AIMS Overview

Release 1.1.0

Daniel R. Reese

March 18, 2016

Acknowledgements

The "Asteroseismic Inference on a Massive Scale" (`AIMS`) project was developed at the University of Birmingham by Daniel R. Reese as one of the deliverables for the SpaceINN network. The SpaceINN network is funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 312844.

DRR would like to thank Diego Bossini, Tiago L. Campante, William J. Chaplin, Hugo R. Coelho, Guy R. Davies, James S. Kuszlewicz, Martin W. Long, Mikkel N. Lund, Andrea Miglio for help, suggestions, comments, and answers to questions during the developpement of the AIMS code

Contents

Acknowledgements **3**

1 Introduction **7**

2 A Bayesian approach **9**
2.1 The priors 9
2.2 The likelihood function 9
2.2.1 Classic observables 10
2.2.2 Seismic observables 10
2.3 Initialisation of walkers 10

3 Frequency combinations and correlations **13**
3.1 Initial assumptions and notation 13
3.2 Frequency combinations 13
3.3 Frequency ratios 14
3.4 Surface correction recipes 15
3.5 The large frequency separation 17

4 Model interpolation **19**
4.1 Linear interpolation between two models 19
4.1.1 Acoustic variables 19
4.1.2 Other variables 21
4.2 Finding interpolation coefficients 21
4.2.1 Age interpolation 21
4.2.2 Track interpolation 22

Bibliography **27**

Chapter 1

Introduction

The purpose of the present document is to provide a broad overview to the `AIMS` program as well as a technical supplement to the documentation included in the `AIMS` program. It explains in more detail where some of the formulas comes from. For a description on how to use `AIMS` or what file formats are accepted by `AIMS`, please refer to the documentation included in the `AIMS` program.

The "Asteroseismic Inference on a Massive Scale" (`AIMS`) program uses a Bayesian approach to estimate stellar properties and associated error bars from a set of seismic and non-seismic constraints. It relies on various components:

- a *pre-calculated* grid of stellar models and associated pulsation frequencies
- an unstructured interpolation scheme within the grid of models
- a Monte Carlo Markov Chain (MCMC) approach for finding the probability distribution functions (PDFs) of the various stellar properties

It is up to the user to calculate the grid of stellar models and their pulsation spectra, using the evolution and pulsation code of their choice. Some adaptations may be needed in order for `AIMS` to read the files which contain the pulsation spectra (see documentation within the `AIMS` code). Once this grid is supplied, `AIMS` reads the pulsation spectra and stores the information in a binary file which can then be used in subsequent runs using `AIMS`.

Model interpolation will be described in Chapter 4. It is a two step process which involves interpolation along an evolutionary track followed by interpolation between evolutionary tracks.

As will be described in the following chapter, the MCMC algorithm is based on the python package `emcee` (Foreman-Mackey et al., 2013). This algorithm finds a representative sample of stellar models which follow the PDFs of the various stellar properties.

Figure 1.1 provides an overview of what `AIMS` needs as inputs, and the sort of outputs it produces.

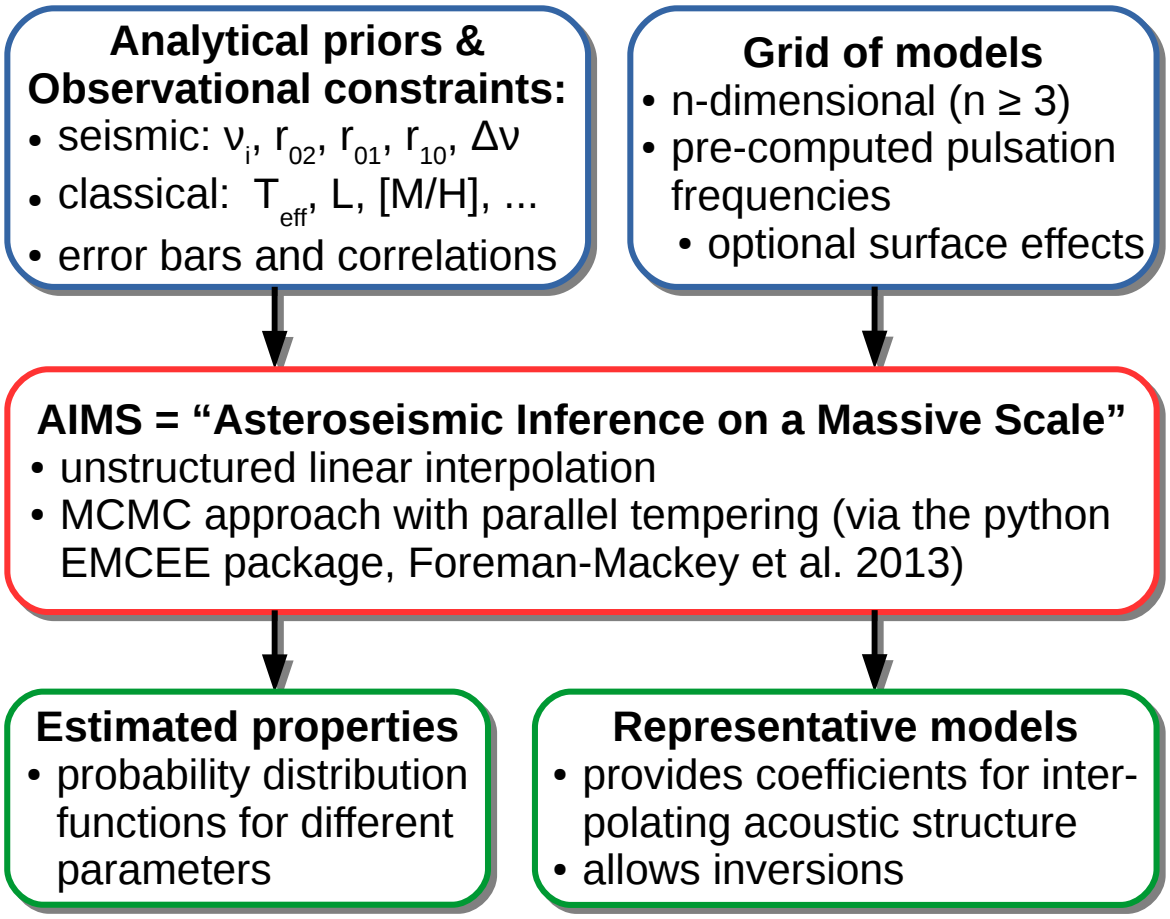

Figure 1.1: Overview of what `AIMS` needs as inputs and what outputs it produces.

Chapter 2

A Bayesian approach

`AIMS` uses a Bayesian approach to finding the probability distribution function (PDF) of the stellar parameters. In order to find a representative sample of models, an Monte Carlo Markov Chain (MCMC) approach based on the python package `emcee.py` (Foreman-Mackey et al., 2013) is used.

When applying a Bayesian approach, it is necessary to define the likelihood function as well as priors. Indeed, the PDF is proportional to the product of the two:

$$p(\theta|\mathcal{O}) \propto \underbrace{p(\mathcal{O}|\theta)}_{likelihood}\,\underbrace{p(\theta)}_{priors} \tag{2.1}$$

where θ is stellar parameters and $\mathcal{O}$ various seismic and classic observables. The priors and likelihood functions will be described in the following sections.

2.1 The priors

The priors represent the *a priori* assumptions on various stellar parameters. In `AIMS`, these priors will apply to the underlying stellar grid parameters, *i.e.* the ones involved in the interpolation process, and the amplitude(s) of surface corrections.

The priors are specified through analytically defined probability distributions. Currently, `AIMS` allows uniform, Gaussian, and a simple version of truncated Gaussian distributions (*c.f.* documentation for the `Distribution` class in `AIMS.py`). At some point in the future, it may be worth introducing new probability distributions if one wants to include, *e.g.*, an initial mass function (IMF). For now, it is probably best applying uninformative, *i.e.* uniform, priors.

2.2 The likelihood function

The likelihood function describes how likely it is to obtain a set of observables for a given set of model parameters. Typically, this function will be a product of probability functions for each of the observables. At this point, we distinguish between classic and seismic observables. These will be specified in an input file as described in the documentation within the `AIMS` code. The format of this file is closely related to that used for the Asteroseismic Modelling Portal (AMP, *e.g.* Metcalfe et al., 2014).

2.2.1 Classic observables

These typically are the luminosity, L, the effective temperature, $T_{\rm eff}$, the metallicity, Z, and any other parameter provided by `AIMS` (see documentation and source code for the `string_to_param()` method in `model.py`). `AIMS` allows the users to apply various probability distributions for these quantities (the same ones as for the priors).

2.2.2 Seismic observables

These typically correspond to anything deduced from the pulsation frequencies[1] (individual frequencies, average and individual frequency separations, frequency ratios). The errors on individual frequencies are assumed to be Gaussian, and independent for now, but inclusion of correlations should not be too difficult (one would have to decide on an input format for the correlation matrix, and would have to modify the `find_covariance` method in `AIMS.py`). When using frequency combinations, `AIMS` automatically calculates the correlations and takes this into account when calculating a χ^2 value based on these. The relevant formulas are given in Chapter 3 (these also include the case where correlations are present between different frequencies, even if these are not currently implemented in `AIMS`).

When calculating the seismic contribution to the likelihood function, `AIMS` proceeds in two steps. The first step involves matching the individual model frequencies to the observations. The use can specify whether the match should be made according to n values, or nearest frequencies, thanks to the `use_n` parameter in `AIMS_configure.py`. If one or more of the observed frequencies are unmatched, the model is rejected. The second step involves calculating the relevant theoretical frequency combinations and comparing this to the observed ones, while taking into account the relevant correlations. This step can be done efficiently and consistently thanks to the mapping found in the first step.

2.3 Initialisation of walkers

The `emcee` program relies on a number of walkers to explore the parameter space. These walkers should be initialised in some way. `AIMS` provides two choices:

1. initialise according to the priors: with this option `AIMS` initialises the walkers through random realisations of the priors. This will lead to a very broad exploration of the parameter space if the priors are uninformative, but could take a long time to converge. Furthermore, the priors for surface effects can be difficult to set up properly (we recommend doing a trial run beforehand).

2. initialise according to a tight ball around the best solution in the grid: with this option `AIMS` scans through the entire grid and finds the best solution. Walkers close to this solution (*i.e.* in a tight ball around this solution) are then initialised using Gaussian realisations with user-specified 1σ values. AIMS also automatically determines relevant ranges on the surface correction amplitudes based on the values obtained for the best solution.

[1] This doesn't include $\nu_{\rm max}$ which instead should be included as a "classic" observable.

In both cases, `AIMS` filters out solutions that fall outside the grid, or are rejected because of missing modes. If the relevant ranges for the realisations are too broad, then the selected parameters will have a high probability of falling outside the grid and will be rejected. This will make the program stall. Accordingly, the `max_iter` parameter was introduced in `AIMS_configure.py` and limits the number of attempts per walker. If this number is exceeded, `AIMS` terminates with an error message.

Chapter 3

Frequency combinations and correlations

3.1 Initial assumptions and notation

In what follows, we will assume that we have a spectrum of pulsation frequencies, ν_i, where i is shorthand for (n, ℓ). These frequencies are typically extracted from a noisy spectrum, so typically a include noise term (or realisation), ε_i. At this point, we will distinguish two cases:

1. the noise realisations are uncorrelated between the different frequencies, and are characterised by 1σ error bars, which we will denote σ_i.

2. the noise realisations are correlated (this is the more general case). We therefore introduce the covariance matrix, E_{ij}. If the noise was, in fact uncorrelated, then $E_{ij} = \sigma_i^2 \delta_i^j$.

3.2 Frequency combinations

We first start by calculating the error properties of a frequency combination (for example the large for small frequency separations). This can be expressed through the following generic formula:

$$s = \sum_i a_i \nu_i \tag{3.1}$$

Given that s is calculated from noisy data, we explicitly include the error realisations in the following formula:

$$s + \varepsilon_s = \sum_i a_i \left(\nu_i + \varepsilon_i\right) = \sum_i a_i \nu_i + \sum_i a_i \varepsilon_i \tag{3.2}$$

where ε_s is the noise realisation on s. Hence, the 1σ error bar on s, σ_s, is given by:

$$\sigma_s^2 = \sum_i a_i^2 \sigma_i^2 \tag{3.3}$$

if the individual errors are uncorrelated. Otherwise, one has the following formula:

$$\sigma_s^2 = \sum_{i,j} a_i a_j E_{ij} \tag{3.4}$$

If one considers the correlation between s, and another frequency combination $s' = \sum_i a'_i \nu_i$, then it is necessary to consider the product between the two.

$$ss' + \varepsilon_{ss'} = (s + \varepsilon_s)(s' + \varepsilon_{s'}) \simeq ss' + s'\varepsilon_s + s\varepsilon_{s'} = ss' + ss'\left(\frac{\varepsilon_s}{s} + \frac{\varepsilon_{s'}}{s'}\right) \tag{3.5}$$

where we have assumed that $\varepsilon_s \ll s$ and $\varepsilon_{s'} \ll s'$.

Hence,

$$\mathrm{Var}(s, s') = \sum_i \left(s' a_i + s a'_i\right)^2 \sigma_i^2 = (ss')^2 \sum_i \left(\frac{a_i}{s} + \frac{a'_i}{s'}\right)^2 \sigma_i^2 \tag{3.6}$$

if the errors are uncorrelated, or

$$\mathrm{Var}(s, s') = \sum_{i,j} \left(s' a_i + s a'_i\right)\left(s' a_j + s a'_j\right) E_{ij} = (ss')^2 \sum_{i,j} \left(\frac{a_i}{s} + \frac{a'_i}{s'}\right)\left(\frac{a_j}{s} + \frac{a'_j}{s'}\right) E_{ij} \tag{3.7}$$

in the general case.

3.3 Frequency ratios

We now turn our attention to frequency ratios, which may be put under the following generic form:

$$r = \frac{\sum_i a_i \nu_i}{\sum_i b_i \nu_i} \tag{3.8}$$

The noise realisation is given by the following approximate formula:

$$\begin{aligned} r + \varepsilon_r &= \frac{\sum_i a_i \left(\nu_i + \varepsilon_i\right)}{\sum_i b_i \left(\nu_i + \varepsilon_i\right)} = \frac{\left(\sum_i a_i \nu_i\right)\left(1 + \sum_i \frac{a_i}{\left(\sum_j a_j \nu_j\right)} \varepsilon_i\right)}{\left(\sum_i b_i \nu_i\right)\left(1 + \sum_i \frac{b_i}{\left(\sum_j b_j \nu_j\right)} \varepsilon_i\right)} \\ &\simeq r + r \sum_i \left(\frac{a_i}{\left(\sum_j a_j \nu_j\right)} - \frac{b_i}{\left(\sum_j b_j \nu_j\right)}\right) \varepsilon_i \end{aligned} \tag{3.9}$$

Hence, the 1σ error bar is:

$$\sigma_r^2 = r^2 \sum_i \left(\frac{a_i}{\left(\sum_j a_j \nu_j\right)} - \frac{b_i}{\left(\sum_j b_j \nu_j\right)}\right)^2 \sigma_i^2 \tag{3.10}$$

or in the general case:

$$\sigma_r^2 = r^2 \sum_{i,j} \left(\frac{a_i}{(\sum_k a_k \nu_k)} - \frac{b_i}{(\sum_k b_k \nu_k)} \right) \left(\frac{a_j}{(\sum_k a_k \nu_k)} - \frac{b_j}{(\sum_k b_k \nu_k)} \right) E_{ij} \tag{3.11}$$

Correlations on the noise realisation between two frequency ratios is given by:

$$rr' + \varepsilon_{rr'} \simeq rr' + r'\varepsilon_r + r\varepsilon_{r'} = rr' + rr' \left(\frac{\varepsilon_r}{r} + \frac{\varepsilon_{r'}}{r'} \right) \tag{3.12}$$

This leads to

$$\mathrm{Var}(r, r') = (rr')^2 \sum_i \left(\frac{a_i}{\left(\sum_j a_j \nu_j\right)} - \frac{b_i}{\left(\sum_j b_j \nu_j\right)} + \frac{a'_i}{\left(\sum_j a'_j \nu_j\right)} - \frac{b'_i}{\left(\sum_j b'_j \nu_j\right)} \right)^2 \sigma_i^2 \tag{3.13}$$

if the errors are uncorrelated, or

$$\begin{aligned} \mathrm{Var}(r, r') \;=\; & (rr')^2 \sum_{i,j} \left(\frac{a_i}{(\sum_k a_k \nu_k)} - \frac{b_i}{(\sum_k b_k \nu_k)} + \frac{a'_i}{(\sum_k a'_k \nu_k)} - \frac{b'_i}{(\sum_k b'_k \nu_k)} \right) \\ & \left(\frac{a_j}{(\sum_k a_k \nu_k)} - \frac{b_j}{(\sum_k b_k \nu_k)} + \frac{a'_j}{(\sum_k a'_k \nu_k)} - \frac{b'_j}{(\sum_k b'_k \nu_k)} \right) E_{ij} \end{aligned} \tag{3.14}$$

in the general case.

3.4 Surface correction recipes

Another typical manipulation when comparing theoretical frequencies with observed ones is to introduce surface corrections (*e.g.* Kjeldsen et al., 2008). This is because stellar models and associated pulsation calculations do not correctly reproduce the structure near the surface in a star. Schematically, such surface corrections take on the following form:

$$\delta\nu_{\mathrm{surf}} = a f(\nu, \vec{\xi}) \tag{3.15}$$

where a is a coefficient which needs to be determined, and f a function which depends on frequency and in some cases on the pulsation mode. Two possible expressions for f are (Kjeldsen et al., 2008):

$$f(\nu) = \left(\frac{\nu}{\nu_0} \right)^b \tag{3.16}$$

where ν_0 is a reference frequency and b an exponent (typically 4.9, based on the solar case Kjeldsen et al. 2008), and (Ball & Gizon, 2014):

$$f(\nu) = \frac{1}{I} \left(\frac{\nu}{\nu_0} \right)^3 \tag{3.17}$$

where I is the normalised mode inertia. A two-term surface correction is also given in Ball & Gizon (2014). We note, in passing, that Kjeldsen et al. (2008) expresses f as a

function of the observed rather than model frequency. However, the two are very similar if the model is a good match to the observations. In what follows (and in `AIMS`), we will use the model frequency for reasons which will become clear later on.

In `AIMS`, the coefficient a (or coefficients if applying the two-term surface correction from Ball & Gizon 2014) are treated as supplementary parameters or dimensions when applying the MCMC algorithm. This approach does not introduce any supplementary correlations on the frequencies. If however, one were to automatically determine optimal coefficient(s) through linear regression, then they would have to take into account supplementary correlations between the now "corrected" observed frequencies. To illustrate this, we start by determining a through a linear regression:

$$a = \frac{\sum_i f(\nu_i, \vec{\xi_i}) \left(\nu_i^{\rm obs} - \nu_i\right)}{\sum_i \left[f(\nu_i, \vec{\xi_i})\right]^2}, \tag{3.18}$$

where quantities with the superscript "obs" come from the observations, and those without from the reference model. These expressions are a linear combination of the observed frequencies, so the 1σ error bar is straightforward to calculate using Eqs. (3.3) or (3.4). Had we used $f(\nu_i^{\rm obs}, \vec{\xi})$ instead, this would have led to a much more complicated dependence of a on the observed frequencies and made it more difficult to calculate its 1σ error bars.

For convenience, we now "correct" the observed frequencies so as to group all of the observational error bars together. This leads to:

$$\nu_{{\rm corr},i}^{\rm obs} = \nu_i^{\rm obs} - a f(\nu_i, \vec{\xi_i}) \tag{3.19}$$

The 1σ error bars on $\nu_{\rm obs}^{\rm corr.}$ are:

$$\sigma_{{\rm corr},i}^2 = \left(1 - \frac{f_i^2}{\sum_j f_j^2}\right)^2 \sigma_i^2 + \sum_{j\neq i} \left(\frac{f_i f_j}{\sum_k f_k^2}\right)^2 \sigma_j^2 \tag{3.20}$$

or, in the general case:

$$\sigma_{{\rm corr},i}^2 = \left(1 - \frac{f_i^2}{\sum_j f_j^2}\right)^2 E_{ii} - 2\sum_{j\neq i} \frac{f_i f_j}{\sum_k f_k^2} \left(1 - \frac{f_i^2}{\sum_k f_k^2}\right) E_{ij} + \sum_{j\neq i}\sum_{j'\neq i} \frac{f_i^2 f_j f_{j'}}{\left(\sum_k f_k^2\right)^2} E_{jj'} \tag{3.21}$$

where $f_i \equiv f(\nu_i, \vec{\xi_i})$. Analogous expressions are also obtained for the variance between these quantities.

If one takes the error bars into account when determining a, the following expression is obtained:

$$a = \frac{\sum_i f_i \left(\nu_i^{\rm obs} - \nu_i\right)/\sigma_i^2}{\sum_i f_i^2/\sigma_i^2} \tag{3.22}$$

This then leads to:

$$\sigma_{{\rm corr},i}^2 = \left(1 - \frac{f_i^2/\sigma_i^2}{\sum_j f_j^2/\sigma_j^2}\right)^2 \sigma_i^2 + \sum_{j\neq i} \left(\frac{f_i f_j/\sigma_j^2}{\sum_k f_k^2/\sigma_k^2}\right)^2 \sigma_j^2 \tag{3.23}$$

If, furthermore, the errors are correlated, the following cost function must be minimised when calculating the value of a:

$$J = \sum_{i,j} \left(\nu_i^{\rm obs} - \nu_i - a f_i\right) \left(\nu_j^{\rm obs} - \nu_j - a f_j\right) E_{ij}^{-1}, \tag{3.24}$$

where (E_{ij}^{-1}) is the inverse of (E_{ij}), the covariance matrix. Minimising J leads to the following expression:

$$a = \frac{\sum_{i,j} f_i \left(\nu_j^{\rm obs} - \nu_j\right) E_{ij}^{-1}}{\sum_{i,j} f_i f_j E_{ij}^{-1}} \tag{3.25}$$

The corresponding corrected frequency is:

$$\nu_{\rm corr,i}^{obs} = \nu_i^{\rm obs} - \sum_j \frac{\sum_k f_i f_k E_{jk}^{-1}}{\sum_{j',k'} f_{j'} f_{k'} E_{j'k'}^{-1}} (\nu_j^{\rm obs} - \nu_j) \tag{3.26}$$

which leads to the following 1σ error bar:

$$\begin{aligned} \sigma_{{\rm corr},i}^2 \;=\; & \left(1 - \frac{\sum_j f_i f_k E_{ik}^{-1}}{\sum_{j',k'} f_{j'} f_{k'} E_{j'k'}^{-1}}\right)^2 E_{ii} - 2 \sum_{j \neq i} \frac{\sum_k f_i f_k E_{jk}^{-1}}{\sum_{j',k'} f_{j'} f_{k'} E_{j'k'}^{-1}} \left(1 - \frac{\sum_k f_i f_k E_{jk}^{-1}}{\sum_{j',k'} f_{j'} f_{k'} E_{j'k'}^{-1}}\right) E_{ij} \\ & + \sum_{j \neq i} \sum_{k \neq i} \frac{f_i^2 \sum_{k'} f_{k'} E_{jk'}^{-1} \sum_{k'} f_{k'} E_{kk'}^{-1}}{\left(\sum_{j',k'} f_{j'} f_{k'} E_{j'k'}^{-1}\right)^2} E_{jk} \end{aligned} \tag{3.27}$$

3.5 The large frequency separation, $\Delta\nu$

The large frequency separation is a particular frequency combination which asymptotically gives the inverse of twice the acoustic radius. A numerical value can by obtained by minimising the following cost function:

$$J = \sum_i \left(\frac{\nu_i - n_i \Delta\nu - \varepsilon_{\ell_i}}{\sigma_i^2}\right)^2, \tag{3.28}$$

where ν_i are the observed frequencies, and (n_i, ℓ_i) the associated radial orders and harmonic degrees, respectively. The parameters $\Delta\nu$ and ε_ℓ correspond to the large frequency separations and ℓ dependant offsets, to be determined via the least-squares calculation. Deriving J with respect to $\Delta\nu$ and ε_ℓ, and setting these partial derivatives to zero leads to the following system:

$$\begin{bmatrix} \langle n^2 \rangle & \langle n \rangle_{\ell=0} & \langle n \rangle_{\ell=1} & \dots & \langle n \rangle_{\ell=L} \\ \langle n \rangle_{\ell=0} & \langle 1 \rangle_{\ell=0} & 0 & \dots & 0 \\ \langle n \rangle_{\ell=1} & 0 & \langle 1 \rangle_{\ell=1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \langle n \rangle_{\ell=L} & 0 & \dots & 0 & \langle 1 \rangle_{\ell=L} \end{bmatrix} \begin{bmatrix} \Delta\nu \\ \varepsilon_{\ell=0} \\ \varepsilon_{\ell=1} \\ \vdots \\ \varepsilon_{\ell=L} \end{bmatrix} = \begin{bmatrix} \langle n\nu \rangle \\ \langle \nu \rangle_{\ell=0} \\ \langle \nu \rangle_{\ell=1} \\ \vdots \\ \langle \nu \rangle_{\ell=L} \end{bmatrix}, \tag{3.29}$$

where $\langle 1 \rangle = \sum_i 1/\sigma_i^2$, $\langle n \rangle = \sum_i n_i/\sigma_i^2$, etc. The subscripts $\ell = 0$, $\ell = 1$, etc. signify that quantity has been summed only over modes with the corresponding ℓ value. Using Cramer's rule, it is possible to solve for $\Delta\nu$:

$$\Delta\nu = \frac{\langle n\nu \rangle - \sum_\ell \frac{\langle \nu \rangle_\ell \langle n \rangle_\ell}{\langle 1 \rangle_\ell}}{\langle n^2 \rangle - \sum_\ell \frac{\langle n \rangle_\ell^2}{\langle 1 \rangle_\ell}}, \tag{3.30}$$

where we have divided the top and bottom by $\prod_\ell \langle 1 \rangle_\ell$. The quantities ε_ℓ are immediately deduced as follows:

$$\varepsilon_\ell = \frac{\langle \nu \rangle_\ell - \langle n \rangle_\ell \Delta\nu}{\langle 1 \rangle_\ell}. \tag{3.31}$$

At this point, is worth noting that if a particular ℓ value has no modes associated with it, then including it in the above solution leads to divisions by 0. If a particular ℓ value has only 1 mode associated with it, then it's contribution cancels out. This is logical, since only 1 point is not enough to determine the line of a slope. Hence, `AIMS` only works with ℓ values associated with at least 2 modes.

If we want to calculate the 1σ value of $\Delta\nu$ and correlations with other frequency combinations, we need to extract the coefficients of the individual modes. This can be done by expanding Eq. 3.30:

$$\Delta\nu = \sum_i c_i \nu_i = \sum_i \frac{n_i - \frac{\langle n \rangle_{\ell_i}}{\langle 1 \rangle_{\ell_i}}}{\left(\langle n^2 \rangle - \sum_\ell \frac{\langle n \rangle_\ell^2}{\langle 1 \rangle_\ell} \right) \sigma_i^2} \nu_i \tag{3.32}$$

These coefficients can then be inserted into Eq. (3.3) to obtain the 1σ value:

$$\sigma_{\Delta\nu}^2 = \sum_i c_i^2 \sigma_i^2 = \frac{1}{\langle n^2 \rangle - \sum_\ell \frac{\langle n \rangle_\ell^2}{\langle 1 \rangle_\ell}} \tag{3.33}$$

One can also apply Eq. (3.6) to find the covariance between $\Delta\nu$ and other seismic constraints.

It is important to stress that Eq. (3.33) only gives the propagation of observational error into the final result, and in no way offers a measure of the dispersion of the frequencies around a straight line. A more complete measure would include both effects.

Chapter 4

Model interpolation

When considering interpolation between models, there are two very important aspects to consider:

1. how does one actually combine two or more models?

2. how does one obtain the appropriate interpolation coefficients?

These questions are addressed in the following sections. It is important to bear in mind that in `AIMS`, these two steps are separate, *i.e.* `AIMS` obtains the interpolation coefficients from the outset, then combines the models using the interpolation coefficients thus obtained. Although slightly more complicated to implement, such an approach is more computationally efficient than applying some interpolation procedure (for example from `numpy`) to each individual parameter within a model, as it avoids calculating the coefficients multiple times.

4.1 Linear interpolation between two models

When interpolating between two stellar models, `AIMS` only interpolates the global properties and the frequencies (by matching frequencies with the same (ℓ, n) values). Nonetheless, the global parameters are interpolated in such a way as to be coherent with a full interpolation of the acoustic structure of the two models, which can be carried out in a separate program using the interpolation coefficients provided by AIMS. Therefore, it is important to start with a description of the interpolation of the acoustic structure and then see what consequences this has on the interpolation of global properties.

Let us start by considering two models, Models 1 and 2. We wish to interpolate between the two, thereby producing a third model, Model 3. Specifically, Model 3 will be a weighted "sum" of Models 1 and 2, the coefficients being a and b, respectively, where $a + b = 1$. In what follows, we will use the indices 1, 2 and 3 to designate these models.

4.1.1 Acoustic variables

Let R_i be the photospheric radii of the different models. A first approach is to consider that the density and Γ_1 profiles of Model 3 is simply the weighted sum of these profiles

from Models 1 and 2. This leads to:

$$\rho_3(xR_3) = a\rho_1(xR_1) + b\rho_2(xR_2), \tag{4.1}$$
$$\Gamma_{1,3}(xR_3) = a\Gamma_{1,1}(xR_1) + b\Gamma_{1,2}(xR_2), \tag{4.2}$$

where $x \in [0,1]$. Furthermore, we want to make sure that if the masses of Models 1 and 2 are identical, than the mass of Model 3 is also the same. A simple way of achieving this is by imposing:

$$M_3 = aM_1 + bM_2 \tag{4.3}$$

If we combine Eqs. (4.1) and (4.3) this uniquely defines the radius of Model 3:

$$\begin{aligned} M_3 &= R_3^3 \int_0^1 4\pi\rho_3(xR_3)x^2\mathrm{d}x \\ &= R_3^3 \int_0^1 4\pi \left[a\rho_1(xR_1) + b\rho_2(xR_2)\right] x^2\mathrm{d}x \\ &= aR_3^3 \int_0^1 4\pi\rho_1(xR_1)x^2\mathrm{d}x + bR_3^3 \int_0^1 4\pi\rho_2(xR_2)x^2\mathrm{d}x \\ &= R_3^3 \left[a\frac{M_1}{R_1^3} + b\frac{M_2}{R_2^3}\right]. \end{aligned} \tag{4.4}$$

This can be put in the following form:

$$\frac{M_3}{R_3^3} = a\frac{M_1}{R_1^3} + b\frac{M_2}{R_2^3}, \tag{4.5}$$

and leads to:

$$R_3 = \sqrt[3]{\frac{M_3}{a\frac{M_1}{R_1^3} + b\frac{M_2}{R_2^3}}}. \tag{4.6}$$

Before going on to describe how the remaining global parameters are obtained, it is useful to discuss the relationship between the non-dimensional density profiles $\bar{\rho}(\bar{r}) = \rho(r)/\rho_{\mathrm{ref}}$, where $\rho_{\mathrm{ref}} = M/R^3$. A few simple calculations show that:

$$\bar{\rho}_3(xR_3) = \bar{a}\bar{\rho}_1(xR_1) + \bar{b}\bar{\rho}_2(xR_2), \tag{4.7}$$

where

$$\bar{a} = a\frac{M_1}{R_1^3}\frac{R_3^3}{M_3}, \qquad \bar{b} = b\frac{M_2}{R_2^3}\frac{R_3^3}{M_3}. \tag{4.8}$$

One can show that $\bar{a} + \bar{b} = 1$ by dividing Eq. 4.5 by M_3/R_3^3.

The pressure can then be obtained by hydrostatic equilibrium, provided a surface value is calculated. We arbitrarily define it as follows:

$$\bar{P}_3^{\mathrm{surf}} = a\bar{P}_1^{\mathrm{surf}} + b\bar{P}_2^{\mathrm{surf}} \tag{4.9}$$

where $\bar{P}^{\mathrm{surf}}$ is the non-dimensional surface values. This is calculated at x^{surf}, the minimum between x_1^{surf} and x_2^{surf} to avoid extrapolating beyond the end of either model. Using the above values of M_3 and R_3, it is straightforward to convert $\bar{P}_3^{\mathrm{surf}}$ into its dimensional counterpart.

4.1.2 Other variables

We interpolate other quantities as follows:

$$x_{0,3} = ax_{0,1} + bx_{0,2}, \tag{4.10}$$
$$z_{0,3} = az_{0,1} + bz_{0,2}, \tag{4.11}$$
$$t_3 = at_1 + bt_2, \tag{4.12}$$
$$T_{\rm eff,3} = aT_{\rm eff,1} + bT_{\rm eff,2}, \tag{4.13}$$

where x_0 and z_0 are the initial hydrogen and metal abundances, t the age of the star, and $T_{\rm eff}$ the effective temperature. The effective temperature is related to the luminosity via the following relation:

$$L = 4\pi\sigma R^2 T_{\rm eff}^4. \tag{4.14}$$

Hence, one could calculate directly the luminosity of model 3 from its radius and effective temperature. In what follows, however, we instead apply the less restrictive law $L \propto R^2 T_{\rm eff}^4$ and allow for differences in the proportionality constant between Models 1 and 2. By interpolating this coefficient, we obtain the following expression for the luminosity of model 3:

$$L_3 = \left(a\frac{L_1}{R_1^2 T_{\rm eff,1}^4} + b\frac{L_2}{R_2^2 T_{\rm eff,2}^4} \right) R_3^2 T_{\rm eff,3}^4. \tag{4.15}$$

4.2 Finding interpolation coefficients

In `AIMS`, the grid of models is stored as a multi-dimensional grid of evolutionary tracks. Each track is described by a set of variables such as mass and metallicity. Each model along a track is characterised by an age. Accordingly, interpolation has been broken down into two steps:

1. **age interpolation**: interpolate to the appropriate age along the relevant tracks.
2. **track interpolation**: interpolate between the relevant tracks using the models from step 1.

One could envisage combining both steps into a single interpolation step. However, it was decided that the above procedures offers a better way of controlling the age parameter, to make sure one doesn't interpolate beyond the end of the grid. Furthermore, it may potentially be more accurate as models are not expected to change much between consecutive steps of an evolutionary sequence.

4.2.1 Age interpolation

Interpolating according to age is a simple 1D interpolation problem. This is done by searching for the closest matching models using dichotomy (in `AIMS`, the models are sorted according to age), then finding the appropriate linear combination coefficients for the two closest models.

What is slightly less trivial is choosing the best age to which to interpolate on a given track. Two options are implemented in `AIMS`, as described in the following 2 paragraphs.

"Simple" age interpolation In this approach, one simply interpolates to the target age on each track. This is the simplest and most naive approach. One disadvantage is that if the target age is beyond the end of any of the relevant tracks, then no result will be obtained. This is illustrated in the right panel of Fig. 4.1.

"Scaled" age interpolation In this approach, one interpolates to the scaled target age. For a given evolutionary track, the scaled age, η, can be defined as follow:

$$\eta = \frac{t - t^{\rm i}}{t^{\rm f} - t^{\rm i}} \tag{4.16}$$

where t is the physical age, $t^{\rm i}$ and $t^{\rm f}$ are the initial and final ages along the track. Hence, this linear transformation maps the age to the interval $[0, 1]$. In this scheme, one therefore finds the initial and final ages of the interpolated track (based on the interpolation coefficients which will be described in the following section), use these to find the scaled age, then calculate the physical age on each track using the reverse transformation:

$$t_j = t_j^{\rm i} + \eta(t_j^{\rm f} - t_j^{\rm i}) \tag{4.17}$$

where the index "j" represents each (non-interpolated) track. It is straightforward to see that the linear combination of the t_j, using the interpolation coefficients associated to each track, does agree with the original target age:

$$\begin{aligned} \sum_j a_j t_j &= \sum_j a_j t_j^{\rm i} + \eta \sum_j a_j (t_j^{\rm f} - t_j^{\rm i}) \\ &= t^{\rm i} + \eta(t^{\rm f} - t^{\rm i}) \\ &= t \end{aligned}$$

where the parameters without the index "j" denote the parameters of the interpolated track.

Figure 4.1 illustrates both approaches, and gives an example where the "simple" age interpolation fails.

4.2.2 Track interpolation

In `AIMS`, an approach based on Delaunay tessellation has been adopted for carrying out interpolation between evolutionary tracks. A tessellation is a partition of an n-dimensional space into a set of simplices (*i.e.* n-polytopes with $n + 1$ nodes) using the input set of points as nodes. An example of a 2D tessellation is illustrated Fig. 4.2. Interpolation based on a tessellation has the advantage of not requiring a structured grid and also facilitates working in an arbitrary number of dimensions provided this is greater or equal to 2 (excluding the age dimension). It may also be more computationally efficient as fewer tracks are combined than if one interpolates along each direction (*i.e.* parameter) separately. Nonetheless, this may also lead to a decrease in the accuracy of the results. Figure 4.2 illustrates a 2D tessellation obtained for the grid of models used in Coelho et al. (2015).

The tessellation is calculated via `numpy`'s Delaunay() method, which comes from the `Qhull` library. Along with the tessellation, `numpy` provides methods for determining in

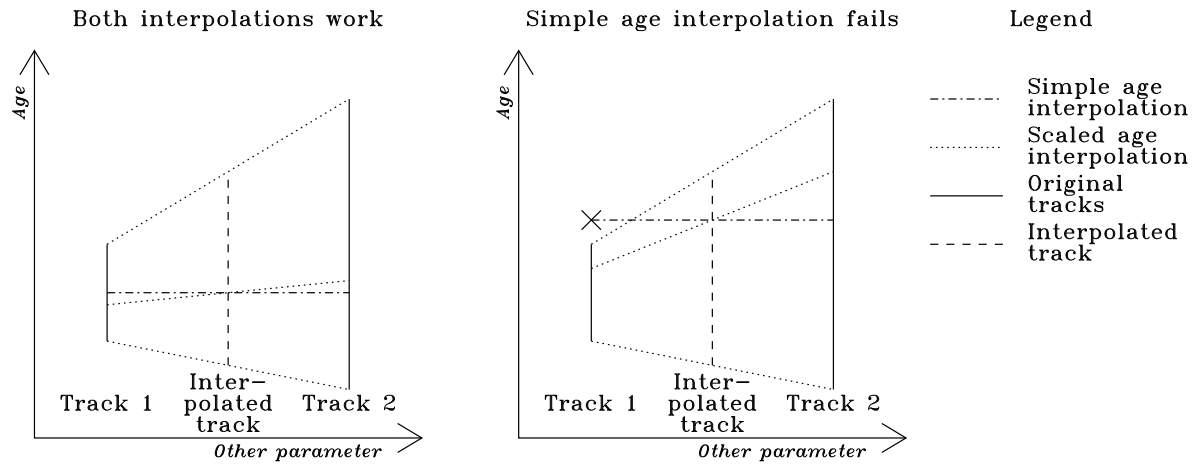

Figure 4.1: Comparison of "simple" and "scaled" age interpolation.

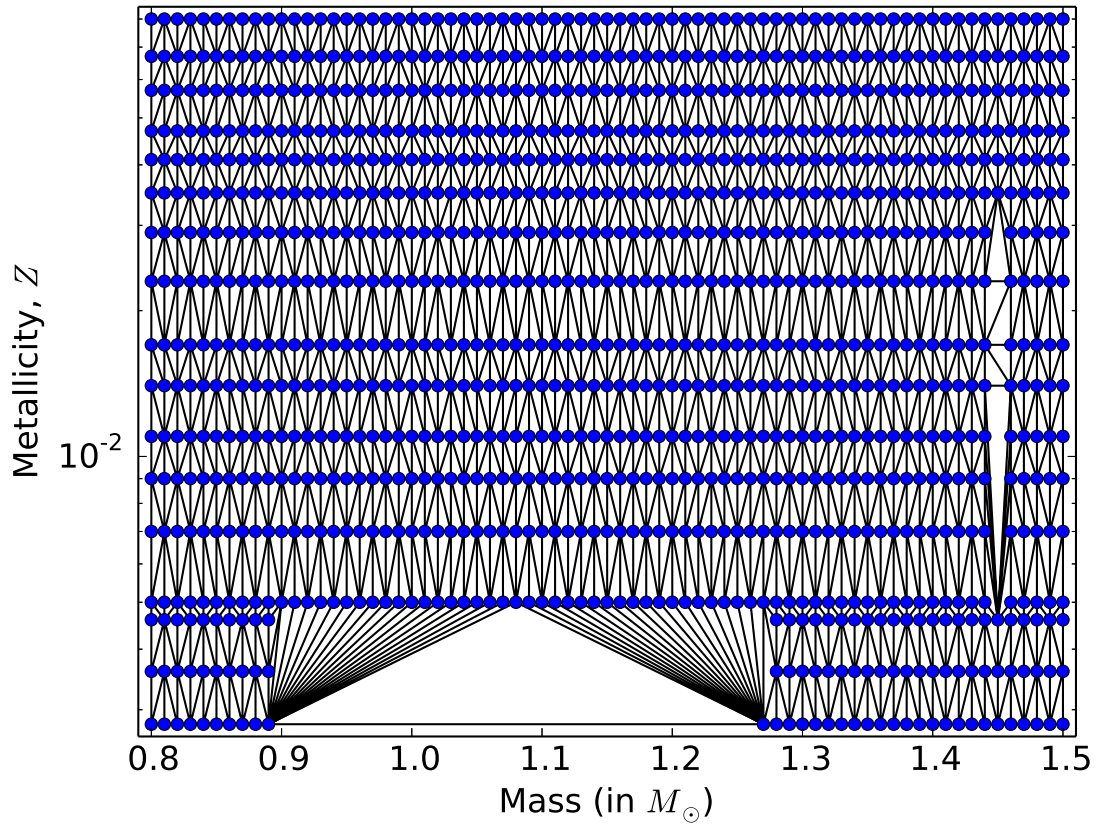

Figure 4.2: An example of a Delaunay tessellation carried out on the grid of models from Coelho et al. (2015).

which simplex (or triangle in the 2D case) a given point lies. The interpolation then consists in carrying out a linear combination of the nodes of the simplices. The relevant coefficients, known as barycentric coordinates[1], are not given directly in `numpy` and must therefore be worked out from the relevant transform matrix included in the tessellation. In what follows, we sketch how such barycentric coordinates are obtained and what some of the properties are.

We start with a 2D simplex, *i.e.* a triangle, as illustrated in the left panel of Fig. 4.3. In order to find the barycentric coordinates for point P, we need to set up a linear system. Let us denote (a, b, c) the barycentric coordinates based on nodes (A, B, C). Accordingly, (a, b, c) obey the following condition (as based on the interpolation of the x and y coordinates):

$$\overrightarrow{OP} = a\overrightarrow{OA} + b\overrightarrow{OB} + c\overrightarrow{OC} \tag{4.18}$$

where O is the origin. Furthermore,

$$1 = a + b + c \tag{4.19}$$

in order to have a proper average of the three nodes. An immediate consequence of this is:

$$\vec{0} = a\overrightarrow{PA} + b\overrightarrow{PB} + c\overrightarrow{PC} \tag{4.20}$$

which is obtained by subtracting $\overrightarrow{OP} = (a+b+c)\overrightarrow{OP}$ from Eq. (4.18). This equation along with Eq. (4.19) forms a system of 3 equations and 3 unknowns, which can subsequently be solved via Cramer's rule. For the coordinate a, this leads to:

$$\begin{aligned} a &= \frac{\begin{vmatrix} 1 & 1 & 1 \\ 0 & B_x - P_x & C_x - P_x \\ 0 & B_y - P_y & C_y - P_y \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 \\ A_x - P_x & B_x - P_x & C_x - P_x \\ A_y - P_y & B_y - P_y & C_y - P_y \end{vmatrix}} = \frac{\begin{vmatrix} B_x - P_x & C_x - P_x \\ B_y - P_y & C_y - P_y \end{vmatrix}}{\begin{vmatrix} 1 & 0 & 0 \\ A_x - P_x & B_x - A_x & C_x - A_x \\ A_y - P_y & B_y - A_y & C_y - A_y \end{vmatrix}} \\ &= \frac{\det\left(\overrightarrow{PB}, \overrightarrow{PC}\right)}{\det\left(\overrightarrow{AB}, \overrightarrow{AC}\right)} \end{aligned} \tag{4.21}$$

where A_x is the x coordinate of point A, A_y the y coordinate of A etc. One will notice that the numerator is twice the area[2] of triangle PBC and that the denominator is twice the area of triangle ABC. Hence is the barycentric coordinate a is simply the ratio of the area of these two triangles as illustrated in the right panel of Fig. 4.3. Such a geometrical interpretation is easily generalised to n dimensions.

One more property which is useful to point out is that the barycentric coordinates are all positive or zero if and only if the point P lies within the triangle (or simplex in

[1]This is `numpy`'s terminology. According to Wolfram, these would be called areal coordinates because of the normalisation condition given in Eq. (4.19, whereas barycentric coordinates could have an arbitrary normalisation.

[2]This is an algebraic value, the sign of which depends on the orientation of $\overrightarrow{PB}$ with respect to $\overrightarrow{PC}$.

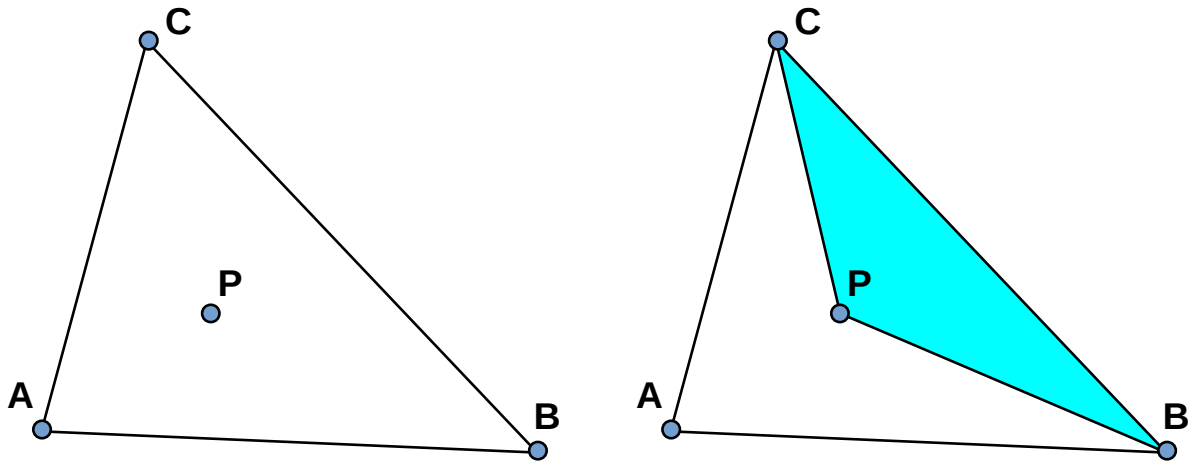

Figure 4.3: (Left panel) A triangle with an arbitrary point within where we would like to carry out interpolation. (Right panel) The barycentric coordinate, a, associated with node A, is the ratio between the surface area of triangles PBC and ABC.

dimension n). Intuitively, one can expect the signs of the numerator and denominator in Eq. (4.21) to be the same of the point P lies within the simplex.

As pointed out above, in `AIMS`, the barycentric coordinates are obtained via the the transform matrix associated with the simplex in which a given point lies (*c.f.* `numpy` documentation for this transform matrix). However, this procedure yields all but one of the coordinates. To deduce the last coordinate, `AIMS` applies Eq. (4.19).

Extrapolation

Sometimes, the MCMC algorithm chooses points which lie outside the grid. In such a situation, it is then necessary to choose the most appropriate simplex (typically the nearest one) and carry out extrapolation (since the point lies outside the simplex) in much the same way as the interpolation described above. When the first version of `AIMS` was released, it was mistakenly thought that `numpy` is also able to find an appropriate simplex for outside points. This turns out to be false. Instead, `numpy` returns the index -1 in such situations. However, given the `python` convention according to which -1 represents the last element of a list, tuple, or array, this mistake did not cause any errors during the execution of `AIMS` and escaped detection. Further tests, however, revealed this behaviour and the need for an new strategy for finding an appropriate simplex.

In the current release of `AIMS`, it was decided to use the nearest simplex (or more simply the one with the nearest facet) as the most appropriate choice for carrying out extrapolation. The distance between a point, P, and facet is defined as the distance between P and the nearest point within the facet. It can be calculated via the recursive algorithm described in Golubitsky et al. (2015). In some cases, this does not lead to a unique solution but rather 2 or more closest simplices. Such a situation is illustrated in Fig. 4.4. As explained in the caption, the search algorithm then selects the facet with the smallest angle, or equivalently, the largest height, which can be calculated as the ratio between the volume of the simplex formed of the point and the outside facet of the nearest simplex, and the surface area of this facet. This approach for selecting

appropriate simplices for outside points is implemented in `ImprovedTessellation.py`, a Python module which extends the `scipy.spatial.Delaunay` class.

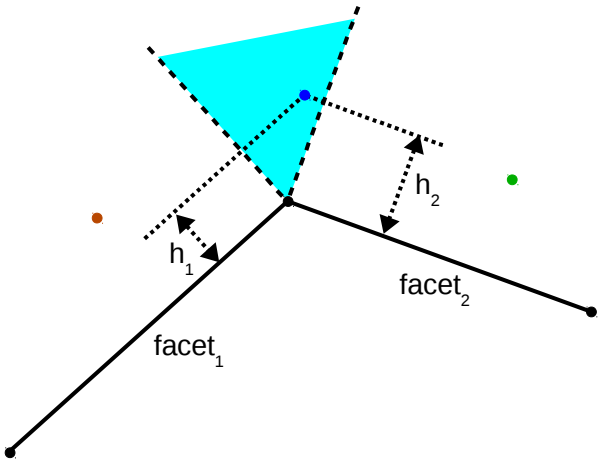

Figure 4.4: Figure illustrating the search for the nearest facet. The red dot is closest to facet 1 and the green dot to facet 2. The light blue shaded area shows the set of points which are at the same distance from both facets. The dark blue dot is one such point. The algorithm which searches for the nearest facet will select facet 2, because the angle between this facet and the dot is smaller. Equivalently, the height h_2 is larger.

Once the nearest simplex has been selected, extrapolation can proceed in much the same way as interpolation (for instance, the extrapolation coefficients will take on the same expressions as the interpolation coefficients). Given that the point lies outside the simplex, at least one of the coefficients involved in the linear combination associated with the extrapolation will be negative, and one can expect the quality of the result to be somewhat worse. Also, it is important to bear in mind that although the interpolation scheme leads to a continuous function within the convex hull, there is no guarantee that this function remains continuous outside the convex hull, where extrapolation is used instead. This could potentially be a source of error. Hence, `AIMS` incorporates a user-defined threshold on the absolute value of negative extrapolation coefficients which limits the region where the extrapolation is allowed to take place. Beyond this limit, no extrapolation is carried out and the point is rejected.

Bibliography

Ball, W. H. & Gizon, L. 2014, A&A, 568, A123

Coelho, H. R., Chaplin, W. J., Basu, S., et al. 2015, MNRAS, 451, 3011

Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, PASP, 125, 306

Golubitsky, O., Mazalov, V., & Watt, S. M. 2015, An Algorithm to Compute the Distance from a Point to a Simplex, poster for some conference, http://www.researchgate.net/publication/267801141__An_Algorithm_to_Compute_the_Distance_from_a_Point_to_a_Simplex

Kjeldsen, H., Bedding, T. R., & Christensen-Dalsgaard, J. 2008, ApJ, 683, L175

Metcalfe, T. S., Creevey, O. L., Doğan, G., et al. 2014, ApJS, 214, 27

Index

Symbols
1σ error bar 13

A
AIMS 7
AMP 9

B
barycentric coordinates 24

C
classic observable 7
classic observables 10
correlation 14, 15
covariance matrix 13

E
emcee 7, 9

F
frequency combination 13
frequency mapping 10
frequency ratio 14

G
grid of stellar models 7
grid of stellar models 7

I
interpolation 7, 19
    age 21
    coefficients 21
    track 21, 22

L
large frequency separation 13, 17
likelihood 9

M
Monte Carlo Markov Chain 7, 9

N
noise realisation 13

P
priors 9
probability distribution function 7
probability distribution function 9
pulsation frequencies 13

S
seismic observable 7
seismic observables 10
simplex 22
SpaceINN 3
surface corrections 15

T
tessellation 22, 23

W
walkers 10