SSB_sample

October 16, 2015

1 Stellar spectra B. LTE Line Formation

1.1 FALC temperature stratification

This second exercise sample gives the tools for you to code the compulsory exercise B. The commands DO NOT contain everything which is reported in the statement, e.g. constants, ranges for the plots, plot titles… Please, check and follow the instructions in the IDL statement and make use of the below described parts of the code. Notice that the functions planck.pro, earth.pro, etc. are easy to convert into Python so they do not appear here (Functions are reported below in case they account for some coding-difficulties). Let me know if you run into troubles and/or find some bugs in the code.

```
In [ ]: # importing useful libraries (you may need more)
        import numpy as np                  # numerical package
        import matplotlib.pyplot as plt     # plotting package
        from matplotlib import rc
        rc('font',**{'family':'serif'})     # This is for Latex writing


        # DEFINE ALL THE CONSTANTS YOU NEED HERE (or any another place, if you prefer)


        # reading falc.dat
        (h, tau5, colm, temp, vturb, nhyd, nprot, nel, ptot, pgasptot,
                dens = np.loadtxt('/where/you/have/the/file/falc.dat',
                usecols=(0,1,2,3,4,5,6,7,8,9,10), unpack=True) )

        # plotting
        fig = plt.figure()
        plt.plot(h, temp)
        # commands for fancy plots as titles, axis-labels...
        # if you want/need to save the plot in some format, you can use (bbox and
        #pad make the figure to be tighten to the plot-box)
        fig.savefig('/where/and/name/of/figure/Myfigure.pdf', bbox_inches='tight',pad_inches=0.106)
        plt.show()
```

1.0.1 2.1 Observed solar continua

```
In [ ]: # to obtain maximums
        print 'max(Ic)= ', np.max(Icont), 'at', wav[np.where(Icont == np.max(Icont))]
```

1.0.2 2.2 continuous extinction

```
In [ ]: # exthmin function : PLEASE, SEE exthmin.pro for comments
        def exthmin(wav,temp,eldens):
            theta = 5040. / temp
```

```
elpress = eldens*k*temp
sigmabf = ( 1.99654 -1.18267E-5*wav +2.64243E-6*wav**2 -4.40524E-10*
    wav**3+3.23992E-14*wav**4 -1.39568E-18*wav**5 +2.78701E-23*wav**6 )
sigmabf *= 1e-18

sigmabf[np.where(sigmabf > 16444)] = 0

graysaha=4.158E-10*elpress*theta**2.5*10.**(0.754*theta)
kappabf=sigmabf*graysaha
kappabf=kappabf*(1.-np.exp(-h*c/(wav*1E-8*k*temp)))

lwav=alog10(wav)
f0 =  -2.2763 -1.6850*lwav +0.76661*lwav**2 -0.0533464*lwav**3
f1 =  15.2827 -9.2846*lwav +1.99381*lwav**2 -0.142631*lwav**3
f2 = ( -197.789 +190.266*lwav -67.9775*lwav^2 +10.6913*lwav**3
      -0.625151*lwav**4 )
ltheta=alog10(theta)
kappaff = 1E-26*elpress*10**(f0+f1*ltheta+f2*ltheta**2)

return kappaff + kappabf
```

1.0.3 2.3 Optical Depth

In [ ]:
```
tau = np.zeros(len(tau5), dtype=float) # initializing tau array
ext = exthmin(500nm!!, temp, e-density)
for i in range(1,len(tau)): # index zero is not accounted for, so tau[0] = 0 because we have al
    tau[i] = tau[i-1] + 0.5*(ext[i]+ext[i-1])*(h[i-1]-h[i])*1e5

plt.plot(h,tau5,'--', label = 'tau5'')
plt.plot(h,tau, label = 'tau')
plt.yscale('log')
plt.show()
```

1.0.4 2.4 Emergent intensity and height of formation

In [ ]:
```
ext, tau, integrand, confunc = np.zeros(len(tau5), dtype=float) #repeat for every parameter if
intt = 0.0  # notice that in the statement this is 'int' but in python means integer
hint = 0.0

for i in range(1,len(tau)):  # the index zero is not accounted for
    ext[i] = exthmin(wl*1e4, temp[i],nel[i])*(nhyd[i]-nprot[i])+0.664e-24*nel[i]
    tau[i] = tau[i-1] + 0.5*(ext[i]+ext[i-1])*(h[i-1]-h[i])*1e5
    integrand[i] = planck(temp[i],wl)*np.exp(-tau[i])

    intt += 0.5*(integrand[i]+integrand[i-1])*(tau[i]-tau[i-1])
    hint += h[i]*0.5*(integrand[i]+integrand[i-1])*(tau[i]-tau[i-1])
    confunc[i] = integrand[i]*ext[i]

hmean = hint / intt
```

1.0.5 2.7 Flux integration

In [ ]:
```
xgauss=[-0.7745966692,0.0000000000,0.7745966692]
wgauss=[ 0.5555555555,0.8888888888,0.5555555555]
fluxspec=np.zeros(len(wav),dtype=float)
```

```
intmu=np.zeros((3,len(wav)),dtype=float)
for imu in range(3):
  mu=0.5+xgauss[imu]/2.
  wg=wgauss[imu]/2.
  for iw in range(len(wav)):
    wl=wav[iw]
    intmu[imu,iw]=intt # THIS "INTMU" FUNCTION MAY BE CREATED WITH THE ABOVE SCRIPT (THE FOR LO
    fluxspec[iw]=fluxspec[iw]+wg*intmu[imu,iw]*mu
    fluxspec *= 2

plt.plot(wav,fluxspec) # + other plot options...
plt.plot(wav, Fcont)
```

1.0.6 Extra material - section 3.4 formulas

```
In [ ]: def parfunc_Na(temp):
            u = np.zeros(3, dtype=float)
            theta = 5040. / temp
            c0=0.30955
            c1=-0.17778
            c2=1.10594
            c3=-2.42847
            c4=1.70721
            logU1 = ( c0 + c1 * np.log10(theta) + c2 * np.log10(theta)**2 + c3 * np.log10(theta)**3
                + c4 * np.log10(theta)**4 )
            u[0]=10**logU1

            u[1]=1
            u[2]=6
            return u
```

Voigt function CAUTION! Be very careful here with the constants and parameters you put to construct this Voigt function. Remember the function for the voigt profile (scipy.wofz) used in SSA exercise. You have a "similar" example here https://www.astro.rug.nl/software/kapteyn/EXAMPLES/kmpfit_voigt.py

```
In [ ]: voigt_NaD = voigt(a_voigt, v_voigt) / dopplerwidth
```

Van der Waals broadening

```
In [ ]: def gammavdw_NaD(temp, pgas, s):
            # do not forget to look at the statement for constants and explanations
            rsq_u = rsq_NaD(s)
            rsq_l = rsq_NaD(1)
            loggvdw=6.33 + 0.4*np.log10(rsq_u - rsq_l) + np.log10(pgas) - 0.7 * np.log10(temp)
            return 10**loggvdw

        def rsq_NaD(s):
            # put constants in statement here
            E_n = np.zeros(3, dtype=float)
            E_n[1] = h*c / 5895.94e-8 * erg2eV
            E_n[2] = h*c / 5889.97e-8 * erg3eV
            Z = 1.
            Rydberg = 13.6
            l = [0., 1., 1.]
```

```
    nstar_sq = Rydberg * Z**2 / (E_ionization - E_n[s-1])
    rsq = nstar_sq / 2. / Z**2 * (5*nstar_sq + 1 - 3*l[s-1]*(l[s-1] + 1))
    return rsq

# Plot the Boltzmann and Saha distributions for checking you are in the right way
```