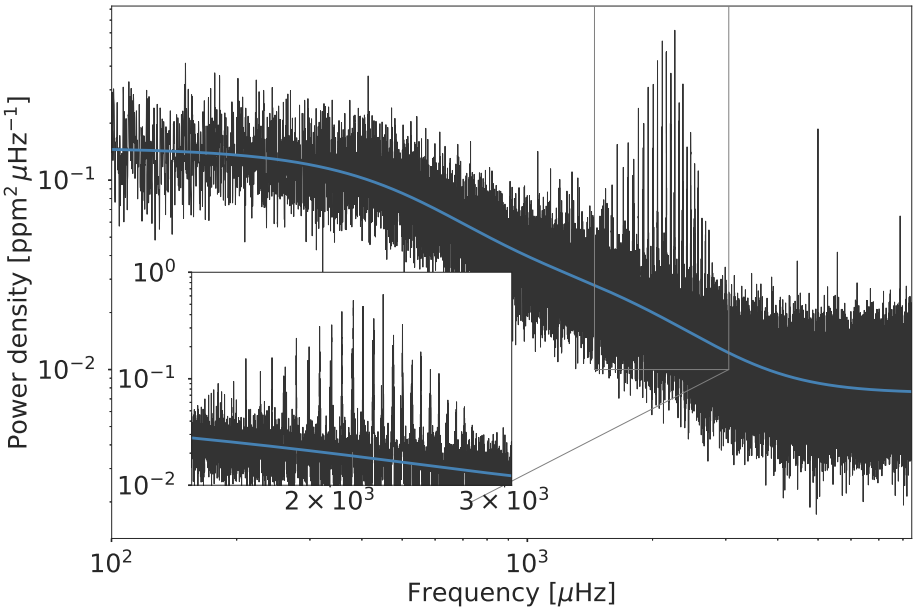


Figure 6.6: The figure shows a zoom in of the power spectrum for 16 Cyg A where the Harvey profile used to estimate the background is displayed (blue line).

Mode Identification

In a star, it is parameters such as temperature, pressure, chemical composition and so on that drives these specific oscillations modes. If these conditions change somehow, e.g. the core of the star goes from being radiation dominated to being convective, it will change the way these modes are driven. That is why it is essential in asteroseismology to identify the individual modes to extract information about the interior of the star.

Mode identification is determining the different modes based on the three numbers (l,n, m) for every frequency in the power spectrum. Mode identification for solar-like stars is relatively easy, however, for other types of stars, e.g. δ Scuti stars, it is more difficult due to the fact that not all modes are excited as in solar-like stars and there is no theory that explains why these modes are not excited. In solar-like stars, the asymptotic relation (Eq. 2.5) makes it simple to identify modes by constructing an

echelle diagram. The echelle diagram is constructed by dividing the power spectrum into parts of the large frequency separation $\Delta\nu$ and thereafter stacking these on top of each. Since the different oscillation modes occur in a specific order in the power spectrum, and this is repeated through the entire power spectrum, the oscillation modes with the same l number will group and form vertical lines in the echelle diagram (see Fig. 2.6). In Fig. 2.6 the frequency modulo $\Delta\nu$ is the rest of the frequencies after division. Comparing Fig. 2.5 and Fig. 2.6, even though it is for two different stars, the typical pattern in echelle diagrams for $l$modes can be seen: l= 2,0,3 and 1. However, if mixed modes are present in the star, this can also be seen in the echelle diagram in terms of a deviation from the vertical lines. In Fig. 6.7 is a echelle diagram based on a model of η Boo where the deviation from the vertical pattern is clear. The echelle diagram shows how the l=1 modes have been shifted from there normal position. The diagram also shows how an extra mode occurs for every mixed mode.

Finding Relavant Peaks

When creating these echelle diagrams, I first need to detect the relevant peaks in the power spectrum. This process can be done using different methods described in Handberg and Campante 2011 and Lund et al. 2017. However, in this thesis I have used the prominence (measure of height relative to the lowest contour line) of each peak in the power spectrum and given a threshold to search above. The peaks and prominences of each peak are found using the Python package scipy.signal.find_peaks and scipy.signal.peak_prominences. I use the prominences to search for the relevant peaks due to the fact that only one data set has been released from TESS for each of the stars in this thesis. Had I used Kepler data, I could have given a simple threshold to search above. This is because so many data sets are available for each Kepler star, so here one can layer the data sets and the relevant peaks will stand out from the noise (a higher signal to noise ratio compared to TESS at the moment). An example of using prominence to detect the relevant peaks can be seen in Fig. 6.8. The reason for using the prominence method is also to detect some relevant peaks that may have a smaller amplitude than a noise peak, but which still have a more significant peak.

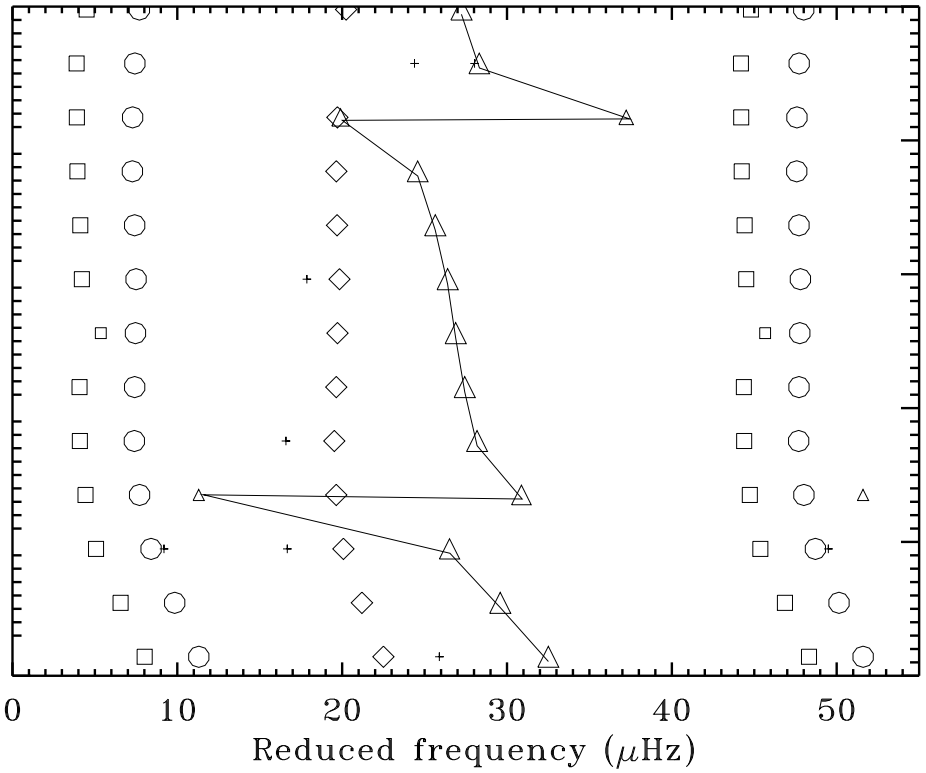


Figure 6.7: Echelle diagram for a model for the star η Boo. Circles are l=0 modes, triangles are l=1 modes, squares are l=2 modes and diamonds are l=3 modes. The diagram shows how mixed l=1 modes are shifted from the regular vertical line pattern and that extra modes are created for every mixed mode. Figure is adapted from Bedding et al. 2014.

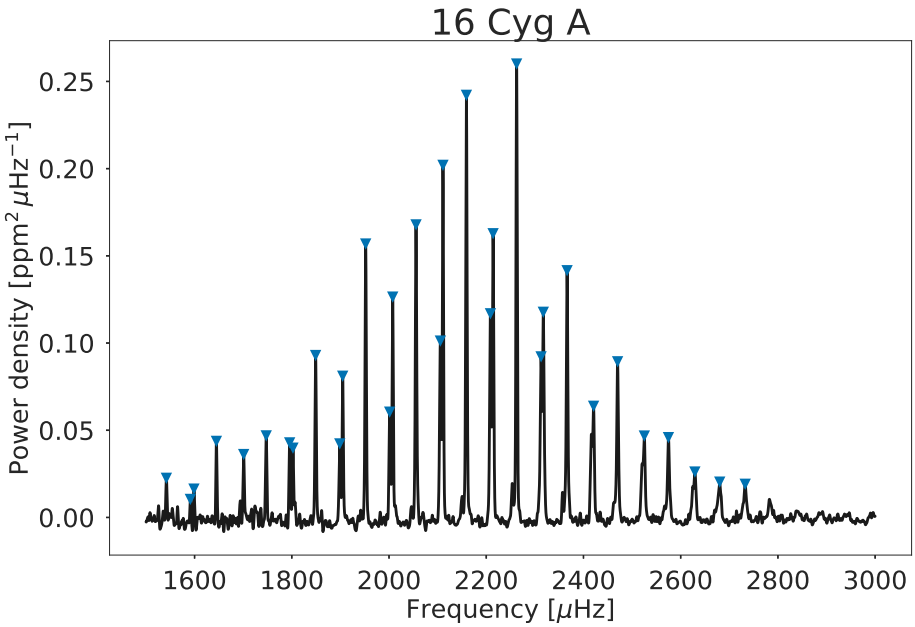


Figure 6.8: Power spectrum for the star 16 Cyg A. The blue triangles are the peaks found using the Python packages scipy.signal.find_peaks and scipy.signal.peak_prominences.

The Large Frequency Separation, $\Delta\nu$

Now that the outliers and transits have been removed and the background noise has been estimated, we can start to detect the repeating patterns that indicates solar-like oscillations, as discussed in section 2.2. We are specifically searching for the parameter called the large frequency seperation ($\Delta\nu$). To find the large frequency separation we need to search for a repeating pattern of peaks in the power spectrum. Since this signal is not necessarily a sinusoidal signal, the autocorrelation (as described in section 5.3) is an excellent tool to detect these peaks.
The autocorrelation is calculated for a specific frequency range where the oscillations are located. The autocorrelation of the power spectrum will tell us if there are any features throughout the power spectrum that are repeating themselves. In our case of solar-like oscillations there will be a series of peaks located at $\frac{1}{2}\Delta\nu$, $\Delta\nu$, $\frac{3}{2}\Delta\nu$ and so on assuming that the

$l = 1$ modes are located exactly between the $l = 0$ modes. Doing a proper detailed fitting of the autocorrelation of the power spectrum, one can also extract informations such as $\delta\nu_{02}$, rotation of the star and more. However, the extraction of $\delta\nu_{02}$ is different here, which will be discussed in section 6.7. Properly fitting the autocorrelation is done as described in Campante et al. 2010, where they create a complete model of the power spectrum and then calculate the autocorrelation of that model-spectrum. This is then fitted to the autocorrelation of the real power spectrum. However, doing a proper fit to the autocorrelation is pretty computationally intensive, due to the fact that it involves calculating the autocorrelation for each evaluation of the likelihood function.
Since it is only the large frequency separation I am interested in, I have used a more simpel model (Handberg 2013) instead of the full autocorrelation model described in Campante et al. 2010, which is defined as

$$\begin{aligned} ac(x) = a_0 \exp\left[\frac{-(x-0.5\Delta\nu^2)}{c^2}\right] + a_1 \exp\left[\frac{-(x-\Delta\nu^2)}{c^2}\right] + a_2 \exp\left[\frac{-(x-1.5\Delta\nu^2)}{c^2}\right] \\ + a_3 \exp\left[\frac{-(x-2\Delta\nu^2)}{c^2}\right] + a_4 \exp\left[\frac{-(x-2.5\Delta\nu^2)}{c^2}\right] + a_5 \exp\left[\frac{-(x-3.0\Delta\nu^2)}{c^2}\right] \\ + a_6 \exp\left[\frac{-(x-3.5\Delta\nu^2)}{c^2}\right] + k. \end{aligned} \tag{6.12}$$

The model consists of a sum of seven Gaussians which are centered at different values of $\Delta\nu$. The function can easily be fitted to the autocorrelation of the power spectrum with standard non-linear least squares fitting routines. From this, a robust value of the large frequency separation can be obtained. Example of such fit can be seen in Figure 6.9.

From investigations of a large sample of stars (both solar-like stars and red giant) there have been found a correlation between the large frequency separation, $\Delta\nu$, and the frequency of maximum amplitude, $\nu_{\max}$ (Stello et al. 2009):

$$\Delta\nu = \Delta\nu_{\odot} \cdot \left(\frac{\nu_{\max}}{\nu_{\max\odot}}\right)^{0.77}, \tag{6.13}$$

where $\Delta\nu_{\odot} = 135.1\ \mu$Hz and $\nu_{\max\odot} = 3150\ \mu$Hz. The relation can be used as a guideline as to where to search for $\Delta\nu$ or as a first guess to put into the fitting routine (Eq. 6.12), which makes the routine even more robust.

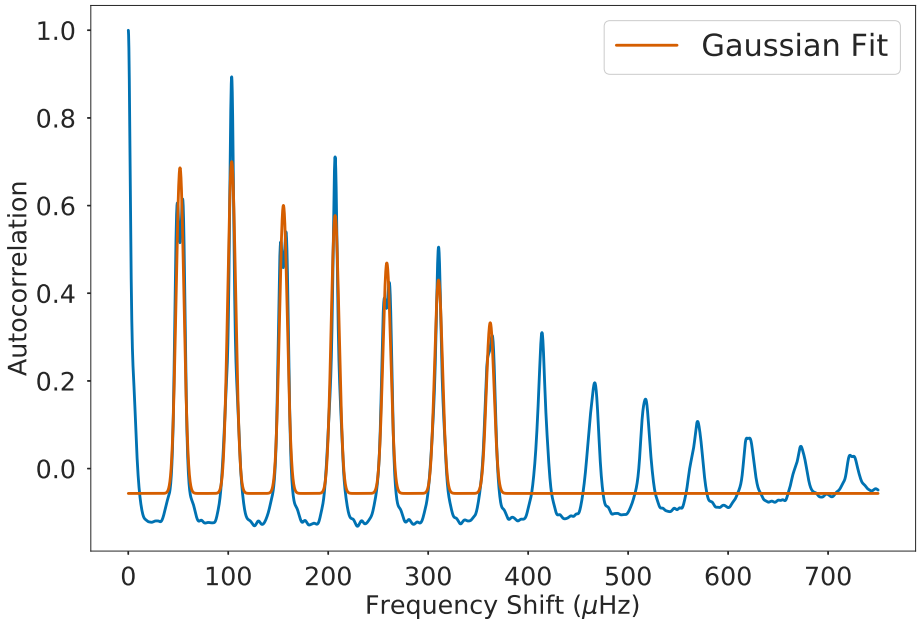


Figure 6.9: Autocorrelation for the star 16 Cyg A with a $\Delta\nu = 103.43\ \mu$Hz. The fitted model (Eq. 6.12) is shown as the red curve.

The Small Frequency Separation, $\delta\nu_{02}$

Now that the large frequency separation ($\Delta\nu$) has been estimated, we move on to estimating another important asteroseismic parameter, the small frequency separation ($\delta\nu_{02}$), which is the distance between modes of l=0 and l=2 (see Fig. 2.5). The small frequency separation in main-sequence stars is sensitive to the speed of sound near the core, which gives an indication of the evolutionary state of the star and hence its age (Chaplin and Miglio 2013). This parameter is one of the sensitive parameters in the scaling relation mentioned in section 2.4.

The parameter is estimated using a so-called matched filter that uses the asymptotic relation (Eq. 2.5 and Eq. 2.7). A matched filter is where you correlate a known signal with an unknown signal to detect the presence of the known signal in the unknown signal (Turin 1960).

I simulate frequencies for l=0 and l=2 using the asymptotic relation and matched with my observed frequencies for the specific star. The match

filter is made using a double Gaussian, to see how well the simulated frequencies match the observed frequencies. The Gaussian is defined as

$$MF = \sum_{i=0} \sum_{j=0} \exp\left(-\frac{\nu_{\mathrm{i}} - \nu_{\mathrm{j}}}{2k}\right)^2, \tag{6.14}$$

where ν_{i} are the observed frequencies from the power spectrum and ν_{j} are the frequencies simulated using the asymptotic relation, k is the variance indicating the width of the peaks. Using the information of the matched filter I can create a density plot which gives me the two parameters ε and the small frequency separation, $\delta\nu_{02}$ (see Fig. 6.10).

The shape of the matched filter, Fig. 6.10, can be explained by the way my matched filter (Eq. 6.14) works, which can be seen in Fig. 6.11. When moving the Gaussian matched filter with some value ε, there can still be a match even though it is not the right match. This is because if we move the Gaussian with some value ε corresponding to $\frac{1}{2}\Delta\nu$, the Gaussian peak for l=0 or l=2 may match with the l=1 peak from the observed frequencies giving a peak in the matched filter, which explains the pattern on the y-axis. When changing the value for $\delta\nu_{02}$, I change the distance between the l=0 and l=2 modes, which means that even if I shift the matched filter with ε, I can compensate for this by increasing $\delta\nu_{02}$. This will ensure that the simulated frequencies still match with the observed frequencies to some extend, which is why the pattern for the matched filter has this zig-zag pattern.

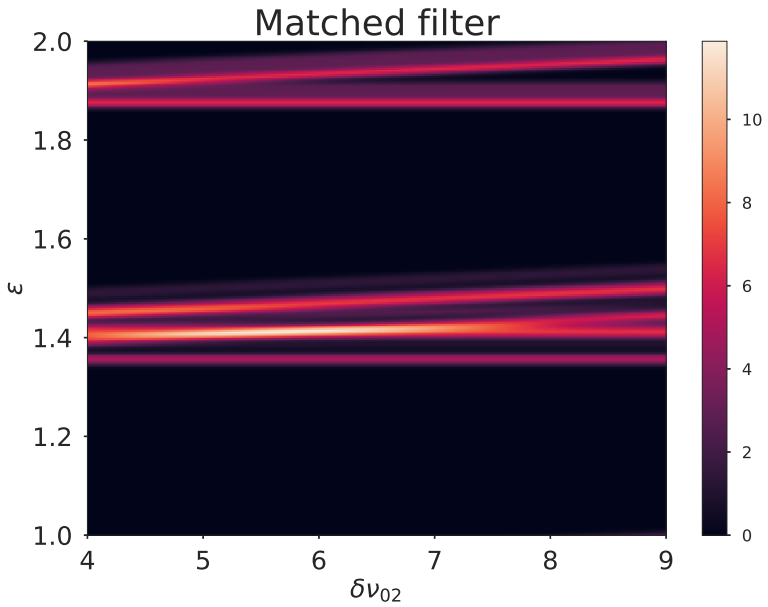


Figure 6.10: Matched filter for 16 Cyg A with a $\delta\nu_{02} = 5.53\ \mu$Hz and $\varepsilon = 1.41$.

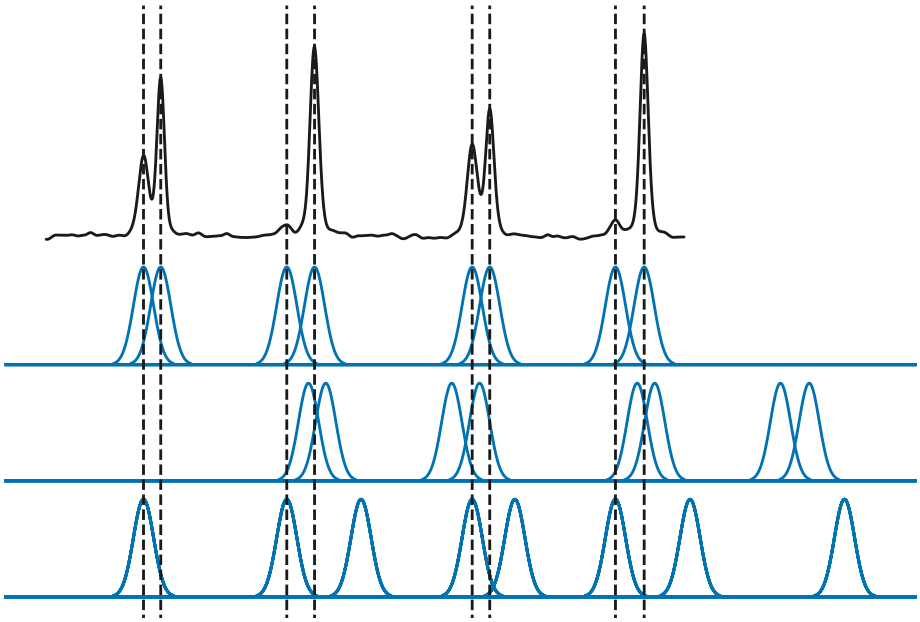


Figure 6.11: Top plot: A zoom in of the power spectrum for 16 Cyg A displaying 2 peaks for l=0,1,2,3. Second top plot: The matched filter with the best match with the data. Second bottom plot: When shifting the matched filter with ε, in this case $\frac{\Delta\nu}{2}$, there is still match between the simulated and the observed frequencies. Bottom plot: Here I have shifted the matched filter with $\frac{\Delta\nu}{2}$, but also increased $\delta\nu_{02}$ to $2 \cdot \delta\nu_{02}$, which shows a better correlation between the simulated and observed frequencies than by only increasing ε.

Chapter 7

Asteroseismic Analysis of Selected TESS Stars

In this chapter I will give a short introduction to each star and present the results from my analysis of the five TESS stars I have worked on in this thesis. In Fig. 7.1 I show the process from light curve to estimating stellar parameters of the stars. The green boxes indicate preparations of the light curve and the power spectrum before performing asteroseismic analysis. The blue boxes indicate the analysis and estimate of essential asteroseismic parameters. The yellow box indicate the calculation of uncertainties on some of the parameters. The red boxes indicates the start and the end of the analysis.

Selected TESS Stars

In this thesis I have analyzed five stars:

- β Hydri
- ν Indi
- TOI-197
- HD 212771
- HD 203949

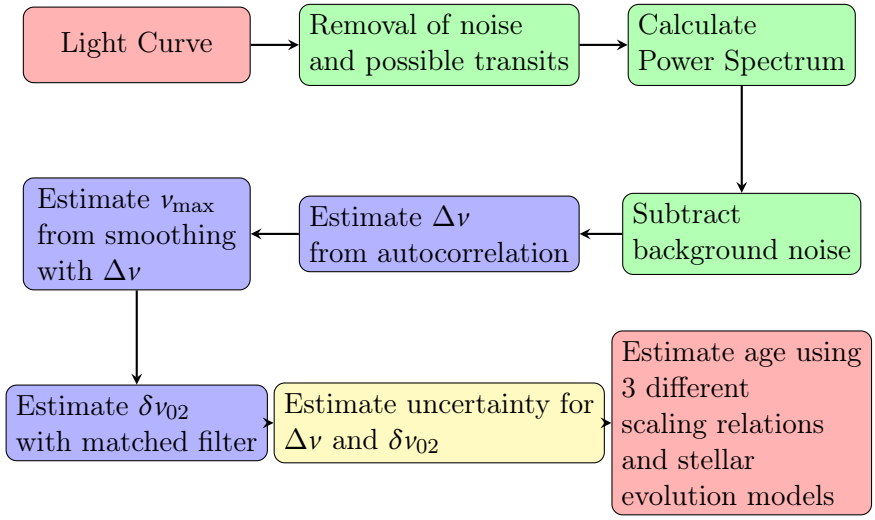


Figure 7.1: Flow chart displaying the data process from raw light curve to age estimation.

These five stars are chosen because they are some of the first data released from TESS that displays stellar oscillations and some of the first TESS stars with a confirmed exoplanet transit. Below I describe the information we know about these selected stars, since many of the stars have been observed previously.

The star β Hydri is a bright subgiant with a magnitude of $V_{\text{mag}} = 2.82$ that is slightly more massive and much more evolved than the Sun (Brandão et al. 2011). The star is one of the oldest stars in the solar galactic neighborhood, and it is frequently seen as a representation of the future of the Sun, making it a particularly interesting object to study (Dravins et al. 1993a; Dravins et al. 1993b; Dravins et al. 1993c). Frandsen 1987 and Edmonds and Cram 1995 both made unsuccessful attempts to detect stellar oscillations in β Hydri by placing upper limits on the p-mode amplitudes. However, Bedding et al. 2001 and Carrier et al. 2001 both confirmed the presence of solar-like oscillations in β Hydri and estimated the large frequency separation $\Delta\nu$ to be about $\sim 55\,\mu$Hz, but were unable to identify individual frequencies. Subsequently, after using the high-precision spectrographs HARPS and UCLES and observing the star for more than a week (Bedding et al. 2007b), both confirmed the oscillations detected

previous and identified 28 oscillations modes that included some mixed modes of spherical degree l=1.

ν Indi is a very bright metal-poor subgiant (only about 3% of solar) with a magnitude of $V_{\text{mag}} = 5.28$ (Bedding et al. 2006b). The star was first thought to be a binary, but was later confirmed to be a single star (Lambert and Mcwilliam 1986). The results from Bedding et al. 2006b indicate that ν Indi is a low mass (compared to the Sun) star and is at least 9 Gyr old.

TOI-197 is the first detection of acoustic oscillations in a star which has an orbiting exoplanet by TESS (Huber et al. 2019). TOI-197 is a bright (V magnitude of $V_{\text{mag}} = 8.2$) subgiant near the base of the red giant branch (oscillates with an average frequency $\sim 430\,\mu Hz$ which is characteristic for a star in the phase of its evolution). The planet orbiting the TOI-197 is characterized as a hot Saturn with a period of 14.3 days (close orbit) and planetary mass of $M_p = 0.191 \pm 0.017\,M_{\text{Jup}}$.

HD 212771 is an evolved known host with long period planet, which was detected using the radial velocity method. HD 2127711 is along with HD 203949 some of the first exoplanets host stars where oscillations have been detected (Campante et al. 2019).
HD 212771 is a bright subgiant (spectroscopically classified by Houk and Smith-Moore 1988) with a TESS magnitude $T_{\text{mag}} = 6.753$. However, based on its period spacing, $\Delta\Pi_1$, and its placement in the $\Delta\Pi_1$-$\Delta\nu$ diagram, it is reclassified as a low-luminosity red-giant branch star based on asteroseismology. The orbiting companion of the star is a Jovian planet with a minimum mass of $M_p \sin i = 2.3 \pm 0.4\,M_{\text{Jup}}$ and is in a 373.3 day orbit (Johnson et al. 2010).

Like HD 212771, HD 203949 is also an evolved star with TESS magnitude of $T_{\text{mag}} = 6.753$. HD 203949 is a classified horizontal branch star (Houk 1982). However, based on the parameters from asteroseismology, there is some confusion on whether the star is a red giant branch star or if it is a red clump star. Due to its low frequency it is not possible to estimate the period spacing, $\Delta\Pi_1$, which would confirm which type of star it is, based on the $\Delta\Pi_1$-$\Delta\nu$ diagram (Campante et al. 2019). The orbiting companion of the star is a relative massive planet with a minimum mass

of $M_p \sin i = 8.2 \pm 0.2\, M_{\mathrm{Jup}}$ and has a period of 184.2 days (Jones et al. 2014).

Analysis of Selected TESS Stars

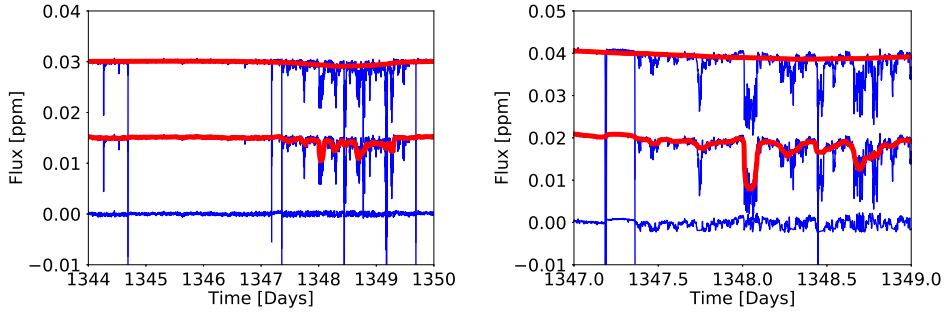
For the analysis of these stars I have written a program in `Python 3` (see appendix G) which can estimate the essential asteroseismic parameters and give an estimate of the mass, radius and age of the stars. The program is not specifically written for TESS-data and can therefore be used for other time series data. It is therefore also easy to add more data from upcoming releases from the TESS satellite. I will now go through the steps presented in Fig. 7.1 and discuss the results obtained from my code.

Removal of Noise and Possible Transits

As described in section 6.2, the first part of the analysis is to remove the different sources of noise either coming from the instruments or the stars themselves that are of no interest for our asteroseimic analysis. Beside these sources of noise, the presence of a exoplanet can also affect the signal from the host star, which is why exoplanet transits are also filtered out from the final time serie data of the star. In Fig. 7.2 the light curves for two the targets in this thesis are displayed (The rest can be seen in appendix A). Here I show the process of removing stellar, instrumental and planetary noise (Fig. 7.2e) and showing the final time serie used for the asteroseismic analysis.

The Power Spectrum and Removing The Background

After creating the time series for each star (by removing the transits etc and transforming the units of the flux) I calculate the power spectrum by performing a Fourier tranformation using Eq. 5.7 - Eq. 5.11 described in chapter 5. To optimize the signal of the power spectrum I first subtract the background, this is to minimize noise from different sources such as photons, granulation and etc. I use a modified Harvey profile (Eq. 6.11 described in section 6.3) to subtract the background, these can be seen in Fig. 7.3 (the rest can be seen in appendix B).

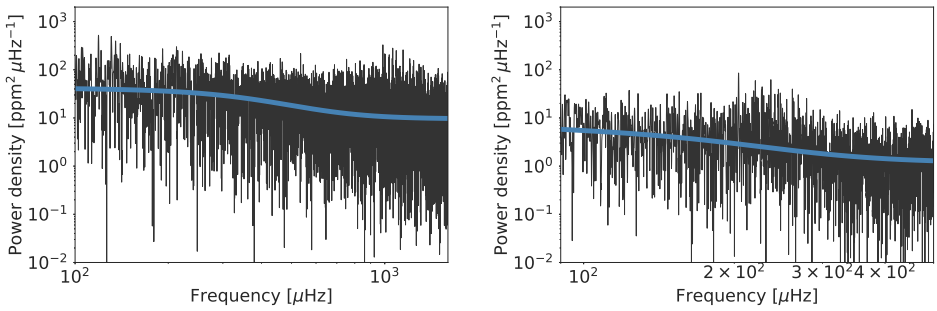


(a) ν Indi (b) HD 203949

Figure 7.2: The light curves for two of the target stars in this thesis. The plots consist of three curves. The top curve is the raw normalized curve where no trends and outliers have been removed. The curve also includes the long time scale median filter (red line) which is looking at any long trends in the light curve. The middle curve is also the raw light curve however with the short time scale median filter which is used to model sharp features (e.g. transits). The curve at the bottom shows the corrected time series after transits and outliers have been removed. One can see a clear exoplanet transit in the plot for HD 203949.

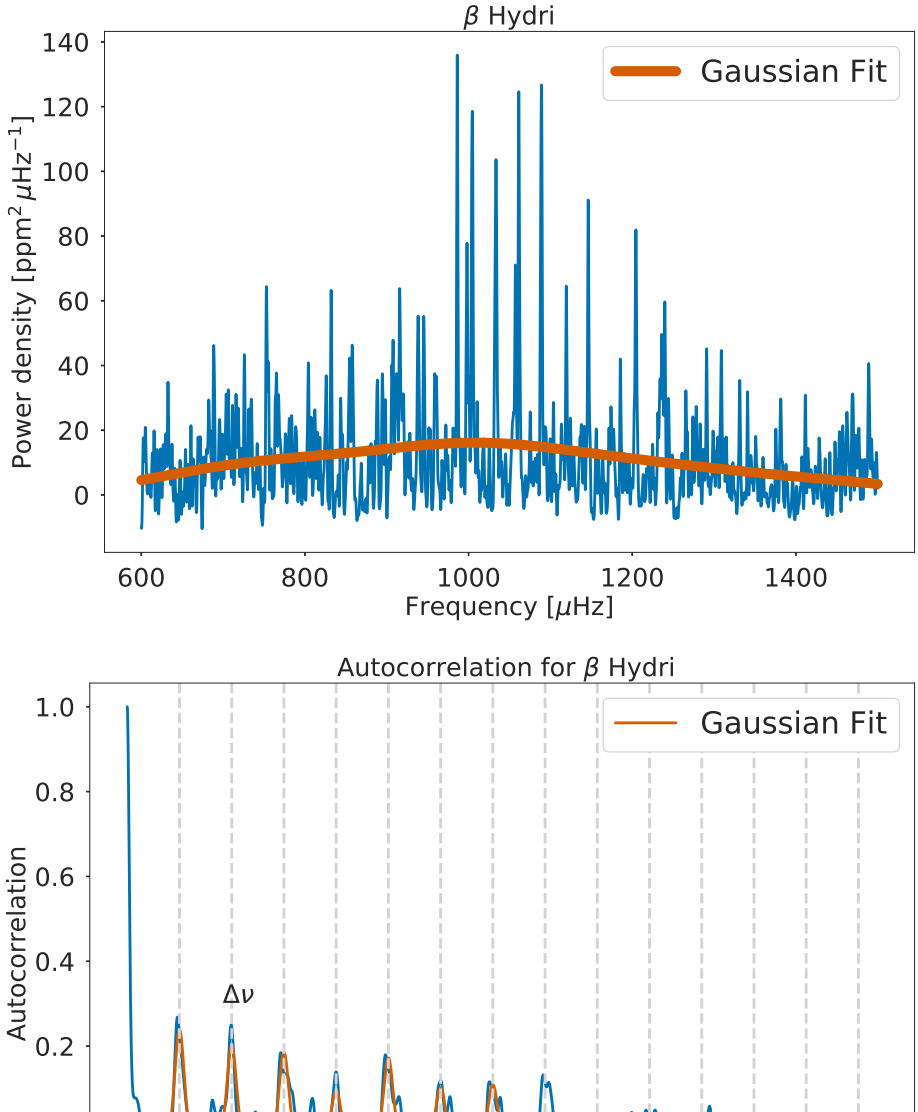
Estimating $\Delta\nu$ Using The Autocorrelation

As mentioned in section 6.6 one can use the autocorrelation (Eq. 5.37) to search for periodic signals in the power spectrum, from which it can be used to estimate the large frequency separation $\Delta\nu$. On top of using the autocorrelation I also fit the autocorrelation with a series of Gaussian functions, these are centered at different values of $\Delta\nu$ ($\frac{\Delta\nu}{2}$, $\Delta\nu$, $\frac{3\Delta\nu}{2}$ etc.). In this case where we have solar-like oscillations, the peaks in the autocorrelation will be located at $\frac{1}{2}\Delta\nu$, $\Delta\nu$, $\frac{3}{2}\Delta\nu$ and so on, so this is incorporated in the series of Gaussian functions.



(a) β Hydri (b) HD 212771

Figure 7.3: Un-smoothed power spectra for TESS stars. The blue line in each plot is the Harvey profile background subtracted from each power spectra. The oscillation envelope is not equally visible for each star.



The upper plot in Fig. 7.4 (the rest can be seen appendix C) shows the power spectrum after the background has been subtracted and has been smoothed with a Gaussian. The lower plot shows the autocorrelation as a function of the frequency. The peaks occur at frequencies where the correlation between the signal from the power spectrum and the same signal shifted k data points are largest. The best Gaussian fit to the autocorrelation will give an estimate of the large frequency separation. In the plot for the autocorrelation I have indicated the location of the large frequency separation for β Hydri and one clearly sees that the peaks occur every $\frac{1}{2}\Delta\nu$, as marked with grey vertical lines.

The value for $\Delta\nu$ can then be used to plot the echelle diagram for the star. As shown in Fig. 2.6 an echelle diagram is made by dividing the power spectrum into parts of length of $\Delta\nu$ and then stacking these on top of each other. Since the power spectrum for solar-like stars have this comb like structure, due to the asymptotic relation, the oscillation modes with the same l number will group and form vertical lines in the echelle diagram. If there are any irregularities in this vertical lines it could be due to the presence of mixed modes.

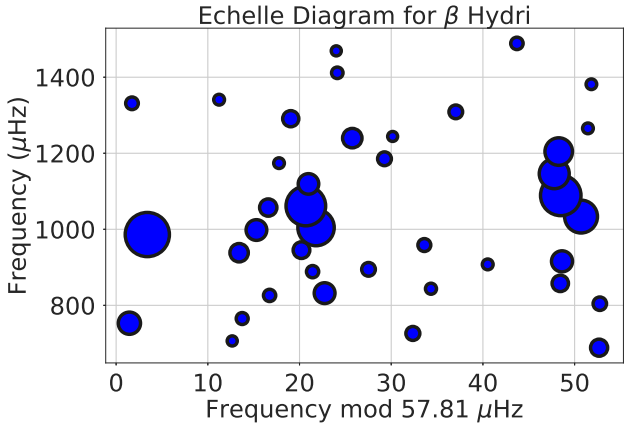


Figure 7.5: Echelle diagram for β Hydri. The sizes indicates the significance of each peak found, where the largest dots are the most significant peaks in the power spectrum. The echelle diagram is made using $\Delta\nu = 57.81$ μHz. One can clearly see an avoided crossing of the l=1 modes in this case.

Figure 7.5 shows the echelle diagram for β Hydri (the rest can be seen in appendix D). The different sizes of the data points indicate how prominent a peak is in the power spectrum. This is done due to the way the peaks are found. The sizes are made to highlight the most significant peaks, due to the fact that some of the peaks in the power spectrum are simply noise. This makes it easier to separate real peaks from noise peaks. A threshold is made to not include the smallest peaks (the peaks with very low prominence). The amount of noise peaks are due to the the low signal to noise ratio, which is a consequence of only having one light curve of each star from TESS available at the time compared to the echelle diagram for 16 Cyg A (see Fig. 2.6), where I had around ~ 25 light curves for this star, increasing the signal to noise ratio. The extra noise peaks also makes it hard to distinct "real" peaks from noise peaks, yet one can still sense the vertical lines showing l=2, l=0 and l=1 modes. This indicates that the $\Delta\nu$ value found with the autocorrelation is the optimal value. In the echelle diagram for β Hydri one can see the vertical line for the l=1 modes is not as straight as for l=2 and l=0 modes. This is due to the presence of a mixed mode creating an avoided crossing as described in section 2.3.

The large frequency separation, $\Delta\nu$, for the rest of the stars, ν Indi, TOI-197, HD 212771 and HD 203949 is also found using the autocorrelation.
It is not always the best fit to the autocorrelation that gives the best estimate for $\Delta\nu$ as seen in Fig. 7.6. The two figures on the right are the autocorrelation and echelle diagram for TOI-197 using a wrong estimated $\Delta\nu$. One can see that the autocorrelation finding the wrong $\Delta\nu$ (Fig. 7.6b) fits several of the peaks compared to the autocorrelation finding the correct $\Delta\nu$ only fits the peak corresponding to the value of $\Delta\nu$. However, if strictly looking at the power spectrum for TOI-197 (see Fig. 7.7) one can see that $\Delta\nu = 28.85$ μHz must be the correct large frequency separation, since the large frequency separation is defined as the distance between two modes of same l value. In Fig. 7.7 I have made a line between two l=0 modes to show that $\Delta\nu = 28.85$ μHz must be the correct value for $\Delta\nu$ and to show that $\Delta\nu = 14.43$ μHz is only $\frac{1}{2}\Delta\nu$. In the echelle diagram for TOI-197 one can also see the clear signature of mixed modes indicating that the star has left the main sequence. In Table 7.1 an overview of $\Delta\nu$ for all the stars can be seen.

Star	$\Delta\nu_{\text{analysis}}$ [μHz]	$\Delta\nu_{\text{litterature}}$ [μHz]	References
β Hydri	57.81 ± 0.15	57.24 ± 0.16	Bedding et al. 2007b
ν Indi	24.86 ± 0.06	24.25 ± 0.25	Bedding et al. 2006b
TOI-197	28.85 ± 0.17	28.94 ± 0.15	Huber et al. 2019
HD 212771	16.47 ± 0.04	16.25 ± 0.19	Campante et al. 2019
HD 203949	4.18 ± 0.14	4.10 ± 0.14	Campante et al. 2019

Table 7.1: Table with $\Delta\nu$ values found using autocorrelation. The uncertainties for $\Delta\nu$ are found using the frequencies of the l=0 modes in the power spectrum and plotting these as function of the order n. The uncertainty is found utilizing a 95% confidence interval, take the difference between the upper and lower boundary and divide by 4 because there is 4σ between them.

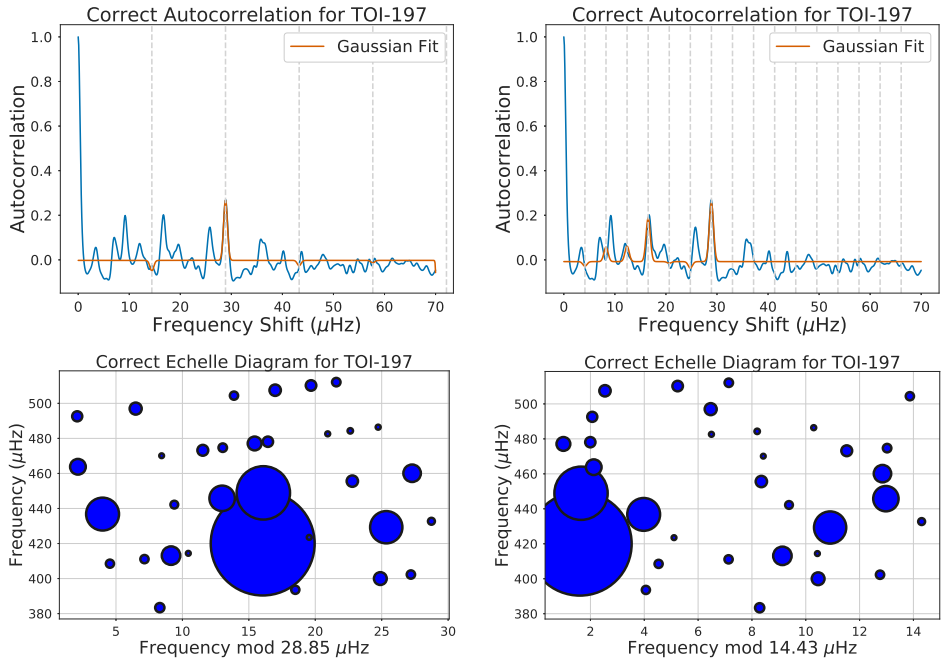


Figure 7.6: Left: Shows the autocorrelation and the echelle diagram for TOI-197 using the correct value for $\Delta\nu$. Right:Shows the autocorrelation and the echelle diagram for TOI-197 using the wrong value for $\Delta\nu$. The correction can be seen in the order for the echelle diagram where we have the typical have l=2,0 whereas using the wrong $\Delta\nu$ gives the order l=0,2.

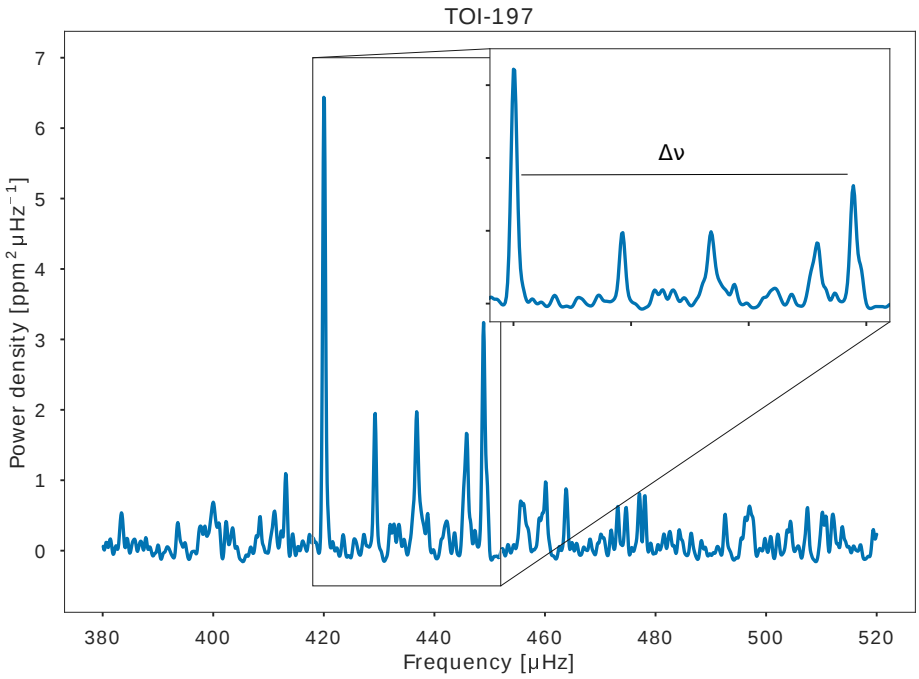


Figure 7.7: Power spectrum for TOI-197. In the zoom-in the distance $\Delta\nu$ is drawn to illustrate that the correct $\Delta\nu$ must be $\Delta\nu = 28.85$ μHz and not $\Delta\nu = 14.43$ μHz.

Estimating $\nu_{\max}$

To find the amplitude of maximum power, $\nu_{\max}$ I smooth the power spectrum using a Gaussian (using the python packages Gaussian1DKernel and convolve from astropy.convolution) where I set the standard deviation to $4 \cdot \Delta\nu$. A result for $\nu_{\max}$, for ν Indi, can be seen in Fig. 7.8 (the rest can be seen in appendix C).

Estimating δ_{02}

As mentioned in section 6.7 the small frequency separation $\delta\nu_{02}$, in main sequence stars, is sensitive to the sound speed near the core, which gives an indication of the evolutionary state of the star, and hence its age. This is because the distance between l=0 and l=2 modes decreases on the main sequence due to the burning of hydrogen in the core and thereby the de-

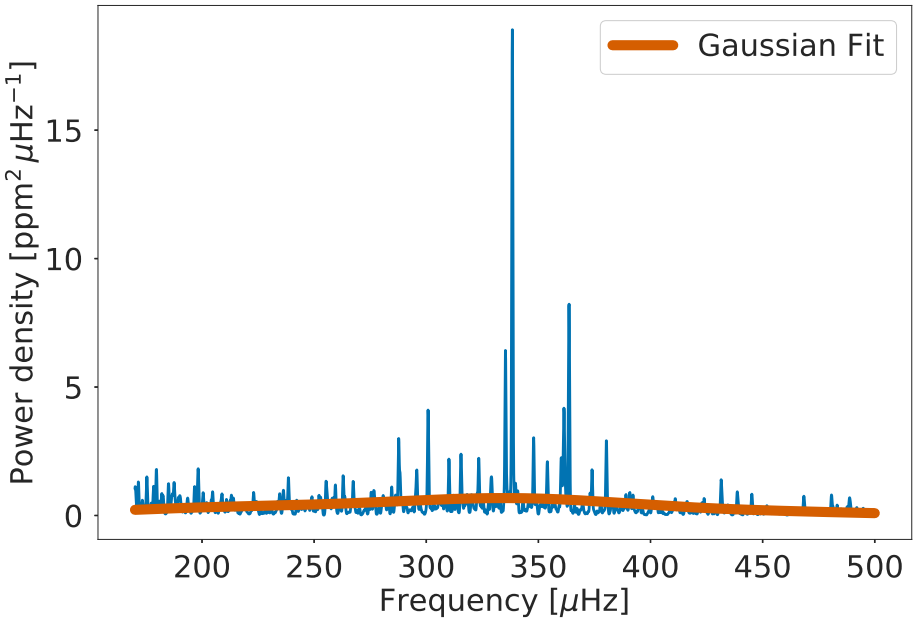


Figure 7.8: Power spectrum for ν Indi. The red line is the Gaussian used to estimate $\nu_{\max}$. The Gaussian gives a $\nu_{\max} = 334.50$ μHz.

crease of hydrogen. The adiabatic sound speed slowly decreases due to the rise in mean molecular weight as a product of the hydrogen to helium conversion. The adiabatic sound speed is inverse proportional to the mean molecular weight $v^2 = \frac{T}{\mu}$, where T is the absolute temperature and μ is the mean molecular weight. A rise in the opacity also means that the energy transfer from the core of the star to the surface gets slowed down, resulting in longer time on the main sequence. This also indicates that two stars with the same mass but with a different metallicity will spend a different amount of time on the main sequence, with the high metallicity star spending the most time there.

However, the stars used in this thesis are mainly subgiants, a red giant branch star and a single red giant branch/red clump star. Whether the distance between l=0 and l=2 decreases is unknown in these types of stars, meaning that $\delta\nu_{02}$ is not a good indicator for the age. Furthermore, the

l=2 modes are strongly mixed when the star is this evolved. However, if the star is in the early stages of subgiant phase or in a phase where the l=2 modes are only weakly mixed with the gravity modes in the center of the star, $\delta\nu_{02}$ may still give a relatively good estimate of the age of the star. If the l=2 modes are strongly mixed, the use of the period spacing ($\Delta\Pi_1$) can give a better estimate of the age. In Fig. 7.9 one can see that the use of δ_{02} as an indication for age decreases as the star gets more evolved ($\Delta\nu$ decreases and thereby also δ_{02}), whereas for stars on the main sequence, such as the Sun, which have a higher $\Delta\nu$ and therefor δ_{02}, the small frequency separation is a good indicator for the age.

The small frequency separation is found using a matched filter as described in section 6.7. The way the matched filter works is that I produce simulated frequencies using the asymptotic relation (Eq. 2.5) and then try to match these with the observed frequencies which is done using Eq. 6.14. In Fig. 7.10 the matched filter for some of the stars are plotted where the match is indicated by the intensity on the white color (rest can be seen in appendix E).

The shape of the matched filter is described in section 6.7. The matched filter for HD 212771 and β Hydri can be seen in Fig. 7.10. The matched filter for β Hydri deviates from the standard zig-zag pattern. This is due to the range sat for $\delta\nu_{02}$ which is only 1 μHz, meaning that you cannot see the larger structure. This is because papers such as White et al. 2011 have given an estimate of these parameters, giving me an idea in which range to search for. The values for $\nu_{\max}$ and $\delta\nu_{02}$ can be seen in Table 7.2

To check the values estimated for $\Delta\nu$ and $\nu_{\max}$ I test them using the relation (Eq. 6.13) from Stello et al. 2009 which can be seen in Fig. 7.11

Estimating Age

Determining the exact age of stars is a part of astronomy that is still quite difficult. Here I present the results and methods used to estimate the mass, radius and age of the stars investigated in this thesis. The age for each star is estimated using two different methods. The first one uses three different scaling relations and the other one use stellar evolution models based on the mass estimated from the scaling relation.

The three scaling relations used are described in section 2.4. The first

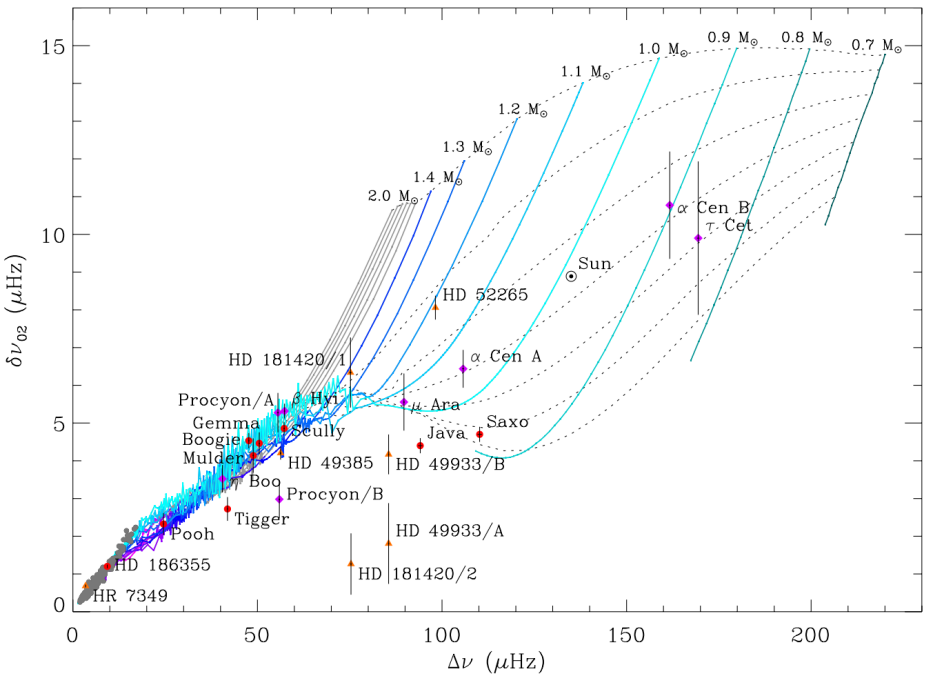


Figure 7.9: C-D diagram with models with near-solar metallicity ($Z_0 = 0.017$), the tracks increase by $0.1\,M_\odot$ from $0.7\,M_\odot$ to $2.0\,M_\odot$ as labeled. The dashed black lines are isochrones, increasing by 2 Gyr from 0 Gyr (zero age main sequence (ZAMS)) at the top to 12 Gyr at the bottom. The stars, as labeled, are from CoRoT (orange triangles), Kepler (red circles) or from the ground (purple diamonds). The gray circles are Kepler red giants (very evolved stars) from Huber et al. 2010. The figure is adapted from White et al. 2011.

scaling relation (Eq. 2.15 and Eq. 2.16) only gives an estimate of the mass and radius, and is the scaling relation that is generally used in other papers. In this thesis I test two new scaling relations derived by Bellinger 2019a and Bellinger 2019b for main sequence and red giant branch stars respectively. These two new scaling relations also gives an estimate of the age besides an estimate for the mass and radius but are not thoroughly tested. The results from the scaling relations can be seen in Table 7.3

From the results it can be seen that the mass and radius is consistent with the reference value for four of the stars, and the same with the age,

Star	$\nu_{\max}$ [μHz]	$\delta\nu_{02}$ [μHz]
β Hydri	1008.0 ± 10.1	5.00 ± 1.01
ν Indi	334.5 ± 3.3	5.18 ± 1.20
TOI-197	435.8 ± 4.4	7.43 ± 0.30
HD212771	225.3 ± 2.3	2.41 ± 0.17
HD 203949	32.4 ± 0.3	0.90 ± 0.70

Table 7.2: Table with values for $\nu_{\max}$ and $\delta\nu_{02}$ for all 5 stars. The uncertainties for $\nu_{\max}$ are found using the typical uncertainty of 1%. For $\delta\nu_{02}$ the uncertainty is found by identifying the modes for l=0 and l=2 using the asymptotic relation, finding the variance and from that the uncertainty. The uncertainty for TOI-197 is found using the typical uncertainty of 4%, because the amount of l=2 modes were to few for this star.

Star	β Hydri	ν Indi	TOI-197	HD212771	HD 203949
$M_{\Delta\nu\text{-scaling}}$ [$M_\odot$]	1.05±0.05	0.97±0.04	1.11±0.05	1.44±0.06	0.90±0.04
$M_{\text{SR for MS}}$ [$M_\odot$]	1.026±0.033	0.801±0.026	1.14±0.04	1.32±0.04	1.37±0.04
$M_{\text{SR for RGB}}$ [$M_\odot$]	1.05±0.05	1.05±0.05	1.13±0.05	1.46±0.07	0.87±0.04
M_{ref} [$M_\odot$]	1.04	0.847±0.043	1.212±0.074	1.42 ś 0.07	1.23 ±0.15 / 1.00±0.16
$R_{\Delta\nu\text{-scaling}}$ [$R_\odot$]	1.792±0.027	3.06±0.05	2.90±0.04	4.59±0.07	9.82±0.15
$R_{\text{SR for MS}}$ [$R_\odot$]	1.783±0.020	2.907±0.033	3.000±0.034	4.61±0.05	12.00±0.14
$R_{\text{SR for RGB}}$ [$R_\odot$]	1.789±0.028	3.14±0.05	2.91±0.05	4.61±0.07	9.63±0.15
R_{ref} [$R_\odot$]	1.786	2.97±0.05	2.943±0.064	4.61 ś 0.09	10.93±0.54 / 10.34±0.55
$\tau_{\text{SR for MS}}$ [Gyr]	7.82±0.95	11.3±1.4	3.30±0.40	6.79±0.83	42.5±5.2
$\tau_{\text{SR for RGB}}$ [Gyr]	2.84±0.46	4.28±0.69	5.64±0.91	2.84±0.51	14.8±2.4
τ_{ref} [Gyr]	7.297	11.4±2.4	4.9 ± 1.1	2.90 ś 0.47	6.45±2.79 / 7.29±3.06

Table 7.3: Table with the estimated values for mass, radius and age for all stars. The reference values for β Hydri are from Brandão et al. 2011, ν Indi from Bedding et al. 2006b, TOI-197 from Huber et al. 2019 and HD 212771 and HD 203949 from Campante et al. 2019. The two values for each parameter for HD 203949 is provided assuming that the star is either a red giant branch or a clump star. The uncertainty from the calculated values in this thesis are estimated based on typical uncertainties for each parameter used in the scaling relations.

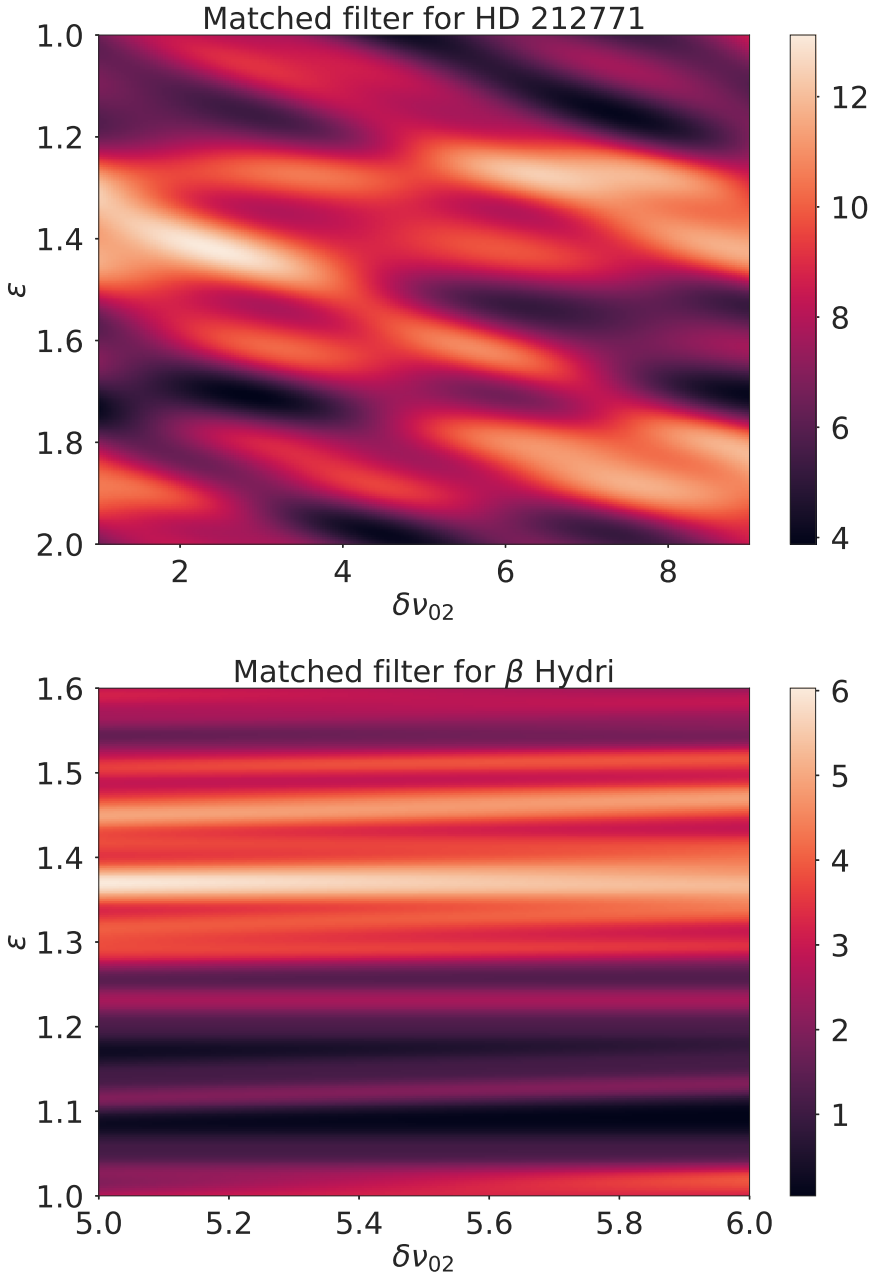


Figure 7.10: Top: Density plot of matched filter for HD 212771 estimating a $\delta\nu02 = 2.41\,\mu$Hz and $\varepsilon = 1.42$. Bottom: Density plot of matched filter for β Hydri estimating a $\delta\nu02 = 5.00\,\mu$Hz and $\varepsilon = 1.37$.

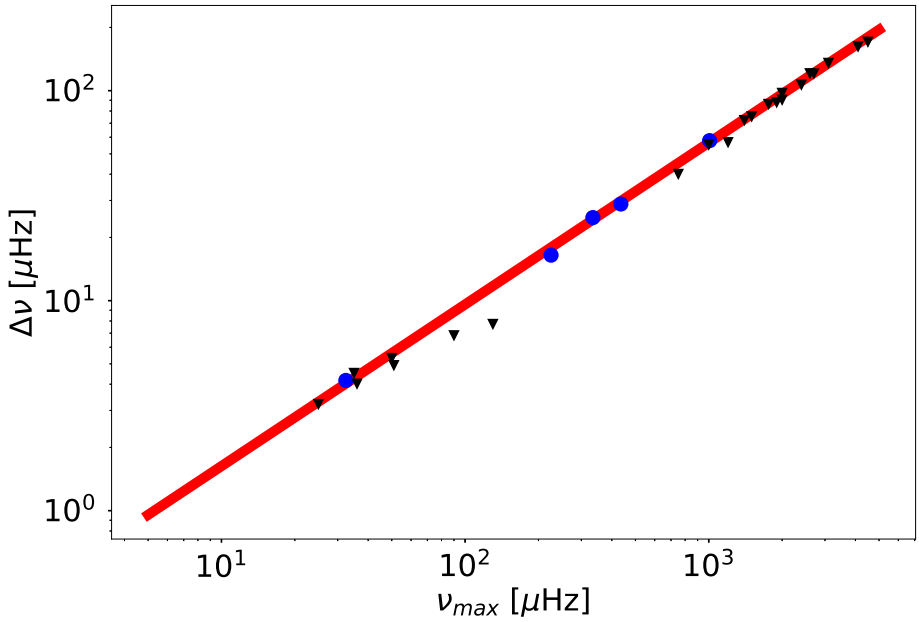


Figure 7.11: Observed $\Delta\nu$ vs $\nu_{\max}$ for the stars from Stello et al. 2009 (black triangles). The stars analyzed in this thesis are marked with a blue circle. The solid red line is the power-law fit (Eq. 6.13).

where one of the results seem to fit the reference value relatively well. The star HD 203949 seems to behave a bit strange compared to the other stars, which can be seen as an inconsistency in the mass, radius and age of the star. The two different values for mass, radius and age are because in Campante et al. 2019 there is a confusion on whether the star is a red clump star of a red giant branch star. Red clump stars are He-core burning stars that are located in the $\Delta\Pi_1$-$\Delta\nu$ diagram around 300 s and 4.1 μHz (Mosser et al. 2012). Based on the $\Delta\nu$ alone it is a red clump star, however, due to the low frequency resolution of the power spectrum, it is not possible to estimate a value for $\Delta\Pi_1$. This prevents a definitive classification of the evolutionary state of the star based on the $\Delta\Pi_1$-$\Delta\nu$ diagram due to the underlying degeneracy for $\Delta\nu \lesssim 10$ μHz (Mosser et al. 2014).

It can also be seen that using the scaling relations for main-sequence

stars on β Hydri and ν Indi leads to a relatively good estimate for the age compared to the reference value even though the two stars are in their subgiant phase. This indicates that the scaling relation may extend further out than only for stars on the main sequence. It can also be seen that the scaling relation for giants seems to give a pretty good estimate for HD 212771, which is an evolved star.

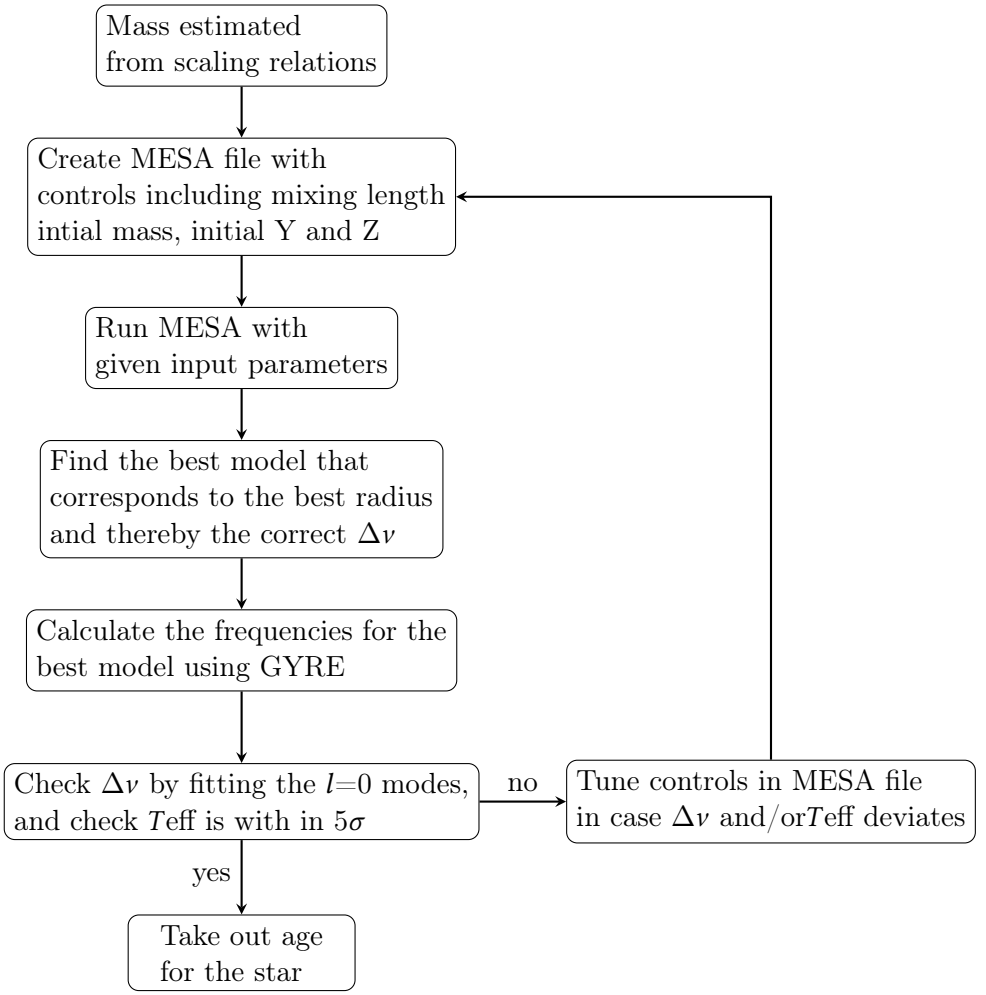


Figure 7.12: Flow chart displaying the process from estimating a mass to estimating an age using stellar evolution models.

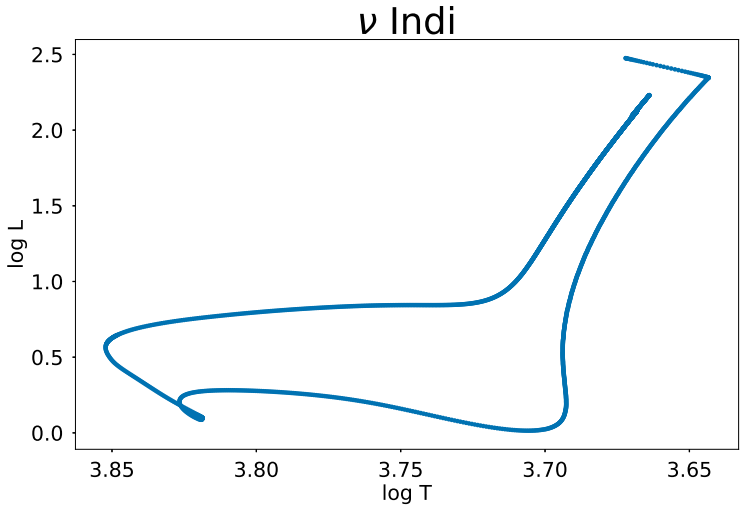


Figure 7.13: Stellar evolution model for ν Indi from pre-main sequence to a pre-determined age using MESA. Mass for the star is 0.94 $M_{\odot}$, α=1.845, Y_{initial}= 0.247261 and Z_{initial} = 0.000686518.

Create MESA File Controls For The Run

Besides estimating the age using scaling relations I also estimated the age using stellar evolution models. This is done using the stellar evolution code `MESA` described in section 4.1. The process from estimating the mass using scaling relations to estimating a mass using the models can be seen in Fig. 7.12. The input parameters for these stellar evolution models are the initial mass and metallicity. The mass have been estimated using scaling relations. Here I take the mean of the mass for each star as initial mass value for the MESA run. The mixing length is set to $\alpha = 1.845$ for these calculations, and the values for initial Y and initial Z are calculated based on the metallicity (Fig. 4.5). An example of the MESA run can be seen for ν indi in Fig. 7.13 (the rest can be seen in appendix F).

In Fig. 7.13 every model for the MESA run is plotted, from which I have to find the best profile. The best profile can be found by finding the profile that matches the estimated value for $\Delta\nu$ best. From Kjeldsen and Bedding 1995 I know that there is a correlation between the large frequency separation, $\Delta\nu$, and the density, ρ (see Eq. 2.11), and the density is given

as $\rho = M/R^3$. This means, if I find the radius that fit the estimated radius for the star from the scaling relations best, I find the best match for $\Delta\nu$. The estimated radius used for comparison are found taking the mean of the three values found using the scaling relations.

When the model with the best match for the radius is found, I calculate the frequencies of the model using the code GYRE (described in section 4.2) which finds the frequencies for l=0,1,2,3 modes. I then use the frequencies for the radial modes (l=0 modes) and fit these to check if the $\Delta\nu$ found matches the one I estimated using the my analysis (see Fig. 7.14). If the found $\Delta\nu$ match (with only a small deviation) then I take the out the age and radius of the chosen model to match with the results and reference. The mass, radius and age for each star can be seen in Table 7.4.

The deviation between the ages found using the stellar models and the reference values can be caused by different parameters. The first thing is the mass, we know that if a star has a higher mass it tends to spend less time on the main sequence giving it a shorter age compared to stars with lower masses that spend longer time on the main-sequence. This trend is also seen for stars in Table 7.4, where the stars that had a higher initial mass in my models seems to give an age that is shorter than the reference value and opposite if the initial mass is lower than the reference values. The only star that differs from this trend is HD 203949 in the case that it is red clump star. Here the mass of the star is $M = 1.00 \pm 0.16 M_\odot$ which is lower than the $M = 1.047\, M_\odot$ I have estimated. This leads to an estimate of the age of the star of $\tau = 7.29 \pm 3.06$ Gyr , which is lower than the $\tau = 10.958$ Gyr my model estimates. This result seems quite short for a star with the same mass of the Sun to go all the way up the giant branch, start its helium burning in the core, to then become a red clump.
To check if the result made sense I created a stellar evolution model (Fig. 7.15) using an initial mass of $M = 1.00 M_\odot$ (the mass for HD 203949 in case it is a red clump star, Campante et al. 2019). Then I tried to find the model with the best match to the radius of HD203949 if it was a red clump star ($R = 10.34\, R_\odot$, Campante et al. 2019). This led to a radius of $R = 10.34\, R_\odot$, so exactly the same radius as in Campante et al. 2019, this gives an estimate of the age of the star to be around $\sim$ 13.01 Gyr, which is closer to the result estimated using my analysis and models, taking the

Star	β Hydri	ν Indi	TOI-197	HD212771	HD 203949
Mass [$M_\odot$]	1.042	0.940	1.130	1.410	1.047
Radius [$R_\odot$]	1.789	3.038	2.938	4.603	10.487
τ [Gyr]	6.150	7.071	6.453	2.910	10.958
$M_{\rm ref}$ [$M_\odot$]	1.04	0.847±0.043	1.212±0.074	1.42 ś 0.07	1.23 ±0.15 / 1.00±0.16
$R_{\rm ref}$ [$R_\odot$]	1.786	2.97±0.05	2.943±0.064	4.61 ś 0.09	10.93±0.54 / 10.34±0.55
$\tau_{\rm ref}$ [Gyr]	7.297	11.4±2.4	4.9 ± 1.1	2.90 ś 0.47	6.45±2.79 / 7.29±3.06
$\Delta\nu_{\rm own\ code}$	57.81 ± 0.15	24.86 ± 0.06	28.85 ± 0.17	16.47 ± 0.04	4.18 ± 0.14
$\Delta\nu_{\rm models}$	58.01 ± 0.14	24.94 ± 0.09	28.60 ± 0.07	16.10 ± 0.04	3.98 ± 0.04

Table 7.4: Table with the mass, radius and age, from creating stellar evolution models with MESA. In the table is also included $\Delta\nu$ for both my own code and the one estimated from the stellar evolution models.

mass difference into account. Their mass is lower than my estimated mas, meaning that the estimated age from the models should also be longer.

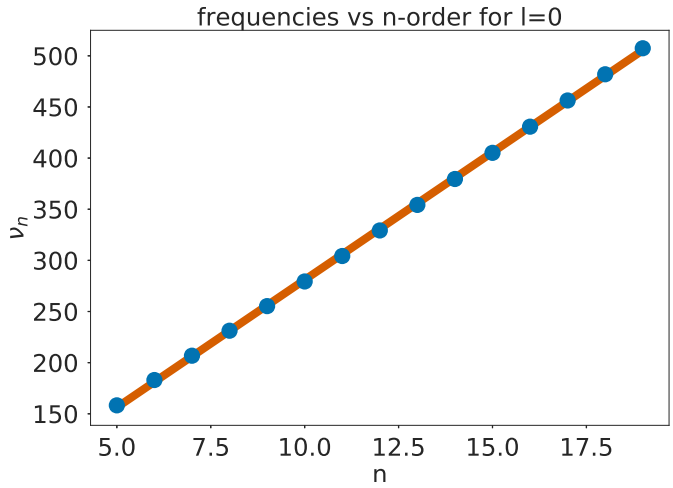


Figure 7.14: Calculated frequencies using GYRE plotted against the n-order (blue points). The solid red line is the fit estimating a $\Delta\nu = 24.94 \pm 0.09$ μHz.

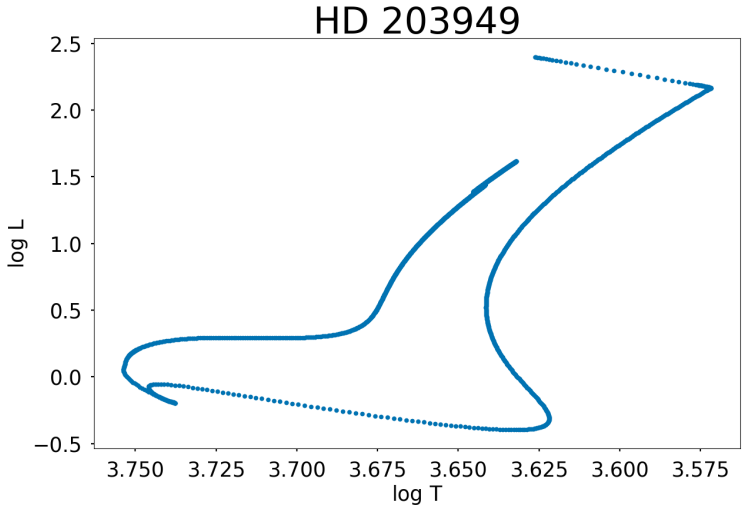


Figure 7.15: Stellar evolution model for HD 203949 from pre-main sequence to a pre-determined age using MESA. Using the mass of the star from Campante et al. 2019 in case it is a red clump star of $M = 1.00 M_{\odot}$. Mixing length is $\alpha = 1.845$ and $Y_{\text{initial}} = 0.279394$ and $Z_{\text{initial}} = 0.0236383$.

Chapter 8

Discussion And Conclusion

Challenges With The Analysis

The process from the raw light curve to the estimation of stellar parameters (mass, radius and age) has been filled with challenges which I will try to discuss here.
The first part of the analysis is the removal of noise from planetary transits, instruments or outliers. This is done using a moving median filter moving in intervals of different lengths (τ_{long} and τ_{short}) from the python package bottleneck. Problem with using this moving median filter is the way it takes the median. Normally, the median is the middle value of the data set in each interval if there is an odd number of data points. In case there is an even number of data points, the median is the mean of the two middle data points. However, when taking the median using bottleneck.move_median the whole median filter is shifted with half the interval (fx $\tau_{\text{long}}/2$), and the filter does not extend out to the ends. This was done manually using extrapolation. A list of ones were made which I multiplied with the first or last data point to extrapolate in the start or the end respectively.

Making the asteroseismic analysis also had some challenges. Estimating the autocorrelation was a great challenge. This was due to the sensitivity of the input parameters, especially the initial guess for $\Delta\nu$ had a huge affect on the fit of the autocorrelation. Second was verifying the estimated $\Delta\nu$ based on looking at the power spectrum and the echelle diagram for

each star.

Mode identification is relatively easy for solar-like stars, however, due to the fact that only one light curve for each stars was released at the time, the signal to noise ratio (SNR) was relatively low. This made the mode identification quite hard due to the difficulty in separating noise peaks from a "real" peak. This led to the use of the prominence of the individual peaks to try to include some of the relatively small but "real" peaks. With a higher signal to noise ratio mode identification would be relatively easier using the asymptotic relation. The problem with low signal to noise ratio is that using the frequencies calculated from the asymptotic relation can fall in a region where it is hard to separate between a noise peak or a "real" peak. An improvement to estimate the individual modes would be to use peak bagging and MCMC. This could separate the real peaks from the noise peaks based on the input parameters for the peak bagging. Using peak bagging you fit the individual peaks with some free parameters such as amplitude, line width, splitting, inclination and more.

Estimating the small frequency separation, $\delta\nu_{02}$, using a matched filter gave some problems due to change in output based on the resolution. Here the struggle was to find a resolution high enough to give a proper estimate of $\delta\nu_{02}$. The matched filter was chosen due to the low signal to noise ratio. If the signal to noise ratio were higher, more the identification of l=2 would be easier and the presence of noise peaks in the echelle diagram would be significantly lower. This would mean that $\delta\nu_{02}$ could be estimated using the echelle diagram.

Age estimation using the scaling relations also had some limitations, due to the dependence of $\delta\nu_{02}$ or the period spacing, $\Delta\Pi_1$. If $\delta\nu_{02}$ was not estimated correctly, the age would deviate significantly from the true age in case it is a main sequence star or a subgiant (only to some extent of subgiants). For the giants in this thesis, $\Delta\Pi_1$ needed to be estimated or another variable combination was used (without the inclusion of $\Delta\Pi_1$). To optimize this code, a analyses of the avoided crossings could be made to estimate $\Delta\Pi_1$, this could optimize the code making it more precise in estimating the age of giants using scaling relation.

The stellar evolution models used in this thesis are somewhat basic.

This is due to the lack of information about stellar evolution parameters such as diffusion, mixing length, convective overshooting etc. However, in these models a solar value for the mixing length is used. The stellar models are solar calibrated, meaning that a MESA run was made to replicate the Sun. This was done to check if the mixing length parameter inserted in the controls produced a radius, luminosity and temperature with minimal deviation from the solar values. This led to a mixing length of $\alpha = 1.845$. A change in mixing length could affect the models and thereby also the calculated age. Optimizing and incorporating relevant controls for each star for the MESA run would optimize the calculation of the age.

Conclusion

Code For Analysis of Stars in Different Stages of Evolution

In this thesis I have developed a code that can estimate the fundamental stellar parameters mass, radius and age for 5 selected TESS stars that are in different stages of their evolution. The code takes as input a raw light curve (or several light curves if available) and prepares this for the asteroseismic analysis by removing noise from different sources (instrumental noise, pointing, outliers, transits etc.). After this, the light curve is Fourier transformed from the time domain to the frequency domain using the least squares method. The age for these 5 stars are estimated using both scaling relations and stellar evolution models. The scaling relations used to estimate the ages of the stars are newly modified versions of the $\Delta\nu$-scaling relation (Kjeldsen and Bedding 1995). These new versions are not as well tested as the $\Delta\nu$-scaling relation. For comparison theoretical stellar evolution models are made to estimate the age of the star, and compare with the results obtained from the analysis.

To use these modified scaling relations, several asteroseismic parameters are needed. After creating the power spectrum and subtracting the background (photon noise, granulation, faculae etc.) (section 6.3), using a modified Harvey profile, these parameters can be estimated. I can conclude that the autocorrelation with a Gaussian fit can give a robust estimate on the large frequency separation, $\Delta\nu$, which relates to the density of the star (section 6.6).
The small frequency separation, $\delta\nu_{02}$, can be estimated relatively precisely using the matched filter (section 6.7). The initial guesses for the matched filter are based on the structure of the echelle diagram of the star.

Testing New Scaling Relations

The masses and radii obtained from the scaling relations (section 2.4) seem to overall match the values in the literature. In two of the stars (β Hydri and ν Indi) where $\delta\nu_{02}$ could be good indicator of the age, the age estimated from my analysis seems to correspond pretty well with the litterature value. This is very interesting since the scaling relation is designed for main sequence stars, and these two stars are subgiants. This indicates that the scaling relation may extend further out than only just for main sequence stars. The second modified scaling relation is for giants, and has the period spacing, $\Delta\Pi_1$, as an indicator for the age compared to $\delta\nu_{02}$ for main sequence stars. Estimating $\Delta\Pi_1$ has not been a part of the work in this thesis, however, different variable combinations are made to perform this scaling relation even if not all parameters are available. In this thesis two giants are analyzed (HD 212771 and HD 203949) where $\Delta\Pi_1$ was only available for one of the stars (HD 212771). The age estimated from using the scaling relation for giants on HD 212771 seems to correspond quite well with the literature value, indicating that the period spacing is a good indicator for the age for giants. The same seems to be the case with TOI-197. The echelle diagram displays many mixed modes for l=1 indicating that the scaling relation for giants may provide the best estimate of the age.

The age estimated using the scaling relation for giants gives a higher age than the value in the literature, but still within the uncertainties. However, this is due to the mass used to estimate the age, my estimated mass is a bit smaller than the literature value meaning that the age will naturally be longer, due to spending longer time on the main sequence. However, the age for HD 203949 is way off with a factor 2 using the scaling relation for giants, which makes no sense since the star is a very evolved star.

Creating Theoretical Stellar Evolution Models For Estimating Age

Using the stellar evolution code MESA and the stellar pulsation code GYRE, theoretical models were made for each star. This was to determine the age only using the mass and metallicity of each star. I can conclude that the ages obtained from the stellar models match the literature value quite well. The deviations in the results between using the scaling relations and the models, are due to the different masses used. In the models, the mass is estimated using the three results obtained from the scaling rela-

tions and taking the mean. This means that if the estimated mean mass of the star is higher than the reference value, then the calculated age will be shorter than the reference value due to shorter time spent on the main sequence. This also means that if the estimated mean mass of the star is lower than the reference value, the age will be longer than the reference value due to fact that the star spends longer time on the main sequence. However, this does not seems to be the case for HD 203949. In the case that HD 203949 is a red clump star, the star has a mass lower than the estimated mean mass I used in my models, meaning that its age should be higher than my calculated value. This led to computing a model of HD 203949 using the mass and metallicity used in Campante et al. 2019, and finding the best model using their radius. This gave an estimate of the age which was a factor 2 off. This result indicates that the literature value for the age of this star is wrong, and that the age obtained using my code and models is closer to the true age of the star. This means that using these newly modified scaling relations, a relatively precise estimate of the age can be obtained. However, these scaling relations need to be tested further.

Bibliography

1. Aerts, Conny, Christensen-Dalsgaard, Jørgen, and Kurtz, Donald W (2010). Asteroseismology. Springer Science & Business Media.

2. Basu, Sarbani and Chaplin, William J (2017). Asteroseismic Data Analysis: Foundations and Techniques. Vol. 4. Princeton University Press.

3. Bedding, Timothy R and Kjeldsen, Hans (2003). "Solar-like oscillations". In: Publications of the Astronomical Society of Australia 20.2, pp. 203–212.

4. Bedding, Timothy R, Pallé, PL, and Esteban, C (2014). "Solar-like oscillations: An observational perspective". In: Asteroseismology 22, p. 60.

5. Bedding, Timothy R et al. (2001). "Evidence for Solar-like Oscillations in β Hydri". In: The Astrophysical Journal Letters 549.1, p. L105.

6. Bedding, Timothy R et al. (2006a). "Solar-like oscillations in the metal-poor subgiant ν Indi: constraining the mass and age using asteroseismology". In: The Astrophysical Journal 647.1, p. 558.

7. Bedding, Timothy R et al. (2006b). "Solar-like oscillations in the metal-poor subgiant ν Indi: constraining the mass and age using asteroseismology". In: The Astrophysical Journal 647.1, p. 558.

8. Bedding, Timothy R et al. (2007a). "Solar-like oscillations in the G2 subgiant β Hydri from dual-site observations". In: The Astrophysical Journal 663.2, p. 1315.

9. Bedding, Timothy R et al. (2007b). "Solar-like oscillations in the G2 subgiant β Hydri from dual-site observations". In: The Astrophysical Journal 663.2, p. 1315.

10. Bedding, Timothy R et al. (2011). "Gravity modes as a way to distinguish between hydrogen-and helium-burning red giant stars". In: Nature 471.7340, p. 608.

11. Bellinger, Earl P, Hekker, Saskia, Angelou, George C, Stokholm, Amalie, and Basu, Sarbani (2019). "Stellar ages, masses, and radii from asteroseismic modeling are robust to systematic errors in spectroscopy". In: Astronomy & Astrophysics 622, A130.

12. Bellinger, Earl Patrick (2019a). "A seismic scaling relation for stellar age". In: Monthly Notices of the Royal Astronomical Society.

13. Bellinger, Earl Patrick (2019b). "A seismic scaling relation for stellar age II: The red giant branch". In: in prep.

14. Bonanno, Alfio and Fröhlich, Hans-Erich (2017). "A new helioseismic constraint on a cosmic-time variation of G". In: arXiv preprint arXiv:1707.01866.

15. Borucki, William J et al. (2011). "Characteristics of planetary candidates observed by Kepler. II. Analysis of the first four months of data". In: The Astrophysical Journal 736.1, p. 19.

16. Borucki, William et al. (2008). "KEPLER: search for Earth-size planets in the habitable zone". In: Proceedings of the International Astronomical Union 4.S253, pp. 289–299.

17. Brandão, IM et al. (2011). "Asteroseismic modelling of the solar-type subgiant star β Hydri". In: Astronomy & Astrophysics 527, A37.

18. Campante, Tiago L., Corsaro, Enrico, Lund, Mikkel N., and Mosser, Benoît (2019). "TESS Asteroseismology of the known red-giant host stars HD 212771 and HD 203949". In: in prep.

19. Campante, TL et al. (2010). "Modelling the autocovariance of the power spectrum of a solar-type oscillator". In: Monthly Notices of the Royal Astronomical Society 408.1, pp. 542–550.

20. Carrier, Fabien et al. (2001). “Research Note: Solar-like oscillations in Hydri: Confirmation of a stellar origin for the excess power”. In: Astronomy & Astrophysics 378.1, pp. 142–145.

21. Chaplin, William J and Miglio, Andrea (2013). “Asteroseismology of solar-type and red-giant stars”. In: Annual Review of Astronomy and Astrophysics 51, pp. 353–392.

22. Christensen-Dalsgaard, J (1984). “Space Research in Stellar Activity and Variability, ed”. In: A. Mangeney & F. Praderie 11.

23. Christensen-Dalsgaard, JØrgen (2004). “Physics of solar-like oscillations”. In: Solar Physics 220.2, pp. 137–168.

24. Christensen-Dalsgaard, Jørgen (1997). Lecture Notes on Stellar Oscillations.

25. Christensen-Dalsgaard, Jørgen (2008). “ADIPLSthe Aarhus adiabatic oscillation package”. In: Astrophysics and Space Science 316.1-4, pp. 113–120.

26. Davies, GR, Broomhall, AM, Chaplin, WJ, Elsworth, Y, and Hale, SJ (2014). “Low-frequency, low-degree solar p-mode properties from 22 years of Birmingham Solar Oscillations Network data”. In: Monthly Notices of the Royal Astronomical Society 439.2, pp. 2025–2032.

27. Dravins, D, Linde, P, Fredga, K, and Gahm, GF (1993a). “The distant future of solar activity: A case study of Beta Hydri. II-Chromospheric activity and variability”. In: The Astrophysical Journal 403, pp. 396–411.

28. Dravins, D, Lindegren, L, Nordlund, A, and Vandenberg, DA (1993b). “The distant future of solar activity: A case study of Beta Hydri. I-Stellar evolution, lithium abundance, and photospheric structure”. In: The Astrophysical Journal 403, pp. 385–395.

29. Dravins, D et al. (1993c). “The distant future of solar activity: A case study of Beta Hydri. III-Transition region, corona, and stellar wind”. In: The Astrophysical Journal 403, pp. 412–425.

30. Eddington, Arthur S (1920). The internal constitution of the stars.

31. Edmonds, PD and Cram, LE (1995). "A Search for global acoustic on α 1 Cen and β Hyi". In: Monthly Notices of the Royal Astronomical Society 276.4, pp. 1295–1302.

32. Frandsen, Søren (1987). "An upper limit on p-mode amplitudes in Beta Hyi". In: Astronomy and Astrophysics 181, pp. 289–292.

33. Frandsen, S et al. (1995). "CCD photometry of the delta-Scuti star kappa^ 2^ Bootis." In: Astronomy and Astrophysics 301, p. 123.

34. Fressin, François et al. (2013). "The false positive rate of Kepler and the occurrence of planets". In: The Astrophysical Journal 766.2, p. 81.

35. Grevesse, N and Sauval, AJ (1998). "Standard solar composition". In: Space Science Reviews 85.1-2, pp. 161–174.

36. Handberg, R and Campante, TL (2011). "Bayesian peak-bagging of solar-like oscillators using MCMC: a comprehensive guide". In: Astronomy & Astrophysics 527, A56.

37. Handberg, R and Lund, MN (2014). "Automated preparation of Kepler time series of planet hosts for asteroseismic analysis". In: Monthly Notices of the Royal Astronomical Society 445.3, pp. 2698–2709.

38. Handberg, Rasmus (2013). "Asteroseismology of solar-like stars". English. PhD thesis.

39. Handberg, Rasmus et al. (2017). "NGC 6819: testing the asteroseismic mass scale, mass loss and evidence for products of non-standard evolution". In: Monthly Notices of the Royal Astronomical Society 472.1, pp. 979–997.

40. Harvey, J (1985). "High-resolution helioseismology". In: Future missions in solar, heliospheric & space plasma physics. Vol. 235.

41. Hekker, S and Christensen-Dalsgaard, J (2017). "Giant star seismology". In: The Astronomy and Astrophysics Review 25.1, p. 1.

42. Hekker, Saskia et al. (2010). "The Octave (Birmingham–Sheffield Hallam) automated pipeline for extracting oscillation parameters of solar-like main-sequence stars". In: Monthly Notices of the Royal Astronomical Society 402.3, pp. 2049–2059.

43. Houk, N and Smith-Moore, M (1988). "Michigan Catalogue of Two-dimensional Spectral Types for the HD Stars. Volume 4, Declinations-26ř. 0 to-12ř. 0." In: Michigan Catalogue of Two-dimensional Spectral Types for the HD Stars. Volume 4, Declinations-26ř. 0 to-12ř. 0.. N. Houk, M. Smith-Moore. Department of Astronomy, University of Michigan, Ann Arbor, MI 48109-1090, USA. 14+ 505 pp. Price US 25.00 (USA, Canada), US 28.00 (Foreign)(1988).

44. Houk, Nancy (1982). "Michigan Catalogue of Two-dimensional Spectral Types for the HD stars. Volume_3. Declinations-40_f0 to-26_f0." In: Michigan Catalogue of Two-dimensional Spectral Types for the HD stars. Volume_3. Declinations-40_f0 to-26_f0., by Houk, N.. Ann Arbor, MI (USA): Department of Astronomy, University of Michigan, 12+ 390 p.

45. Howell, Steve B et al. (2014). "The K2 mission: characterization and early results". In: Publications of the Astronomical Society of the Pacific 126.938, p. 398.

46. Huber, Daniel et al. (2009). "Automated extraction of oscillation parameters for Kepler observations of solar-type stars". In: arXiv preprint arXiv:0910.2764.

47. Huber, Daniel et al. (2010). "Asteroseismology of red giants from the first four months of Kepler data: global oscillation parameters for 800 stars". In: The Astrophysical Journal 723.2, p. 1607.

48. Huber, Daniel et al. (2019). "A Hot Saturn Orbiting An Oscillating Late Subgiant Discovered by TESS". In: The Astronomical Journal 157.6, p. 245.

49. Huber, D et al. (2011). "Testing scaling relations for solar-like oscillations from the main sequence to red giants using Kepler data". In: The Astrophysical Journal 743.2, p. 143.

50. Jenkins, Jon M et al. (2010). “Overview of the Kepler science processing pipeline”. In: The Astrophysical Journal Letters 713.2, p. L87.

51. Jetsu, Lauri and Porceddu, Sebastian (2015). “Shifting Milestones of Natural Sciences: The Ancient Egyptian Discovery of Algols Period Confirmed”. In: PloS one 10.12, e0144140.

52. Johnson, John Asher et al. (2010). “Retired A Stars and Their Companions. IV. Seven Jovian Exoplanets from Keck Observatory”. In: Publications of the Astronomical Society of the Pacific 122.892, p. 701.

53. Jones, MI, Jenkins, James Stewart, Bluhm, P, Rojo, P, and Melo, CHF (2014). “The properties of planets around giant stars”. In: Astronomy & Astrophysics 566, A113.

54. Karoff, Christoffer (2008). “Observational asteroseismology”. In:

55. Keller, CU, Schüssler, M, Vögler, A, and Zakharov, V (2004). “On the origin of solar faculae”. In: The Astrophysical Journal Letters 607.1, p. L59.

56. Kjeldsen, H and Bedding, TR (1995). “Amplitudes of stellar oscillations: the implications for asteroseismology.” In: Astronomy and Astrophysics 293, pp. 87–106.

57. Lambert, DL and Mcwilliam, Ai (1986). “Isotopic abundances of magnesium in the metal-poor subgiant V INDI”. In: The Astrophysical Journal 304, pp. 436–442.

58. Lissauer, Jack J et al. (2011). “Architecture and dynamics of Kepler’s candidate multiple transiting planet systems”. In: The Astrophysical Journal Supplement Series 197.1, p. 8.

59. Lomb, Nicholas R (1976). “Least-squares frequency analysis of unequally spaced data”. In: Astrophysics and space science 39.2, pp. 447–462.

60. Lund, Mikkel N, Chaplin, William J, and Kjeldsen, Hans (2012). “A new method to detect solar-like oscillations at very low S/N using statistical significance testing”. In: Monthly Notices of the Royal Astronomical Society 427.2, pp. 1784–1792.

61. Lund, Mikkel Nørup (2015). “Asteroseismology: Rotation and Convection in stars”. English. PhD thesis.

62. Lund, Mikkel N et al. (2017). “Standing on the shoulders of Dwarfs: The Kepler asteroseismic LEGACY sample. I. Oscillation mode parameters”. In: The Astrophysical Journal 835.2, p. 172.

63. Marcy, Geoffrey W et al. (2014). “Masses, radii, and orbits of small Kepler planets: the transition from gaseous to rocky planets”. In: The Astrophysical Journal Supplement Series 210.2, p. 20.

64. Mathur, S et al. (2010). “Determining global parameters of the oscillations of solar-like stars”. In: Astronomy & Astrophysics 511, A46.

65. Matthews, Jaymie M (2007). “One small satellite, so many light curves: Examples of delta Scuti asteroseismology from the MOST space mission^ 1”. In: Communications in Asteroseismology 150, p. 333.

66. Metcalfe, Travis S et al. (2010). “A precise asteroseismic age and radius for the evolved Sun-like star KIC 11026764”. In: The Astrophysical Journal 723.2, p. 1583.

67. Michel, Eric et al. (2008). “CoRoT measures solar-like oscillations and granulation in stars hotter than the Sun”. In: Science 322.5901, pp. 558–560.

68. Mosser, B et al. (2012). “Probing the core structure and evolution of red giants using gravity-dominated mixed modes observed with Kepler”. In: Astronomy & Astrophysics 540, A143.

69. Mosser, B et al. (2014). “Mixed modes in red giants: a window on stellar evolution”. In: Astronomy & Astrophysics 572, p. L5.

70. Paxton, Bill et al. (2010). “Modules for experiments in stellar astrophysics (MESA)”. In: The Astrophysical Journal Supplement Series 192.1, p. 3.

71. Press, William H and Rybicki, George B (1989). “Fast algorithm for spectral analysis of unevenly sampled data”. In: The Astrophysical Journal 338, pp. 277–280.

72. Pra, Andrej et al. (2016). “NOMINAL VALUES FOR SELECTED SOLAR AND PLANETARY QUANTITIES: IAU 2015 RESOLUTION B3”. In: Astronomical Journal (Online) 152.2.

73. Ricker, George R et al. (2016). “The transiting exoplanet survey satellite”. In: Space Telescopes and Instrumentation 2016: Optical, Infrared, and Millimeter Wave. Vol. 9904. International Society for Optics and Photonics, 99042B.

74. Scargle, Jeffrey D (1989). “Studies in astronomical time series analysis. III-Fourier transforms, autocorrelation functions, and cross-correlation functions of unevenly spaced data”. In: The Astrophysical Journal 343, pp. 874–887.

75. Stello, D, Bruntt, H, Preston, H, and Buzasi, D (2008). “Oscillating K giants with the WIRE satellite: determination of their asteroseismic masses”. In: The Astrophysical Journal Letters 674.1, p. L53.

76. Stello, D, Chaplin, WJ, Basu, S, Elsworth, Y, and Bedding, TR (2009). “The relation between Δ v and v max for solar-like oscillations”. In: Monthly Notices of the Royal Astronomical Society: Letters 400.1, pp. L80–L84.

77. Tassoul, Monique (1980). “Asymptotic approximations for stellar nonradial pulsations”. In: The Astrophysical Journal Supplement Series 43, pp. 469–490.

78. Townsend, RHD and Teitler, SA (2013). “GYRE: an open-source stellar oscillation code based on a new Magnus Multiple Shooting scheme”. In: Monthly Notices of the Royal Astronomical Society 435.4, pp. 3406–3418.

79. Turin, George (1960). “An introduction to matched filters”. In: IRE transactions on Information theory 6.3, pp. 311–329.

80. Walker, Gordon et al. (2003). “The MOST asteroseismology mission: Ultraprecise photometry from space”. In: Publications of the Astronomical Society of the Pacific 115.811, p. 1023.

81. White, Timothy R et al. (2011). “Calculating asteroseismic diagrams for solar-like oscillations”. In: The Astrophysical Journal 743.2, p. 161.

Appendix A

Timeseries for β Hydri, TOI-197 and HD 212771

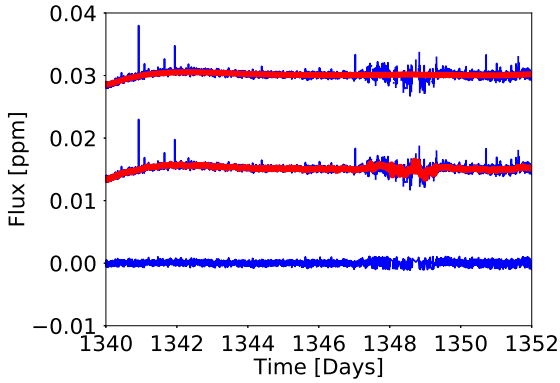


Figure A.1: The light curve for β Hydri. The plot consists of three curves. The top curve is the raw normalized curve where no trends and outliers have been removed. The curve also includes the long time scale median filter (red line) which is looking at any long trends in the light curve. The middle curve is also the raw light curve however with the short time scale median filter which is used to model sharp features (e.g. transits). The curve at the bottom shows the corrected time series after transits and outliers have been removed.

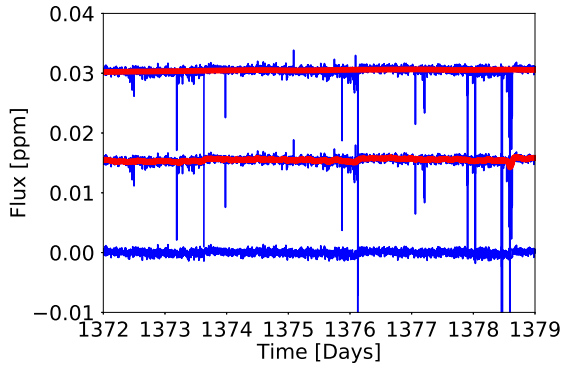


Figure A.2: The light curve for TOI-197. The plot consists of three curves. The top curve is the raw normalized curve where no trends and outliers have been removed. The curve also includes the long time scale median filter (red line) which is looking at any long trends in the light curve. The middle curve is also the raw light curve however with the short time scale median filter which is used to model sharp features (e.g. transits). The curve at the bottom shows the corrected time series after transits and outliers have been removed.

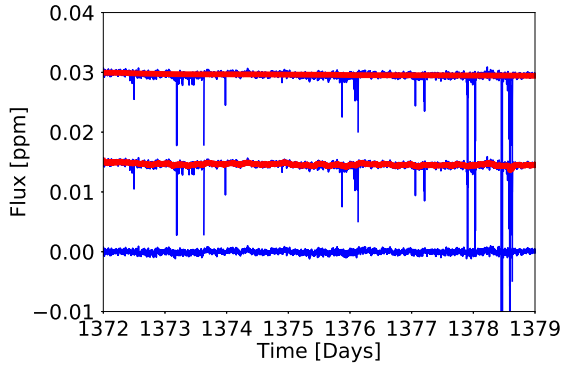


Figure A.3: The light curve for HD 212771. The plot consists of three curves. The top curve is the raw normalized curve where no trends and outliers have been removed. The curve also includes the long time scale median filter (red line) which is looking at any long trends in the light curve. The middle curve is also the raw light curve however with the short time scale median filter which is used to model sharp features (e.g. transits). The curve at the bottom shows the corrected time series after transits and outliers have been removed.

Appendix B

Timeseries for ν Indi, TOI-197 and HD 212771

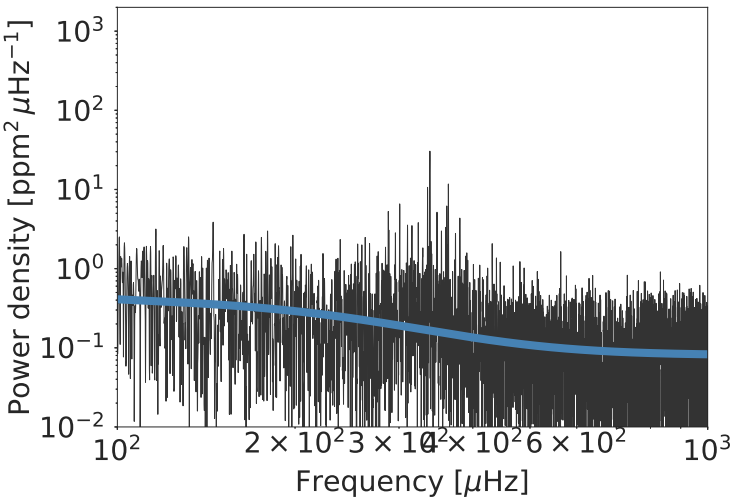


Figure B.1: Un-smoothed power spectra for ν Indi. The blue line in each plot is the Harvey profile background subtracted from each power spectra.

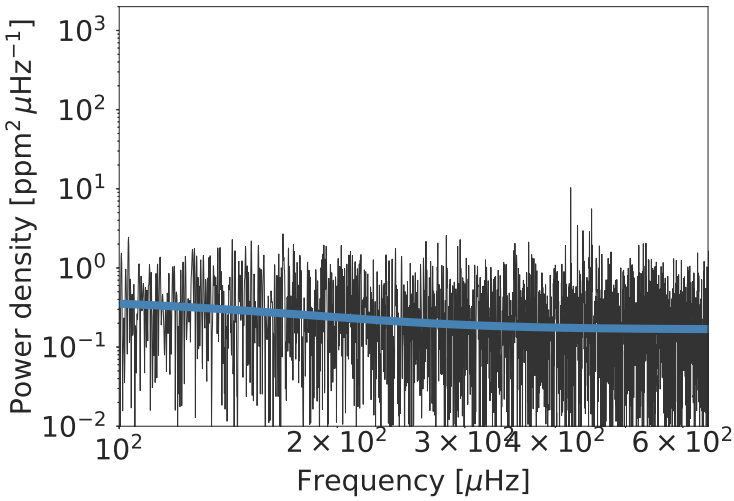


Figure B.2: Un-smoothed power spectra for TOI-197. The blue line in each plot is the Harvey profile background subtracted from each power spectra.

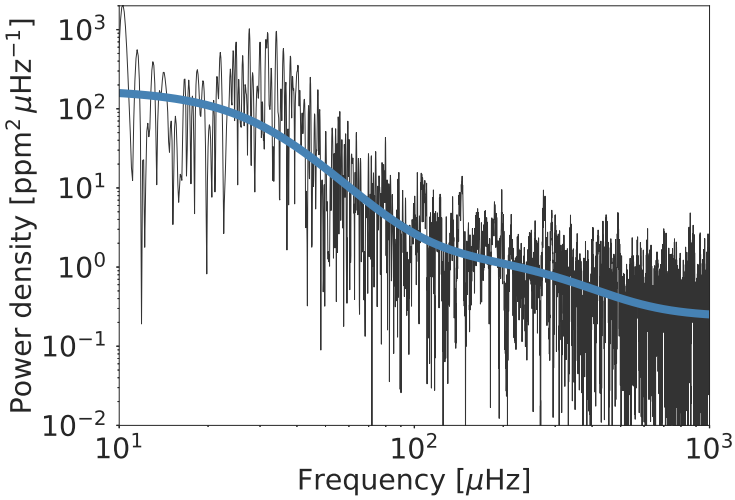


Figure B.3: Un-smoothed power spectra for HD 203949. The blue line in each plot is the Harvey profile background subtracted from each power spectra.

Appendix C

Power Spectrum with ν_{max} For TOI-197, HD 212771 and HD 203949

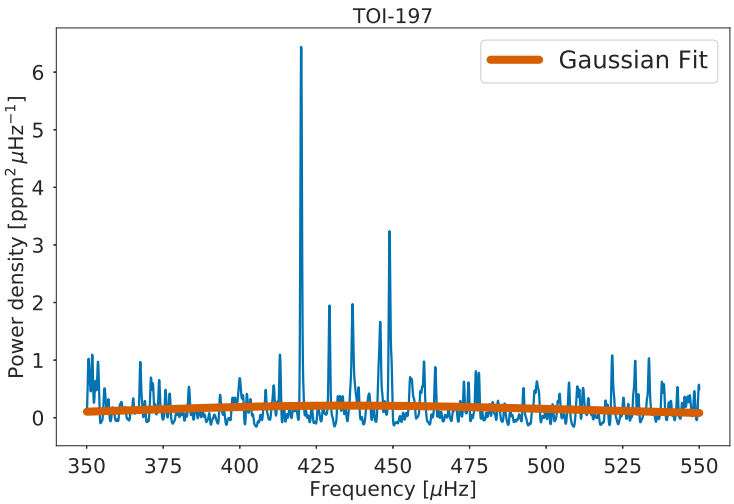


Figure C.1: Powerspectrum for TOI-197 with a Gaussian (red line) to indicate the frequency of maximum power.

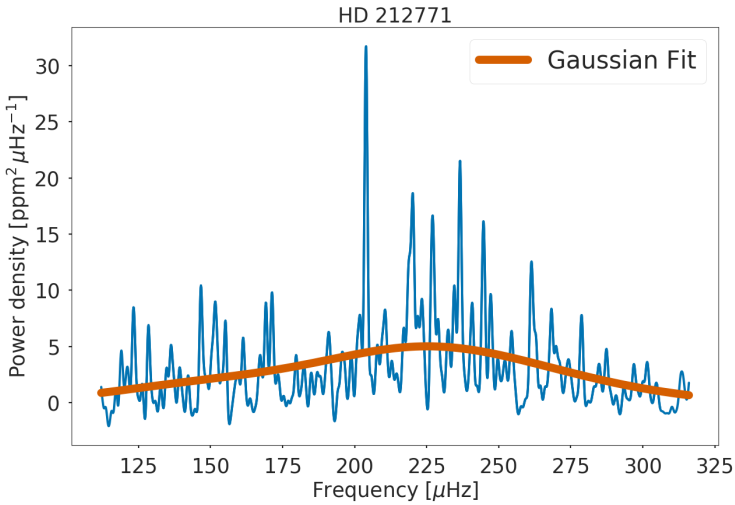


Figure C.2: Powerspectrum for HD 212771 with a Gaussian (red line) to indicate the frequency of maximum power.

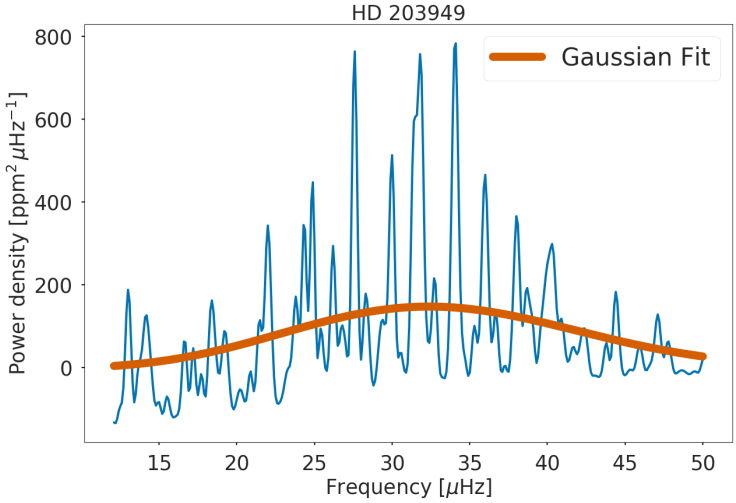


Figure C.3: Powerspectrum for HD 203949 with a Gaussian (red line) to indicate the frequency of maximum power.

Appendix D

Autocorrelation and Echelle Diagram For ν Indi, HD 212771 and HD 203949

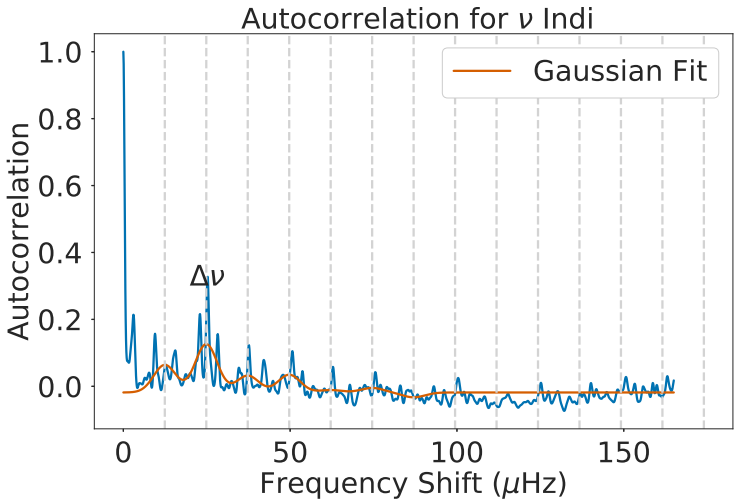


Figure D.1: Autocorrelation for ν Indi with the location for $\Delta\nu$ displayed.

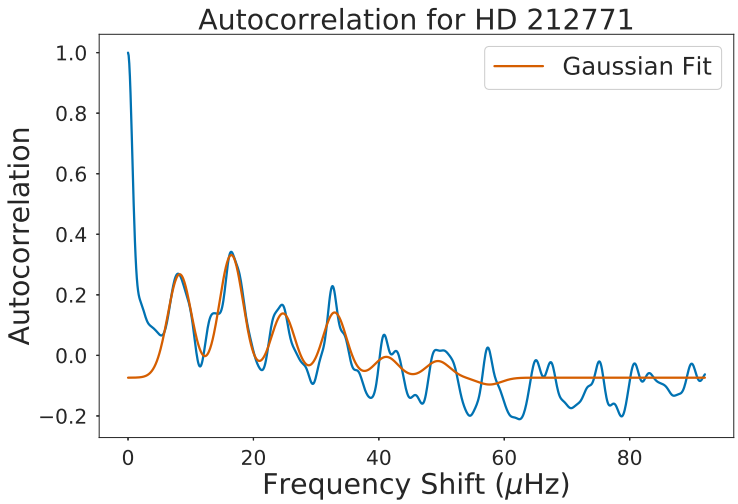


Figure D.2: Autocorrelation for HD 212771.

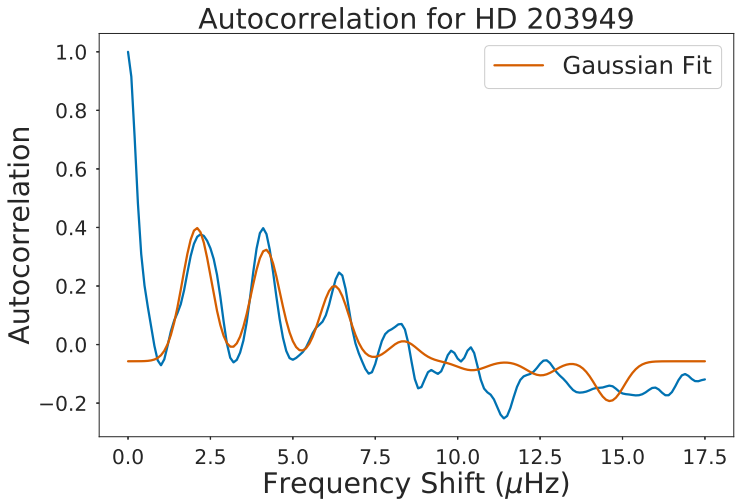


Figure D.3: Autocorrelation for HD203949.

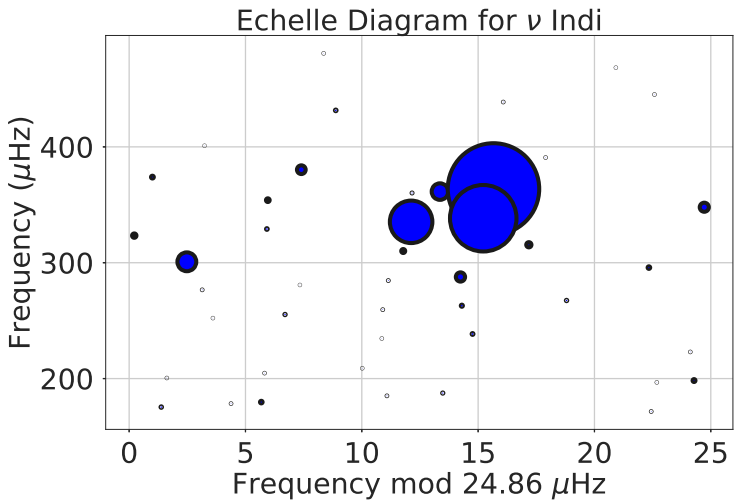


Figure D.4: Echelle diagram for ν Indi.

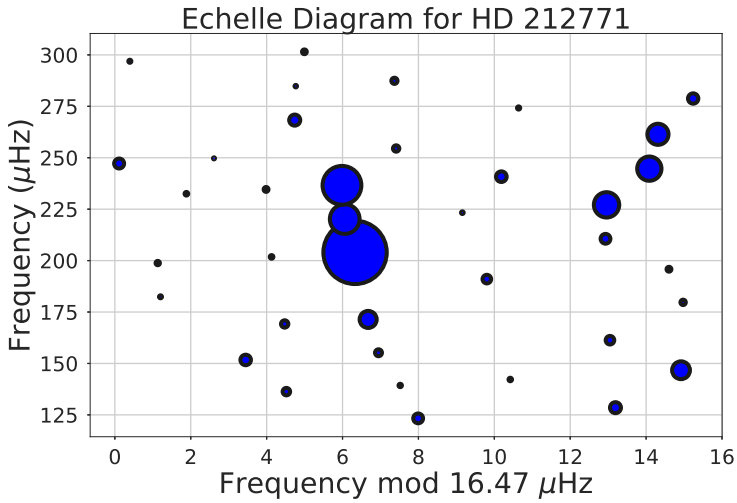


Figure D.5: Echelle diagram for HD 212771.

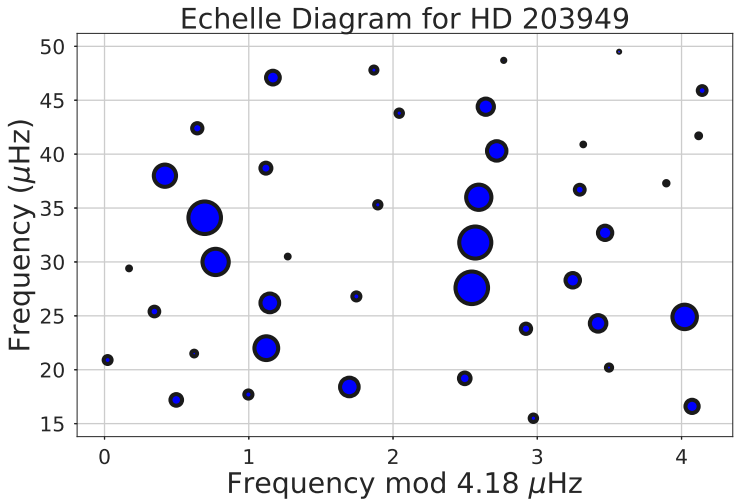


Figure D.6: Echelle diagram for HD 203949.

Appendix E

Matched Filter For ν Indi, TOI-197 and HD 203949

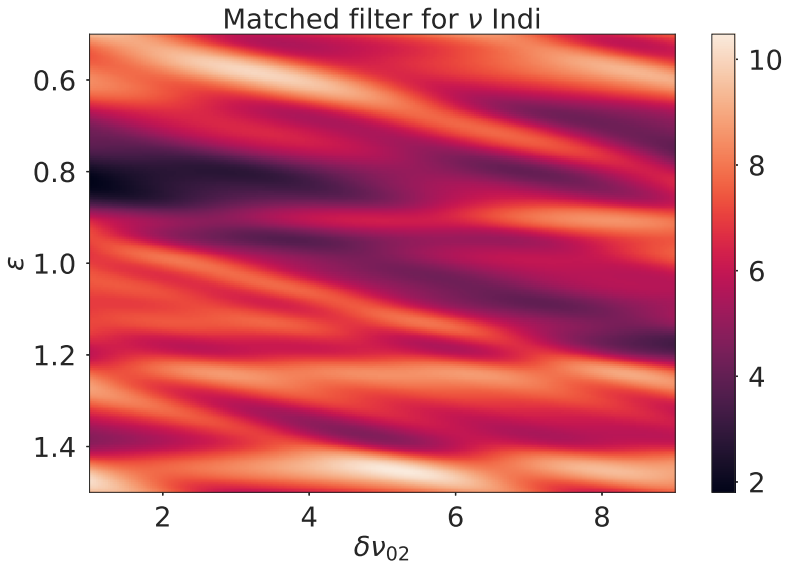


Figure E.1: Density plot of matched filter for ν Indi estimating a $\delta\nu02 = 5.18\,\mu$Hz and $\varepsilon = 1.45$.

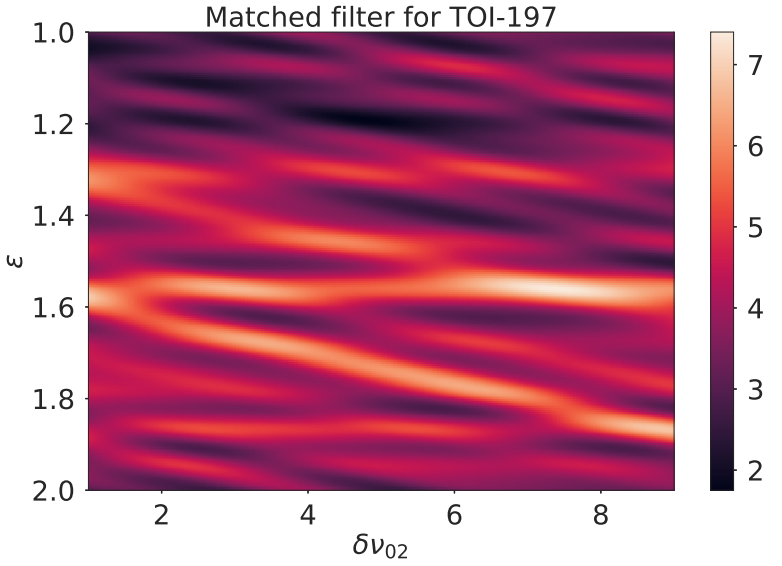


Figure E.2: Density plot of matched filter for TOI-197 estimating a $\delta\nu02 = 7.43\,\mu$Hz and $\varepsilon = 1.56$.

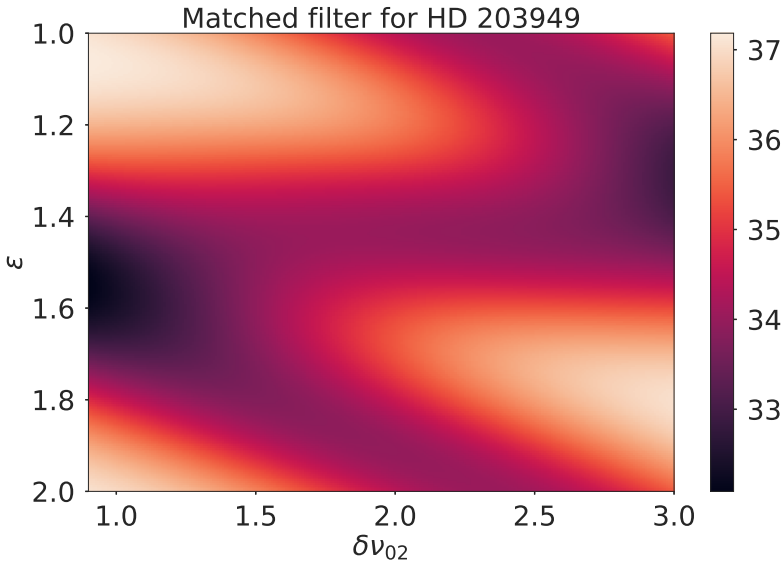


Figure E.3: Density plot of matched filter for HD 203949 estimating a $\delta\nu02 = 0.9\,\mu$Hz and $\varepsilon = 1.06$.

Appendix F

Stellar Evolution Models For β Hydri, TOI-197, HD 212771 and HD 203949

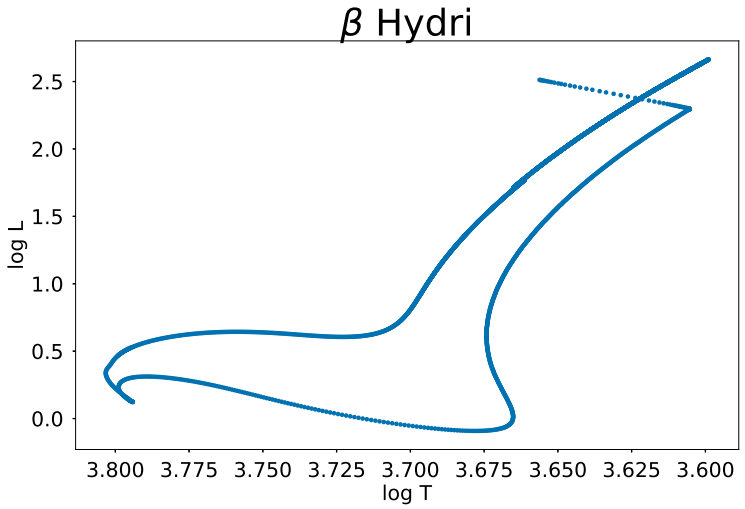


Figure F.1: Stellar evolution model for ν Indi from pre-main sequence to a pre-determined age using MESA. Mass for the star is 1.042 $M_{\odot}$, α=1.845, Y_{initial}=0.255229 and Z_{initial} = 0.0063781.

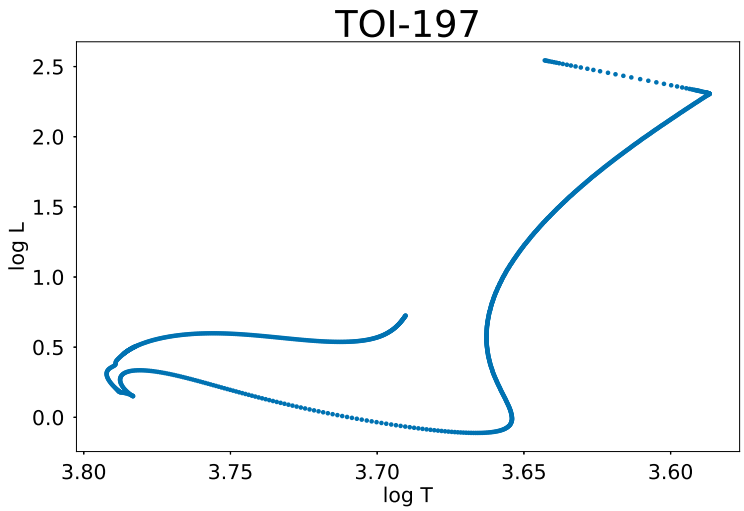


Figure F.2: Stellar evolution model for ν Indi from pre-main sequence to a pre-determined age using MESA. Mass for the star is 1.13 $M_{\odot}$, α=1.845, Y_{initial}=0.265544 and Z_{initial} = 0.0137456.

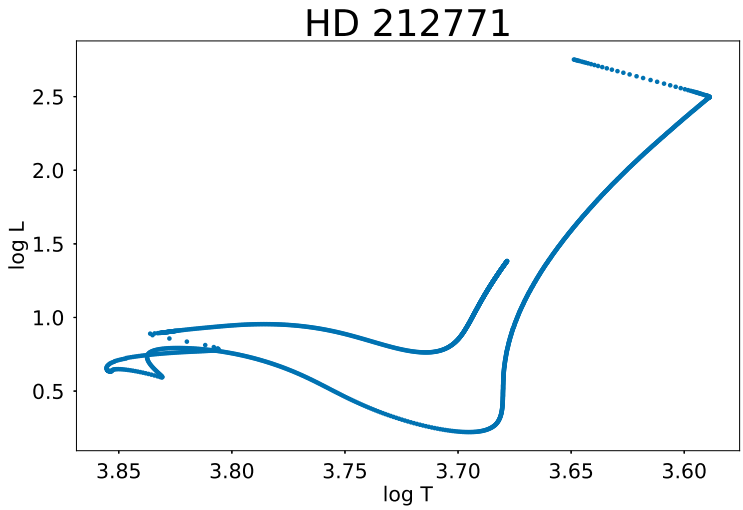


Figure F.3: Stellar evolution model for ν Indi from pre-main sequence to a pre-determined age using MESA. Mass for the star is 1.41 $M_{\odot}$, α=1.845, Y_{initial}=0.264714 and Z_{initial} =0.0131529.

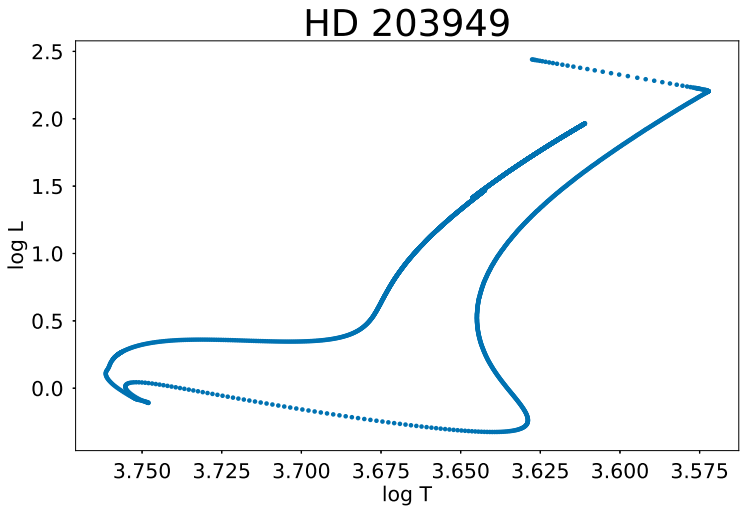


Figure F.4: Stellar evolution model for ν Indi from pre-main sequence to a pre-determined age using MESA. Mass for the star is 1.047 $M_{\odot}$, α=1.845, $Y_{\rm initial}$=0.279394 and $Z_{\rm initial}$ =0.0236383.

Appendix G

Code

Removing Noise and Transits

```
import numpy as np
import statistics as stat
import math
from scipy.stats import norm
import pandas as pd
import bottleneck as bn
from astropy.stats import sigma_clip
import astropy
from astropy.io import fits
from filter import running_median_filter, binary_search,
    find_time, running_median, isOdd


import matplotlib.pyplot as plt
plt.style.use('seaborn-poster')
plt.style.use('seaborn-colorblind')


names_of_all_stars= np.loadtxt('list_of_all_quaters.txt')


#for i in range(len(names_of_all_stars)):
for i in names_of_all_stars:


  # If file is a .fits file use this
```

```
    #hdulist = fits.open('/home/simonbanerjee/Dropbox/Speciale/
      Lightcurve_to_powerspectrum/16_Cyg_A/kplr100003551
      -2011208035123_slc.fits' %i, mode='readonly')
    #hdulist = fits.open('/home/simonbanerjee/Dropbox/Speciale/
      Lightcurve_to_powerspectrum/16_Cyg_A/kplr%d-%d_slc.fits' %(
      names_of_all_stars[i,0],names_of_all_stars[i,1]), mode='
      readonly')
    hdulist = fits.open('/home/simonbanerjee/Dropbox/Speciale/
      Lightcurve_to_powerspectrum/nu_indi/tess00%d-s01-c0120-dr01
      -v04-tasoc_lc.fits' %i, mode='readonly')

    #hdulist = fits.open('/home/simonbanerjee/Dropbox/Speciale/
      Lightcurve_to_powerspectrum/KIC8751420/kplr100004295
      -2013131215648_slc.fits', mode='readonly')

    LightCurve = hdulist['LIGHTCURVE'].data
    hdulist.close()

    time = LightCurve.field('TIME')
    flux = LightCurve.field('FLUX_RAW')

    """
    # If file is a .dat use this
    data = np.loadtxt('/home/simonbanerjee/Dropbox/Speciale/
      Lightcurve_to_powerspectrum/KIC8478994/kplr008478994-%d_slc
      .dat' %i)
    time = data[:,0]
    rawflux = data[:,1]
    rawflux_error= data[:,2]
    flux = data[:,3]
    flux_error= data[:,4]
    """


    flux[np.isneginf(flux)] = np.nan


    #Normalizing by dividing y-axis values with the mean value
    flux=flux/np.nanmean(flux)

    # We will try to find tau long and and tau short using the
      find_time routine
    # The routine checks when the sign change depending on the
      chosen time target
    days_for_tau_long = 2 # 2 days
    days_for_tau_short = 2/24 # 2 hours
    tau_long= find_time(time,days_for_tau_long)
```

```
    tau_short= find_time(time,days_for_tau_short)
1060
    # The isOdd function makes sure that the number find_time
      spits outs is always odd
1062 tau_long=isOdd(tau_long)
    tau_short=isOdd(tau_short)
1064
    #Using Bottlenecks move_median fitler
1066 # Bottleneck takes the median by setting the midpoint at the
      end and the taking all the previuos points in that chunk

1068 tau_long_flux = bn.move_median(flux,tau_long,min_count=1)
    tau_short_flux= bn.move_median(flux-tau_long_flux,tau_short,
      min_count=1)+tau_long_flux
1070
    #I remove a chunk of data in the start of the data due to the
       way bottleneck calculates the median
1072 #I use my find_time function to do this
    tau_long_data_removal =find_time(time,days_for_tau_long)
1074 tau_short_data_removal =find_time(time,days_for_tau_short)

1076 # I make sure that my data starts from tau and ends with the
      last data point
    tau_long_flux = tau_long_flux[tau_long_data_removal:-1]
1078 tau_short_flux= tau_short_flux[tau_short_data_removal:-1]
    tau_long_time = time[tau_long_data_removal:-1]
1080 tau_short_time= time[tau_short_data_removal:-1]

1082 #Now I shift my data with tau/2 so that my data fits with the
       median filter
    tau_long_time = tau_long_time-days_for_tau_long/2
1084 tau_short_time= tau_short_time-days_for_tau_short/2

1086 #Here I look at how many points there is in the start and the
       end between the data set and the median filter
    tau_long_number_points_start=find_time(time,tau_long_time[0]-
      time[0])
1088 tau_long_number_points_end=find_time(time,time[-1]-
      tau_long_time[-1])

1090 tau_short_number_points_start=find_time(time,tau_short_time
      [0]-time[0])
    tau_short_number_points_end=find_time(time,time[-1]-
      tau_short_time[-1])
1092
    #Now I make some lists of ones that I am going to multiply
      with the first data point of the median filter (for left
```

```
    side) and the last data point of the median filter (right
    side)
1094 tau_long_left_extrapol = np.ones(tau_long_number_points_start
    +1)
    tau_long_right_extrapol = np.ones(tau_long_number_points_end)
1096 tau_short_left_extrapol = np.ones(
    tau_short_number_points_start+1)
    tau_short_right_extrapol = np.ones(
    tau_short_number_points_end)
1098
    # Now I multiply the ones that makes up the left hand side
    with the inital value of the median filter
1100 # and I multiply the ones that makes up the rigth hand side
    with the final value of the median filter
    tau_long_left_extrapol  = tau_long_left_extrapol  *
    tau_long_flux[0]
1102 tau_long_right_extrapol = tau_long_right_extrapol *
    tau_long_flux[-1]
    tau_short_left_extrapol = tau_short_left_extrapol *
    tau_short_flux[0]
1104 tau_short_right_extrapol= tau_short_right_extrapol*
    tau_short_flux[-1]

1106 # I make sure that the median times goes out in both ends
    tau_long_time_start = np.linspace(time[0],tau_long_time[0],
    tau_long_number_points_start+1)
1108 tau_long_time_end   = np.linspace(tau_long_time[-1], time
    [-1],tau_long_number_points_end)
    tau_long_time       = np.concatenate([tau_long_time_start,
    tau_long_time, tau_long_time_end])
1110
    tau_short_time_start= np.linspace(time[0],tau_short_time[0],
    tau_short_number_points_start+1)
1112 tau_short_time_end  = np.linspace(tau_short_time[-1], time
    [-1],tau_short_number_points_end)
    tau_short_time      = np.concatenate([tau_short_time_start,
    tau_short_time, tau_short_time_end])
1114
    # Now I make sure that the median fluxes goes out in both
    ends using a constant value in the start and a constant
    value in the end
1116 tau_long_flux = np.concatenate([tau_long_left_extrapol,
    tau_long_flux, tau_long_right_extrapol])
    tau_short_flux = np.concatenate([tau_short_left_extrapol,
    tau_short_flux, tau_short_right_extrapol])
1118
```

```
1120 #Plotting the data with the two different median filters

1122 plt.figure
     plt.plot(time,flux,'k-',time,tau_long_flux,'r-')
1124 plt.title(r'$\tau$-long')
     plt.xlabel('time [Days]')
1126 plt.ylabel('Flux')
     plt.show()
1128 #plt.savefig('median_filter_tau_long.png')


1130


1132 plt.figure
     plt.plot(time,flux,'k-',time,tau_short_flux,'r-')
1134 plt.title(r'$\tau$-short')
     plt.xlabel('time [Days]')
1136 plt.ylabel('Flux')
     plt.show()
1138 #plt.savefig('median_filter_tau_short.eps')


1140
     # To detect the presence of sharp features, I construct this
       diagnostic signal
1142 w_flux= (tau_long_flux/tau_short_flux)-1

1144 #Plotting the diagnostic signal vs time to see sharp features
     plt.figure
1146 plt.plot(time,w_flux)
     plt.title('w [diagnostic signal] vs time')
1148 plt.xlabel('Time [Days]',fontsize = 25)
     plt.ylabel('w', fontsize = 25)
1150 plt.savefig('/home/simonbanerjee/Dropbox/Speciale/
       Lightcurve_to_powerspectrum/16_Cyg_A/w_vs_time/w_vs_time_%d
       .eps'%names_of_all_stars[i,1])
     plt.clf()
1152 plt.show()


1154
     #I calculate the MAD (moving absolute deviation) of the
       diagnostic signal
1156 sigma_w_flux =running_median_filter(np.fabs(w_flux-
       running_median_filter(w_flux,tau_long)),tau_long)

1158 #Plotting the MAD to see if it's correct
     plt.figure
1160 plt.plot(time,sigma_w_flux)
     plt.title('MAD vs time')
```

```
1162    plt.xlabel('time [days]')
        plt.ylabel('MAD')
1164    plt.show()

1166    #Creating the final filter as weighted mean between the short
           and long filters
        # First we create the c values, which is the turnover
          function between the two filters
1168    # which returns 1 or 0 given sigma_w/mean_sigma_w.

1170    # These values are used as default
        mu_to = 5 # Center of the graph
1172    sigma_to = 1 # standard deviation

1174    sigma_ratio_flux= w_flux/sigma_w_flux

1176    c_flux= norm.cdf(sigma_ratio_flux,mu_to,sigma_to) # c values
          are calculated using the cumulatative distribution function

1178    # Now I make the final filter
        #filter_flux = c_flux*tau_short_flux+(1-c_flux)*tau_long_flux
1180    #this one should be used normally
        flux_filt=(10**6)*((flux/filter_flux)-1)
1182
        #used to make a figure
1184    flux_filt=((flux/filter_flux)-1)


1186
        plt.figure()
1188    plt.plot(time,flux_filt,'b')
        plt.title(r'Filter applied')
1190    plt.xlabel('Time [Days]',fontsize = 25)
        plt.ylabel('Flux [ppm]',fontsize = 25)
1192    plt.show()
        #plt.savefig('/home/simonbanerjee/Dropbox/Speciale/
          Lightcurve_to_powerspectrum/16_Cyg_A/final_filter/
          final_filter_%d.png'%names_of_all_stars[i,1])
1194
        #Now it's time to do some sigma clipping
1196    # Astropy has a sigma clipping function, however this
          function doesn't work with NaN or inf values, which means
          these must be deleted first

1198    finite_filter = np.isfinite(flux_filt) # makes a filter that
          checks in the flux column for only finite values (No, NaN,
          inf or is inf)
        #aplly filter to both time and flux
```

```
    time_filt= time[finite_filter]
    flux_filt= flux_filt[finite_filter]
    #Performing the sigma clipping to the data with only finite
      values
    filter_sigma_cliping = astropy.stats.sigma_clip(flux_filt,
      sigma=4, iters=5,stdfunc=astropy.stats.mad_std)
    time_filt= time_filt[~filter_sigma_cliping.mask]
    flux_filt = flux_filt[~filter_sigma_cliping.mask]
    #print(flux_filt.shape)

    plt.figure()
    plt.plot(time_filt,flux_filt, '-', color='#1f77b4', label="
      flux_filt")
    plt.title('Corrected time series data')
    plt.xlabel('Time [days]')
    plt.ylabel('Corrected flux [ppm]')
    plt.legend(loc='best', numpoints=1)
    plt.tight_layout() #This makes sure that the legends do not
      overlap
    plt.show()
    #plt.savefig('/home/simonbanerjee/Dropbox/Speciale/
      Lightcurve_to_powerspectrum/16_Cyg_A/timeseries/timeseries_
      %d.png'%names_of_all_stars[i,1])
    plt.clf()
    plt.show()
    np.savetxt('/home/simonbanerjee/Dropbox/Speciale/
      Lightcurve_to_powerspectrum/nu_indi/time_series_for_%d.txt'
       %names_of_all_stars[i],np.transpose([time_filt,flux_filt])
      )
```

/home/simonbanerjee/Dropbox/Speciale/Lightcurve_to_powerspectrum/timeseries_to_thesis.py.

Creating The Power Spectrum

```
#numpy
import os
import numpy as np
import math, os, sys, time, random
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
#functions
from time_series_powerspectrum import power_spectrum
#plt.rc('text', usetex=True)
#plt.rc('font', family='serif')
##########################################################
```

```
names_of_all_stars= np.loadtxt('list_of_all_quaters.txt')
for i in names_of_all_stars:
    data = np.loadtxt('/home/simonbanerjee/Dropbox/Speciale/
    Lightcurve_to_powerspectrum/16_Cyg_A/time_series_for_%d.txt
    ' %i)
    time = data[:,0]
    amplitude = data[:,1]
    amplitude = amplitude/np.mean(amplitude)
    #amplitude= (amplitude)*10**6

    #weight=data[:,2]

    freq, power, _, _ = power_spectrum(time, amplitude)
    freq =freq.reshape(-1)
    power = power.reshape(-1)
    np.savetxt('/home/simonbanerjee/Dropbox/Speciale/
    Lightcurve_to_powerspectrum/16_Cyg_A/power_spec_16_Cyg_A/
    powerspectrum_for_%d.txt' %i,np.transpose([freq,power]))


    plt.figure()
    plt.plot(freq, power, '-')
    #plt.title(r'KIC8478994' )
    plt.xlabel(r'Frequency [$\mu$Hz]')
    plt.ylabel(r'Power density [ppm$^2\, \mu$Hz$^{-1}$]')
    #plt.savefig('Tidsserie_Analyse_Kepler_star_2.png')
    plt.show()
```

/home/simonbanerjee/Dropbox/Speciale/Lightcurve_to_powerspectrum/powerspectrum.py.

Asteroseismic Analysis

```
import numpy as np
import statistics as stat
import math
import astropy
import matplotlib.pyplot as plt
import glob
import scipy
import time
import sys
#sys.path.insert(0, '/home/simonbanerjee/Dropbox/Speciale/
    Lightcurve_to_powerspectrum')
sys.path.insert(0, r"/home/simonbanerjee/Dropbox/Speciale/
    Lightcurve_to_powerspectrum")
from astropy.convolution import Gaussian1DKernel, convolve
#from scipy.signal import find_peaks
```

```
from peak_finder import peak_finder
from filter import autocorr, running_mean_filter, isOdd,
    find_time
from scipy.optimize import curve_fit
from astropy.modeling import models, fitting
from time_series_powerspectrum import power_spectrum,
    calculate_v_nl, calculate_MF, matched_filter,
    fit_large_freq_separation
from time_series_powerspectrum import background_fit_2,
    background_fit, logbackground_fit, gridsearch,
    running_median,  linear_regression
from math import e
from uncertainties import ufloat
from scipy import optimize
from scipy.optimize import curve_fit
#matplotlib_setup()
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import
    zoomed_inset_axes
from scipy.signal import find_peaks, peak_prominences
from scipy import stats
import statsmodels.api as sm


from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused
    import
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

#matplotlib_setup()
import matplotlib.pyplot as plt
import seaborn as sns

# Activate Seaborn color aliases
#sns.set_palette('colorblind')
#sns.set_color_codes(palette='colorblind')
#plt.style.use('ggplot')
#sns.set_context('poster')
#sns.set_style("ticks")
plt.style.use('seaborn-poster')
plt.style.use('seaborn-colorblind')
#sns.set()

stepsize = 0.1 #mHz
#nyquist=4200 #mHz
data= np.loadtxt(r"/home/simonbanerjee/Dropbox/Speciale/
    Lightcurve_to_powerspectrum/nu_indi/
```

```
    powerspectrum_for_317019578.txt")
#data= np.loadtxt(r"power_other_half.txt")
freq = data[:,0]
power = data[:,1]



power_final_smoothed =power
powerden=power_final_smoothed

##
    ###########################################################################

"""
Calculating the background using a modified Harvey profile
"""
##
    ###########################################################################


nu_max = 340
P_n = np.arange(0.1, 0.25, step=0.05)  #[np.median(powerden[
    freq > f]) for f in np.arange(2000, 6000, step=500)]
guess_sigma_0 =  [n * np.sqrt(np.mean(powerden ** 2)) for n in
    np.arange(10, 50, step=5)]
guess_tau_0 = [n * (1 / nu_max) for n in np.arange(0.01, 0.2,
    step=0.05)]
guess_sigma_1 = [n * np.sqrt(np.mean(powerden ** 2)) for n in
    np.arange(10, 50, step=5)]
guess_tau_1 = [n * (1 / nu_max) for n in np.arange(0.01, 0.2,
    step=0.05)]

print('Parameterspace is %f-%f, %f-%f, %f-%f, %f-%f, and %f-%f'
     % (
    np.min(guess_sigma_0), np.max(guess_sigma_0), np.min(
    guess_tau_0), np.max(guess_tau_0),
    np.min(guess_sigma_1), np.max(guess_sigma_1), np.min(
    guess_tau_1), np.max(guess_tau_1),
    np.min(P_n), np.max(P_n)))

# Cut out around the signals in order not to overfit them
minimum = 500
maximum = 1500

filt = (freq > minimum) & (freq < maximum)
freq_filt = freq[~filt]
powerden_filt = powerden[~filt]
```

```
1086
1088 freq_filt, powerden_filt, ws = running_median(freq_filt,
         powerden_filt, bin_size=1e-3)
1090 def cost(popt):
         return np.mean((logbackground_fit(freq_filt, *popt) - np.
         log10(powerden_filt)) ** 2)
1092
     freq_fit, powerden_fit, ws = running_median(freq, powerden,
         bin_size=1e-4)
1094
     z0 = [guess_sigma_0, guess_tau_0, guess_sigma_1, guess_tau_1,
         P_n]
1096 popt = gridsearch(logbackground_fit, freq_fit, np.log10(
         powerden_fit), z0)
     # popt = [52.433858, 0.000885, 81.893752, 0.000167, 0.220056]
1098
     print('Best parameter for background were: s_0= %f, t_0= %f,
         s_1= %f, t_1 =%f, P_n= %f' % tuple(popt))
1100 # Fit
     #z0 = [guess_sigma_0, guess_tau_0, guess_sigma_1, guess_tau_1]
1102 popt, pcov = curve_fit(logbackground_fit, freq_fit, np.log10(
         powerden_fit), p0=popt, maxfev=10000)
1104 print('Best parameter for background were: s_0= %f, t_0= %f,
         s_1= %f, t_1 =%f, P_n= %f' % tuple(popt))
1106 print('Cost = %f' % cost(popt))
     freq_plot = freq#[::1000]
1108 powerden_plot = powerden#[::1000]
     #rpowerden_plot = rpowerden[::1000]
1110
     f1=plt.figure(1)
1112 plt.loglog(freq_plot, powerden_plot, '0.2', basex=10, basey=10,
          linewidth=0.5)
     plt.loglog(freq_plot, background_fit(freq_plot, *popt), '
         steelblue', linestyle='-', basex=10,
1114                 basey=10, linewidth=10)
     #plt.loglog(freq_plot, popt[4] + background_fit_2(freq_plot, *
         popt[:2]), 'steelblue', linestyle='--',
1116 #                basex=10, basey=10)
     #plt.loglog(freq_plot, popt[4] + background_fit_2(freq_plot, *
         popt[2:4]), 'steelblue', linestyle='--',
1118 #                basex=10, basey=10)
     #plt.loglog(freq_plot, np.ones(len(freq_plot)) * popt[4], '
         royalblue', linestyle='--')
```

```
# plt.title(r'The power density spectrum of %s' % starname)
plt.xlabel(r'Frequency [$\mu$Hz]',fontsize = 35)
plt.ylabel(r'Power density [ppm$^2\, \mu$Hz$^{-1}$]', fontsize=35)
plt.xlim([100,1000 ])
plt.ylim([10 ** (-2), 2 * 10 ** (3)])
ax = f1.add_subplot(111)
ax.tick_params(axis='both', which='major', labelsize=35)
ax.tick_params(axis='both', which='minor', labelsize=35)
#plt.title(r'$\beta$ Hydri', fontsize=35)
plt.tight_layout()
#plt.savefig('/home/simonbanerjee/Dropbox/Speciale/Master_Thesis/background_nu_indi.eps')

#plt.show()
#plt.savefig('%s_backgroundfit_%s_%s.pdf' % (starname,
                #minfreq, maxfreq), bbox_inches='tight')

powerden_background_corrected = powerden - background_fit(freq_plot, *popt)


##############################################################################

"""
Smoothing the power spectrum using a Gaussian
"""
##############################################################################

xstart=1700
xend= 5000
#xstart=850
#xend=2500

#std=4
std=2
g= Gaussian1DKernel(stddev=std,mode='integrate')
power_final_smoothed=convolve(power,g)
powerden=power_final_smoothed

##############################################################################

#saving file of the weigthed powerspectrum to use to find nu_max in file nu_max.py
```

```
#np.savetxt('weighted_powerspec_TIC71134596.txt',np.transpose([freq,power_final_smoothed]))


##############################################################################

freq_window=freq[xstart:xend]
#power_final = power_final[35000:50000]
power_final_smoothed=power_final_smoothed[xstart:xend]

##############################################################################

"""
Finding the peaks
"""
##############################################################################


"""
comparorder = 1
height1 = 1.0
height2 = 1.0
height3 = 0.8
height4 = 0.5


freq_peak1 , power_peak1 = peak_finder(freq_window[xstart-xstart:2800-xstart],power_final_smoothed[xstart-xstart:2800-xstart],height1,comparorder)
freq_peak2 , power_peak2 = peak_finder(freq_window[2850-xstart:3300-xstart],power_final_smoothed[2850-xstart:3300-xstart],height2,comparorder)
freq_peak3 , power_peak3 = peak_finder(freq_window[3350-xstart:3850-xstart],power_final_smoothed[3350-xstart:3850-xstart],height3,comparorder)
freq_peak4 , power_peak4 = peak_finder(freq_window[3900-xstart:6000-xstart],power_final_smoothed[3900-xstart:6000-xstart],height4,comparorder)

freq_peak =np.concatenate((freq_peak1,freq_peak2,freq_peak3,freq_peak4), axis=0)
```

```
power_peak = np.concatenate((power_peak1,power_peak2,
    power_peak3,power_peak4), axis=0)
"""

power_peak, properties = find_peaks(power_final_smoothed,
    prominence=0.7)
prominences = peak_prominences(power_final_smoothed, power_peak
    )[0]
print(prominences)


##
    ###############################################################################

"""
Performing the autocorrelation and estimating Dnu
"""
##
    ###############################################################################



#Running the autocorrelation
freq_lags = round(len(freq_window)/2)
ACF = autocorr(power_final_smoothed,lags=freq_lags)

# autocorr don't give you x-values, so these you must obtain
#I know the timesteps and the range of my frequencies
lags= np.arange(len(ACF))*stepsize
#lags = lags[500:-1]
#ACF = ACF[500:-1]
lags = lags
ACF = ACF
#max_ACF = max(ACF)  # Find the maximum y value
#delta_nu = lags[ACF.argmax()]  # Find the x value
    corresponding to the maximum y value
#np.savetxt('delta_nu.txt',np.fabs(delta_nu))
#print("delta nu is %.20f microhertz using autocorrelation
    before subtracting background" %delta_nu)

p1=0.35
p2= 0.06
p3= 0.2
p4= 0.06
p5= 0.05
p6= 0.05
```

```
p7= 0.05
c= 2.5
delta_nu_fit_me = 25
K=0.05

fit_me = p1*np.exp(-(lags-0.5*delta_nu_fit_me)**2/c**2) + p2*np
    .exp(-(lags-delta_nu_fit_me)**2/c**2)+p3*np.exp(-(lags-1.5*
    delta_nu_fit_me)**2/c**2) + p4*np.exp(-(lags-2*
    delta_nu_fit_me)**2/c**2)+ p5*np.exp(-(lags-2.5*
    delta_nu_fit_me)**2/c**2) + p6*np.exp(-(lags-3*
    delta_nu_fit_me)**2/2**2)+p7*np.exp(-(lags-3.5*
    delta_nu_fit_me)**2/c**2) + K

popt, pcov = curve_fit(fit_large_freq_separation, lags, ACF, p0
    =[p1, p2, p3, p4, p5, p6, p7, delta_nu_fit_me, K, c])
print(popt)
print("delta nu is %.20f microhertz using autocorrelation " %
    popt[7])
delta_nu = popt[7]
#delta_nu =25
freq_mod_delta_nu =np.mod(freq_window[power_peak],delta_nu)



f2=plt.figure(2)
plt.plot(lags,ACF)
plt.plot(lags, fit_large_freq_separation(lags, *popt), 'r-',
    label='Gaussian Fit')
xcoords = [0.5*delta_nu, delta_nu, 1.5*delta_nu, 2*delta_nu,
    2.5*delta_nu, 3*delta_nu, 3.5*delta_nu, 4*delta_nu, 4.5*
    delta_nu, 5*delta_nu, 5.5* delta_nu, 6*delta_nu, 6.5*
    delta_nu, 7*delta_nu]
for xc in xcoords:
    plt.axvline(x=xc,color='lightgrey', linestyle='--')
plt.legend(loc='upper right', prop={'size': 35})
plt.text(delta_nu-5, 0.3, r'$\Delta \nu$',fontsize=35)
plt.xlabel(r'Frequency Shift ($\mu$Hz)', fontsize=35)
plt.ylabel(r'Autocorrelation',fontsize=35)
ax2 = f2.add_subplot(111)
ax2.tick_params(axis='both', which='major', labelsize=35)
ax2.tick_params(axis='both', which='minor', labelsize=35)
plt.title(r' Autocorrelation for $\nu$ Indi',fontsize=35)
plt.tight_layout()
#plt.savefig('/home/simonbanerjee/Dropbox/Speciale/
    Master_Thesis/autocorr_nu_indi.eps')

#plt.show()
```

```
#plt.xlabel(r'Frequency mod %0.2f $\mu$hertz' %delta_nu,
    fontsize =35)
#plt.ylabel(r'Frequency ($\mu hertz$)',fontsize =35)
#ax4 = f4.add_subplot(111)
#ax4.tick_params(axis='both', which='major', labelsize=35)
#ax4.tick_params(axis='both', which='minor', labelsize=35)


##############################################################################
"""
Plotting the power spectrum and echelle diagram
"""
##############################################################################



f3=plt.figure(3)
color ='Blues'
#plt.subplot(2, 1, 1)
#plt.plot(freq_window,np.sqrt(power[xstart:xend]),'#d6d8db')
plt.plot(freq_window,power_final_smoothed,'k-')
plt.plot(freq_window[power_peak], power_final_smoothed[
    power_peak], "v")
#plt.plot(freq_peak,np.sqrt(power_peak),'x')
#plt.title('Smoothed, stddev=%d'%std)
plt.xlabel(r'$Frequency \mu$Hz')
plt.ylabel(r'Power density [ppm$^2\, \mu$Hz$^{-1}$]')
plt.title(r'$\nu$ indi')
#plt.grid(True)

#plt.subplot(2, 1, 2)
f4=plt.figure(4)
#plt.scatter(freq_mod_delta_nu,freq_peak, c=power_peak, cmap=
    color)
#plt.scatter(delta_nu+freq_mod_delta_nu,freq_peak, c=power_peak
    , cmap=color)
#plt.plot(freq_mod_delta_nu,freq_window[power_peak],'bo',
    markersize= 15)
#plt.plot(identified_modes, freq_interpolation, 'kv')
#N=len(freq_mod_delta_nu)
#np.savetxt('echelle_data.txt',np.transpose([freq_mod_delta_nu,
    freq_window[power_peak], prominences]))
#area = (30 * np.random.rand(N))**2  # 0 to 15 point radii
echelle_non_sig= np.loadtxt('echelle_data.txt')
```

```
1296 echelle_most_sig = np.loadtxt('echelle_data_most_sig.txt')
     x_non = echelle_non_sig[:,0]
1298 y_non = echelle_non_sig[:,1]
     prom_non= echelle_non_sig[:,2]
1300 x_most = echelle_most_sig[0]
     y_most= echelle_most_sig[1]
1302 prom_most= echelle_most_sig[2]


1304

     plt.scatter(x_non,y_non,s=(prom_non**4.5),c='blue', edgecolors=
         'k', linewidths=5,alpha=0.5)
1306 plt.scatter(x_most,y_most,s=(prom_most**3),c='blue', edgecolors
         ='k', linewidths=5,alpha=0.5)

1308 #np.savetxt('echelle_diagram_for_l0.txt',np.transpose([
         freq_mod_delta_nu,freq_window[power_peak]]))

1310 #plt.plot(delta_nu+freq_mod_delta_nu,freq_peak,'bo')

1312 plt.title(r'Echelle Diagram for $\nu$ Indi',fontsize =35)
     plt.xlabel(r'Frequency mod %0.2f $\mu$Hz' %delta_nu,fontsize
         =35)
1314 plt.ylabel(r'Frequency ($\mu$Hz)',fontsize =35)
     ax4 = f4.add_subplot(111)
1316 ax4.tick_params(axis='both', which='major', labelsize=35)
     ax4.tick_params(axis='both', which='minor', labelsize=35)
1318 plt.grid(True)
     plt.tight_layout()
1320 #plt.show()
     plt.savefig('/home/simonbanerjee/Dropbox/Speciale/Master_Thesis
         /echelle_nu_indi.eps')
1322



1324

     ##
         ########################################################################

1326 """
     Finding nu_max
1328 """
     ##
         ########################################################################


1330



1332 std=(4*(delta_nu))/(2*np.sqrt(2*np.log(2))) # I smooth the the
         with Gaussian kernal and a std of 4*delta_nu
```

```
xstart_after_background=xstart
xend_after_background=xend
freq=freq[xstart_after_background:xend_after_background]
powerden=powerden[xstart_after_background:xend_after_background]

g1= Gaussian1DKernel(stddev=std/stepsize,mode='integrate')
power_gauss2=convolve(powerden,g1)
max_power_smoothed2 = max(power_gauss2)  # Find the maximum y value
nu_max = freq[power_gauss2.argmax()]  # Find the x value corresponding to the maximum y value
print("nu max is %.2f microhertz using gaussian smooth after subtracting the background" %nu_max)


f5= plt.figure(5)
ax3 = plt.gca()
plt.plot(freq,powerden,'b-')
plt.plot(freq, (power_gauss2),'r-',label="Gaussian Fit", linewidth = 10)
plt.legend(prop={'size': 30})
plt.xlabel(r'Frequency [$\mu$Hz]', fontsize= 30)
plt.ylabel(r'Power density [ppm$^2\, \mu$Hz$^{-1}$]',fontsize=30)
ax4 = f4.add_subplot(111)
ax4.tick_params(axis='both', which='major', labelsize=30)
ax4.tick_params(axis='both', which='minor', labelsize=30)
plt.tight_layout()
#plt.savefig('/home/simonbanerjee/Dropbox/Speciale/Master_Thesis/numax_nu_indi.eps')
#plt.show()

##############################################################################

"""
Using the matched filter to estimate dnu02
"""
##############################################################################

#Now I wanna find the small frequency seperation dnu_02 using a matched filter
dnu02_start= 1
dnu02_end=9
epsilon_start= 1.5
```

```
epsilon_end= 0.5
resolution = 250

dnu02 = np.linspace(dnu02_start, dnu02_end, resolution)
epsilons = np.linspace(epsilon_start, epsilon_end, resolution)
MF = np.zeros((len(dnu02), len(epsilons)), dtype='float64')
    # opsamler data

iter = 0
for k, epsilon in enumerate(epsilons):
    percentage = iter/len(epsilons)*100
    print("The match filter is %f%% done" %percentage)
    iter=iter+1
    for j, dnu in enumerate(dnu02):
        MF[k,j] = matched_filter(epsilon, dnu, freq_window[
    power_peak],delta_nu)



X, Y = np.meshgrid(epsilons, dnu02)
value_epsilon, value_dnu02 = np.unravel_index(np.argmax(MF), MF
    .shape)
print("epsilon;", epsilons[value_epsilon])
print("dnu02;", dnu02[value_dnu02])

epsilon= epsilons[value_epsilon]
dnu02 = dnu02[value_dnu02]
#np.savetxt('epsilon_and_dnu02_beta_hydri.txt',(epsilon,dnu02))


f5 = plt.figure(5)
ax = f5.gca(projection='3d')
x=np.linspace(dnu02_start,dnu02_end,num=len(MF))
y=np.linspace(epsilon_start,epsilon_end, num= len(MF))
X, Y = np.meshgrid(x,y)
Z=MF
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)
f5.colorbar(surf, shrink=0.5, aspect=5)
ax.set_xlabel('\n' + r'$\delta\nu_{02}$', linespacing=4)
ax.set_ylabel('\n' + r'$\varepsilon$', linespacing=4)
plt.tight_layout()
#plt.savefig('/home/simonbanerjee/Dropbox/Speciale/
    Master_Thesis/dnu02_3D_nu_indi.eps')



```

```
f6=plt.figure(6)
plt.imshow(MF, origin='lower', extent=[dnu02_start, dnu02_end,
    epsilon_start, epsilon_end])
plt.title(r'Matched filter for $\nu$ Indi',fontsize =30)
plt.xlabel(r'$\delta\nu_{02}$',fontsize=30)
plt.ylabel(r'$\varepsilon$',fontsize=30)
ax6 = f6.add_subplot(111)
ax6.tick_params(axis='both', which='major', labelsize=30)
ax6.tick_params(axis='both', which='minor', labelsize=30)
cbar=plt.colorbar()
cbar.ax.tick_params(labelsize=30)
plt.axis('auto')
plt.tight_layout()
#plt.savefig('/home/simonbanerjee/Dropbox/Speciale/
    Master_Thesis/dnu02_2D_nu_indi.eps')

##
    ###########################################################################

"""
Calculating the uncertainty for the asteroseismic parameters
    Delta_nu and
delta_nu02
"""
##
    ###########################################################################


##
    ###########################################################################

#uncertainty for Delta nu
##
    ###########################################################################

#np.savetxt('echelle_diagram_only_0.txt',np.transpose([
    freq_mod_delta_nu,freq_window[power_peak]]))
v0n_data=np.loadtxt("echelle_diagram_for_0.txt")
v0n_mod=v0n_data[:,0]
v0n=v0n_data[:,1]

n=(v0n/delta_nu)-epsilon
n=[ round(elem, 0) for elem in n ]
n=np.asarray(n)
print(n)

```

```
1446
Polynomial = np.polynomial.Polynomial
1448
# The data: conc = [P] and absorbance, A
1450
cmin, cmax = min(n), max(n)
1452 pfit, stats = Polynomial.fit(n, v0n, 1, full=True, window=(cmin
    , cmax),
                                                    domain=(
    cmin, cmax))
1454 print('Raw fit results:', pfit, stats, sep='\n')

1456 A0, m = pfit
resid, rank, sing_val, rcond = stats
1458 rms = np.sqrt(resid[0]/len(n))

1460 print('Fit: v0n = {:.3f}[n] + {:.3f}'.format(m, A0),
      '(rms residual = {:.4f})'.format(rms))
1462
print(linear_regression(n,v0n,0.95))
1464 confidence_interval_Delta_nu= linear_regression(n,v0n,0.95)
uncertainty_for_Delta_nu = (confidence_interval_Delta_nu[1]-
    confidence_interval_Delta_nu[0])/4
1466 print(r" The uncertainty for Delta nu is %0.2f plus/minus %0.2f
    " %(delta_nu,uncertainty_for_Delta_nu))


1468
f7 = plt.figure(7)
1470 plt.plot(n, v0n, 'ko')
plt.plot(n, pfit(n), color='r')
1472 plt.title('order vs freq for l=0')
plt.xlabel(r'n')
1474 plt.ylabel(r'$v_n$')
#plt.show()
1476


1478 ##
    ##############################################################################

#uncertainty for delta_nu02
1480 ##
    ##############################################################################

#np.savetxt('echelle_diagram_only_2.txt',np.transpose([
    freq_mod_delta_nu,freq_window[power_peak]]))
1482 v2n_data=np.loadtxt("echelle_diagram_for_2.txt")
v2n_mod = v2n_data[:,0]
```

```
v2n=v2n_data[:,1]

n2=(v2n/delta_nu)-epsilon
n2=[ round(elem, 0) for elem in n2 ]
n2=np.asarray(n2)
print(n2)


##############################################################################

#Interpolation to get more points
"""
In this case for l=0 I have order
[ 7. 11. 13. 14. 17.]

for l=2 I have
[ 8.  9. 10. 11. 12. 13. 14.]

so in this case I will need to estimate the delta_nu_mod_freq values for order
n= 12 for l=0
this is done by taking the two surrounding values and taking the mean of them,
it is rough but it will do
not done for this star
"""

l0_11 = v0n_mod[1]
l0_12 = np.mean([v0n_mod[1],v0n_mod[2]])
l0_13 = v0n_mod[2]
l0_14 = v0n_mod[3]


l2_10= v2n_mod[2]
l2_11 = v2n_mod[3]
l2_12 = v2n_mod[4]
l2_13 = v2n_mod[5]



modes_l0= [l0_11,l0_12, l0_13, l0_14]
modes_l2= [l2_10, l2_11, l2_12, l2_13]
diff_delta_nu02= np.subtract(modes_l0,modes_l2)
print(diff_delta_nu02)


```

```
N=len(diff_delta_nu02)
for i in range(len(diff_delta_nu02)):
    mu_delta_nu02=np.sum(diff_delta_nu02)/6

print(mu_delta_nu02)

diff_delta_nu02[:] = [x - mu_delta_nu02 for x in
    diff_delta_nu02]
delta_nu_02_minus_mu_delta_nu02 = np.abs(diff_delta_nu02)
print(delta_nu_02_minus_mu_delta_nu02)

#calculating the variance
var_delta_nu02 = (1/(N-1))*np.sum(
    delta_nu_02_minus_mu_delta_nu02)**2
print(var_delta_nu02)
N=5
print(N)
#uncertainty for delta_nu02
uncertainty_for_delta_nu02 = np.sqrt(var_delta_nu02/N)
print(r" The uncertainty for delta_nu02 is %0.4f plus/minus
    %0.2f" %(dnu02,uncertainty_for_delta_nu02))


##
    ###########################################################################
#End of uncertainty for asteoseismic parameters
##
    ###########################################################################


##
    ###########################################################################
"""
Estimating mass, radius and age using 3 different scaling
    relations
"""
##
    ###########################################################################



#Performing scaling relations using Earl Bellinger new approach
    for main-sequence stars
```

```
1562
1564 Teff_nu_indi =  5300 # K from bedding 2006: SOLAR-LIKE
     OSCILLATIONS IN THE METAL-POOR SUBGIANT : INDI:CONSTRAINING
     THE MASS AND AGE USING ASTEROSEISMOLOGY
     Fe_H_nu_indi = -1.4 # dex from bedding 2006: SOLAR-LIKE
     OSCILLATIONS IN THE METAL-POOR SUBGIANT : INDI:CONSTRAINING
     THE MASS AND AGE USING ASTEROSEISMOLOGY
1566 age_Sun = ufloat(4.569, 0.006)

1568 # Enter some data, for example a solar twin
     # ufloat holds a measurement and its uncertainty
1570 nu_max_star= ufloat(nu_max, nu_max * 0.01) # muHz, with 1%
     uncertainty
     delta_nu_star= ufloat(delta_nu, delta_nu * 0.001) # muHz, with
     0.1% uncertainty
1572 d02_nu_star= ufloat(dnu02, dnu02 * 0.04) # muHz, with 4%
     uncertainty
     Teff_star= ufloat(Teff_nu_indi,Teff_nu_indi * 0.01) # K, with
     1% uncertainty
1574 Fe_H_star= ufloat(Fe_H_nu_indi, 0.1) # dex, 0.1 dex uncertainty

1576
     # Take the powers from Table 1, here given with more precision
1578 #P        = [       alpha,            beta,             gamma,
           delta,         epsilon]
     P_age     = [-6.55598425,      9.05883854,      -1.29229053,
     -4.24528340,     -0.42594767]
1580 P_mass    = [ 0.97531880,     -1.43472745,                 0,
     1.21647950,      0.27014278]
     P_radius = [ 0.30490057,     -1.12949647,                 0,
     0.31236570,      0.10024562]
1582 P_R_seis = [ 0.88364851,     -1.85899352,                 0,
                0,               0]

1584 # Apply the scaling relation
     def scalingrelations(nu_max, Delta_nu, delta_nu, Teff, Fe_H, P=
     P_age,
1586              nu_max_Sun = ufloat(3090, 30),       # muHz
                  Delta_nu_Sun = ufloat(135.1, 0.1),  # muHz
1588              delta_nu_Sun = ufloat(8.957, 0.059),# muHz
                  Teff_Sun = ufloat(5772, 0.8),        # K
1590              Fe_H_Sun = ufloat(0,     0)):        # dex

1592      alpha, beta, gamma, delta, epsilon = P
```

```
1594     # Equation 5
         return ((nu_max / nu_max_Sun) ** alpha *
1596         (Delta_nu / Delta_nu_Sun) ** beta *
             (delta_nu / delta_nu_Sun) ** gamma *
1598         (Teff / Teff_Sun) ** delta *
             (e**Fe_H / e**Fe_H_Sun) ** epsilon)
1600

1602 scaling_mass = scalingrelations(nu_max_star, delta_nu_star,
         d02_nu_star, Teff_star, Fe_H_star, P=P_mass)
     scaling_radius = scalingrelations(nu_max_star, delta_nu_star,
         d02_nu_star, Teff_star, Fe_H_star, P=P_radius)
1604 scaling_age = scalingrelations(nu_max_star, delta_nu_star,
         d02_nu_star, Teff_star, Fe_H_star, P=P_age) * age_Sun

1606 print('Scaling relations using Earl Bellingers scaling
         relations for main sequence stars')
     print('Mass:',scaling_mass,'[solar units]')
1608 print('Radius:', scaling_radius, '[solar units]')
     print('Age:', '{:.2u}'.format(scaling_age), '[Gyr]')
1610



1612



1614

     """
1616 #Performing scaling relations using Earl Bellinger new approach
         for giants
     """
1618



1620

     # Enter some data, for example a solar twin
1622 # ufloat holds a measurement and its uncertainty
     nu_max_giant= ufloat(nu_max, nu_max * 0.01) # muHz, with 1%
         uncertainty
1624 delta_nu_giant= ufloat(delta_nu, delta_nu * 0.001) # muHz, with
          0.1% uncertainty
     #period_spacing_giant= ufloat(1,  0.04) # muHz, with 4%
         uncertainty
1626 Teff_star_giant= ufloat(Teff_nu_indi,Teff_nu_indi * 0.01) # K,
         with 1% uncertainty
     Fe_H_star_giant= ufloat(Fe_H_nu_indi, 0.1) # dex, 0.1 dex
         uncertainty
1628


1630 # Take the powers from Table 1, here given with more precision
```

```
#P       = [       alpha,              beta,
    delta,          epsilon]
P_age_giant     = [      -8.65,              11.68,
           -10.35,          0.2914]
P_mass_giant    = [      3.176,             -4.195,
           1.076,          -0.0704]
P_radius_giant = [       1.079,             -2.091,
           0.3709,         -0.02565]


# Apply the scaling relation
def scalingrelations_giant(nu_max, Delta_nu, Teff, Fe_H,
    P_giant=P_age_giant,
            nu_max_Sun = ufloat(3090, 30),      # muHz
            Delta_nu_Sun = ufloat(135.1, 0.1),  # muHz
            Teff_Sun = ufloat(5772, 0.8),       # K
            Fe_H_Sun = ufloat(0,    0)):        # dex

    alpha_giant, beta_giant, delta_giant, epsilon_giant =
    P_giant

    # Equation 5
    return ((nu_max / nu_max_Sun) ** alpha_giant *
        (Delta_nu / Delta_nu_Sun) ** beta_giant *
        (Teff / Teff_Sun) ** delta_giant *
        (e**Fe_H / e**Fe_H_Sun) ** epsilon_giant)


scaling_mass_giant = scalingrelations_giant(nu_max_giant,
    delta_nu_giant,  Teff_star_giant, Fe_H_star_giant, P_giant=
    P_mass_giant)
scaling_radius_giant = scalingrelations_giant(nu_max_giant,
    delta_nu_giant, Teff_star_giant, Fe_H_star_giant, P_giant=
    P_radius_giant)
scaling_age_giant = scalingrelations_giant(nu_max_giant,
    delta_nu_giant,  Teff_star_giant, Fe_H_star_giant, P_giant=
    P_age_giant) * age_Sun

print('Scaling relations using Earl Bellingers scaling
    relations for giants')
print('Mass:',scaling_mass_giant,'[solar units]')
print('Radius:', scaling_radius_giant, '[solar units]')
print('Age:', '{:.2u}'.format(scaling_age_giant), '[Gyr]')
```

```
"""
#Performing scaling relations using the normal way
"""



# Enter some data, for example a solar twin
# ufloat holds a measurement and its uncertainty
nu_max_star_normal= ufloat(nu_max, nu_max * 0.01) # muHz, with 1% uncertainty
Teff_star_normal= ufloat(Teff_nu_indi, Teff_nu_indi * 0.01) # K, with 1% uncertainty
delta_nu_star_normal= ufloat(delta_nu, delta_nu * 0.001) # muHz, with 0.1% uncertainty

#P                = [       alpha,             beta,             gamma]
P_mass_normal    = [          3,                -4,               3/2]
P_radius_normal = [          1,                -2,               1/2]


# Apply the scaling relation
def scalingrelations_normal(nu_max, Delta_nu, Teff, P_normal=P_mass_normal,
            nu_max_Sun = ufloat(3090, 30),       # muHz
            Delta_nu_Sun = ufloat(135.1, 0.1),   # muHz
            Teff_Sun = ufloat(5772, 0.8)):        # K

    alpha_normal, beta_normal, gamma_normal = P_normal

    # Equation 5
    return ((nu_max / nu_max_Sun) ** alpha_normal *
        (Delta_nu / Delta_nu_Sun) ** beta_normal *
        (Teff / Teff_Sun) ** gamma_normal)


scaling_mass_normal = scalingrelations_normal(
    nu_max_star_normal, delta_nu_star_normal, Teff_star_normal,
     P_normal=P_mass_normal)
scaling_radius_normal = scalingrelations_normal(
    nu_max_star_normal, delta_nu_star_normal, Teff_star_normal,
     P_normal=P_radius_normal)

print('Scaling relations using the standard method')
```

```
print('Mass:',scaling_mass_normal,'[solar units]')
print('Radius:', scaling_radius_normal, '[solar units]')
```

/home/simonbanerjee/Dropbox/Speciale/Lightcurve_to_powerspectrum/nu_indi/nu_indi.py

Subcodes Including Functions To Main Code

```
import numpy as np

#*****************************************************

from collections import deque,Counter
from bisect import insort, bisect_left
from itertools import islice

from scipy.interpolate import interp1d
from scipy import arange, array, exp
import scipy
from scipy.signal import medfilt
import bottleneck as bn
from bottleneck import move_median


def RunningMedian(seq, M):
    """
     Purpose: Find the median for the points in a sliding
    window (odd number in size)
             as it is moved from left to right by one point at
     a time.
       Inputs:
            seq -- list containing items for which a running
    median (in a sliding window)
                 is to be calculated
           M -- number of items in window (window size) --
    must be an integer > 1
      Otputs:
         medians -- list of medians with size N - M + 1
       Note:
          1. The median of a finite list of numbers is the "
    center" value when this list
             is sorted in ascending order.
          2. If M is an even number the two elements in the
    window that
             are close to the center are averaged to give the
    median (this
             is not by definition)
    """
```

```
    seq = iter(seq)
    s = []
    m = M // 2

    # Set up list s (to be sorted) and load deque with first
    window of seq
    s = [item for item in islice(seq,M)]
    d = deque(s)

    # Simple lambda function to handle even/odd window sizes
    median = lambda : s[m] if bool(M&1) else (s[m-1]+s[m])*0.5

    # Sort it in increasing order and extract the median ("
    center" of the sorted window)
    s.sort()
    medians = [median()]

    # Now slide the window by one point to the right for each
    new position (each pass through
    # the loop). Stop when the item in the right end of the
    deque contains the last item in seq
    for item in seq:
        old = d.popleft()           # pop oldest from left
        d.append(item)              # push newest in from right
        del s[bisect_left(s, old)] # locate insertion point and
     then remove old
        insort(s, item)             # insert newest such that
    new sort is not required
        medians.append(median())
    return medians

def binary_search(array, target):
    lower = 0 # mimimum value
    upper = len(array)-1 # maximum value
    while lower < upper:   # use < instead of <=
        x = lower + (upper - lower) // 2
        val = array[x]
        if target == val:
            return x
        elif target > val:
            if lower == x:   # these two are the actual lines
                break        # you're looking for
            lower = x
        elif target < val:
            upper = x

def running_median_filter(x, N):
```

```
    """
    >>> running_median_filter([0, 3, 9, 18, 24], 5)
    array([  0.,   3.,   9.,  18.,  24.])
    """
    assert N % 2 == 1
    # Code cannot run unless timetarget is an odd number. Time
    target can be even, like 6 days,
    #but the number of data points in the data it gives has to
    be odd, for example like 6 days is after 7613 data point
    x = np.asarray(x)
    extra_count = (N - 1) // 2
    left = []
    right = []
    #This for-loop makes an extrapolation in both ends of the
    median filter dataself.
    #This makes sure that the data sets is of equal length
    #The way it works is following
    #Say you're time target is 11 days then there is 5 days in
    each ends
    #Here we take make tiny chunks of the points in the end of
    the data and then take the median of that
    # Then we take a tinier chunk and a tinier chunk and moving
     more and more towards the center of the end pieces
    for i in range(extra_count):
        left.append(np.median(x[:2*i+1]))
        right.append(np.median(x[-2*i-1:]))
    right.reverse()
    y = running_median(x, N)
    return np.concatenate([left, y, right])


def running_median(x, N):
    """
    >>> running_median([0, 1, 2, 3, 4], 3)
    array([1, 2, 3])
    """
    assert N % 2 == 1
    # Code cannot run unless timetarget is an odd number. Time
    target can be even, like 6 days,
    #but the number of data points in the data it gives has to
    be odd, for example like 6 days is after 7613 data point
    x = np.asarray(x)
    filt = scipy.signal.medfilt(x, N) # Using the scipy median
    filter
    margin = (N - 1) // 2
    return filt[margin:-margin]
```

```
1112 def find_time(array, array_target):
         i=0 #start from zero
1114     sign=-1 #check when sign changes positive to negative
         while sign<0 and i< len(array):
1116         sign = array[i] - array[0] - array_target
             i= i+1
1118     return i-2

1120 def isOdd(x):
         if x % 2 == 0:
1122         return x+1
         else:
1124         return x

1126 def autocorr(data, lags=40):
         """
1128     Autocorrelation done using Python statistics module. The
         function is basically an interface to the existing funciton
          - not a re-implementationself.
         Not all of the functionalit of the original function is
         used.
1130
             Arguments:
1132             - 'data' : the data which to run the
         autocorrelation on.
                 - 'lags' : Number of lags to return autocorrelation
          for (default is 40 in the original function)
1134
         """
1136     #importing the function
         from statsmodels.tsa.stattools import acf
1138
         #run the autocorrelation
1140     corr = acf(data,nlags=lags)

1142     #return desired values
         return corr
1144
     def running_mean(x, N):
1146     """
         This calculates the running mean using the cumulative sum
         in a
1148     specified window.
         (see wikipedia -> Moving average -> Cumulative moving
         average.)
1150     Arguments:
             - 'x': Data series
```

```
        - 'N': The size of the window
    The output has N-1 fewer entries than x.
    >>> running_mean([0, 1, 2, 3, 4], 2)
    array([ 0.5,  1.5,  2.5,  3.5])
    """
    x = np.asarray(x)
    cumsum = np.cumsum(np.insert(x, 0, 0))
    return (cumsum[N:] - cumsum[:-N]) / N

def running_mean_filter(x, N):
    """
    This fixes the length of the running mean so it has the
    same length
    as the inputted data series by repeating the last entry $N
    -1$ times.
    Arguments:
        - 'x': Data series
        - 'N': The size of the window, which need to be a
    uneven number.
    >>> running_mean_filter_2([0, 3, 9, 15, 18], 5)
    array([  0.,   4.,   9.,  14.,  18.])
    """
    assert N % 2 == 1
    x = np.asarray(x)
    extra_count = (N - 1) // 2
    left = []
    right = []
    for i in range(extra_count):
        left.append(np.mean(x[:2*i+1]))
        right.append(np.mean(x[-2*i-1:]))
    right.reverse()
    y = running_mean(x, N)
    return np.concatenate([left, y, right])
```

/home/simonbanerjee/Dropbox/Speciale/Lightcurve_to_powerspectrum/filter.py.

```
"""
This file defines a function to calculate the power spectrum of
    a star
"""
# Import modules
import numpy as np
from time import time as now
import os
import scipy.signal
# from mpl_toolkits.mplot3d import Axes3D
import matplotlib
```

```
from matplotlib import cm
import itertools
from math import e
from uncertainties import ufloat
import scipy.stats
from scipy.ndimage import gaussian_filter

# Make nice plots
def matplotlib_setup():
    """ The setup, which makes nice plots for the report"""
    fig_width_pt = 328
    inches_per_pt = 1.0 / 72.27
    golden_mean = (np.sqrt(5) - 1.0) / 2.0
    fig_width = fig_width_pt * inches_per_pt
    fig_height = fig_width * golden_mean
    fig_size = [fig_width, fig_height]
    matplotlib.rc('text', usetex=True)
    matplotlib.rc('figure', figsize=fig_size)
    matplotlib.rc('font', size=8, family='serif')
    matplotlib.rc('axes', labelsize=8)
    matplotlib.rc('legend', fontsize=8)
    matplotlib.rc('xtick', labelsize=8)
    matplotlib.rc('ytick', labelsize=8)
    matplotlib.rc('text.latex', preamble=
                  r'\usepackage[T1]{fontenc}\usepackage{lmodern
    }')

#matplotlib_setup()
import matplotlib.pyplot as plt
import seaborn as sns

# Activate Seaborn color aliases
sns.set_palette('colorblind')
sns.set_color_codes(palette='colorblind')
#plt.style.use('ggplot')
#sns.set_context('poster')
sns.set_style("ticks")


def power_spectrum(time, amplitude, weight=None, minfreq=None,
    maxfreq=None,
                   oversample=None, memory_use=None, freq=None)
    :
    """
    This function returns the power spectrum of the desired
    star.
    Arguments:
```

```
        - 'time': Time in megaseconds from the timeserie
    analysis.
        - 'amplitude': Photometry data from the timeserie
    analysis.
        - 'weight': Weights for each point in the time series.
        - 'minfreq': The lower bound for the frequency interval
        - 'maxfreq': The upper bound for the frequency interval
        - 'oversample': The resolution of the power spectrum.
        - 'memory_use': The amount of memory used for this
    calculation.
        - 'freq': Override minfreq, maxfreq, ... and use these
    frequencies instead.
    """
    # The default longest wavelength is the length of the time
    series.
    if minfreq is None:
        #minfreq = 1 / (time[-1] - time[0]) #takes the
    difference between the last data point and the first data
    point
        #minfreq = 1 / (time[-1] - time[0]) #takes the
    difference between the last data point and the first data
    point
        #minfreq = 775 # this is in microhertz
        minfreq = 0.1 # this is in microhertz


    # The default greatest frequency is the Nyquist frequency.
    if maxfreq is None:
        #maxfreq = 1 /(2 * np.median(np.diff(time))) #This the
    expression for the Nyquist frequency
        #maxfreq = 1 /(2 * np.median(np.diff(time*86400)))
    *10**6 #This the expression for the Nyquist frequency
        #maxfreq= 4200 # is in microhertz
        maxfreq = 8400 # is in microhertz
    # By default oversample 4 times
    if oversample is None:
        oversample = 4 # This is used when we make our timestep
     interval
        #oversample=100
    # By default use 500000 memory cells (8 bytes each).
    if memory_use is None:
        memory_use = 500000
    if weight is None:
        weight = np.ones(amplitude.shape)
    else:
        weight = np.asarray(weight)
        assert weight.shape == amplitude.shape
```

```
1088
    if freq is None:
1090        # Generate cyclic frequencies
            #step = 1 / (oversample*(time[-1] - time[0])) #Here we
        can see how the oversample from before is used to determine
         the time step interval
1092        #step = 1 / (oversample *((time[-1] - time[0])*86400))
        *10**6 #Here we can see how the oversample from before is
        used to determine the time step interval
            step = 0.1
1094        freq = np.arange(minfreq, maxfreq, step)
        print("Computing power spectrum for frequencies " +
1096          "[%g, %g] with step %g" %
              (freq.min(), freq.max(), np.median(np.diff(freq))))
1098
    # Generate list to store the calculated power
1100    alpha = np.zeros((len(freq),))
    beta = np.zeros((len(freq),))
1102
    # Convert frequencies to angular frequencies
1104    nu = 2 * np.pi * freq
    #nu=freq
1106    # Iterate over the frequencies
    timerStart = now()
1108
    # After this many frequencies, print progress info
1110    print_every = 75e6 // len(time)

1112    # Define a chunk for the calculation in order to save time
    chunksize = memory_use // len(time)
1114    chunksize = max(chunksize, 1)

1116    # Ensure chunksize divides print_every
    print_every = (print_every // chunksize) * chunksize
1118
    for i in range(0, len(nu), chunksize):
1120        # Define chunk
            j = min(i + chunksize, len(nu))
1122        rows = j - i

1124        # Info-print
            if i % print_every == 0:
1126            elapsedTime = now() - timerStart
                if i == 0:
1128                totalTime = 0.004 * len(nu)
                else:
1130                totalTime = (elapsedTime / i) * len(nu)
```

```
1132            print("Progress: %.2f%% (%d of %d)  "
                      "Elapsed: %.2f s  Total: %.2f s"
1134                  % (np.divide(100.0*i, len(nu)), i, len(nu),
                          elapsedTime, totalTime))
1136
        """
1138        The outer product is calculated. This way, the product between
        time and ang. freq. will be calculated elementwise; one column
1140        per frequency. This is done in order to save computing time.
        """
1142        nutime = np.outer(time*86400/10**6, nu[i:j]) #the time is in megaseconds and frequency is in  microhertz and converted from cycles pr day

1144        """
        An array with the measured amplitude is made so it has the same size
1146        as "nutime", since we want to multiply the two.
        """
1148        amplituderep = amplitude.reshape(-1, 1)
        weightrep = weight.reshape(-1, 1)
1150
        # The Fourier subroutine
1152        sin_nutime = np.sin(nutime)
        cos_nutime = np.cos(nutime)
1154
        # Making the Fourier Transformation
1156        s = np.sum(weightrep * sin_nutime * amplituderep, axis=0)
        c = np.sum(weightrep * cos_nutime * amplituderep, axis=0)
1158        ss = np.sum(weightrep * sin_nutime ** 2, axis=0)
        cc = np.sum(weightrep * cos_nutime ** 2, axis=0)
1160        sc = np.sum(weightrep * sin_nutime * cos_nutime, axis=0)

1162        alpha[i:j] = ((s * cc) - (c * sc)) / ((ss * cc) - (sc ** 2)) # calculates the alpha values
        beta[i:j] = ((c * ss) - (s * sc)) / ((ss * cc) - (sc ** 2)) # calculates the beta values
1164
    alpha = alpha.reshape(-1, 1)
1166    beta = beta.reshape(-1, 1)
```

```
    freq = freq.reshape(-1, 1)#*86400*10**6
    power = alpha ** 2 + beta ** 2 # calculates the Power
    elapsedTime = now() - timerStart
    print('Computed power spectrum in %.2f s' % (elapsedTime))
    return (freq, power, alpha, beta)



def calculate_v_nl_modes(delta_nu, epsilon, dnu, nmin, nmax):
  """
  Solves the asymptotic relation
    Depening on which small seperation you look for dnu should
    be divided with something else. For dnu_02 it is 6
  """
  v_nl = [delta_nu*(n+l/2+epsilon)-(dnu/6)*(l*(l+1)) for n,l in
     itertools.product(range(nmin, nmax), (0,1,2,3))]
  return v_nl

def calculate_v_nl(delta_nu, epsilon, dnu, nmin, nmax):
  """
  Solves the asymptotic relation
    Depening on which small seperation you look for dnu should
    be divided with something else. For dnu_02 it is 6
  """
  v_nl = [delta_nu*(n+l/2+epsilon)-(dnu/6)*(l*(l+1)) for n,l in
     itertools.product(range(nmin, nmax), (0,2))]
  return v_nl

def calculate_MF(included_peak, v_nl, resnu):
    #Calculates the best matching values for given peaks and
    v_nl
    MF = 0
    for nui in included_peak:
        for nuni in v_nl:
            MF += np.exp(-((nui-nuni)/(2*resnu))**2)
    return MF


def matched_filter(epsilon, dnu02, included_peak, d_nu):
    delta_nu = d_nu          # insert your large frequency
    splitting here.
    nmin = 0             # degree of note that it run over.
    These may be modifies since
                         # 0 and 45 is quit extreme but serves
    as a prove of concept.
    nmax = 45
    resnu = 0.5         # resolution of the peak. Depending of
```

```
    how good your dataset is
                        # you might be able to make this more
    narrow (i.e 0.5) or enlargen
                        # if you can't fint something good

    v_nls = calculate_v_nl(delta_nu, epsilon, dnu02, nmin, nmax
    )
    return calculate_MF(included_peak, v_nls, resnu)



# In order to perform the fit, we choose to weight the data by
    fitting the model to logaritmic bins.
def running_median(freq, powerden, weights=None, bin_size=None,
     bins=None):
    if bin_size is not None and bins is not None:
        raise TypeError('cannot specify both bin_size and bins
    ')
    freq = np.squeeze(freq)
    powerden = np.squeeze(powerden)
    n, = freq.shape
    n_, = powerden.shape
    assert n == n_

    if weights is None:
        weights = np.ones(n, dtype=np.float32)

    # Sort data by frequency
    sort_ind = np.argsort(freq)
    freq = freq[sort_ind]
    powerden = powerden[sort_ind]
    weights = weights[sort_ind]

    # Compute log of frequencies
    log_freq = np.log10(freq)
    # Compute bin_size
    if bin_size is None:
        if bins is None:
            bins = 10000
        df = np.diff(log_freq)
        d = np.median(df)
        close = df < 100*d
        span = np.sum(df[close])
        bin_size = span / bins
    bin_index = np.floor((log_freq - log_freq[0]) / bin_size)
    internal_boundary = 1 + (bin_index[1:] != bin_index[:-1]).
```

```
    nonzero()[0]
    boundary = [0] + internal_boundary.tolist() + [n]

    bin_freq = []
    bin_pden = []
    bin_weight = []
    for i, j in zip(boundary[:-1], boundary[1:]):
        bin_freq.append(np.mean(freq[i:j]))
        bin_pden.append(np.median(powerden[i:j]))
        bin_weight.append(np.sum(weights[i:j]))
    return np.array(bin_freq), np.array(bin_pden), np.array(
    bin_weight)

# Eq. 1 in mentioned paper
def background_fit_2(nu, sigma, tau):
    k1 = ((4 * sigma ** 2 * tau) /(1 + (2 * np.pi * nu * tau)
    ** 2 +(2 * np.pi * nu * tau) ** 4))
    return k1

def background_fit(nu, sigma_0, tau_0, sigma_1, tau_1, P_n):
    k1 = background_fit_2(nu=nu, sigma=sigma_0, tau=tau_0)
    k2 = background_fit_2(nu=nu, sigma=sigma_1, tau=tau_1)
    return P_n + k1 + k2

def logbackground_fit(nu, sigma_0, tau_0, sigma_1, tau_1, P_n):
        assert nu.all() > 0
        assert np.all(np.isfinite(nu)) == True

        xs = background_fit(nu, sigma_0, tau_0, sigma_1, tau_1,
     P_n)
        invalid = xs <= 0
        xs[invalid] = 1
        log_xs = np.log10(xs)
        log_xs[invalid] = -10000  # return a very low number
    for log of something negative
        return log_xs

def gridsearch(f, xs, ys, params):
        # Save l2-norm in a dictionary for the tuple of chosen
    parameters
        score = {}
        dxs = np.diff(np.log10(xs))
        dxs = np.concatenate([dxs, [dxs[-1]]])
        for p in itertools.product(*params):
            print('\rNow %f %f %f %f %f, Done %f' %
                  (*p, len(score)/np.product([len(x) for x in
    params])),
```

```
                    end='')
            zs = f(xs, *p)
            score[p] = np.sum((ys- zs) ** 2)
        print('')
        return min(score.keys(), key=lambda p: score[p])

def fit_large_freq_separation(data, a0, a1, a2, a3, a4, a5, a6,
    delta_nu, k,c):
    return (a0*np.exp(-(data-0.5*delta_nu)**2/c**2) + a1*np.exp
    (-(data-delta_nu)**2/c**2)+
            a2*np.exp(-(data-1.5*delta_nu)**2/c**2) + a3*np.exp
    (-(data-2*delta_nu)**2/c**2)+
            a4*np.exp(-(data-2.5*delta_nu)**2/c**2) + a5*np.exp
    (-(data-3*delta_nu)**2/c**2)+
            a6*np.exp(-(data-3.5*delta_nu)**2/c**2) + k)


def linear_regression(x, y, prob):
    """
    Return the linear regression parameters and their <prob>
    confidence intervals.
    ex:
    >>> linear_regression([.1,.2,.3],[10,11,11.5],0.95)
    """
    x = np.array(x)
    y = np.array(y)
    n = len(x)
    xy = x * y
    xx = x * x

    # estimates

    b1 = (xy.mean() - x.mean() * y.mean()) / (xx.mean() - x.
    mean()**2)
    b0 = y.mean() - b1 * x.mean()
    s2 = 1./n * sum([(y[i] - b0 - b1 * x[i])**2 for i in range(
    n)])
    print('b0 = ',b0)
    print('b1 = ',b1)
    print('s2 = ',s2)

    #confidence intervals

    alpha = 1 - prob
    c1 = scipy.stats.chi2.ppf(alpha/2.,n-2)
    c2 = scipy.stats.chi2.ppf(1-alpha/2.,n-2)
    print('the confidence interval of s2 is: ',[n*s2/c2,n*s2/c1
```

```
    ])

    c = -1 * scipy.stats.t.ppf(alpha/2.,n-2)
    bb1 = c * (s2 / ((n-2) * (xx.mean() - (x.mean())**2)))**.5
    print('the confidence interval of b1 is: ',[b1-bb1,b1+bb1])
    Delta_nu_lower = b1-bb1
    Delta_nu_upper = b1+bb1

    bb0 = c * ((s2 / (n-2)) * (1 + (x.mean())**2 / (xx.mean() -
     (x.mean())**2)))**.5
    print('the confidence interval of b0 is: ',[b0-bb0,b0+bb0])
    return Delta_nu_lower, Delta_nu_upper

def echelle(freq, power , delta_nu, save=None):

    fres = (freq[-1] - freq[0]) / (len(freq)-1)
    numax = (delta_nu / 0.263) ** (1 / 0.772)
    nmax = int(np.round(((numax - freq[0]) / delta_nu) - 1))
    nx = int(np.round(delta_nu / fres))
    print(nx)
    assert nx % 2 == 0  # we shift by nx/2 pixels below
    dnu = nx * fres
    print(dnu)
    ny = int(np.floor(len(power) / nx))

    startorder = nmax - 7
    print(startorder)
    endorder = nmax + 3
    # print("%s pixel rows of %s pixels" % (endorder-startorder
    , nx))

    start = int(startorder * nx)
    endo = int(endorder * nx)
    #start = 0
    #endo = len(freq)
    print(start)
    print(endo)
    apower = power[start:endo]
    pixeldata = np.reshape(apower, (-1, nx))

    def plot_position(freqs):
        o = freqs - freq[start]
        x = o % dnu
        y = start * fres + dnu * np.floor(o / dnu)
        return x, y

    h = plt.figure()
```

```
    plt.xlabel(r'Frequency mod $\Delta\nu$ [$\mu$Hz]' % dnu)
    plt.ylabel(r'Frequency [$\mu$Hz]')
    # Subtract half a pixel in order for data points to show up
    # in the middle of the pixel instead of in the lower left
    corner.
    plt.xlim([-fres/2, dnu-fres/2])
    plt.ylim([start * fres, endo * fres])
    for row in range(pixeldata.shape[0]):
        bottom = (start + (nx * row)) * fres
        top = (start + (nx * (row + 1))) * fres
        blur_data = gaussian_filter(pixeldata[row:row+1], 75)
        plt.imshow(blur_data, aspect='auto', cmap='Blues',
                   interpolation='gaussian', origin='lower',
                   extent=(-fres/2, dnu-fres/2, bottom, top))
    if save is None:
        plt.savefig('/home/simonbanerjee/Dropbox/Speciale/
    Master_Thesis/%s_echelle_%0.2f.eps' % ('16_Cyg_a', delta_nu
    ),
                    bbox_inches='tight')
    return h, plot_position
```

/home/simonbanerjee/Dropbox/Speciale/Lightcurve_to_powerspectrum/time_series_powers