AIMS tutorial

Asteroseismic **I**nference on a **M**assive **S**cale

Tutorial agenda

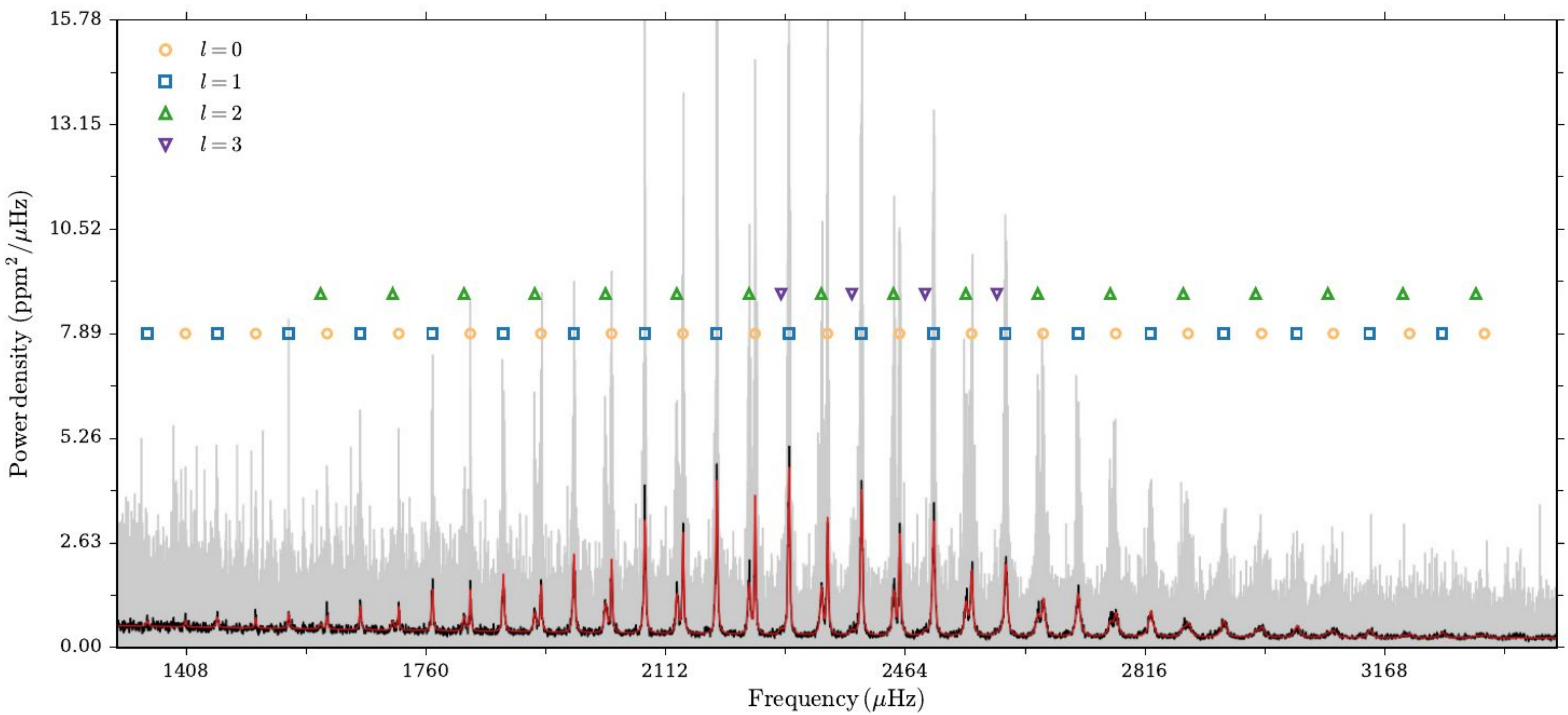
- Model fitting overview
- AIMS overview

- Exercises:
  - Modelling of 2/3 solar-like stars
  - Comparison between grids
  - Make your own binary grid
  - Test the interpolation
  - Modelling with different inputs (frequencies, ratios, average seismic parameters)

The Data

The Data

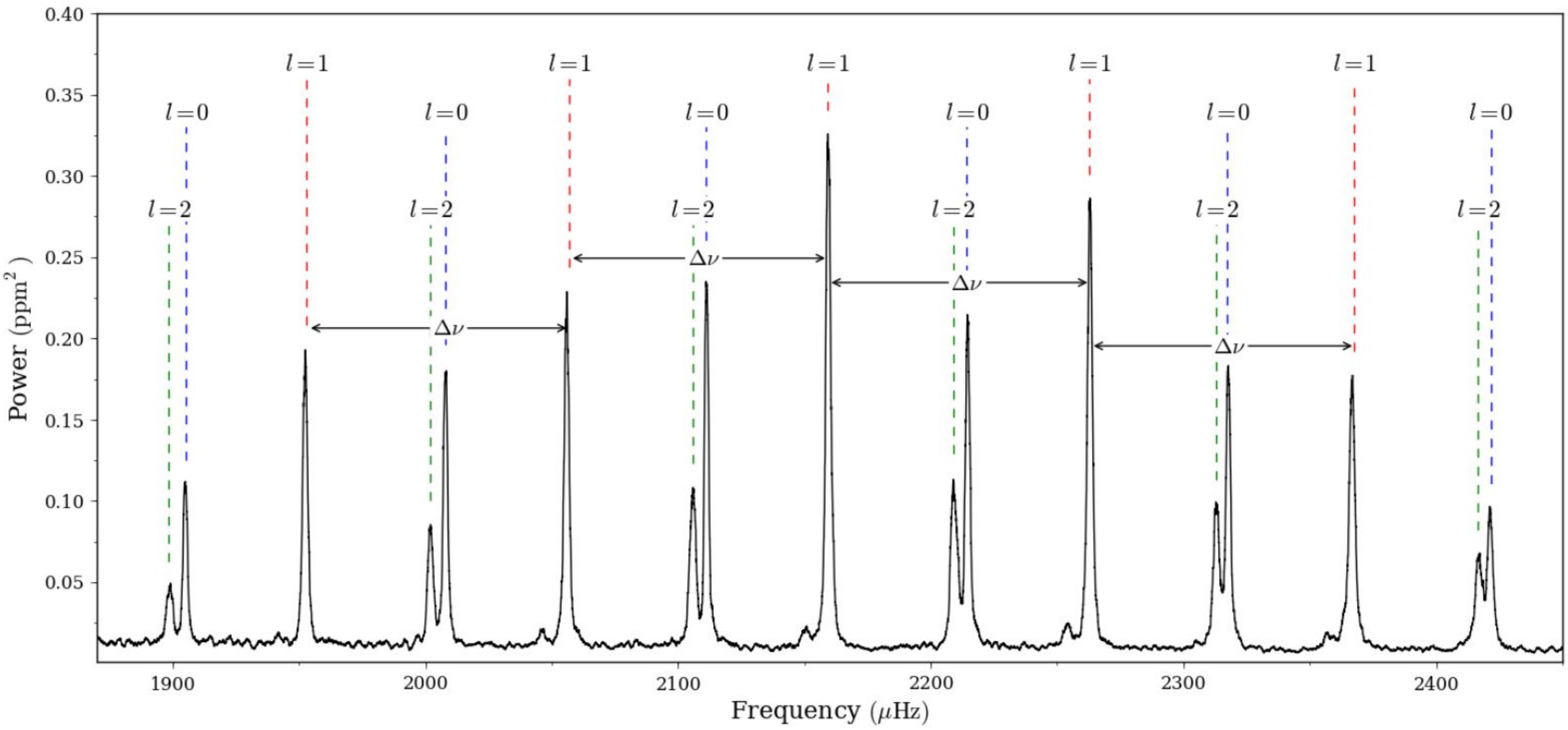
Individual frequencies



The Data

Average seismic parameters

$\Delta\nu$, ν_{max}, δ_{02}



Stellar model fitting

- “Boutique/à la carte” modelling -

Individual frequencies:

$$\chi^2_\nu = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{\nu_i^{(\mathrm{obs})}-\nu_i^{(\mathrm{mod})}}{\sigma_i}\right)^2$$

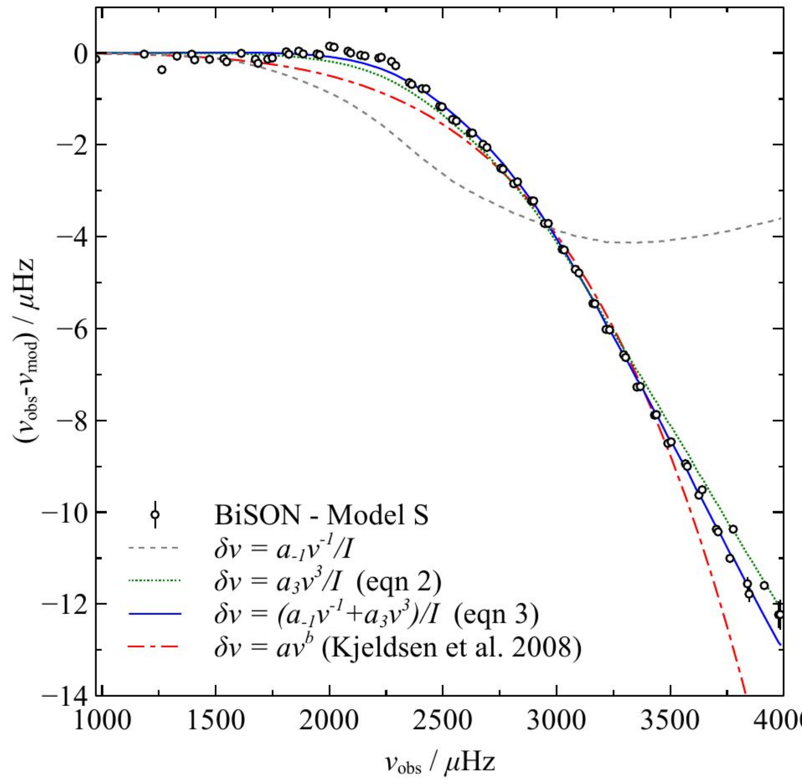
Stellar model fitting

- “Boutique/à la carte” modelling -

<u>Individual frequencies:</u>

$$\chi^2_\nu = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{\nu_i^{(\mathrm{obs})} - \nu_i^{(\mathrm{mod})}}{\sigma_i}\right)^2$$

Surface correction



Ball & Gizon (2014)

Stellar model fitting

- “Boutique/à la carte” modelling -

Individual frequencies:

$$\chi^2_\nu = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{\nu_i^{(\mathrm{obs})} - \nu_i^{(\mathrm{mod})}}{\sigma_i}\right)^2$$

$$\chi^2 = \chi^2_\nu + \left(\frac{T_{\mathrm{eff}}^{(\mathrm{obs})} - T_{\mathrm{eff}}^{(\mathrm{mod})}}{\sigma(T_{\mathrm{eff}})}\right)^2 + \left(\frac{[\mathrm{Fe/H}]^{(\mathrm{obs})} - [\mathrm{Fe/H}]^{(\mathrm{mod})}}{\sigma([\mathrm{Fe/H}])}\right)^2$$

Stellar model fitting

- “Boutique/à la carte” modelling -

Individual frequencies:

$$\chi^2_\nu = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{\nu_i^{(\mathrm{obs})} - \nu_i^{(\mathrm{mod})}}{\sigma_i}\right)^2$$

$$\chi^2 = \chi^2_\nu + \left(\frac{T_{\mathrm{eff}}^{(\mathrm{obs})} - T_{\mathrm{eff}}^{(\mathrm{mod})}}{\sigma(T_{\mathrm{eff}})}\right)^2 + \left(\frac{[\mathrm{Fe/H}]^{(\mathrm{obs})} - [\mathrm{Fe/H}]^{(\mathrm{mod})}}{\sigma([\mathrm{Fe/H}])}\right)^2$$

Likelihood function (assumes Gaussian errors):

$$\mathcal{L} = \left(\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma_i}\right) \times \exp(-\chi^2/2)$$

Stellar model fitting

Frequency difference ratios:

$$r_{01}(n) = \frac{d_{01}(n)}{\Delta\nu_1(n)}, \quad r_{10}(n) = \frac{d_{10}(n)}{\Delta\nu_0(n+1)} \qquad r_{02}(n) = \frac{\nu_{n,0} - \nu_{n-1,2}}{\Delta\nu_1(n)}$$

$$d_{01}(n) = \tfrac{1}{8}\left(\nu_{n-1,0} - 4\nu_{n-1,1} + 6\nu_{n,0} - 4\nu_{n,1} + \nu_{n+1,0}\right)$$

$$d_{10}(n) = -\tfrac{1}{8}\left(\nu_{n-1,1} - 4\nu_{n,0} + 6\nu_{n,1} - 4\nu_{n+1,0} + \nu_{n+1,1}\right)$$

Stellar model fitting

<u>Frequency difference ratios:</u>

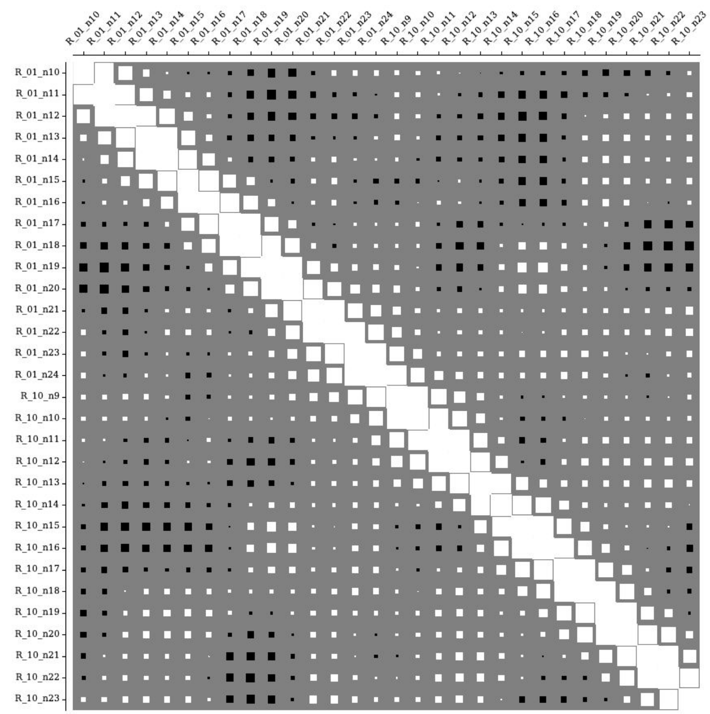
$$r_{01}(n) = \frac{d_{01}(n)}{\Delta\nu_1(n)}, \quad r_{10}(n) = \frac{d_{10}(n)}{\Delta\nu_0(n+1)} \qquad r_{02}(n) = \frac{\nu_{n,0} - \nu_{n-1,2}}{\Delta\nu_1(n)}$$

$$d_{01}(n) = \tfrac{1}{8}\left(\nu_{n-1,0} - 4\nu_{n-1,1} + 6\nu_{n,0} - 4\nu_{n,1} + \nu_{n+1,0}\right)$$

$$d_{10}(n) = -\tfrac{1}{8}\left(\nu_{n-1,1} - 4\nu_{n,0} + 6\nu_{n,1} - 4\nu_{n+1,0} + \nu_{n+1,1}\right)$$

$$\xcancel{\chi^2 = \frac{1}{N}\sum_k \left(\frac{x_{\rm obs}(k) - x_{\rm model}(k)}{\sigma(x_{\rm obs}(k))}\right)^2}$$

$$\chi^2 = \frac{1}{N}\left(\vec{x}_{\rm obs} - \vec{x}_{\rm model}\right)^T \mathbf{C}^{-1} \left(\vec{x}_{\rm obs} - \vec{x}_{\rm model}\right)$$



Lund et al. (2014)

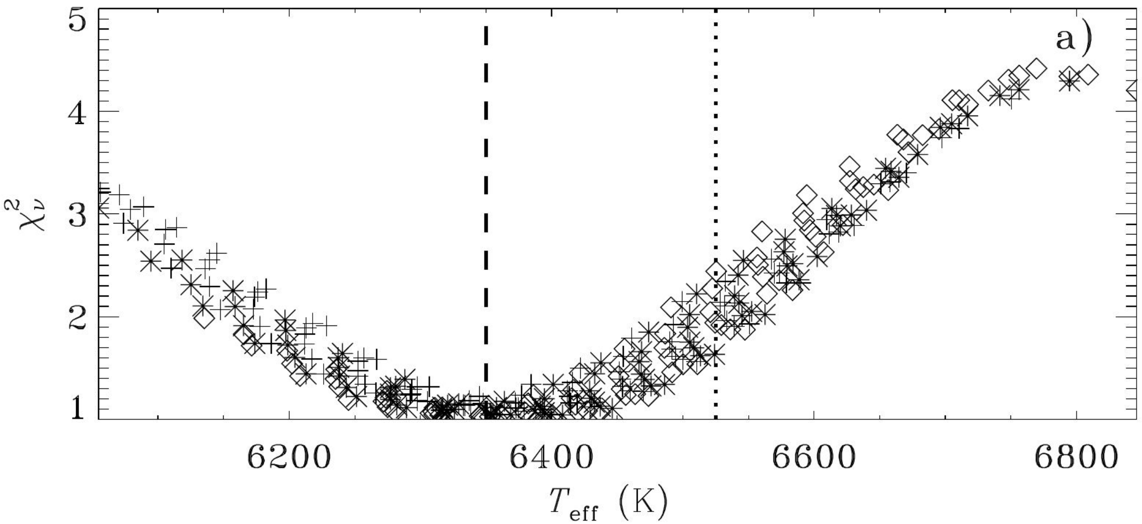
Optimisation methods

Maximum Likelihood

$$\mathcal{L} = (\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i}) \times \exp(-\chi^2/2)$$

Minimise: -ln(ℒ)

- Downhill Simplex
- Genetic algorithm (AMP)
- Minimum in grid



Christensen-Dalsgaard et al. (2011)

Optimisation methods

Bayesian Inference

(Tiago's and Enrico's talks)

Priors

$$p\left(\mathbf{v}|\mathcal{O}\right) \propto p\left(\mathbf{v}\right) \mathcal{L}\left(\mathcal{O}|\mathbf{v}\right)$$

$$\mathcal{L}\left(\mathcal{O}|\mathbf{v}\right) = \frac{1}{(2\pi)^{1/2}\sqrt{|\mathbf{C}|}} \exp\left(-\chi^2/2\right)$$

<u>Marginalise for posterior probability distribution:</u>

$$p\left(x|\mathcal{O}\right) = \int \delta(x(\mathbf{v}) - x) p\left(\mathbf{v}|\mathcal{O}\right) w_v d^3 v$$

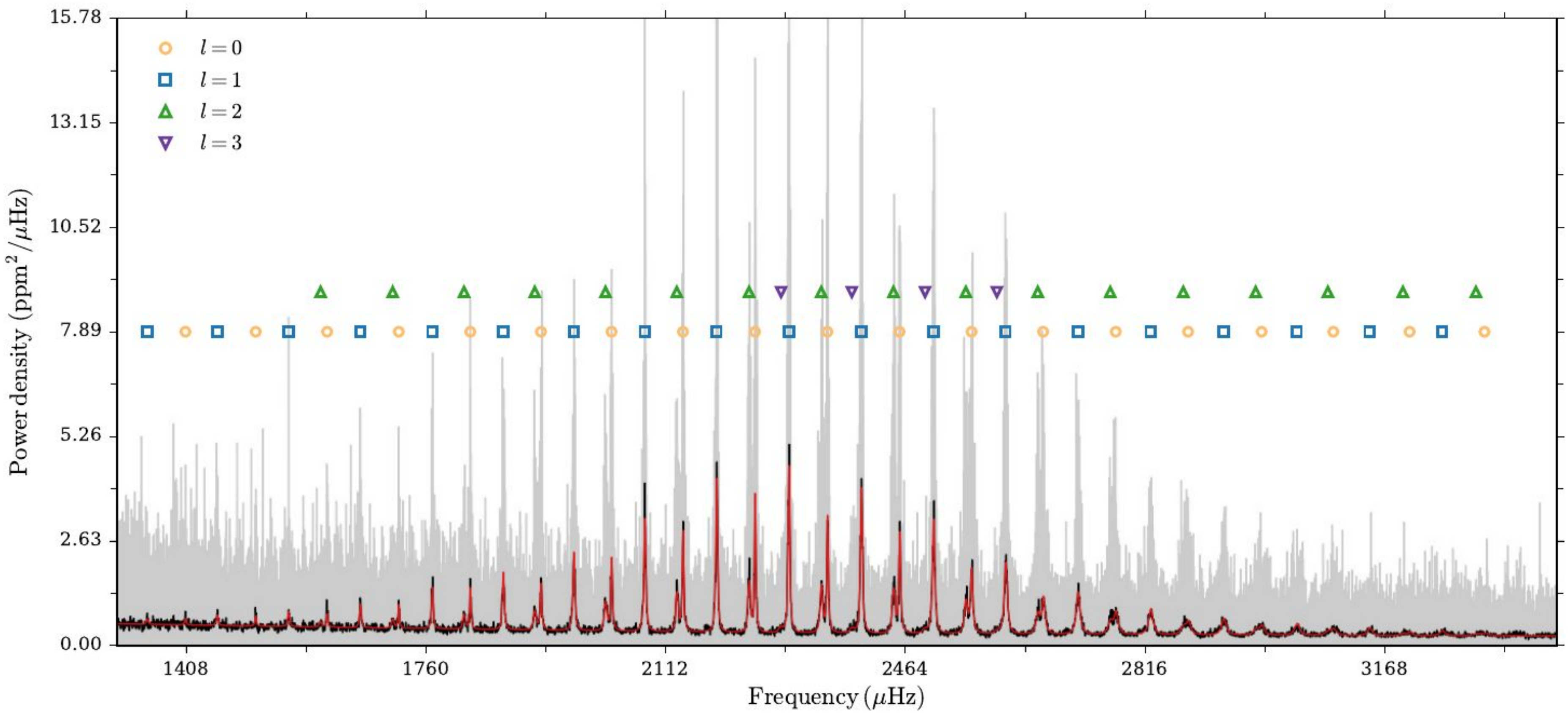
Weights (sampling volume correction).
- BASTA (Victor's method)

With MCMC the marginalisation is easy, simply make histogram

Nested sampling would also be an option

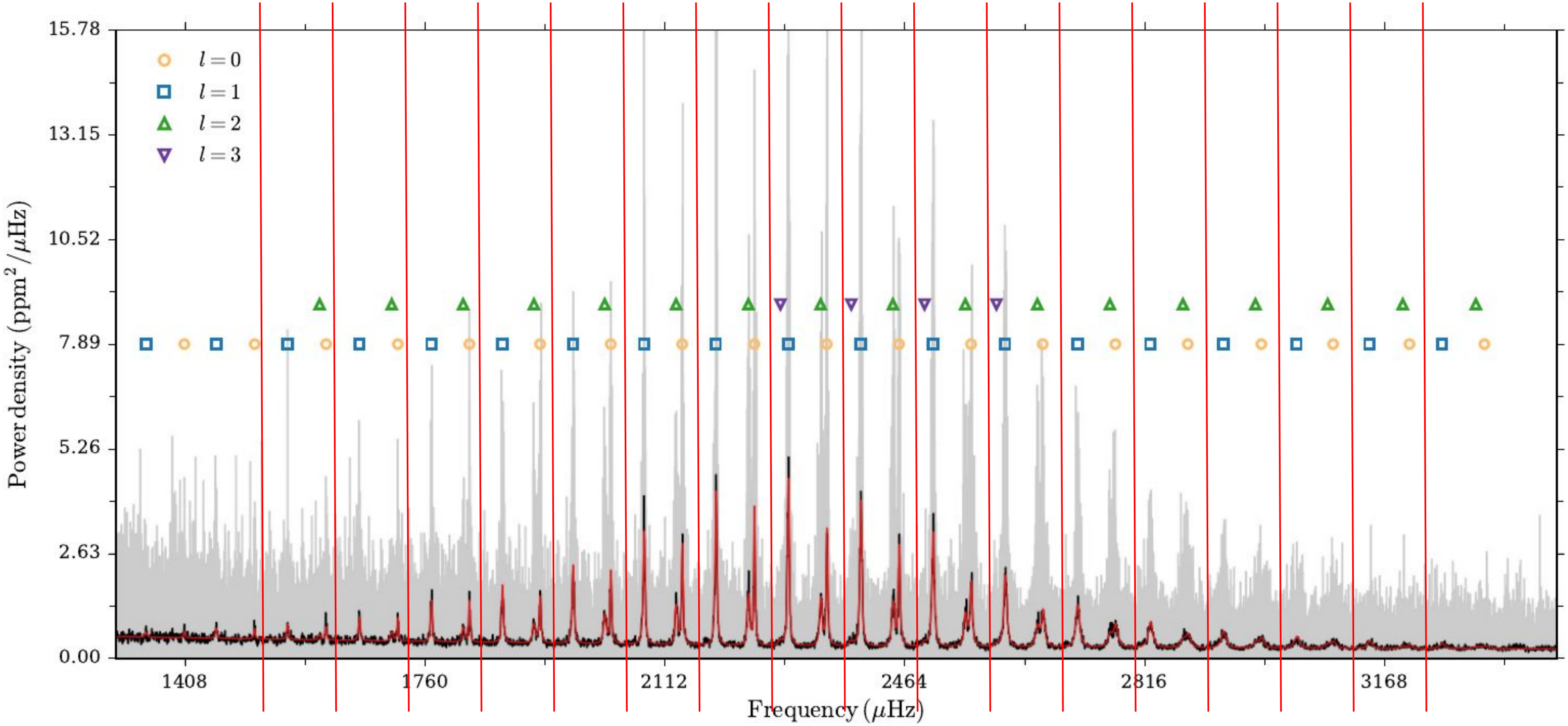
Stellar model fitting

The Echelle diagram:



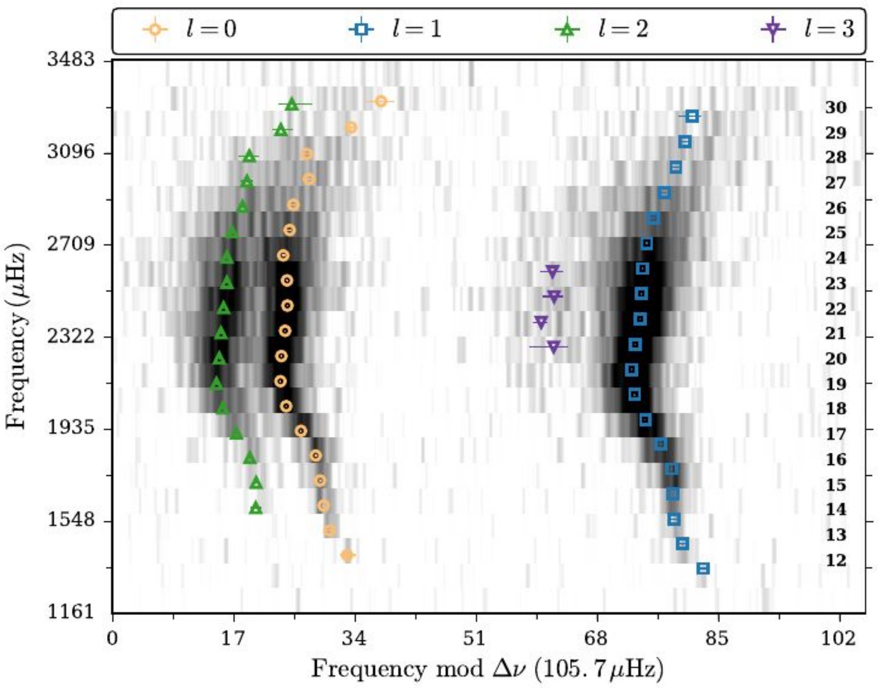
Stellar model fitting

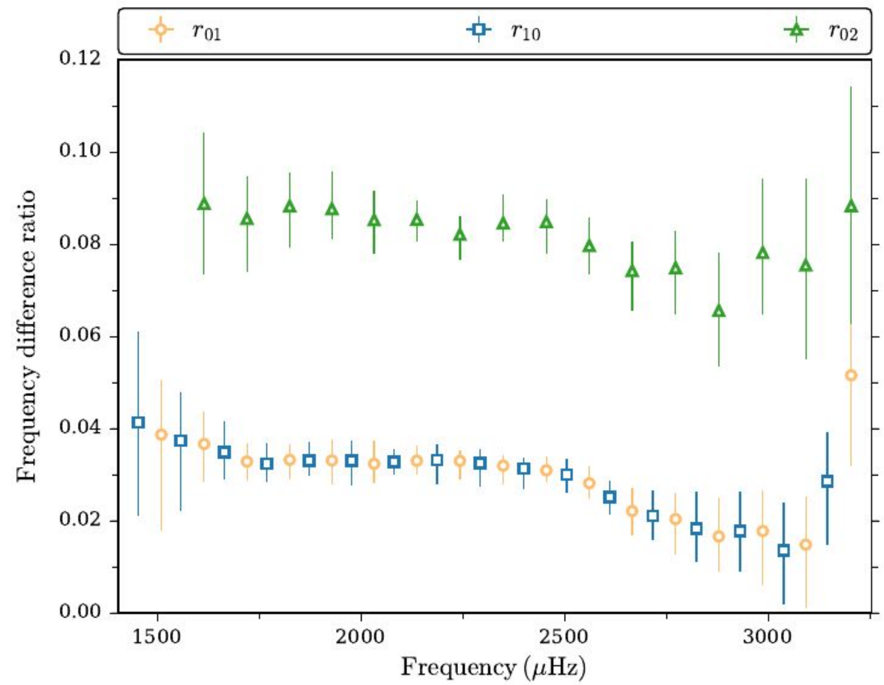
<u>The Echelle diagram:</u>



Stellar model fitting

The Echelle diagram:





Things to remember

<u>In model fitting/optimisation:</u>

- Surface correction
- Solar reference values for seismic parameters
- Correction to $\Delta\nu$?

<u>In constructing model grids (Victor's talk):</u>

- Solar reference value for Z/[Fe/H]
- Choice of opacities, nuclear reaction rates, EOS
- Treatment of convection
- Overshoot or not?
- Microscopic diffusion?
- $\Delta Y/\Delta Z$? = 1-3

$$[\mathrm{Fe/H}] = \log\left(\frac{Z/X}{Z_\odot/X_\odot}\right)$$

(Some of the) stellar model fitting codes on the market

- ASTFIT (J. Christensen-Dalsgaard) - The ASTEC Fitting method
  - ASTEC (Aarhus STellar Evolution Code) and ADIPLS (Aarhus adiabatic oscillation package)
- BASTA (V. Silva Aguirre) - BAyesian STellar Algorithm
  - GARSTEC (Garching stellar evolution code) and ADIPLS
- YMCM (S. Basu) - The Yale Monte Carlo Method
  - YREC (Yale Stellar Evolution Code)
- AMP (T. Metcalfe) - Asteroseismic Modelling Portal
- BeSPP (A. Serenelli) - Bellaterra Stellar Parameters Pipeline
  - GARSTEC and ADIPLS
- C2kSMO (Y. Lebreton) - Cesam2k Stellar Model Optimization Pipeline
  - Cesam2k evolutionary code and Líege Oscillations Code
- GOE (W. Ball) - GOEttingen pipeline
  - MESA (Modules for experiments in stellar astrophysics) and ADIPLS
- + V&A, SEEK, RADIUS, YB, RadEx10, OSM, Machine Learning methods (Earl's poster), …

- <u>Today: AIMS (MESA/CESTAM + ADIPLS)</u>

Tutorial agenda

- Model fitting overview ✔
- AIMS overview

- Exercises:
  - Modelling of 2/3 solar-like stars
  - Comparison between grids
  - Make your own binary grid
  - Test the interpolation
  - Modelling with different inputs (frequencies, ratios, average seismic parameters)

AIMS overview

Analytical priors & Observational constraints:
- seismic: ν_i, r_{02}, r_{01}, r_{10}, $\Delta\nu$
- classical: T_{eff}, L, [M/H], ...
- error bars and correlations

Grid of models
- n-dimensional ($n \geq 3$)
- pre-computed pulsation frequencies
  - optional surface effects

AIMS = “Asteroseismic Inference on a Massive Scale”
- unstructured linear interpolation
- MCMC approach with parallel tempering (via the python EMCEE package, Foreman-Mackey et al. 2013)

Estimated properties
- probability distribution functions for different parameters

Representative models
- provides coefficients for inter-polating acoustic structure
- allows inversions

Python coding + Sphinx documentation

Project Summary

Description

Name: "Asteroseismic Inference on a Massive Scale" (*AIMS*)

Goals:
- estimate stellar parameters and credible intervals/error bars
- chose a representative set or sample of reference models
- be computationally efficient

Inputs:
- classic constraints and error bars (Teff, L, ...)
- seismic constraints and error bars (individual frequencies)

Requirements:
- a *precalculated* grid of models including:
  - the models themselves
  - parameters for the model (M, R, Teff, age, ...)
  - theoretical frequency spectra for the models

Methodology:
- applies an MCMC algorithm based on the python package emcee. Relevant articles include:
  - Bazot et al. (2012, MNRAS 427, 1847)
  - Gruberbauer et al. (2012, ApJ 749, 109)
- interpolates within the grid of models using Delaunay tessellation (from the scipy.spatial package which is based on the Qhull library)
- modular approach: facilitates including contributions from different people

Contributors

Author:

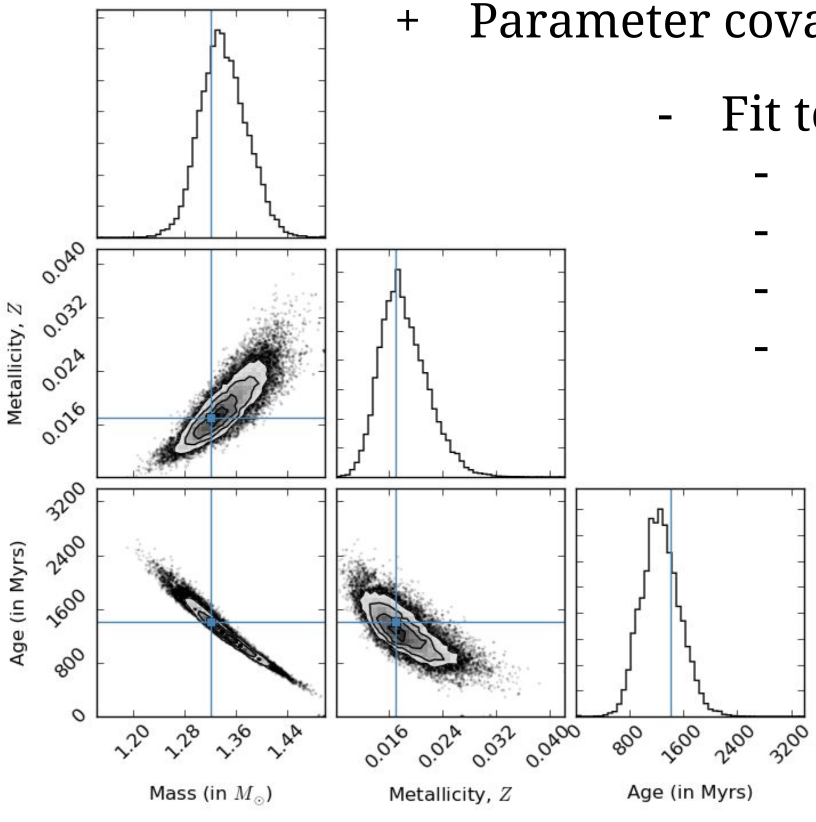
- Daniel R. Reese

AIMS overview

Basic idea: combine MCMC sampler (emcee) with interpolation in a grid of pre-computed models

- ➢ Start MCMC from scan of grid values
- ➢ Parameter values from marginalised posterior distributions

  \+ Parameter covariances



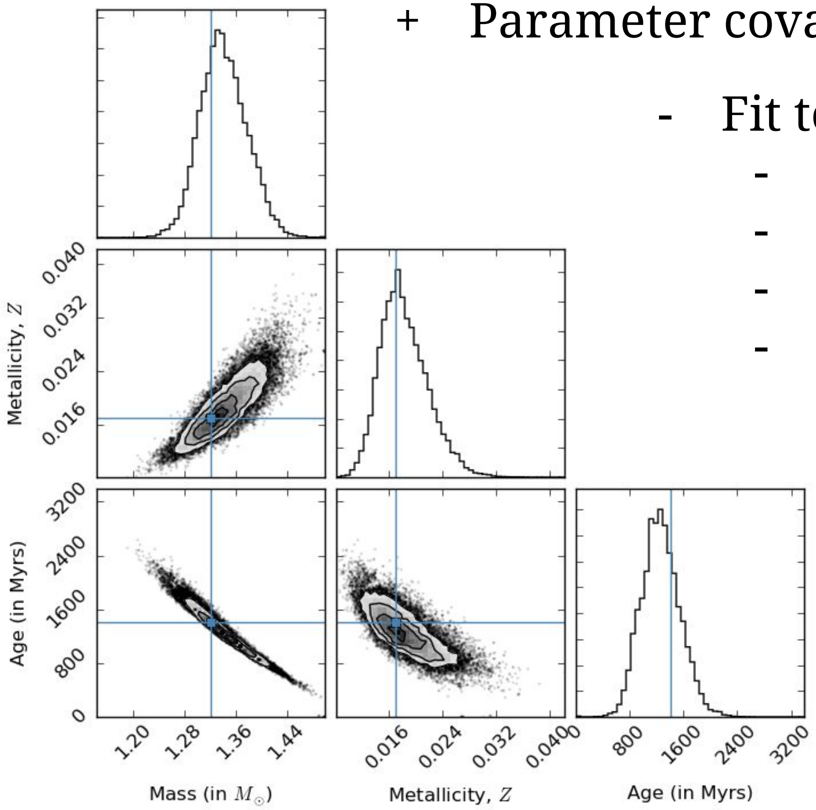
- Fit to:
  - Average seismic parameters ($\Delta\nu$, ν_{max})
  - Individual frequencies (ν_i)
  - Ratios (r_{02}, r_{01}, r_{10})
  - Classical parameters (L, T_{eff}, [M/H], …)

AIMS overview

Basic idea: combine MCMC sampler (emcee) with interpolation in a grid of pre-computed models

- ➢ Start MCMC from scan of grid values
- ➢ Parameter values from marginalised posterior distributions

+ Parameter covariances



- Fit to:
  - Average seismic parameters ($\Delta\nu$, ν_{max})
  - Individual frequencies (ν_i)
  - Ratios (r_{02}, r_{01}, r_{10})
  - Classical parameters (L, T_{eff}, [M/H], ...)

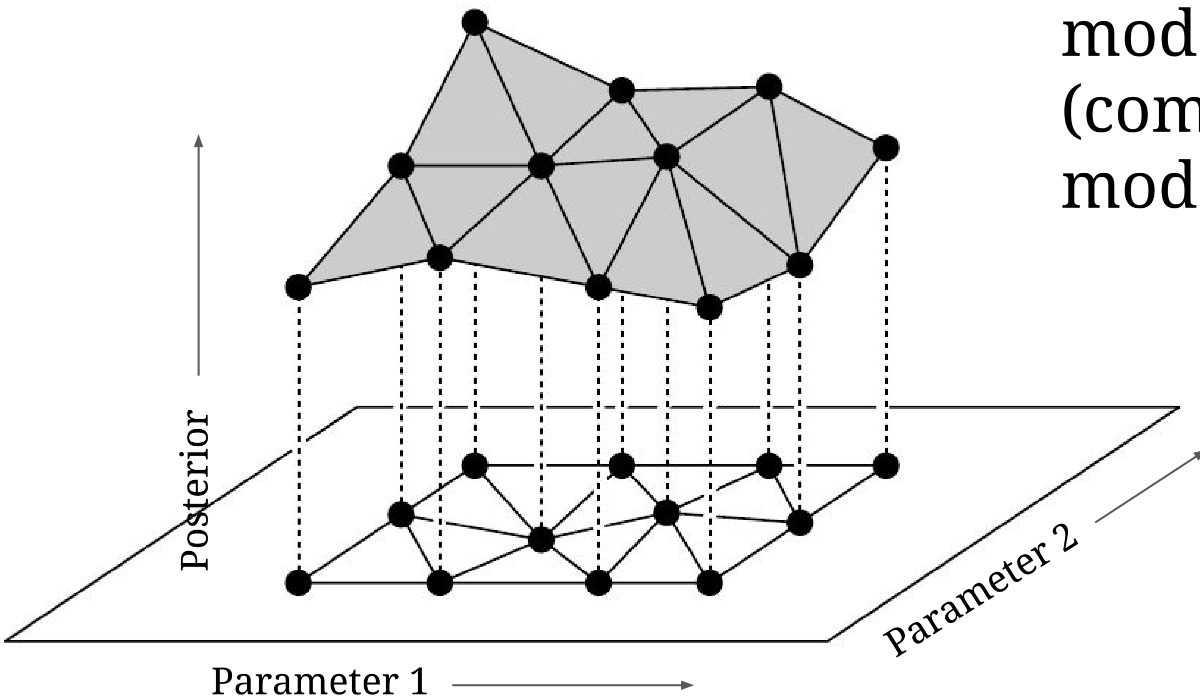
Goodman & Weare (2010) - Affine invariant ensemble sampler

Dan Foreman-Mackey (2013) - emcee

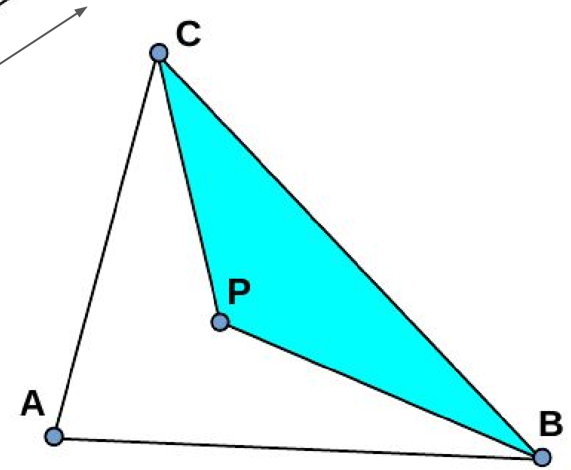
Interpolation in model grid for MCMC:

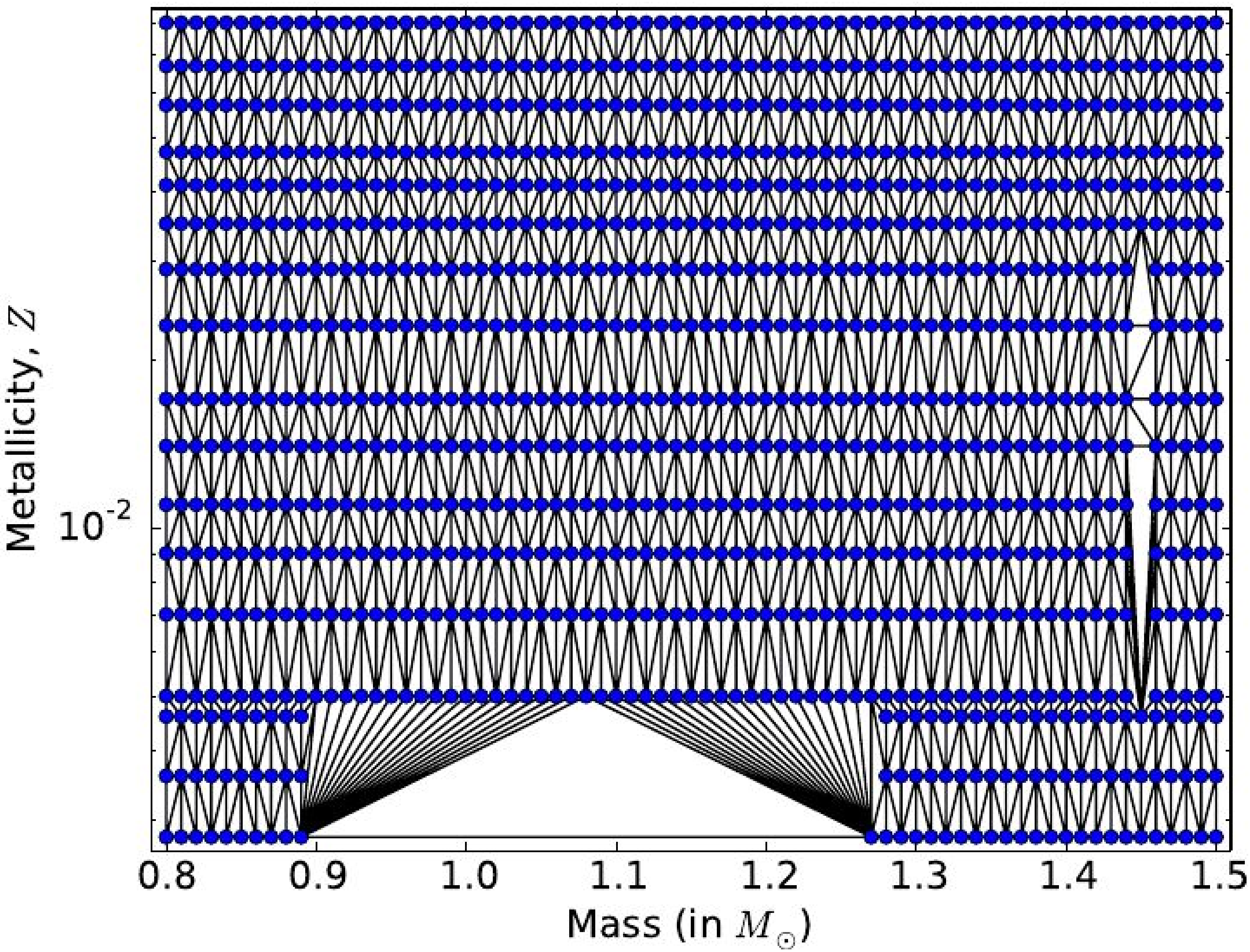
- Tessellate grid by Delaunay triangulation
- Linear barycentric interpolation on matching simplex

- User defined model grid (comparison of model codes)



Credit: Modified from “Computational Geometry: Algorithms and Applications”





Metallicity, Z
10^{-2}
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5
Mass (in $M_\odot$)

Tutorial agenda

- Model fitting overview ✔
- AIMS overview ✔

- Exercises:
  - Modelling of 2/3 solar-like stars
  - Comparison between grids
  - Make your own binary grid
  - Test the interpolation
  - Modelling with different inputs (frequencies, ratios, average seismic parameters)

Tutorial agenda

- Modelling of 2/3 solar-like stars

Star	T_{eff}	[Fe/H]	ν_{max}	$\Delta\nu$	log(g)	# modes
Star 1	5752 ± 77	-0.01 ± 0.05	3078 ± 13	134.90 ± 0.10	4.438 ± 0.005	39
Star 2	5309 ± 77	-0.54 ± 0.10	4198 ± 23	173.61 ± 0.16	4.539 ± 0.005	46
Star 3	6246 ± 77	-0.42 ± 0.10	1474 ± 5	75.18 ± 0.21	4.113 ± 0.008	53

File	Evolution code	Mass range	Z range	α_{MLT} range	Reference
data_cestam (891 MB)	CESTAM	0.7-1.2 $M_\odot$	0.0031 - 0.0927	1.3 - 1.9	
data_cestam_reduced (195 MB)	CESTAM	0.7-1.2 $M_\odot$	0.0031 - 0.0927	1.3 - 1.9	
data_mesa (253 MB)	MESA	0.8-1.5 $M_\odot$	0.0028 - 0.0800	1.8	Coelho et al. (2015)

- Modelling of 2/3 solar-like stars

Observational file

- Here “Star2.obsfile”
- Contains “l, (n), fre, dfre”
- (n) - check configuration file
- Spectroscopic parameters:
  - Teff/T
  - M/Fe_H/M_H
  - Dnu + numax
  - Log_g

- “Variable”, (prior), central val, error

```
#degree, order, frequency, error (symme
1  13  2598.38349518  0.588930971174
1  14  2774.02257403  1.12950118521
1  15  2945.51465181  0.174796956987
0  16  3035.59834241  0.152819169864
1  16  3117.85702049  0.13616411132
2  16  3196.23333897  0.655304455845
0  17  3207.0969515   0.113306722612
1  17  3289.78679165  0.085381332088
2  17  3369.65793372  0.363673651676
0  18  3379.36763931  0.0817260650742
1  18  3462.13300567  0.0700789385769
2  18  3542.20003263  0.245928123605
0  19  3551.64830171  0.0716555620972
1  19  3635.36065763  0.0666677131599
2  19  3714.75021773  0.169769022403
0  20  3724.28709494  0.0598254798124
1  20  3808.11186574  0.0769476874871
2  20  3888.75730197  0.172895253967
0  21  3897.2886574   0.0852799206882
1  21  3981.22024022  0.0927397666405
2  21  4061.81327253  0.151013507373
0  22  4070.35749431  0.0866667722671
1  22  4154.53837551  0.0982679152894
2  22  4234.77993254  0.256177939189
0  23  4243.87985003  0.100700324383
1  23  4328.1845516   0.144582054148
2  23  4411.17043124  0.496163259932
0  24  4417.59908676  0.192987434784
1  24  4502.34701305  0.168688734844
2  24  4586.69141157  0.751255993716
0  25  4592.83433385  0.30610113887
1  25  4677.23770887  0.216795586472
2  25  4761.52420681  1.15075937971
0  26  4766.43550485  0.384255148087
1  26  4851.57981076  0.378157893908
2  26  4938.47618549  0.720941690606
0  27  4941.06382251  0.569888776476
1  27  5028.83630555  0.679177768703
2  27  5111.8689779   1.08006272761
0  28  5117.76835496  0.742857648052
1  28  5199.58948529  0.853596000986
2  28  5284.77005007  1.28040966299
0  29  5290.19741305  0.927684826524
1  29  5378.16460302  0.730111471586
1  30  5551.01661272  0.781712156089
1  31  5729.69285433  0.615970903812
numax 4198 23
Dnu 173.62 0.16
Teff 5309 77
M -0.54  0.10
#log_g 4.539 0.005
```

```
#########################   Parallelisation   ##############################
# NOTE: this is currently implemented with multiprocessing, which duplicates
#       the memory in each process.  To be more memory efficient, turn off
#       parallelisation using the "parallel" parameter.
nprocesses  = 4      # number of processes (if running in parallel)
parallel    = True   # specifies whether to run in parallel

#########################   EMCEE control parameters   #####################
ntemps      = 5      # number of temperatures
nwalkers    = 100     # number of walkers (this number should be even)
nsteps0     = 1000     # number of burn-in steps
nsteps      = 2000     # number of steps
thin        = 10     # thinning parameter (1 out of thin steps will be kept ...)
thin_comb   = 100    # thinning parameter for output linear combinations of models
PT          = True   # use parallel tempering?

#########################   Initialisation   ###############################
tight_ball   = True  # initialise with a tight ball around best solution
max_iter     = 1000  # maximum number of iterations to find walker

# Ranges used around tight ball configuration for walkers.
# NOTES:
#   - these ranges will be re-centred around the parameters of the
#     best model in the grid
#   - the ranges on parameters related to surface amplitudes will be reset by AIMS
#   - exact names should be used as keys, since AIMS accesses
#     the relevant distributions by using the name as a key.
#   - it doesn't matter if there are supplementary parameters
#     which don't intervene. AIMS will simply ignore them.

tight_ball_range = {}
tight_ball_range["Mass"]     = ("Gaussian", [0.0, 0.30])
tight_ball_range["Z"]        = ("Gaussian", [0.0, 0.002])
tight_ball_range["log_Z"]    = ("Gaussian", [0.0, 0.05])
tight_ball_range["Age"]      = ("Gaussian", [0.0, 200.0])
tight_ball_range["alpha_MLT"]= ("Gaussian", [0.0, 0.2])
tight_ball_range["A_surf"]   = ("Gaussian", [0.0, 1.0])  # will be reset by AIMS
tight_ball_range["A3_surf"]  = ("Gaussian", [0.0, 1.0])  # will be reset by AIMS
tight_ball_range["Am1_surf"] = ("Gaussian", [0.0, 1.0])  # will be reset by AIMS
tight_ball_range["alpha_surf"] = ("Gaussian", [0.0, 1.0])  # will be reset by AIMS
tight_ball_range["b_Kjeldsen2008"] = ("Gaussian", [0.0, 1.0])  # will be reset by AIMS
tight_ball_range["beta_Sonoi2015"] = ("Gaussian", [0.0, 1.0])  # will be reset by AIMS
```

AIMS_configure.py

AIMS_configure.py

```
#########################   Radial orders   ################################
use_n        = True # use radial orders when comparing observations with models?
read_n       = True # read radial orders from input file?
assign_n     = False # use best model to reassign the radial order?
                    # NOTE: this supersedes "read_n"
#########################   Constraints   ##################################
# Determines the type of surface correction to include.  Options include:
#   - None: don't use any surface corrections
#   - "Kjeldsen2008": use surface corrections based on Kjeldsen et al. (2008)
#   - "Kjeldsen2008_scaling": use surface corrections based on Kjeldsen et al. (2008);
#                     The b exponent is determined based on the scaling relation
#                     from Sonoi et al. (2015)
#   - "Kjeldsen2008_2": use surface corrections based on Kjeldsen et al. (2008);
#                     The b exponent is a free parameter
#   - "Ball2014": use one-term surface corrections based on Ball & Gizon (2014)
#   - "Ball2014_2": use two-term surface corrections based on Ball & Gizon (2014)
#   - "Sonoi2015": use the surface corrections based on Sonoi et al. (2015)
#   - "Sonoi2015_scaling": use the surface corrections based on Sonoi et al. (2015);
#                     The beta exponent is determined based on the scaling relation
#                     from Sonoi et al. (2015)
#   - "Sonoi2015_2": use the surface corrections based on Sonoi et al. (2015);
#                     The beta exponent is a free parameter
surface_option = "Kjeldsen2008"
#surface_option = "Ball2014_2"
b_Kjeldsen2008 = 4.9  # exponent used in the Kjeldsen et al. surface corrections
beta_Sonoi2015 = 4.0  # exponent used in the Sonoi et al. surface corrections

# Set of seismic constraints to be used. Options include:
#   - "nu": individual frequencies
#   - "r02", "r01", "r10": various frequency ratios
#   - "dnu0": individual large frequency separation using l=0
#   - "avg_dnu": average large frequency separation using all l
#   - "avg_dnu0": average large frequency separation using l=0
# NOTE: combining "nu" with the other constraints leads to a (nearly)
#       singular covariance matrix and is not expected to give good
#       results.
#seismic_constraints = ["r02","r01","r10","avg_dnu0","nu_min0"]
seismic_constraints = ["nu"]
```

AIMS_configure.py

```
#########################    Weighting    ########################################
# Determines what type of weighting to apply to seismic and classic contraints.
# Options include:
#     - None: no weighting
#     - "Absolute": absolute weight applied to constraints
#     - "Relative": weights applied after normalising the classic and seismic
#                   constraints to have the same weight.
# NOTE: even with the relative weighting, classic_weight is kept as absolute.
weight_option = "Relative"
seismic_weight = 1.0
classic_weight = 1.0
```

AIMS_configure.py

```
#########################   Input   ########################################
write_data    = False                  # set this to True if you want to write a
                                      # binary grid file
mode_format   = "simple"              # specifies the format of the files with
                                      # the mode frequencies.  Options include:
                                      #   - "simple": the original AIMS format
                                      #   - "agsm": the agsm format from ADIPLS
npositive     = True                  # if True, only save modes with n >= 0 in
                                      # binary grid file
cutoff        = 5.0                   # remove frequencies above this value times
                                      # the acoustic cutoff-frequency
agsm_cutoff   = False                 # if True, only keep frequencies with icase=10010
                                      # (i.e. below the cutoff frequency as determined
                                      # by ADIPLS) in agsm files.  This test is in
                                      # addition to the above user-defined cutoff.
list_grid     = "list_z0131_reduced"     # file with list of models and characteristics.
                                      # only used when constructing binary file with
                                      # the model grid (i.e. write_data == True)
grid_params = ("Mass", "Z")
#grid_params = ("Mass", "alpha_MLT")
#grid_params = ("Mass", "alpha_MLT", "Z")    # primary grid parameters (excluding age)
                                      # only used when constructing binary file with
                                      # the model grid (i.e. write_data == True)
                                      # These parameters are used to distinguish
                                      # evolutionary tracks

#binary_grid = "../grid_files/data_z0131"
binary_grid = "../grid_files/data_mesa"
#binary_grid = "../grid_files/data_cestam"
                                      # binary file with model grid
                                      # this file is written to if write_data == True
                                      # this file is read from if write_data = False
```

Check the grid you use with: `>> analyse_grid.py grid`

This will tell you how to set `grid_param` and `user_params`

Note: `analyse_grid.py` Should be in AIMS folder because it depends on `AIMS.py`

AIMS_configure.py

```
#########################   Priors    ######################################
priors = {}                      # The priors will be defined thanks to this
                                 # dictionary.  It is very important to use
                                 # as the keys, the exact names of the
                                 # parameters which will be used in the MCMC
                                 # run (i.e. the grid parameter names,
                                 # including age).  Priors should not be
                                 # set on the amplitudes of surface term
                                 # corrections (if these are included) as it
                                 # will be done automatically by AIMS.
priors["Mass"] = ("Uniform", [0.8, 2])
priors["Z"]    = ("Uniform", [0.0028, 0.08])
priors["Age"]  = ("Uniform", [0.0, 5e3])
#########################   Interpolation    ###############################
scale_age = True                 # use a scaled age when interpolating
#########################   Interpolation tests    #########################
test_interpolation = False       # decide whether to test the interpolation.
                                 # If True, interpolation tests are carried
                                 # out for the above binary grid, and written
                                 # in binary format to a file which can
                                 # subsequently be analysed using plot_test.py.
interpolation_file = "interpolation_test"  # Name of the file to which to
                                 # write the results from the interpolation
                                 # tests.  This file can be analysed using
                                 # plot_test.py.
```

AIMS_configure.py

```
##########################   Output   ########################################
# choice of parameters: Mass, Radius, Luminosity, LogLum, Z, X, Xc,
#                        Fe_H, M_H, Age, Teff, Dnu, Rho, logg
output_params = ("Radius","Rho","Teff","Luminosity","Fe_H","Y","Z","numax")
output_dir    = "results"      # name of the root folder with the results
with_combinations = True       # decide whether to write file with model combinations
with_walkers     = True        # decide whether to plot walkers
with_echelle     = True        # decide whether to plot echelle diagrams
with_histograms = True         # decide whether to plot histograms
with_triangles  = True         # decide whether to make triangle plots
plot_extensions = ['png']      # extensions (and formats) for all simple plots
tri_extensions  = ['png']      # extensions (and formats) for triangle plots
# supported formats: eps, jpeg, jpg, pdf, pgf, png, ps, raw, rgba, svg, svgz, tif, tiff
```

Additional notes:

- You can in `AIMS_configure.py` set “`with_osm`” to “`False`” if you do not want to write output files for OSM (Optimal Stellar Model, by R. Samadi).
- You should make sure that the priors you set corresponds to the parameters of your grid. Again, check your grid using `analyse_grid.py`

- Modelling of 2/3 solar-like stars

Running AIMS:

```
>> setenv OMP_NUM_THREADS 1 (tcsh)
              or
>> export OMP_NUM_THREADS=1 (bash)
>> ./AIMS.py Star1.obsfile
```

Do not run your chains too long at this stage.

- Modelling of 2/3 solar-like stars

Running AIMS:

```
>> setenv OMP_NUM_THREADS 1 (tcsh)
           or
>> export OMP_NUM_THREADS=1 (bash)
>> ./AIMS Star1.obsfile
```

Results from run in:

- "results_big.txt"
- "results.txt"
- "samples.txt"
- "samples_big.txt"
- "best_MCMC_model.txt"
- "best_grid_model.txt"

Do not run your chains too long at this stage.

- Modelling of 2/3 solar-like stars

Running AIMS:

```
>> setenv OMP_NUM_THREADS 1 (tcsh)
            or
>> export OMP_NUM_THREADS=1 (bash)
>> ./AIMS Star1.obsfile
```

Results from run in:

- “results_big.txt”
- “results.txt”
- “samples.txt”
- “samples_big.txt”
- “best_MCMC_model.txt”
- “best_grid_model.txt”

Now try running with cestam grid

Update your AIMS configuration file

- Modelling of 2/3 solar-like stars

Running AIMS:

```
>> setenv OMP_NUM_THREADS 1 (tcsh)
            or
>> export OMP_NUM_THREADS=1 (bash)
>> ./AIMS Star1.obsfile
```

Results from run in:

- “results_big.txt”
- “results.txt”
- “samples.txt”
- “samples_big.txt”
- “best_MCMC_model.txt”
- “best_grid_model.txt”

Watch out for l=3 modes in Star1 with cestam grid

Updated versions of emcee have issues with the autocorrelation calculation of the MCMC chains, so you might want to comment out line 1810 in AIMS.py to remove this calculation

- Modelling of 2/3 solar-like stars

Results for Star1?:

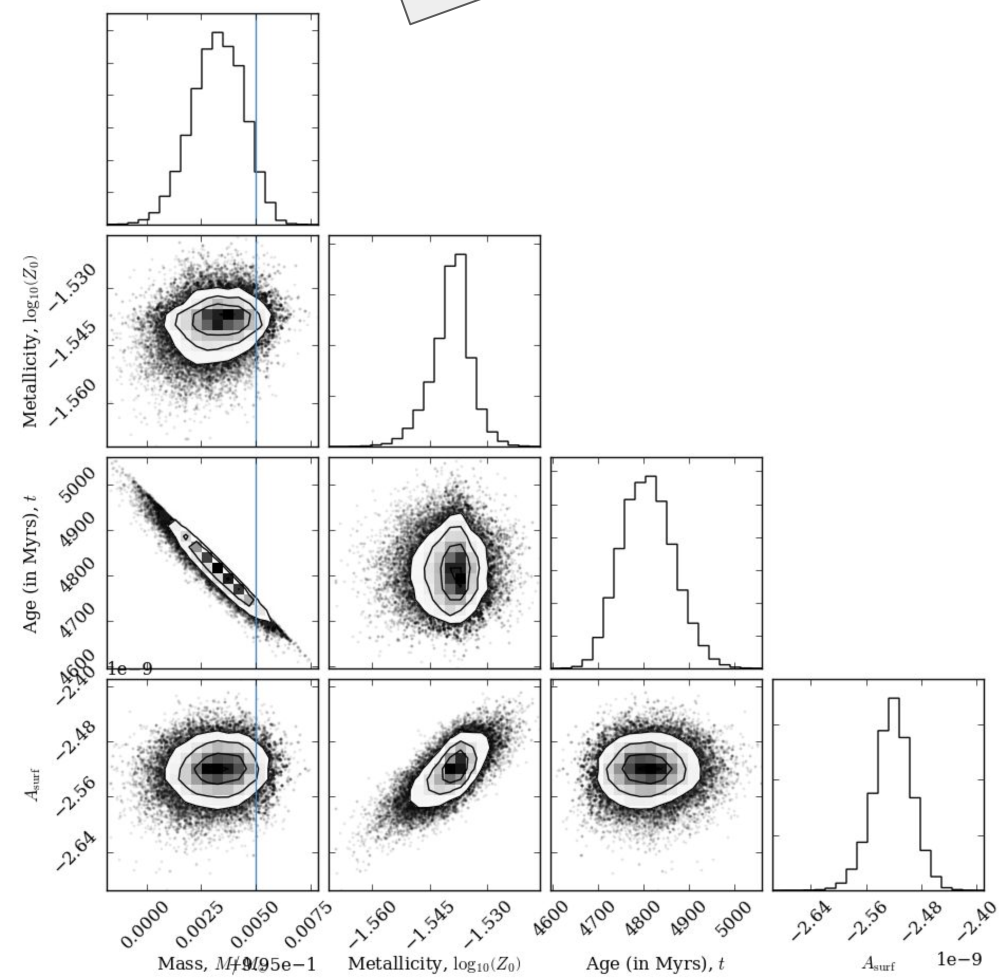
Mass: ?
Radius: ?
Age: ?

- Modelling of 2/3 solar-like stars

Notice small errors - likely from initialisation.

Results for Star1?:

Mass: ~9.98±0.012 $M_{\odot}$
Radius: ~0.9998±0.001 $R_{\odot}$
Age: ~4809±56 Myr



- Modelling of 2/3 solar-like stars

Results for Star1?:

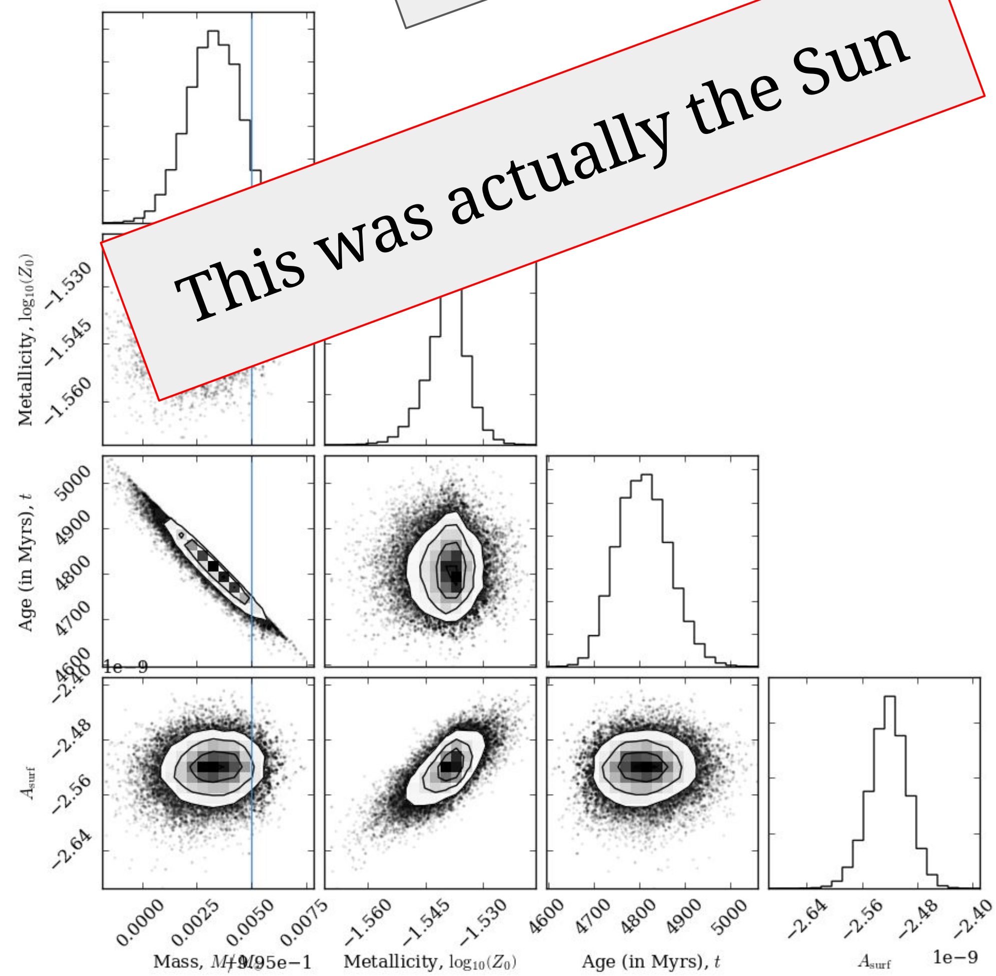
Mass: ~9.98±0.012 $M_{\odot}$

Radius: ~0.9998±0.001 $R_{\odot}$

Age: ~4809±56 Myr

Notice small errors - likely from initialisation.

This was actually the Sun



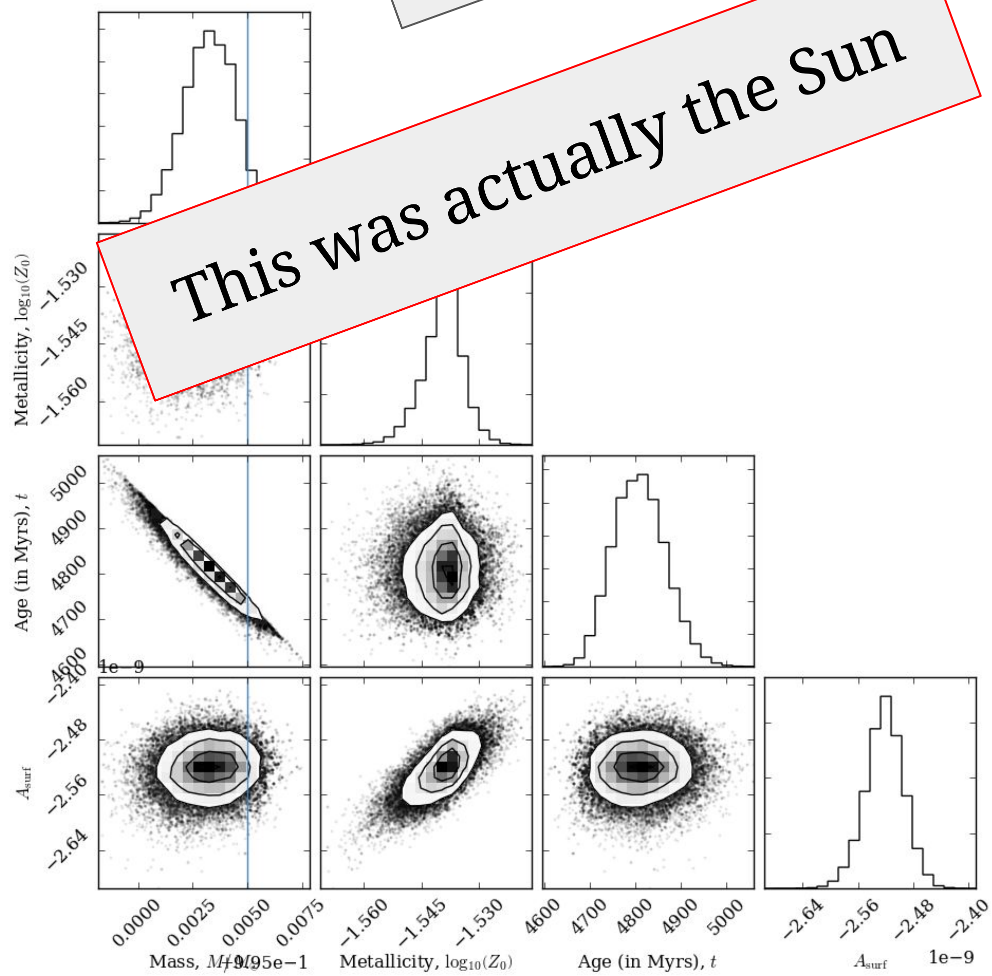
- Modelling of 2/3 solar-like stars

Results for Star1?:

Mass: ~9.98±0.012 $M_\odot$
Radius: ~0.9998±0.001 $R_\odot$
Age: ~4809±56 Myr

Notice small errors - likely from initialisation.

This was actually the Sun



You might want to try initialising with `tight_ball=False`

- Priors on surface correction should not be "`uninformative`"
- Will take longer to converge

- Modelling of 2/3 solar-like stars

Results for Star2?:

Mass: ?
Radius: ?
Age: ?

- # Modelling of 2/3 solar-like stars

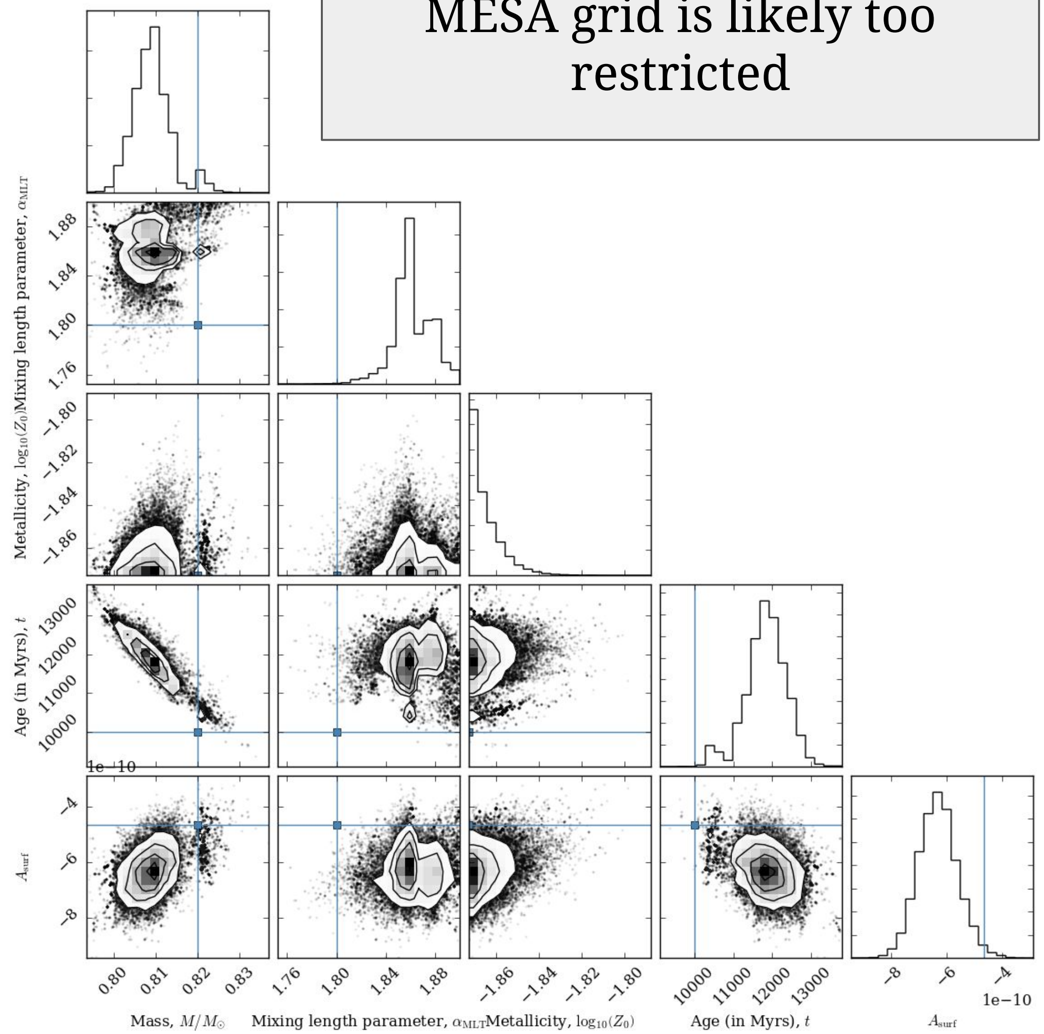
Results for Star2?:

Mass: 0.809±0.0004 $M_\odot$

Radius: 0.789±0.002 $R_\odot$

Age: 11838±530 Myr

MESA grid is likely too restricted



- Modelling of 2/3 solar-like stars

Results for Star3?:

Mass: ?

Radius: ?

Age: ?

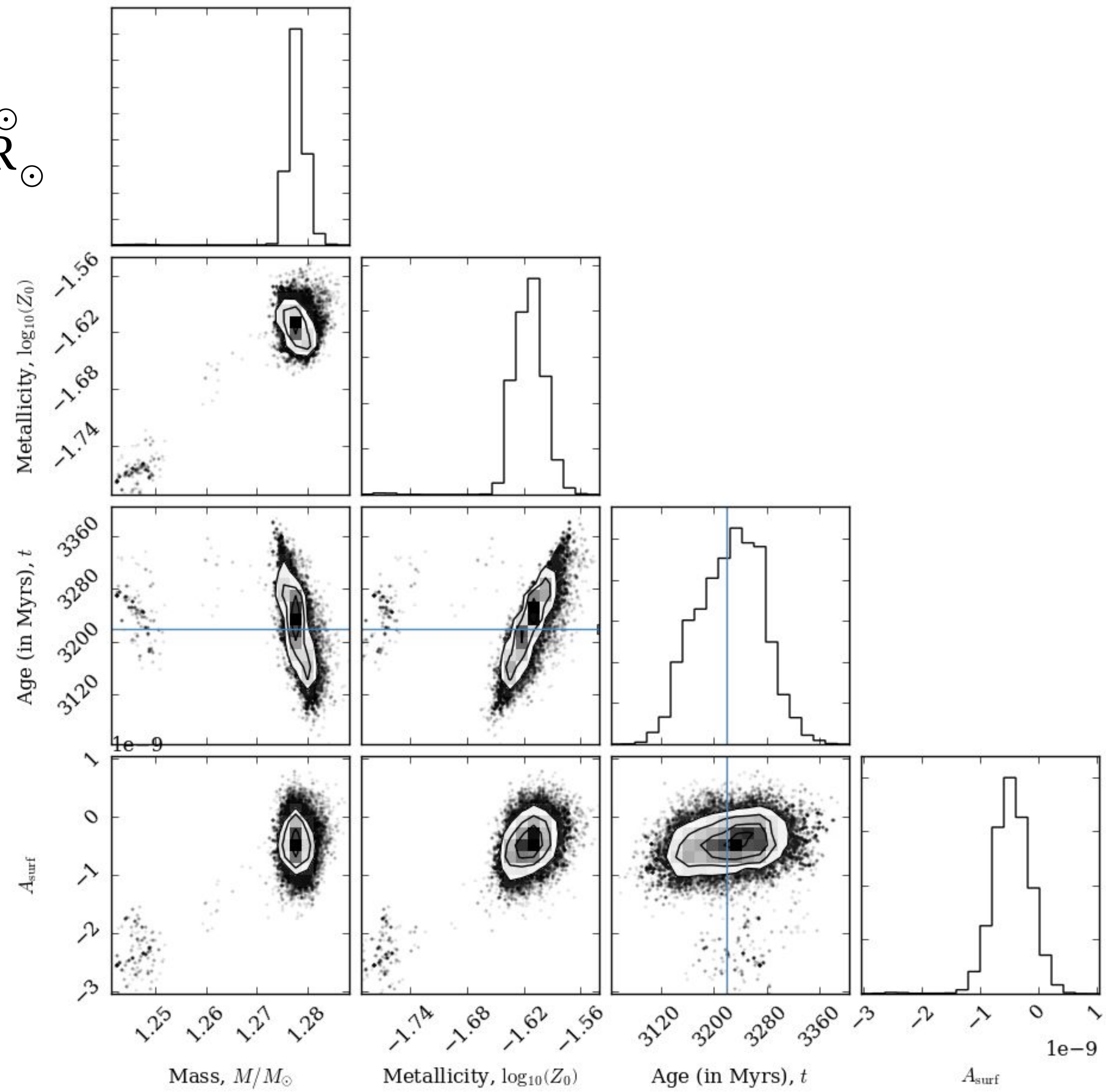
- Modelling of 2/3 solar-like stars

Results for Star3?:

Mass: ~1.278±0.003 $M_\odot$

Radius: ~1.617±0.002 $R_\odot$

Age: ~3224±50 Myr



Tutorial agenda

- Model fitting overview ✔
- AIMS overview ✔

- Exercises:
    - Modelling of 2/3 solar-like stars ✔
    - Comparison between grids ✔
    - Make your own binary grid
    - Test the interpolation
    - Modelling with different inputs (frequencies, ratios, average seismic parameters)

- # Make your own binary grid

`AIMS_configure.py`

```
#########################   Input   ########################################
write_data    = False           # set this to True if you want to write a
                               # binary grid file
mode_format   = "simple"       # specifies the format of the files with
                               # the mode frequencies.  Options include:
                               #   - "simple": the original AIMS format
                               #   - "agsm": the agsm format from ADIPLS
npositive     = True           # if True, only save modes with n >= 0 in
                               # binary grid file
cutoff        = 5.0            # remove frequencies above this value times
                               # the acoustic cutoff-frequency
agsm_cutoff   = False          # if True, only keep frequencies with icase=10010
                               # (i.e. below the cutoff frequency as determined
                               # by ADIPLS) in agsm files.  This test is in
                               # addition to the above user-defined cutoff.
list_grid     = "list_z0131_reduced"    # file with list of models and characteristics.
                               # only used when constructing binary file with
                               # the model grid (i.e. write_data == True)
grid_params = ("Mass", "Z")
#grid_params = ("Mass", "alpha_MLT")
#grid_params = ("Mass", "alpha_MLT", "Z")   # primary grid parameters (excluding age)
                               # only used when constructing binary file with
                               # the model grid (i.e. write_data == True)
                               # These parameters are used to distinguish
                               # evolutionary tracks

#binary_grid = "../grid_files/data_z0131"
binary_grid = "../grid_files/data_mesa"
#binary_grid = "../grid_files/data_cestam"
                               # binary file with model grid
                               # this file is written to if write_data == True
                               # this file is read from if write_data = False
```

Check the grid you use with: `>> analyse_grid.py grid`

Should be in AIMS folder

- Make your own binary grid

AIMS_configure.py

```
##########################   User-defined parameters   ######################
# This variable allows the user to introduce supplementary parameters in
# addition to the parameters hard-coded in to AIMS.  These parameters
# can then be used as output parameters (see output_params) and/or even
# as grid parameters used to define evolutionary tracks (see grid_params).
#
# This variable must be a list (or tuple) of pairs of strings.  The first
# string corresponds to the name of the variable which should be used, for
# instance, in the grid_params and output_params variables.  The second
# string is the fancy latex name for this variable.  Allowance needs to
# be made for a prefix and and postfix (hence the two "%s").  These will
# be replaced by appropriate strings if, for instance, one asks for the
# log of this parameter.

#user_params = (("Xc", r'Central hydrogen, $%sX_c%s$'),)
user_params = (("alpha_MLT", r'Mixing length parameter, $%s\alpha_{\mathrm{MLT}}%s$'), \
               ("Zs", r'Surface metallicity, $%sZ_s%s$'), \
               ("Xs", r'Surface hydrogen, $%sX_s%s$'), \
               ("Zc", r'Central metallicity, $%sZ_c%s$'), \
               ("Xc", r'Central hydrogen, $%sX_c%s$'))

#user_params = (('alpha_MLT', r'Mixing length parameter, $%s\alpha_{\mathrm{MLT}}%s$'), \
#               ('Zs', 'Surface metallicity, $%sZ_s%s$'), \
#               ('Xs', 'Surface hydrogen, $%sX_s%s$'),     \
#               ('Zc', 'Central metallicity, $%sZ_c%s$'), \
#               ('Xc', 'Central hydrogen, $%sX_c%s$'),     \
#               ('Tc', 'Central temperature, $%sT_c%s$ (in K)'))
```

Check the grid you use with: `>> analyse_grid.py grid`

Should be in AIMS folder

- # Make your own binary grid

- Make a grid of models with your favourite stellar evolution code
- Compute oscillation mode frequencies for each model
- Make a “model list”, listing the parameters of your models
  - First line gives prefix to the folder location of the oscillation files for each model (and possibly a post fix to specify ending of filenames - default is “.freq”)
  - Following columns should contain:
    - Stellar mass (in “g”)
    - Stellar radius (in “cm”)
    - Stellar luminosity (in “g cm^2 s^{-3}”)
    - Initial metallicity (Z_0)
    - Initial Hydrogen (X_0)
    - Stellar age (in “Myr”)
    - The effective temperature (in “K”)
    - Following columns are parameters specified in “`user_params`” in `AIMS_configure.py`

- # Make your own binary grid

Use grid of models provided on website (z0131)

- Check the `user_params` and `grid_params` under the Notes

```
data_mesa
=========
grid_params = ('Mass', 'log_Z')
user_params = (('alpha_MLT', r'Mixing length parameter, $%s\alpha_{\mathrm{MLT}}%s$'), \
               ('Zs', 'Surface metallicity, $%sZ_s%s$'), \
               ('Xs', 'Surface hydrogen, $%sX_s%s$'),    \
               ('Zc', 'Central metallicity, $%sZ_c%s$'), \
               ('Xc', 'Central hydrogen, $%sX_c%s$'))

Number of dims.:   3
Number of tracks:  1086
Number of models:  72154
Number of modes:   11713596
Range on parameter Mass: 8.00004e-01 to 1.50001e+00
Range on parameter log_Z: -2.55284e+00 to -1.09691e+00
```

- Make your model list file (`list_z0131_reduced`)
  - Remember to change prefix in file

```
>> ./AIMS.py
```

- Make your own binary grid

Use grid of models provided on website (z0131)

- Check the `user_params` and `grid_params` under the Notes
- Make your model list file (`list_z0131_reduced`)
  - Remember to change prefix in file

```
>> ./AIMS.py
```

Remove "`z0131_22_15_00_087`" from `list_z0131_reduced`

Tutorial agenda

- Model fitting overview ✔
- AIMS overview ✔

- Exercises:
    - Modelling of 2/3 solar-like stars ✔
    - Comparison between grids ✔
    - Make your own binary grid ✔
    - Test the interpolation
    - Modelling with different inputs (frequencies, ratios, average seismic parameters)

- Test the interpolation

- Test the interpolation

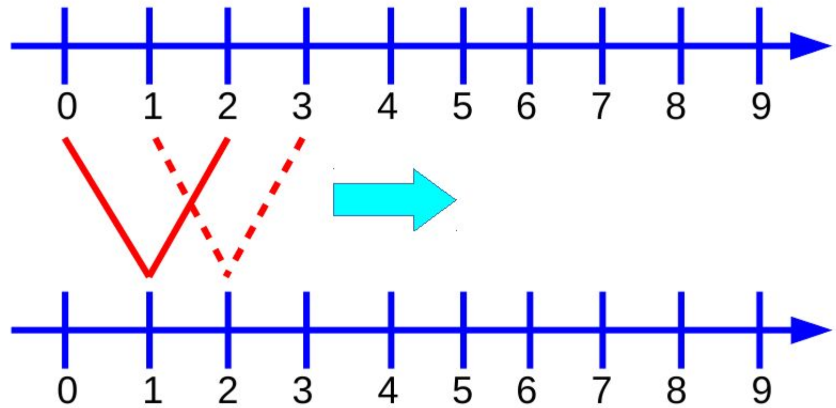
`AIMS_configure.py`

```
#########################   Priors    ######################################
priors = {}                       # The priors will be defined thanks to this
                                  # dictionary.  It is very important to use
                                  # as the keys, the exact names of the
                                  # parameters which will be used in the MCMC
                                  # run (i.e. the grid parameter names,
                                  # including age).  Priors should not be
                                  # set on the amplitudes of surface term
                                  # corrections (if these are included) as it
                                  # will be done automatically by AIMS.
priors["Mass"] = ("Uniform", [0.8, 2])
priors["Z"]    = ("Uniform", [0.0028, 0.08])
priors["Age"]  = ("Uniform", [0.0, 5e3])
#########################   Interpolation    ###############################
scale_age = True                  # use a scaled age when interpolating
#########################   Interpolation tests    #########################
test_interpolation = False        # decide whether to test the interpolation.
                                  # If True, interpolation tests are carried
                                  # out for the above binary grid, and written
                                  # in binary format to a file which can
                                  # subsequently be analysed using plot_test.py.
interpolation_file = "interpolation_test"  # Name of the file to which to
                                  # write the results from the interpolation
                                  # tests.  This file can be analysed using
                                  # plot_test.py.
```

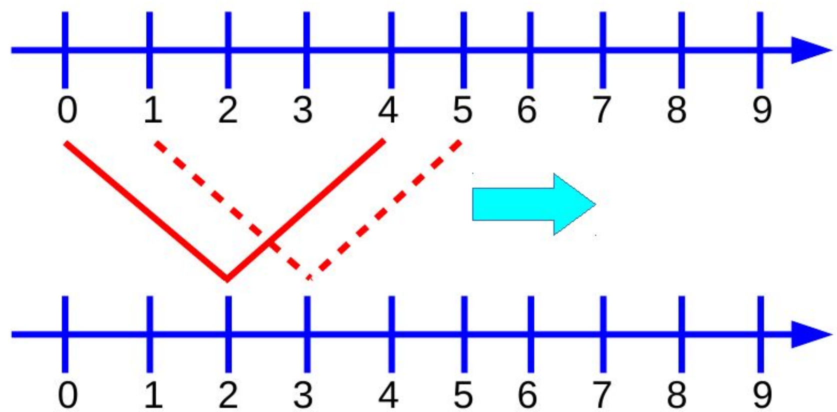
```
>> ./AIMS.py
```

```
>> ./plot_interpolation_test.py interpolation_test
```

nincr = 1



nincr = 2



Tutorial agenda

- Model fitting overview ✔
- AIMS overview ✔

- Exercises:
  - Modelling of 2/3 solar-like stars ✔
  - Comparison between grids ✔
  - Make your own binary grid ✔
  - Test the interpolation ✔
  - Modelling with different inputs (frequencies, ratios, average seismic parameters)

Examples of what not to do:

- Incorrect grid parameters
- Bad “n” values for your model frequencies
- Remember if you have “n” values in your obs file, and correct configuration file accordingly
- Does your model grid cover sufficient “n”+”l” ranges to reproduce your observations?
- Check corner plot to see if your parameter space is too constrained (model grid + priors)
- Priors should match your grid (try `analyse_grid.py`)

The END