SSB_sample

October 18, 2016

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 maxima
        print 'max(Ic)= ',np.max(Icont),'at',wav[np.where(Icont == np.max(Icont))]
```

1.0.2 2.2 continuous extinction

```
In [ ]: def exthmin(wav,temp,eldens):
            # H-minus extinction, from Gray 1992
            # input:
            #  wav = wavelength [Angstrom] (float or float array)
            #  temp = temperature [K]
            #  eldens = electron density [electrons cm-3]
            # output:
            #  H-minus bf+ff extinction [cm^2 per neutral hydrogen atom]
            #  assuming LTE ionization H/H-min

            # physics constants in cgs (all cm)
            k=1.380658e-16   # Boltzmann constant [erg/K]
            h=6.626076e-27   # Planck constant [erg s]
            c=2.997929e10    # velocity of light [cm/s]

            # other parameters
            theta=5040./temp
            elpress=eldens*k*temp

            # evaluate H-min bound-free per H-min ion = Gray (8.11)
            # his alpha = my sigma in NGSB/AFYC (per particle without stimulated)
            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  # cm^2 per H-min ion
            if size(wav) > 1:
                sigmabf[np.where(wav > 16444)] = 0 # H-min ionization limit at lamb
            elif (size(wav) == 1):
                if wav > 16444:
                    sigmabf=0

            # convert into bound-free per neutral H atom assuming Saha = Gray p135
            # units: cm2 per neutral H atom in whatever level (whole stage)
            graysaha=4.158E-10*elpress*theta**2.5*10.**(0.754*theta) # Gray (8.12)
            kappabf=sigmabf*graysaha                          # per neutral H atom
            kappabf=kappabf*(1.-np.exp(-h*c/(wav*1E-8*k*temp))) # correct stimulate

            # check Gray's Saha-Boltzmann with AFYC (edition 1999) p168
            # logratio=-0.1761-np.log10(elpress)+np.log10(2.)+2.5*np.log10(temp)-th
            # print 'Hmin/H ratio=',1/(10.**logratio) # OK, same as Gray factor SB
```

```
    # evaluate H-min free-free including stimulated emission = Gray p136
    lwav=np.log10(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=np.log10(theta)
    kappaff = 1E-26*elpress*10**(f0+f1*ltheta+f2*ltheta**2)    # Gray (8.13)

    return kappabf+kappaff
```

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)):
            tau[i] = tau[i-1] + 0.5*(ext[i]+ext[i-1])*(h[i-1]-h[i])*1e5
        # index zero is not accounted for, so tau[0] = 0 because we have already in

        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 [ ]: # SSB 2.4 page 16 emergent intensity, contribution function and mean height
        sigma_Thomson = 6.648E-25    # Thomson cross-section [cm^2]
        wl = 0.5    # wavelength in micron, 1 micron = 1e-6 m = 1e-4 cm = 1e4 Angstr
        ext = np.zeros(len(tau5))
        tau = np.zeros(len(tau5))
        integrand = np.zeros(len(tau5))
        contfunc = np.zeros(len(tau5))
        intt = 0.0
        hint = 0.0

        for i in range(1, len(tau5)):
            ext[i] = (exthmin(wl*1e4, temp[i], nel[i])*(nhyd[i]-nprot[i])
                      + sigma_Thomson*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*1e-4)*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])
            contfunc[i] = integrand[i]*ext[i]
        # note : exthmin has wavelength in [Angstrom], planck in [cm]
        hmean = hint / intt
```

```
print ('computed continuum intensity wl =%g : %g erg s-1 cm-2 ster-1 cm-1'
       %(wl, intt))
w = np.where(wav == wl)
print ('observed continuum intensity wav=%g : %g erg s-1 cm-2 ster-1 cm-1'
       %(wav[w], Icont[w]*1e10*1e4))
```

1.0.5 2.7 Flux integration

In [ ]:
```
# SSB 2.7 page 17: flux integration
# ===== three-point Gaussian integration intensity -> flux
# abscissae + weights n=3 Abramowitz & Stegun page 916
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.     # rescale xrange [-1,+1] to [0,1]
    wg=wgauss[imu]/2.         # weights add up to 2 on [-1,+1]
    for iw in range(0,len(wav)):
        wl=wav[iw]
        ext = np.zeros(len(tau5))
        tau = np.zeros(len(tau5))
        integrand = np.zeros(len(tau5))
        intt = 0.0
        for i in range(1, len(tau5)):
            ext[i] = (exthmin(wl*1e4, temp[i], nel[i])*(nhyd[i]-nprot[i])
                     + sigma_Thomson*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*1e-4)*np.exp(-tau[i]/mu)
            intt += 0.5*(integrand[i]+integrand[i-1])*(tau[i]-tau[i-1])/mu
        intmu[imu,iw]=intt
        fluxspec[iw]=fluxspec[iw] + wg*intmu[imu,iw]*mu

fluxspec *= 2    # no np.pi, Allen 1978 has flux F, not {\cal F}

figname='ssb_2.7_fluxintegration'
f=plt.figure(figname)
plt.plot(wav,fluxspec*1e-14, label='computed from FALC')
plt.plot(wav,Fcont, label='observed (Allen 1978)')
plt.legend(loc='upper right')
plt.title('observed and computed continuum flux')
plt.ylabel(r'astrophysical flux [10$^{14}$ erg s$^{-1}$ cm$^{-2}$ ster$^{-1
plt.xlabel('wavelength [$\mu$m]')
plt.grid(True)
plt.show()
f.savefig(figname+'.pdf',bbox_inches='tight')
f.savefig(figname+'.png',bbox_inches='tight')
```

1.0.6 Extra material - section 3.4 formulas

```
In [ ]: def parfunc_Na(temp):
            # partition functions Na
            # input: temp (K)
            # output: float array(3) = partition functions U1,U2,U3
            u=np.zeros(3)
            # partition function Na I: follow Appendix D of Gray 1992
            # log(U1(T)) = c0 + c1 * log(theta) + c2 * log(theta)^2 +
            #                c3 *log(theta)^3 + c4 log(theta)^4
            # with theta=5040./T
            theta=5040./temp
            # partition function Na I : Appendix D of Gray (1992)
            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
            # partition function Na II and Na III: approximate by the
            # statistical weights of the ion ground states
            u[1]=1   # from Allen 1976
            u[2]=6   # from Allen 1976
            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 [ ]: from scipy import special

        def voigt(gamma,x):
            z = (x+1j*gamma)
            V = special.wofz(z).real
            return V

        voigt_NaD = voigt(a_voigt, v_voigt) / dopplerwidth
```

Van der Waals broadening

```
In [ ]: def gammavdw_NaD(temp, pgas, s):
            # Van der Waals broadening for Na D1 and Na D2
            # s=2 : Na D1
            # s=3 : Na D2
            # using classical recipe by Unsold
```

```
    # following recipe in SSB
    rsq_u = rsq_NaD(s)
    rsq_l = rsq_NaD(1) # lower level D1 and D2 lines is ground state s=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):
    # compute mean square radius of level s of Na D1 and Na D2 transitions
    #  -> needed for van der Waals broadening in SSB
    # s=1 : ground state, angular momentum l=0
    # s=2 : Na D1 upper level l=1
    # s=3 : Na D2 upper level l=1
    h=6.62607e-27                   # Planck constant (erg s)
    c=2.99792e10                    # light speed [cm/s]
    erg2eV=1/1.60219e-12            # erg to eV conversion
    E_ionization = 5.139            # [eV] ionization energy
    E_n=np.zeros(3)                 # energy level: E_n[0]=0 : ground state
    E_n[1]=h*c/5895.94e-8 * erg2eV # Na D1: 2.10285 eV
    E_n[2]=h*c/5889.97e-8 * erg2eV # Na D2: 2.10498 eV
    Z=1.                            # ionization stage, neutral Na: Na I
    Rydberg=13.6                    # [eV] Rydberg constant
    l=[0.,1.,1.]                    # angular quantum number
    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 that
# you are at the right track
```