

# Manual for ANIPAR (Automatic Near-Infrared PARameter finder

Pedro Sarmento

March 11, 2021

## **Abstract**

This manual contains instructions to use all the routines included with ANIPAR. A detailed description of the associated files is included as well.

This folder contains every code required to create a grid of synthetic spectra, normalize the spectra of M dwarfs, selecting the best normalization for each star, running turbospectrum for that normalization, and determining stellar parameters for them.

The pipeline uses code from iSpec, the radiative transfer code Turbospectrum, and MARCS GES stellar models, together with a custom line list and a linemask, to determine these stellar parameters.

This file contains instructions on how to use the codes included with the pipeline, and can serve as step-by-step instruction booklet to obtain stellar parameters from raw spectra.

Contained are codes and instructions to download APOGEE spectra, converting them into usable iSpec format, normalizing them, finding the best normalization through  $\chi^2$  minimization, and finding Turbospectrum's best fit and final parameters for the spectra.

# Chapter 1

## Table of Contents

### 1.1 Python routines

- `build_download_list.py` - creates a list of APOGEE stars in a format to download
- `fits_to_txt.py` - converts spectra from fits format into an iSpec readable format.
- `SynthesizeSpectralGrid.py` - creates a Grid of synthetic spectra from the combinations available in `isochrones.dat`
- `normalized_Mdwarfs_grid.py` - used for normalization of M dwarf spectra with the different template spectra
- `ChiSqCalc.py` - Calculates the  $\chi^2$  between the spectra in `/normalized_spectra` and the templates used to generate them, outputs list to a table
- `BestTemplateCalc.py` - Automatically finds best template per star + all other potential templates within a given margin, outputs list to a table
- `ispec.ParameterFinder.py` - Calculates best fits + parameters for spectra listed in a given table

### 1.2 Auxiliary files

- `isochrones.dat` - PARSEC isochrones calculated for 1Gyr, header deleted
- `isochrones_header.dat` - PARSEC isochrones calculated for 1Gyr, header included, not actually used in the code but important for reference
- `linemask.txt` - linemask created from star Ross-128, containing the lines best suited for parameter determination
- `barber_Mdwarf_final.list` - Final line list for M dwarfs, containing important elemental transitions and over 1 000 000 water lines relevant for M dwarf spectra
- `My_stars.fits` - example list of interesting stars for download (planet-host M dwarfs)

## 1.3 Folders

- /default\_output/ -i every code will place its output by default inside this folder
- /Grid/ -i Contains a grid of synthetic spectra with the parameters from isochrones.dat; a new Grid can be created by running Synthesize\_SpectralGrid.py again, with a new set of isochrones
- /input/; /ispec/; /synthesizer/ -i contains files required for our run of ispec, do not edit unless you know what you are doing
- /raw spectra/ -i place raw spectra (txt, ispec format) in this folder, normalizer\_Mdwarfs\_grid.py will take spectra from here as input
- /normalized\_spectra/ -i place normalized spectra (txt, ispec format) in this folder, code xx will take spectra from here as input

## Chapter 2

# How to use the code

This chapter includes a list of all the requirements needed to run the code, as well as an individual description of each of the routines, how it works, and what their outputs are. Each python routine is made to run individually, with the user needing to finish each step before initiating the following one. The routines are described in the order they are meant to be used, but an advanced user can run only their selected parts of the full code.

To run a routine, the user can simply load it in an IDLE environment, such as spyder or Anaconda, edit the variable names necessary, and then run it in an IPython console.

### 2.1 Requirements

The main requirements to run the full code are:

- An installation of Python 2.7 - see Van Rossum and Drake Jr ((1995)), preferably with an easy to edit IDLE environment, such as Spyder or Anaconda.
- An installation of iSpec\_v2020.10.01 - see Blanco-Cuaresma et al. ((2014)) and Blanco-Cuaresma ((2019)). One is included with this distribution of the code. If using your own iSpec installation, replace the routines `synth.py`, `lines.py` and `atmospheres.py` with the custom edited ones included with this distribution.
- *Turbospectrum* - from Alvarez and Plez ((1998)) and Plez ((2012)) and MARCS.GES - see Gustafsson et al. ((2008)) stellar models with logg values up to 5.5. These files are included in the iSpec distribution and in this folder, but, if you are using a custom iSpec distribution, please make sure they are available.

Additionally, please make sure the following python modules are installed:

- os - <https://docs.python.org/2/library/os.html>
- sys - <https://docs.python.org/2/library/sys.html>
- logging - <https://docs.python.org/2.7/library/logging.html>

```

7  from math import *
8  import numpy as np
9
10 from astropy.io import fits
11
12 hdulist2 = fits.open('stars_not_found.fits') #file name containing stars for download
13 #must be in ASPCAP table format - table with all stars available at https://www.sdss.org/dr14/irspec/spectro\_data/
14
15 tldata2 = hdulist2[1].data
16 x2 = tldata2['APOGEE_ID']
17 #y2 = tldata2['LOCATION_ID']
18 z2 = tldata2['FIELD']
19 t2 = tldata2['TELESCOPE']
20 stri = '# ' + https://data.sdss.org/sas/dr16/apogee/spectro/aspcap/r12/l33/apo25m/ #DR14, must be changed if wanting to download spectra from other DR
21 ns = 100
22
23 for j, name in enumerate(x2):
24     stri += 'https://data.sdss.org/sas/dr16/apogee/spectro/aspcap/r12/l33/'
25     # name = name[10:]
26     # loc = y2[j]
27     fie = z2[j]
28     tel = t2[j]
29     stri += tel + '/' + fie + '/aspcapStar-r12-' + str(name) + '.fits'
30     stri += '\n'
31
32 f = open('down_list_stars_not_found.txt', 'w') #saving to file
33 f.write(stri)
34 f.close()
35
36 #after running this code and creating the file, the spectra can be downloaded by opening a command line and executing the code:
37 #wget -i down_list_example.txt
38 #The spectra will appear in the folder where the code is executed

```

Figure 2.1: Code for the routine build\_download\_list\_dr16.py

- numpy - <https://numpy.org/>
- multiprocessing - <https://docs.python.org/2/library/multiprocessing.html>
- matplotlib - <https://matplotlib.org/>
- astropy - <https://www.astropy.org/>
- scipy - <https://www.scipy.org/>
- shutil - <https://docs.python.org/2.7/library/shutil.html>

## 2.2 build\_download\_list.py

This first small routine is designed to create a download list for the stars the user wants to characterize. It takes as an input a list of stars in the format of ASPCAP's tables with the full results (download available at [https://www.sdss.org/dr16/irspec/spectro\\_data/](https://www.sdss.org/dr16/irspec/spectro_data/)) and gives as an output a list of the stars online locations. The default name for the list is down\_list\_example.py. The full code is shown here in Fig.2.1.

This version produces the locations of the spectra from APOGEE DR14, but it can be upgraded to the another Data Release by changing the default link. To do so, simply edit the "stri" variable to contain the new locations required.

After creating the download list, the files can be obtained by opening a command line and running the command:

```
wget -i down_list_example.txt
```

The command will download all spectra available to the directory the user is currently in. It is recommended that these files are saved in the folder "/fits\_files/", as the following routines will call the spectra from that directory.

For downloading spectra from APOGEE DR16, the routine build\_download\_list\_dr16.py is available as well. The format is similar to this one, but the paths are specific to that Data Release.

```

16 wvl0=4.179 #constants required for the wavelength conversion
17 wldelta=6*10**-6
18
19 loc = 'fits_files/' #source for spectra, make sure this folder contains ONLY spectra, delete any extra files in the folder
20
21 stars = os.listdir(loc)
22
23 dest = 'raw_spectra/' #destination for the txt format spectra
24 src = ''
25
26 convert = True #if true, the code will convert the vacuum wavelength of APOGEE spectra into air wavelength
27 #this is necessary if the user wants to run the next steps of the normalization + synthetic spectra matching
28
29 #This simple routine converts spectra from fits files (as they are downloaded from ASPCAP) into a file readable by iSpec
30
31 #####
32 #Important: if any downloaded spectra has a name with a format different than: spec_apStar-r8-2M00182256+4401222 (+ _ind or _glo).fits.txt,
33 #change it into that format by including the 2Mass ID of the star in the relevant place
34 #Otherwise the following routines may FAIL
35 #####
36
37 #reading the spectra
38 def read_spec(filename):
39     sp = fits.open(filename)
40     wavelength=np.arange(8575, dtype='float')
41     wavelength = []
42     for w in wavelength:
43         wavelength.append(10**(wvl0+wldelta*w))
44     wavelength = wavelength * u.AA
45     flux = sp[1].data
46     error = sp[2].data
47     return wavelength, flux,error
48
49 #finding the 2Mass id in the spectra's name
50 def namefinder(aspicap):
51     ap = False
52     apogee = ''
53     for c in aspcap:
54         if c == '2' and ap == False:
55             apogee += c
56             ap = True
57         elif ap == True:
58             if c == '.':
59                 ap = False
60             else:
61                 apogee += c
62     return apogee
63
64 #####
65 #air-vacuum conversion
66
67 def v2a(wave):
68     wave = wave/1000 #converting units into micron
69     a = 0
70     b1 = 5.792105*10**-2
71     b2 = 1.67917*10**-3
72     c1 = 738.9133

```

Figure 2.2: Code for the routine fits\_to\_txt.py

## 2.3 fits\_to\_txt.py

This second routine is used to convert the spectra downloaded into a format that can be useable by iSpec. By default, spectra are taken from the folder “/fits\_files/” and saved to “/raw\_spectra/”, but these destinations can be changed by renaming the variables “loc” and “dest”, respectively.

The conversion of most downloaded APOGEE spectra will result in two different spectra being created, one with the suffix “ind” and one with “glo”. These represent the individual and global weighting for the errors being considered in the spectra (more information at <https://www.sdss.org/dr16/irspec/spectra/>).

An important point to note here regarding some stars with weird APOGEE names, is that the names of the output files should follow the format

spec\_apStar-r8-2M00182256+4401222 (+ \_ind or \_glo).fits.txt

. Otherwise, the following routines may fail to retrieve the proper 2mass ID from the name of the files.

## 2.4 Synthesize\_SpectralGrid.py

This auxiliary routine is used to create synthetic spectra with a given parameter combination. It is based on the “synthesize\_spectrum()” function available on iSpec’s example.py file, and, as is, can create a grid of synthetic spectra based on the parameter combinations found on isochrones.dat.





```

21 #####
22 #input options
23
24 inputs = True
25
26 if inputs == True:
27
28     loc = '#location for the normalized spectra and output, can be customized, using own folders by default'
29
30     filename = 'Normalized_Mdwarfs_data.fits' #data for normalized M dwarfs will be stored here
31
32     #txt_folder = 'raw_txt/'
33     txt_folder = 'raw_spectra/' #location of raw spectra to normalize
34
35     #stars = os.listdir(loc + 'norm_stars_data/')
36     stars = os.listdir(loc + txt_folder)
37
38     norm = "Template" #renormalize the spectra?
39

```

Figure 2.4: Code for the routine `normalized_Mdwarfs_grid.py` - input section

```

316 #####
317
318 if __name__ == '__main__':
319
320     print('Starting by opening Aspcap')
321     #Aspcap = open_aspcap(True)
322     print('Creating fits')
323
324     MasterTable = create_fits()
325
326     stars.sort()
327
328     p = Pool(10) #this step parallelizes the process, set your number of parallel threads for the code here
329     table = p.map(Normalizing_Stars, stars)
330

```

Figure 2.5: Code for the routine `normalized_Mdwarfs_grid.py` - main section

The code can be parallelized for increased efficiency and speed. In order to do so, simply change the number in line 351

```
p = Pool(10)
```

to the number of parallel threads available for your computation (see Figure 2.5).

## 2.6 ChiSqCalc.py

The following two routines could have been implemented as a single one, but are separated for ease of customization. They are dedicated to the identification of the best templates to normalize each of the stars being characterized.

ChiSqCalc.py, the first routine, will calculate the  $\chi^2$  differences between each spectrum in folder “/normalized\_spectra/” and the template used to normalize that spectrum. The routine will provide as an output a fits files summarizing both the  $\chi^2$  measured in a selected linemask and the  $\chi^2$  obtained by comparing the full available wavelength of the spectrum. This linemask is the one used developed specifically for M dwarfs, but can be customized by creating a new one and changing the variable “linemaskname” - see Figure 2.6. The output full table is saved to “Mdwarf\_ChiSq\_Table.fits” by default.

Due to the complexity and number of functions, you may want to test them before running the code for the stellar sample. In that case, simply set the

```
if __name__ == '__main__'
```

(line 164, see Figure 2.7) condition to 'different than'.

```

23 #####
24 #INPUT
25
26 loc = ''
27
28 filename = 'Mdwarf_ChiSq_Table.fits' #output table's file name
29
30 linemaskname = 'linemask.txt'
31
32 folder = 'normalized_spectra/' #source for normalized spectra
33
34 stars = os.listdir(loc + folder)
35
36 #####

```

Figure 2.6: Code for the routine ChiSqCalc.py - input section

```

160 #####
161 BigTable = create_fits()
162
163 if __name__ == '__main__': #set this to false to test the functions themselves, without running the full code
164
165     for i,star in enumerate(stars): #running across all normalized spectra
166
167         print( star, i)
168
169         starname, tef, mh, lg = extract_Params(star) #getting the template parameters from the name of the normalized spectra
170         template = open_template(tef, mh, lg) #finding the template used to normalize the spectra
171         spectra = ispec.read_spectrum(loc + folder + star) #opening the template
172
173         linemasks = create_segments_around_linemasks(spectra) #applying the linemask to the spectra
174
175         template = resample_spectrum(template) #resampling both template and normalized spectra for comparison
176         spectra = resample_spectrum(spectra)
177
178         waveobs = spectra['waveobs']
179         comparing_mask = create_comparing_mask(waveobs, linemasks) #masking the compared regions according to linemask
180         mask = np.where(comparing_mask==1.0)
181
182         ChiSqMask = ChiSqCalc(spectra,template,mask) #calculating Chi Sq in linemask regions
183         ChiSqFull = ChiSqCalc(spectra,template) #calculating Chi Sq in every wavelength region
184
185         line = [starname, tef, mh, lg, ChiSqMask, ChiSqFull] #creating a line summarizing the results
186         BigTable.add_row(line) #adding it to table
187
188 BigTable.write(loc + filename,overwrite=True) #writing table to file
189
190

```

Figure 2.7: Code for the routine ChiSqCalc.py - main section

```

20 #This small code finds the best templates for each star, according to the ChiSq values calculated by ChiSqCalc.py
21 #it can output not only the best template for each star, but also every template within a specified margin of ChiSq value
22
23 loc = ''
24
25 Output_filename = 'Mdwarf_BestTemplates.fits' #output table
26 Input_filename = 'Mdwarf_ChiSq_Table.fits' # Input table
27
28 margin = 1.10 #margin, set to 1.0 to include only best template per star
29 #margin will be multiplied by the lowest ChiSq to find all templates within those values
30
31 def open_aspcap():
32     hdulist = fits.open(Input_filename) #input table
33     tbdata = hdulist[1].data
34     return tbdata
35
36 def create_fits():
37     ##### create table
38     columns = ('APOGEE_ID','T_Norm','MH_Norm','logg_Norm','ChiSqMask','ChiSqFull')
39
40     dtypes = ('S20','f8','f8','f8','f8','f8')
41
42     MasterTable = Table(names = columns, dtype = dtypes)
43
44     return MasterTable
45
46 BigTable = open_aspcap()
47 NewTable = create_fits()
48
49 BigTable.sort()
50
51 CharStars = []
52 BestChiSq = []
53 BestFits = []
54
55 for line in BigTable: #finding best combination per star
56     if line['APOGEE_ID'] not in CharStars:
57         BestFits.append(line)
58         CharStars.append(line['APOGEE_ID'])
59         BestChiSq.append(line['ChiSqMask'])
60     elif line['ChiSqMask'] < BestChiSq[-1]:
61         BestFits[-1] = line
62         CharStars[-1] = line['APOGEE_ID']
63         BestChiSq[-1] = line['ChiSqMask']
64
65 for line in BigTable: #adding other combinations within the margin used
66     star = line['APOGEE_ID']
67     for k,comp_star in enumerate(CharStars):
68         if comp_star == star:
69             if line['ChiSqMask'] < BestChiSq[k]*margin and line['ChiSqMask'] != BestChiSq[k]:
70                 BestFits.append(line)
71
72 for line in BestFits:
73     NewTable.add_row(line)
74
75 NewTable.write(loc + Output_filename,overwrite=True) #writing results to file

```

Figure 2.8: Code for the routine BestTemplate.py

## 2.7 BestTemplateCalc.py

This simple routine works as the second half of the previous one, as it simply scans through the full table “Mdwarf\_ChiSq\_Table.fits” generated by ChiSqCalc.py and provides as the output a smaller table “Mdwarf\_BestTemplates.fits” containing only the best templates found for each star. It does not look into the available spectra, but rather only filters the input table and selects the best templates for each star.

This routine can be used to find not only the template with the lowest  $\chi^2$  for each star, but every other template within a given error margin as well. This margin can be customized by changing the variable “margin” in line 28 - see Figure 2.8. The output will be saved to the fits table “Mdwarf\_BestTemplates.fits”, but that can be customized as well.

## 2.8 ispec\_ParameterFinder.py

This final, more complex routine, is designed to actually determine the stellar parameters for its selected stars. It is based on the function “determine\_astrophysical\_parameters\_using\_synth\_” available in the file example.py in the original iSpec distribution.

```

21 #####
22 #input options
23
24 inputs = True
25
26 if inputs == True:
27     free_params = ['teff', "logg", "MH", 'R', 'vmic', 'vsini'] #list of free parameters for the code
28
29     L = 'fusion Mdwarf_New' #Linelist to use, can't actually be changed here due to the way Turbospectrum works
30     Code = 'Turbospectrum' #code to use
31     #Do not change these lines!
32
33     Using_Normalized_stars = True
34
35     loggRanger = 0.3 #specifications on the allowed range for each parameter.
36     #Subtract 0.1 from logg and [M/H], 100K from Teff due to boundary conditions inherent to iSpec
37     #Actual limits would therefore be from 4.5 to 4.9 when using loggRanger = 0.3 and initial value of 4.7
38     teffRanger = 450
39     mhRanger = 0.45
40
41     #output filename
42     filename0 = '7stars_LoggRange_' + str(loggRanger-0.1) + '_TeffRange_' + str(teffRanger-100) + '_MHRange_' + str(mhRanger-0.1)
43
44     #Input table name
45     Input_Table = 'Mdwarf_BestTemplates.fits'
46
47     loc = ''
48     tmp_dir = None
49     if Using_Normalized_stars == True:
50
51         prefix = ''
52
53         folder = prefix + 'normalized_spectra/' #default location for input spectra
54
55         save_path = loc + prefix + 'default_output/' #default location for output best fit spectra
56
57     norm = False #Set to true if you want to renormalize the spectrum, not recommended
58
59     linemasklist = ['NewList_3200'] #linemask to use, can select multiple and code iterates over all of them
60
61     codes = ['turbospectrum']
62
63 #####

```

Figure 2.9: Code for the routine ispec\_ParameterFinder.py - input section

The code will run, by default, on every star + template combination listed in the input table `Mdwarf_BestTemplates.fits` - created by the previous routine `BestTemplateCalc.py`. In order for the code to run properly, each of the normalized spectra must be available at the folder “/normalized\_spectra/”. The code will output a best fit synthetic spectrum for each in the fits file, saving them to “/default\_output/”, as well as a master table summarizing the templates used and parameters derived for each star in the list, together with the  $\chi^2$  values measured for each combination.

Many things in this routine can be customized, such as the parameters left as free as iSpec runs, the radiative transfer code, the limits in the freedom of the three main parameters ( $T_{\text{eff}}$ ,  $\log g$  and  $[M/H]$ ) - see Figure 2.9. The linelist used is actually defined inside “/ispec/spectrum.py”, as due to the complicated way the interaction between Turbospectrum and iSpec is set up, it could not be changed in this location, and the variable “L” simply denotes the name of the linelist to present in the output table.

Another important thing to note about the boundary conditions for  $T_{\text{eff}}$ ,  $\log g$  and  $[M/H]$  is that iSpec will include an additional safety margin inside its code, so the margins written here are 0.1 dex above the desired boundaries for  $\log g$  and  $[M/H]$ , and 100 K for  $T_{\text{eff}}$ .

The output spectra is, by default, saved to “/default\_output/”, but this destination can be customized as well.

The body of the routine has many important parts, including a vacuum to air wavelength conversion, a function to generate the linemask segments, functions designed to create the structure of the final output table, and functions designed to find the names and parameters of the studied spectra from their file names. The user is advised not to edit any of these functions unless they know exactly what they are doing.

```

380 #####
381 # This code creates the output table and oversees the best fit routines
382
383 def parameters(star):
384     starname,t,m,l = extract_Params(star) #getting template parameters from spectrum name
385     star_spectrum = ispec.read_spectrum(folder + star) #opening normalized spectra
386
387     star_params = [starname,0,0,0,0,0, 'ASPCAP']
388
389     #Resample spectrum
390     wavelengths = np.arange(np.min(star_spectrum['waveobs']), np.max(star_spectrum['waveobs']), 0.001)
391     star_spectrum = ispec.resample_spectrum(star_spectrum, wavelengths, method='linear', zero_edges=True)
392
393     #-----
394
395     line = star_params
396     for i in range(25): #creating the new line to add to big results table
397         line.append(0)
398     line[6] = estimate_snr_from_flux(star_spectrum,5)
399     line[11] = l
400     line[9] = str(free_params)
401
402     run = True #set to false if you just want to test your setup, set to True to actually calculate Best fits
403     print('Will work on star:')
404     print(starname)
405
406
407     line[7] = Code
408     print('_____')
409     print('Using code ', Code)
410     print('Using star ', starname)
411     print('_____')
412
413     for spt in linemasklist: #iterating over linemasks
414         line[8] = spt
415         marg = 1.0
416         line_regions = create_segments_around_linemasks(star_spectrum, spectral_type = spt)#,min_dpt,max_dpt, resolution)
417
418         segments = ispec.create_segments_around_lines(line_regions, margin=marg)
419         segments.sort()
420
421         line[10] = t
422         line[12] = m
423         line[13] = L
424         print('-----')
425         print('Starting new run')
426         print('Spectral type of linemask used - ', spt)
427         print('Temp - ', t)
428         print('logg - ', l)
429         print('Metals - ', m)
430         print('Linelist - ', L)
431         #print segments
432         #print line_regions
433         if run == True:

```

Figure 2.10: Code for the routine ispec\_ParameterFinder.py - control section

The main function that controls the inputs and outputs and aggregates the actual results from the synthetic spectra is “parameters(star)” - see Figure 2.10. This function will operate the “determine\_astrophysical\_parameters\_using\_synth\_spectra()” function with the desired inputs and combine them into the output table.

The main code is included at the end of the routine and it has parallelization utilities built within. To create multiple threads and run the synthesis of multiple stars at the same time, the user simply has to edit the number at line 481 -

```
p = Pool(4)
```

to their desired number of threads.

```

463 #####
464
465 if __name__ == '__main__': #This is the part that actually runs everything, separated to allow for parallelization of the whole process
466
467     print ('Starting by opening Aspcap')
468     #Aspcap = open_aspcap(True)
469     print ('Creating fits')
470     filename = filename0 + '.fits'
471
472     MasterTable = create_fits() #creating big table to house results
473
474     StarsList = open_Stars() #opening table with spectra that will be characterized
475
476     stars = []
477     for star in StarsList:
478         stars.append('Normalized ' + str(star['APOGEE_ID']) + ' ' + str(int(star['T_Norm'])) + '.0 '
479                     + str(star['MH_Norm']) + ' ' + str(star['logg_Norm']) + '.fits')
480     #this step creates names of spectra that will be characterized as per indications from StarsList about their templates
481     #Do not change names of spectra so that this step will run smoothly
482     stars.sort()
483
484     p = Pool(4) #parallelizing the process, set your number of parallel threads for the code here
485     #Set to 1 to run everything in series
486
487     table = p.map(parameters, stars)
488     for line in table:
489         MasterTable.add_row(line)
490
491     MasterTable.write(loc + filename, overwrite=True) #writing big table to file

```

Figure 2.11: Code for the routine ispec.ParameterFinder.py - main section

## Chapter 3

# Important warnings

Many of the routines will default their outputs to “/default\_output/”, and the files must then be moved so the next routines can be run. This is done so the user can run each routine individually while learning how to operate the code, but it can easily be customized by changing the output locations in the input sections of each code.

In addition, the names of the output spectra by a given routine are very important to the following routines, as they contain information about the stars being analyzed, the templates used for the normalization, or the output parameters given by the code. The names of the output files of intermediate routines such as `normalized_Mdwarfs_grid.py` should not be changed unless the user knows exactly what they are doing.

Finally, the routines included are a complement to the papers Sarmiento et al. ((2020)) and Sarmiento et al. ((2021)), and a complete read of those papers is suggested for a full understanding of them.

# Bibliography

- R. Alvarez and B. Plez. Near-infrared narrow-band photometry of M-giant and Mira stars: models meet observations. 330:1109–1119, Feb. 1998.
- S. Blanco-Cuaresma. Modern stellar spectroscopy caveats. *mnras*, 486(2):2075–2101, Jun 2019. doi: 10.1093/mnras/stz549.
- S. Blanco-Cuaresma, C. Soubiran, U. Heiter, and P. Jofré. Determining stellar atmospheric parameters and chemical abundances of fgk stars with ispec. *Astronomy & Astrophysics*, 569:A111, 2014.
- B. Gustafsson, B. Edvardsson, K. Eriksson, U. G. Jørgensen, Å. Nordlund, and B. Plez. A grid of MARCS model atmospheres for late-type stars. I. Methods and general properties. *aap*, 486:951–970, Aug. 2008. doi: 10.1051/0004-6361:200809724.
- B. Plez. Turbospectrum: Code for spectral synthesis. *Astrophysics Source Code Library*, 1:05004, 2012.
- P. Sarmiento, E. D. Mena, B. Rojas-Ayala, and S. Blanco-Cuaresma. Derivation of parameters for 3748 fgk stars using h-band spectra from apogee data release 14. *arXiv preprint arXiv:2001.01995*, 2020.
- P. Sarmiento, E. D. Mena, B. Rojas-Ayala, and S. Blanco-Cuaresma. Determination of spectroscopic parameters for 308 m dwarf stars from their apogee data release 14 h-band spectra. *arXiv:2103.04848*, 2021.
- G. Van Rossum and F. L. Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.