In [ ]:
```
import os
import glob
import numpy as np
import matplotlib.pyplot as plt
#import reduction as rdc
import pandas as pd

from astropy.io import fits
import sys
from astropy.wcs import WCS
from astropy.io import fits as pyfits

import timeit
from timeit import default_timer as timer

import pickle
import uncertainties
from uncertainties import ufloat
```

In [ ]:
```
#A macro do estilo e e fontes é:

def estilo():
    from matplotlib import rcParams
    import matplotlib.font_manager as font_manager
    font_dirs = '/home/rafael/anaconda3/lib/python3.8/site-packages/matpl
    #font_dirs = '/Untitled/anaconda3/lib/python3.6/site-packages/matplot
    #font_dirs = '/usr/share/matplotlib/mpl-data/fonts/ttf/'
    font_files = font_manager.findSystemFonts(fontpaths=font_dirs)
    font_list = font_manager.createFontList(font_files)
    #font_list = FontManager.addfont(font_files)
    font_manager.fontManager.ttflist.extend(font_list)
    rcParams['font.family'] = 'Courier Prime'
    rcParams['mathtext.fontset'] = 'dejavuserif'
    rcParams['axes.linewidth'] = 1.2
estilo()

def sm_ted():
    plt.minorticks_on()
    plt.tick_params(axis='both',which='minor', direction = "in",top = Tru
    plt.tick_params(axis='both',which='major', direction = "in",top = Tru
    plt.tick_params(axis='both',which='minor', direction = "in",bottom =
    plt.tick_params(axis='both',which='major', direction = "in",bottom =
```

In [ ]:
```
# iSpec

import logging
import multiprocessing
from multiprocessing import Pool

#ispec_dir = '/home/rafael/Programs/iSpec/iSpec_v20201001'
ispec_dir = os.path.dirname(os.path.realpath('/home/rafael/programs/iSpec
#ispec_dir = os.path.dirname(os.path.realpath('/home/rafael/Programs/iSpe
sys.path.insert(0, os.path.abspath(ispec_dir))
import ispec

def read_write_spectrum(fname):
    #--- Reading spectra -------------------------------------------------
    logging.info("Reading spectra")
```

```
    star_spectrum = ispec.read_spectrum(fname)
    ##--- Save spectrum -----------------------------------------------
    logging.info("Saving spectrum...")
    #ispec.write_spectrum(star_spectrum, "example_sun.fits")
    return star_spectrum

def plot(fname):
    star_spectrum = ispec.read_spectrum(fname)
    mu_cas_spectrum = ispec.read_spectrum(ispec_dir + "/input/spectra/exa
    #--- Plotting (requires graphical interface) -----------------------
    logging.info("Plotting...")
    ispec.plot_spectra([star_spectrum, mu_cas_spectrum])
    ispec.show_histogram(star_spectrum['flux'])

def determine_radial_velocity_with_mask(fname):
    mu_cas_spectrum = ispec.read_spectrum(fname)
    #--- Radial Velocity determination with linelist mask --------------
    logging.info("Radial velocity determination with linelist mask...")
    # - Read atomic data
    mask_file = "/home/rafael/programs/iSpec/iSpec_v20201001/input/lineli
    #mask_file = "mask.lst"
    #mask_file = ispec_dir + "input/linelists/CCF/Atlas.Arcturus.372_926n
    #mask_file = ispec_dir + "input/linelists/CCF/Atlas.Sun.372_926nm/mas
    #mask_file = ispec_dir + "input/linelists/CCF/HARPS_SOPHIE.A0.350_109
    #mask_file = ispec_dir + "input/linelists/CCF/HARPS_SOPHIE.F0.360_698
    #mask_file = ispec_dir + "input/linelists/CCF/HARPS_SOPHIE.G2.375_679
    #mask_file = ispec_dir + "input/linelists/CCF/HARPS_SOPHIE.K0.378_679
    #mask_file = ispec_dir + "input/linelists/CCF/HARPS_SOPHIE.K5.378_680
    #mask_file = ispec_dir + "input/linelists/CCF/HARPS_SOPHIE.M5.400_687
    #mask_file = ispec_dir + "input/linelists/CCF/Synthetic.Sun.350_1100n
    #mask_file = ispec_dir + "input/linelists/CCF/VALD.Sun.300_1100nm/mas
    ccf_mask = ispec.read_cross_correlation_mask(mask_file)

    models, ccf = ispec.cross_correlate_with_mask(mu_cas_spectrum, ccf_ma
                            lower_velocity_limit=-200, upper_velocity_lim
                            velocity_step=1.0, mask_depth=0.01, \
                            fourier=False)

    # Number of models represent the number of components
    components = len(models)
    # First component:
    rv = np.round(models[0].mu(), 2) # km/s
    rv_err = np.round(models[0].emu(), 2) # km/s

    return rv,rv_err


def correct_radial_velocity(fname,rv):
    mu_cas_spectrum = ispec.read_spectrum(fname)
    #--- Radial Velocity correction ------------------------------------
    logging.info("Radial velocity correction...")
    #rv = -96.40 # km/s
    mu_cas_spectrum = ispec.correct_velocity(mu_cas_spectrum, rv)

    return mu_cas_spectrum
```

In [ ]:

```
def rotation(S,bv,tal):
    #tal = 12.8 # TGEC
    logCcf = (1.13*(bv**3) - 3.91*(bv**2) + 2.84*(bv) - 0.47)
    x = 0.63 - bv
```

```
    deltalogC = (0.135*x) - (0.814*(x**2)) + (6.03*(x**3))
    logCcf_ = (logCcf + deltalogC)
    Ccf_ = 10**logCcf_
    Rhk = 1.340*(10**(-4))*Ccf_*S
    logRphot = (-4.02 - 1.40*(bv))
    Rphot = 10**logRphot
    Rhk_ = Rhk - Rphot
    y = np.log10(10**(5)*Rhk_)
    logptal = 0.324 - (0.400*y) - (0.283*(y**2)) - (0.1325*(y**3))
    Prot = tal * 10**(logptal)
    rosby = Prot / tal
    return Prot,rosby,np.log10(Rhk_)

def Ro(Rhk_):
    """
    Compute the Rossby number from the observational activity index S-ind
    This method was derived from Noyes et al. 1984
    """
    y = np.log10(10**(5)*Rhk_)
    Ro = logptal = 0.324 - (0.400*y) - (0.283*(y**2)) - (0.1325*(y**3))
    return Ro

def Rhk_(S,bv):
    """
    Compute the chromospheric activity index R'hk from the observational
    This method was derived from Noyes et al. 1984
    """
    #bv=0.65
    logCcf = (1.13*(bv**3) - 3.91*(bv**2) + 2.84*(bv) - 0.47)
    x = 0.63 - bv
    deltalogC = (0.135*x) - (0.814*(x**2)) + (6.03*(x**3))
    logCcf_ = (logCcf + deltalogC)
    Ccf_ = 10**logCcf_
    Rhk = 1.340*(10**(-4))*Ccf_*S
    logRphot = (-4.02 - 1.40*(bv))
    Rphot = 10**logRphot
    Rhk_ = Rhk - Rphot
    return Rhk_

def Rhk(S):
    bv=0.65
    logCcf = (1.13*(bv**3) - 3.91*(bv**2) + 2.84*(bv) - 0.47)
    x = 0.63 - bv
    deltalogC = (0.135*x) - (0.814*(x**2)) + (6.03*(x**3))
    logCcf_ = (logCcf + deltalogC)
    Ccf_ = 10**logCcf_
    Rhk = 1.340*(10**(-4))*Ccf_*S
    return Rhk

def Rphot(S):
    """
    Compute the chromospheric activity index R'hk from the observational
    This method was derived from Noyes et al. 1984
    """
    bv=0.65
    logCcf = (1.13*(bv**3) - 3.91*(bv**2) + 2.84*(bv) - 0.47)
    x = 0.63 - bv
    deltalogC = (0.135*x) - (0.814*(x**2)) + (6.03*(x**3))
    logCcf_ = (logCcf + deltalogC)
    Ccf_ = 10**logCcf_
```

```
    Rhk = 1.340*(10**(-4))*Ccf_*S
    logRphot = (-4.02 - 1.40*(bv))
    Rphot = 10**logRphot
    return Rphot


def tau (bv):
    """
    Compute the convective turner overtime from the observational the col
    This method was derived from Noyes et al. 1984
    """
    x = 1 - bv
    if x>0:
        k = 1.362 - 0.166*x + 0.025*(x**2) -5.323*(x**3)
    else:
        k = 1.362 - 0.14*x
    Tau = 10**k
    return Tau

def S_index(wav,flux):
    new = np.array((wav, flux)).T

    lamh = 3933.663
    lamk = 3968.469
    lamv = 3900.000
    lamr = 4000.000
    deltalamca = 2.18/2
    deltalamcont = 20/2


    num_points = len(new[:,0])
    deltalam = 0

    fluxh, fluxk, fluxr, fluxv = 0,0,0,0
    weighth, weightk, weightr, weightv = 0,0,0,0

    for i in range(0, num_points):
        if new[i,0] > lamh-deltalamca and new[i,0] < lamh+deltalamca:
            fluxh += new[i,1]*(deltalamca - np.fabs(new[i,0] - lamh))/(de
            weighth += (deltalamca - np.fabs(new[i,0] - lamh))/(deltalamc
        elif new[i,0] > lamk-deltalamca and new[i,0] < lamk+deltalamca:
            fluxk += new[i,1]*(deltalamca - np.fabs(new[i,0] - lamk))/(de
            weightk += (deltalamca - np.fabs(new[i,0] - lamk))/(deltalamc
        elif new[i,0] > lamr-deltalamcont and new[i,0] < lamr+deltalamcon
            fluxr += new[i,1]
            weightr += 1
        elif new[i,0] > lamv-deltalamcont and new[i,0] < lamv+deltalamcon
            fluxv += new[i,1]
            weightv += 1

    fluxv /= weightv
    fluxr /= weightr
    fluxh /= weighth
    fluxk /= weightk

    result = (fluxh + fluxk)/(fluxv + fluxr)
    #S_sophie = np.round(result,4)
    return result
```

S-index dos espectros SOPHIE

In [ ]:
```
# plot SOPHIE spectra with RV corrected

pasta = os.getcwd()
array = ["Name;S_Sophie;Smw;err_Smw;BJD;RV;RV_err;Instrument"]

start = timer()
count = 0

print("Name;S_Sophie;Smw;err_Smw;BJD;RV;RV_err;Instrument")
for root, dirs, files in os.walk(pasta):
    for file in files:
        if file.endswith(".fits"):
            fname = os.path.join(root, file)
            sp = fits.open(fname)
            header = sp[0].header
            #flux = sp[0].data
            #wcs = WCS(header)
            index = np.arange(header['NAXIS1'])
            date = header['Date']
            BJD = header['OHP DRS BJD']
            star = str(header['OBJNAME'])
            #instrumento = str(header['INSTRUME'])
            instrumento = 'SOPHIE'

            #wavelength = wcs.wcs_pix2world(index[:,np.newaxis], 0)
            #wavelength = wavelength.flatten()

            rv,rv_err = determine_radial_velocity_with_mask(fname)
            print(f'Estrela: {star}, RV = {rv} Km/s, err_RV = {rv_err} Km
            c = 299792458
            spectrum = correct_radial_velocity(fname,rv)
            #spectrum = read_write_spectrum(path)

            deltalam = 0
            lamh = 3933.663
            lamk = 3968.469
            lamv = 3900.000
            lamr = 4000.000
            deltalamca = 2.18/2
            deltalamcont = 20/2



            wavelength = 10*spectrum['waveobs']
            FLUX = spectrum['flux']

            S = np.round(S_index(wavelength,FLUX),4)
            a1 = ufloat(0.71, 0.03)
            a2 = ufloat(0.72,0.02)
            b1 = ufloat(0.039, 0.006)
            b2 = ufloat(0.021,0.006)
            a = [a1,a2]
            b = [b1,b2]
            a = np.mean(a)
            b = np.mean(b)
```

```
            S_ = (1/a)*S - (b/a)
            Smw = np.round(S_.nominal_value, 4)
            err_Smw = np.round(S_.s,4)


            array = np.append(array,[('{};{};{};{};{};{};{};{}'.format(st
            print ('{};{};{};{};{};{};{};{}'.format(star,S,Smw,err_Smw,BJ


            fig=plt.figure(figsize=(9,6))
            plt.tick_params(axis='both', which='major', labelsize=22)

            plt.axvspan(lamh-deltalamca, lamh+deltalamca, facecolor='g',
            plt.axvspan(lamk-deltalamca, lamk+deltalamca, facecolor='g',
            plt.axvspan(lamr-deltalamcont, lamr+deltalamcont, facecolor='
            plt.axvspan(lamv-deltalamcont, lamv+deltalamcont, facecolor='

            plt.title(star+" BJD:"+str(BJD),  {'color': 'k','fontsize': 2
            plt.xlabel(r'Wavelength ($\AA$)',  {'color': 'k','fontsize':
            xp1 = 3920*(1.+deltalam)
            xp2 = 3981*(1.+deltalam)
            plt.xlim([xp1, xp2])
            plt.ylim([-0.5,3.0])
            plt.text(lamk-0.7, 1.05, '$K$',  {'color': 'k','fontsize': 20
            plt.text(lamh-0.7, 1.05, '$H$',  {'color': 'k','fontsize': 20

            plt.xlabel(r'Wavelength ($\AA$)', {'color': 'k','fontsize': 2
            plt.ylabel(r'$S_{MW}$', {'color': 'k','fontsize': 20})
            plt.plot(wavelength, FLUX/np.median(FLUX), "-b", lw=1)
            sm_ted()
            plt.tight_layout()
            plt.show()
            count+=1

np.savetxt('Resultados_calib_Boisse2010.csv', array, fmt="%s")

end = timer()
timed = np.round(end - start,3)
ts = np.round(timed/count,3)
print(f"Time exec: {timed} s")
print(f'Time per spectrum = {ts} s')
```

In [ ]:

```
# plot SOPHIE spectra with RV corrected

pasta = os.getcwd()
array = []

print("Name;S_Sophie;Smw;err_Smw;BJD;Data;RV;RV_err;Instrument")
for root, dirs, files in os.walk(pasta):
    for file in files:
        if file.endswith(".fits"):
            fname = os.path.join(root, file)
            sp = fits.open(fname)
            header = sp[0].header
            #flux = sp[0].data
            #wcs = WCS(header)
            index = np.arange(header['NAXIS1'])
            date = header['Date']
            BJD = header['OHP DRS BJD']
            star = str(header['OBJNAME'])
```

```
#instrumento = str(header['INSTRUME'])
instrumento = 'SOPHIE'

#wavelength = wcs.wcs_pix2world(index[:,np.newaxis], 0)
#wavelength = wavelength.flatten()

rv,rv_err = determine_radial_velocity_with_mask(fname)
print(f'Estrela: {star}, RV = {rv} Km/s, err_RV = {rv_err} Km
c = 299792458
spectrum = correct_radial_velocity(fname,rv)
#spectrum = read_write_spectrum(path)

deltalam = 0
lamh = 3933.663
lamk = 3968.469
lamv = 3900.000
lamr = 4000.000
deltalamca = 2.18/2
deltalamcont = 20/2



wavelength = 10*spectrum['waveobs']
FLUX = spectrum['flux']

S = S_index(wavelength,FLUX)
a = ufloat(1.195, 0.035)
b = ufloat(0.008, 0.007)
S_ = (a*S + b)
Smw = np.round(S_.nominal_value, 4)
err_Smw = np.round(S_.s,4)


array = np.append(array,[('{};{};{};{};{};{};{};{}'.format(st
print ('{};{};{};{};{};{};{};{}'.format(star,S,Smw,err_Smw,BJ


fig=plt.figure(figsize=(9,6))
plt.tick_params(axis='both', which='major', labelsize=22)

plt.axvspan(lamh-deltalamca, lamh+deltalamca, facecolor='g',
plt.axvspan(lamk-deltalamca, lamk+deltalamca, facecolor='g',
plt.axvspan(lamr-deltalamcont, lamr+deltalamcont, facecolor='
plt.axvspan(lamv-deltalamcont, lamv+deltalamcont, facecolor='

plt.title(star+" BJD:"+str(BJD),  {'color': 'k','fontsize': 2
plt.xlabel(r'Wavelength ($\AA$)',  {'color': 'k','fontsize':
xp1 = 3920*(1.+deltalam)
xp2 = 3981*(1.+deltalam)
plt.xlim([xp1, xp2])
plt.ylim([-0.5,3.0])
plt.text(lamk-0.3, 1.05, '$K$')
plt.text(lamh-0.3, 1.05, '$H$')

plt.xlabel(r'Wavelength ($\AA$)', {'color': 'k','fontsize': 2
plt.ylabel(r'$S_{MW}$', {'color': 'k','fontsize': 20})
plt.plot(wavelength, FLUX/np.median(FLUX), "-b", lw=1)
sm_ted()
plt.tight_layout()
plt.show()
```

```
np.savetxt('Resultados.csv', array, fmt="%s")
```

In [ ]:
```
array.T
```

In [ ]:
```
MM = pd.read_csv('Table_Lubin et al. 2010.csv', delimiter=';')
MM['B-V'] = MM['B-V'] / 1000
del MM['Delta Mv']
del MM['[Fe/H]']
MM['HD'] = MM['HD'].astype(str)
MM['HD'] = 'HD'+MM['HD']
MM['HIP'] = MM['HIP'].astype(str)
MM['HIP'] = 'HIP'+MM['HIP']

MM
```

In [ ]:
```

```

In [ ]:
```
star = 'HD138573'

bv = 0.656
velstar = -35.662

S_index(star,bv,velstar)
```