AST4310 - Stellar Spectra B

Simen Nyhus Bastnes
simennb

11. November 2016

Introduction

In this exercise, we will study the formation of spectral lines in the solar atmosphere assuming Local Thermodynamical Equilibrium (LTE).

1 Stratification of the solar atmosphere

We study the radial stratification of the solar atmosphere on the basis of the standard model FALC.

1.1 FALC temperature stratification

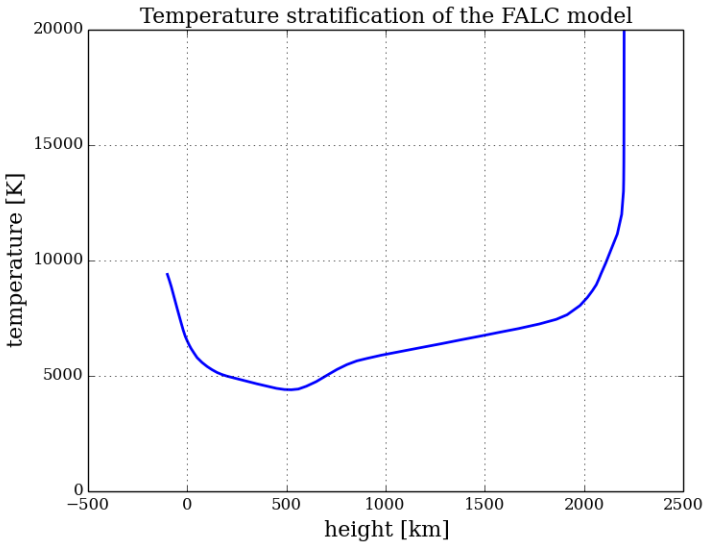
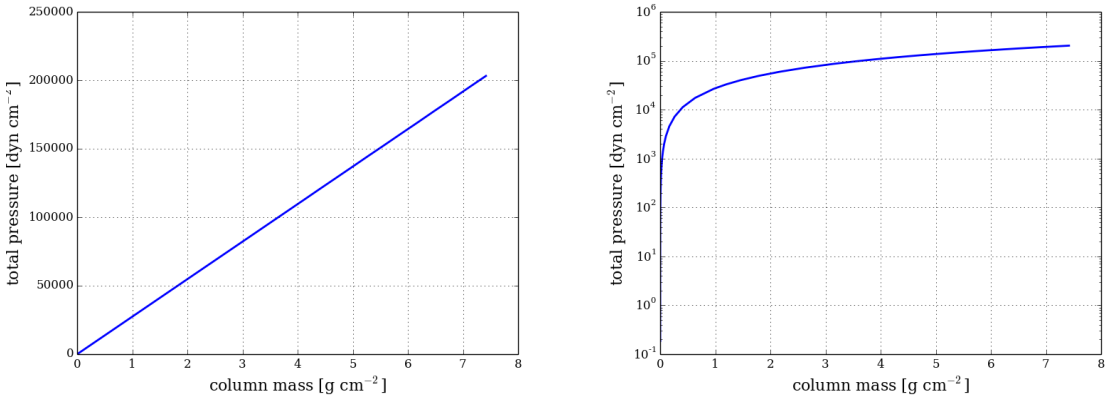


Figure 1: Temperature stratification of the FALC model. The height scale has $h = 0$ at the location with $\tau_{500} = 1$. The photosphere reaches until $h = 525$ km (temperature minimum). The chromosphere higher up has a mild temperature rise out to $h \approx 2100$ km, where it increases steeply as the corona starts.

1.2 FALC density stratification

1.2.1 Total pressure against column mass



(a) Total pressure against column mass, linearly scaled (b) Total pressure against column mass, log. y-axis

Figure 2: In the panels in the figure, we have plotted the total pressure against the column mass, both linearly and with a logarithmic y-axis. As we see, the total pressure scales linearly. We can then write the relationship between pressure and column mass as $P = cm$, where c is a constant with the units $[P/m] = [\mathrm{cm/s^2}]$, which is acceleration. Since it is constant throughout the atmosphere of the sun, we can safely assume that it represents the gravitational acceleration at the surface of the Sun. We calculate this to be $c = g_{\mathrm{surface}} = 27396.5\ \mathrm{cm/s^2}$

1.2.2 Complete mixing

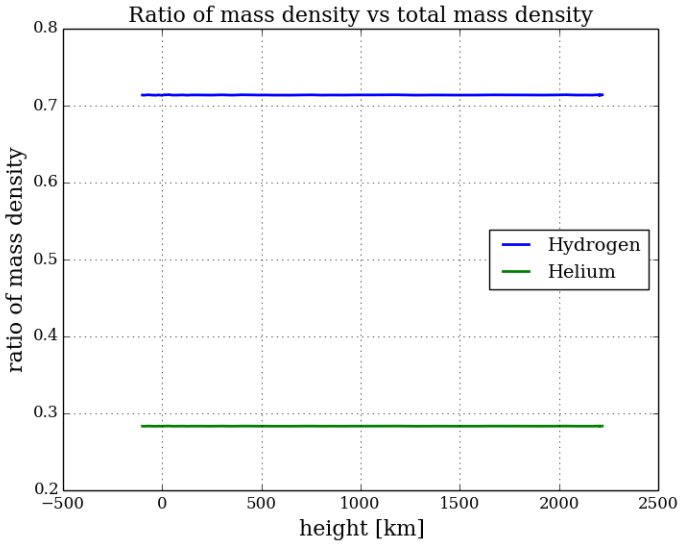
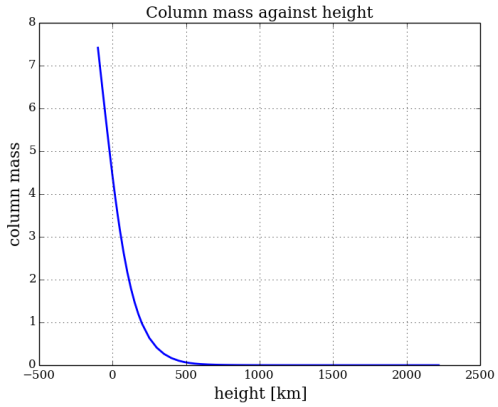
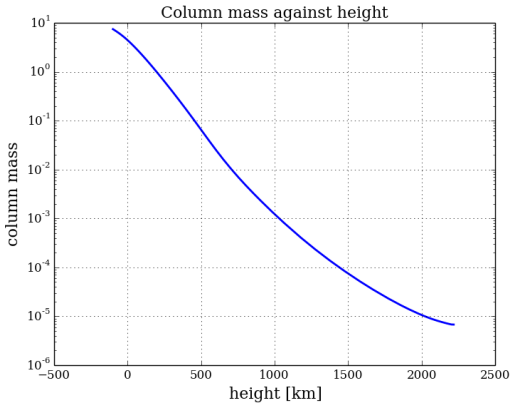


Figure 3: Plot of the relative abundance of hydrogen and helium. We see that the variations at different heights are very small for the elements, so the assumption of complete mixing holds up. The remaining fraction, we call metals, and calculate in our code that the fraction of metal is approximately $2.761 \cdot 10^{-3}$. We can then say that the sun consists of a bit over 70% hydrogen, a bit less than 30% helium, and $\approx 0.3\%$ metals.

1.2.3 Column mass against height



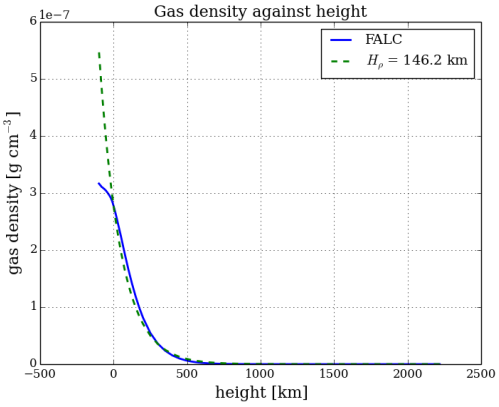
(a) Column mass against height, linearly scaled



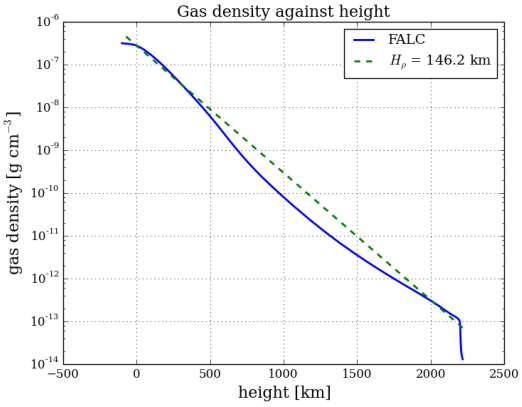
(b) Column mass against height, log. y-axis

Figure 4: Column mass plotted against height for both linear and logarithmic y-axis. We observe that the column mass goes as $m \propto c^h$, since the mass density indreases towards the center of the Sun. From figure 1, we know that the temperature does not increase logarithmically inwards, which affects the mass density, making the column mass not a fully straight line when plotted with logaritmic y-axis.

1.2.4 Gas density against height



(a) Density against height, linearly scaled



(b) Density against height, log. y-axis

Figure 5: Density against height in both linear and ylog plots. Overplotted in dashed lines is the density given by $\rho(0)\exp(-h/H_\rho)$, with $H_\rho = 146.2$ km in the deep photosphere.

1.2.5 Gas pressure against height

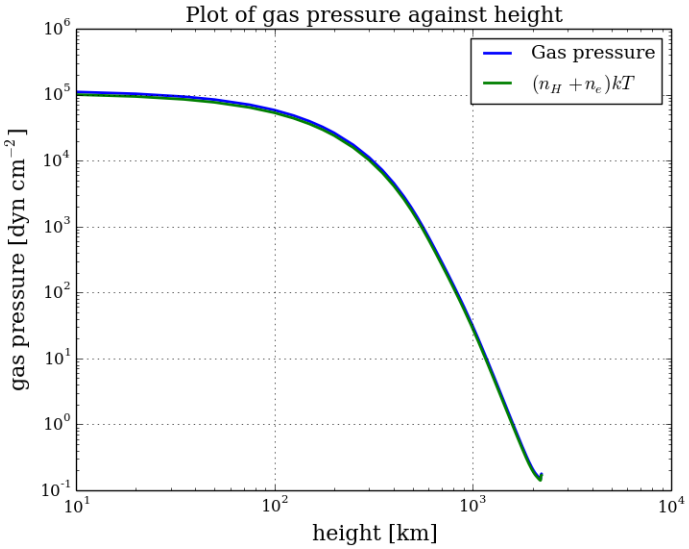
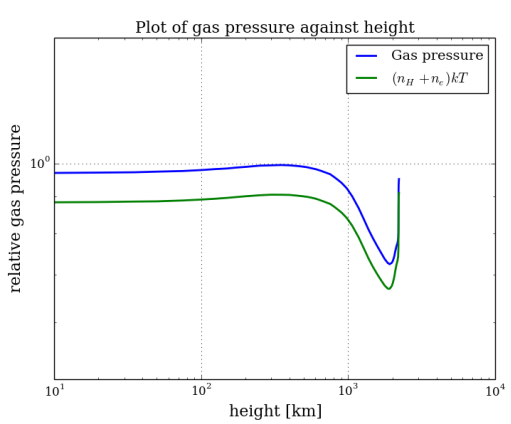
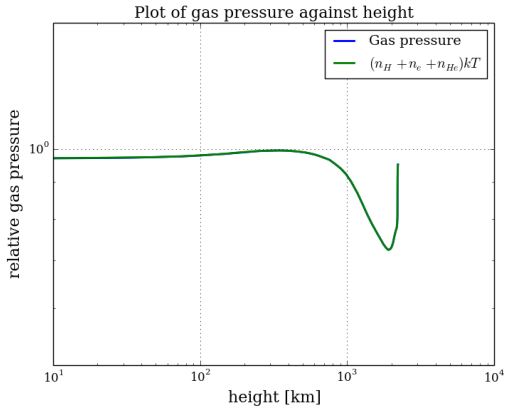


Figure 6: Gas pressure from FALC plotted against height. Overplotted is the ideal gas law for a gas consisting only of hydrogen and free electrons. We see that they are overlap quite a lot, but looking closer at it would be a good idea

1.2.6 Column mass against height



(a) Relative gas pressure against height, log x, log y.



(b) Relative gas pressure with He added to IG, log x, log y.

Figure 7: Relative gas pressure (P_g/P_{tot}) against height. Overplotted is ideal gas for hydrogen and free electrons. On the panel to the right, helium is also added to the ideal gas law. We see that by adding helium, the ideal gas law overlaps the FALC data pretty much perfectly.

1.2.7 Number densities against height

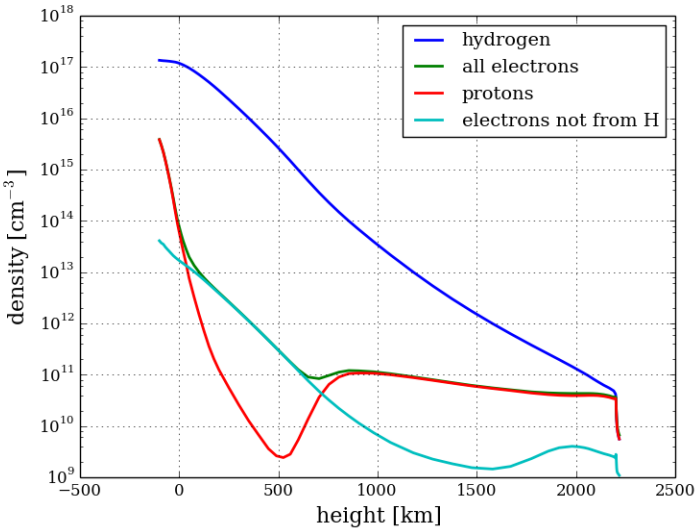


Figure 8: Plot of number densities for the different components of the Sun, against height, plotted with logarithmic y-axis. We see that there is a dip the proton density at $h \approx 500$ km, which we remember was the temperature minimum in our temperature stratification plot. It makes sense that the free proton density is at its lowest there, as the gas is coldest, and less hydrogen will be ionized. All number densities decrease with increasing height, as the mass density drops. The cyan curve, representing electrons not coming from hydrogen is parallel to the hydrogen density in the photosphere, and low chromosphere, where the temperature is fairly low and changes slowly. This implies that for low heights, the metal abundance compared to hydrogen is constant. As we start going towards the corona, the cyan curve has a bump as the temperature increases, and metals are ionized more, releasing more electrons.

1.2.8 Ionization fraction of hydrogen

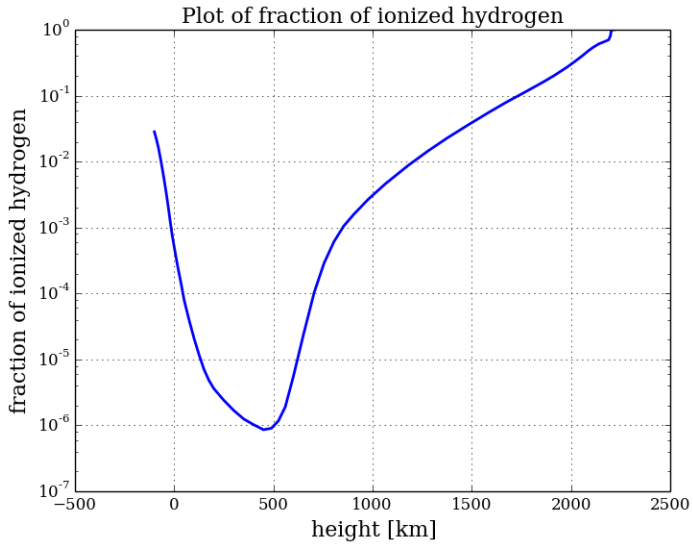


Figure 9: Plot of the fraction of ionized hydrogen to total hydrogen, against height, with logarithmic y-axis. We see that it reaches a minimum around $h = 500$ km, as one would expect, since that is where figure 1 of the temperature reaches a minimum.

1.2.9 Photon density and particle density

We approximate the photon density to $N_{\text{photons}} \approx 20T^3$, which holds for thermodynamic equilibrium and isotropic radiation. For lower parts of the atmosphere, these are satisfied, but higher up, the density is too low for TE, so we approximate it to $N_{\text{photons}} \approx 20T_{\text{eff}}^3/2\pi$, where $T_{\text{eff}} = 5770$ K.

At the lowest part of the atmosphere, $h = -100$ km:

$$\begin{aligned} N_{\text{hydrogen}} &= 1.351 \cdot 10^{17}\,\text{cm}^{-3} \\ N_{\text{photons}} &= 1.661 \cdot 10^{13}\,\text{cm}^{-3} \\ \frac{N_{\text{photons}}}{N_{\text{hydrogen}}} &= 1.230 \cdot 10^{-4} \end{aligned}$$

At the highest part of the atmosphere, $h = 2218.2$ km:

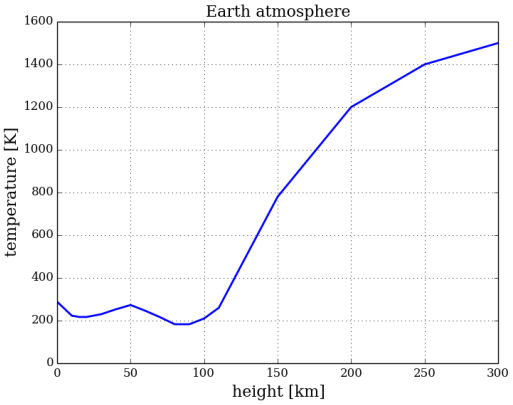
$$\begin{aligned} N_{\text{hydrogen}} &= 5.575 \cdot 10^{9}\,\text{cm}^{-3} \\ N_{\text{photons}} &= 6.115 \cdot 10^{11}\,\text{cm}^{-3} \\ \frac{N_{\text{photons}}}{N_{\text{hydrogen}}} &= 109.68 \end{aligned}$$

We see that in the lower atmosphere, the photon density is insignificant compared to the hydrogen density, while being completely dominant in the upper part of the atmosphere. The medium there is

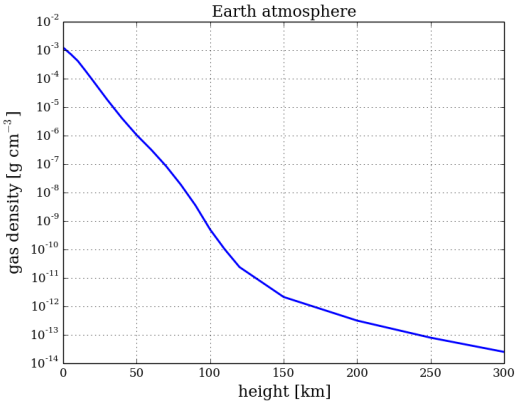
insensitive to these photons as the density is so low that collisions are very very rare. The Lyα line is a resonant line for hydrogen, so the hydrogen there can absorb and emit Lyα.

1.3 Comparison with the Earth's atmosphere

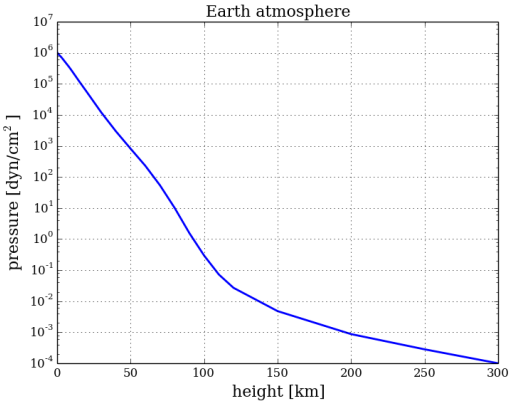
1.3.1 Data from Earth's atmosphere



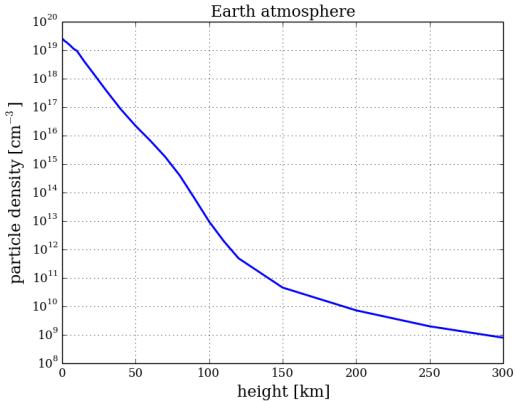
(a) Temperature against height



(b) Gas density against height, log y.



(c) Gas pressure against height, log y.



(d) Particle density against height, log y.

Figure 10: We see that the temperature is fairly stable until $h = 120$ km, where it starts increasing for some reason. The other panels decrease with increasing height like you would expect.

1.3.2 Normalized pressure, gas and number densities

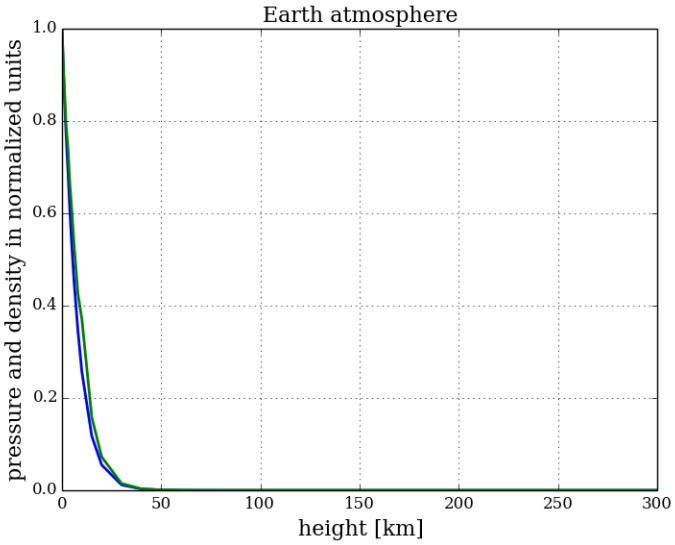


Figure 11: Plot of normalized pressure, gas and number densities against height. For lower heights, they all decrease fairly identical, but at around $h = 120$ km, the slope of the logarithmic curves change. This was also the height where the temperature also started to change its behavior.

1.3.3 Mean molecular weight

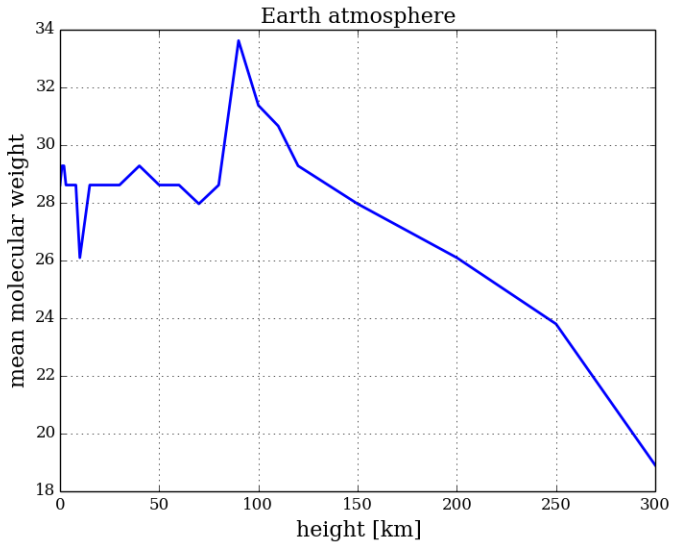


Figure 12: The mean molecular weight plotted against height. We see that for heights over $h = 120$ km (the same height again), the mean molecular weight decreases with increasing height. This is due to the fact that the heavier elements will flow downwards, while lighter elements will be buoyant, and rise upwards.

1.3.4 Calculating and comparing to the Sun

Like we did with the Sun, we can calculate the density scale height for the lower atmosphere. This gives us $H_\rho = 10$ km. Using this (and the fact that Mount Everest is 8.848 km tall), we calculate that it is 42.2% harder to breath at Mount Everest than at the surface.

We compare the density scale height with the one we found for the lower Sun surface

$$\frac{N_{\text{Earth}}}{N_{\text{Sun}}} = 197.575$$

Using that the standard gravity at the Earth's atmosphere is $g_{\text{E}} = 980.665$ cm s^{-2}, we estimate the atmospheric column mass to be

$$m = 1043.468\,\text{g cm}^{-2}$$

The column mass at the base of the stellar photosphere was $m = 4.404$ g cm^{-2}, so the one on earth is considerable higher.

The sunshine photon density at earth is given by

$$N_{\text{phot}} = \pi \frac{R^2}{D^2} N_{\text{phot}}^{\text{top}}$$

with $N_{\text{phot}}^{\text{top}} = 6.115 \cdot 10^{11}\,\text{cm}^{-3}$, R the solar radius and D the distance sun-earth. This gives us

$$N_{\text{phot}} = 4.152 \cdot 10^{7}\,\text{cm}^{-3}$$

the particle density in the air is

$$N_{\text{part}} = 2.570 \cdot 10^{19}\,\text{cm}^{-3}$$

and the local thermal photon production

$$N_{\text{phot}}^{\text{loc}} = 4.778 \cdot 10^{8}\,\text{cm}^{-3}$$

The local thermal photon production is higher than the one coming from the sun, so could maybe indicate that the LTE assumption does not hold that well in low Earth atmosphere.

2 Continuous spectrum from the solar atmosphere

2.1 Observed solar continua

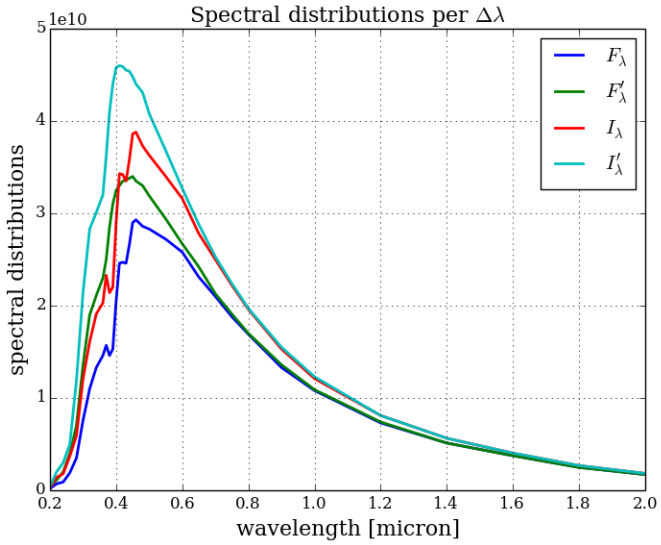


Figure 13: Plot of the four spectral distributions against wavelength. They have the same unit as intensity is a measure of flux per solid angle.

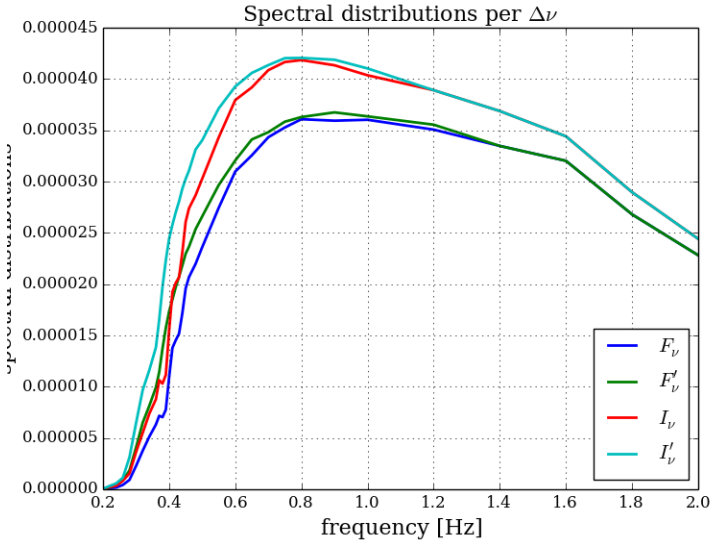


Figure 14: Plot of the four spectral distributions per frequency bandwidth $\Delta\nu = 1$ Hz.

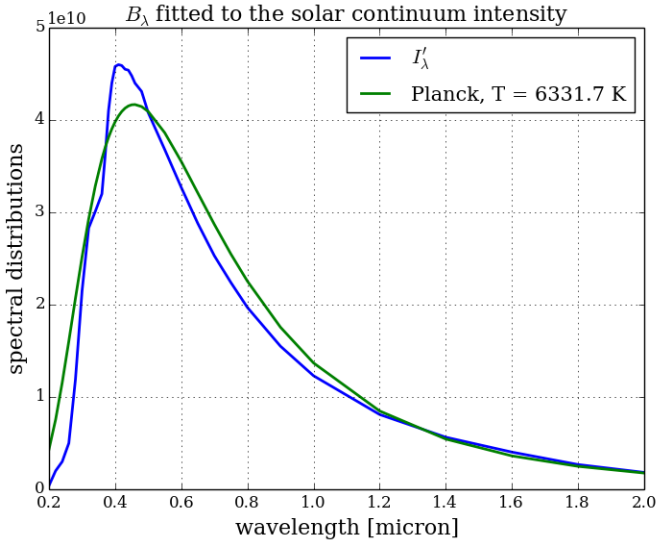


Figure 15: Plot of continuum intensity with a Planck function fitted to it. The Planck function gives us the best fit for temperature $T = 6331.7$ K.

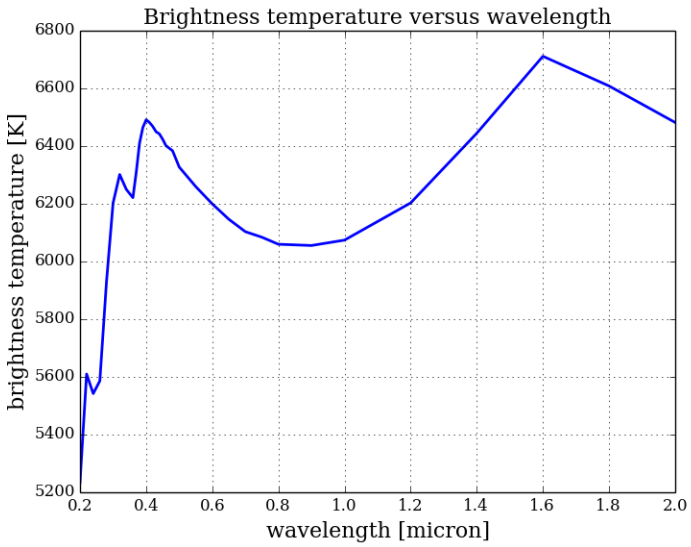


Figure 16: Plot of the inverted Planck function versus wavelength. This gives us the brightness temperature T_b for a wavelength. The shape of this curve looks a lot like an inverse version of figure 5 from the assignment. It peaks at around $\lambda = 1.6\,\mu$m, which means that the Sun should be fairly transparent to this wavelength, and that most of the radiation escapes, resulting in a high brightness temperature.

2.2 Continuous extinction

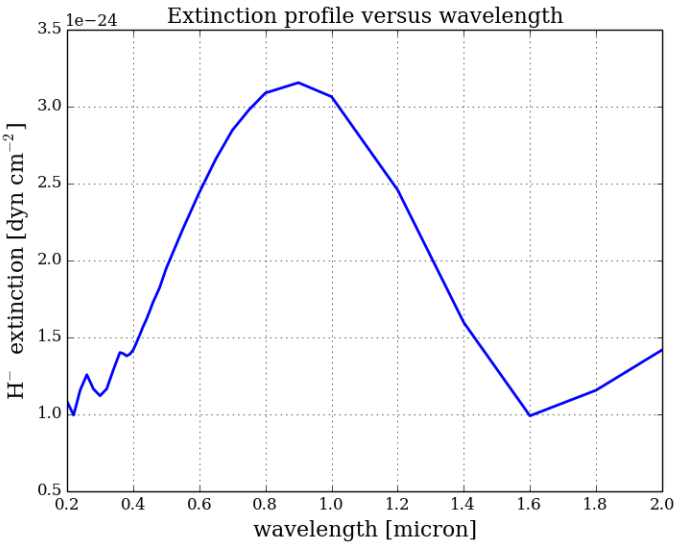


Figure 17: Extinction coefficient plotted against wavelength. We see that this looks quite a lot like Gray's version (figure 5 in SSB). However after the peak, it does not just decay over $\simeq \lambda^3$, as is more close to the total extinction rather than just from H^-_{bf}. We see that this is close to the solar brightness temperature plot with wavelength, or more strictly inverse of each other.

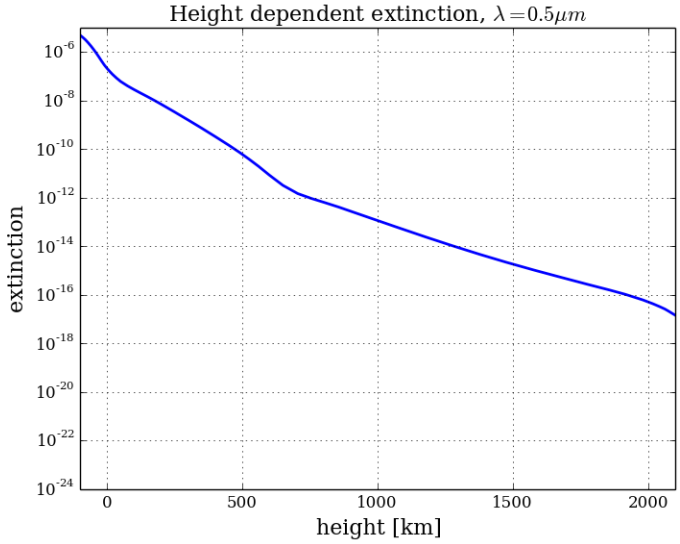


Figure 18: H^- extinction per cm with height for $\lambda = 0.5\,\mu$m, log y.

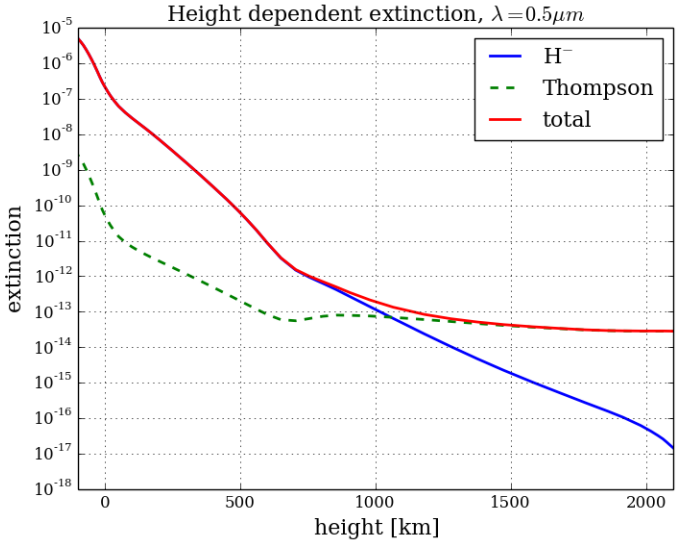


Figure 19: Extinction per cm with height for $\lambda = 0.5\,\mu$m, log y. Unlike the last figure, H^- extinction is now measured per neutral hydrogen atom. Thompson scattering off free electrons is also added to the extinction per cm by the constant $\sigma^{\mathrm{T}} = 6.648 \times 10^{-25}$ cm^2, and multiplying with the electron number density to get correct units. Also plotted, is the total continuous extinction.

For low heights, we see that the H^- extinction dominates, and for higher heights, the Thompson scattering dominates. This follows from how the ionization fraction of hydrogen increases after it reaches the minimum at $h \approx 500$ km, and the number density of neutral hydrogen decreases.

2.3 Optical depth

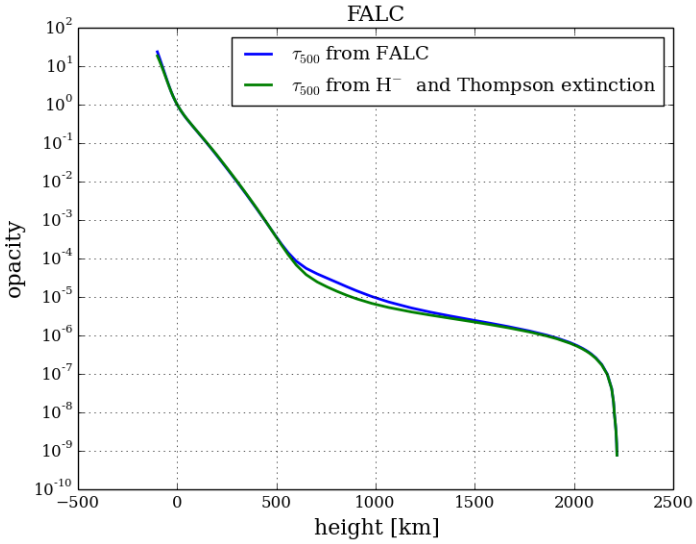


Figure 20: Integrated extinction at $\lambda = 500$ nm to get the optical depth scale $\tau_\lambda(h_0)$, plotted against height. Overplotted is the FALC τ_{500} scale. We see that our code reproduces the τ_{500} results very well for low and high heights, but not so well in between.

2.4 Emergent intensity and height of formation

We can now compute the intensity of the radiation that emerges from the center of the solar disk, by integration as explained in the exercise text. Doing this for $\lambda = 500$ nm, gives us the following computed intensity, which we can compare to the continuum intensity we looked at earlier.

$$I_{\text{computed}} = 4.26826 \cdot 10^{14} \text{ergs}^{-1}\text{cm}^{-2}\text{ster}^{-1}$$
$$I_{\text{observed}} = 4.08 \cdot 10^{14} \text{ergs}^{-1}\text{cm}^{-2}\text{ster}^{-1}$$

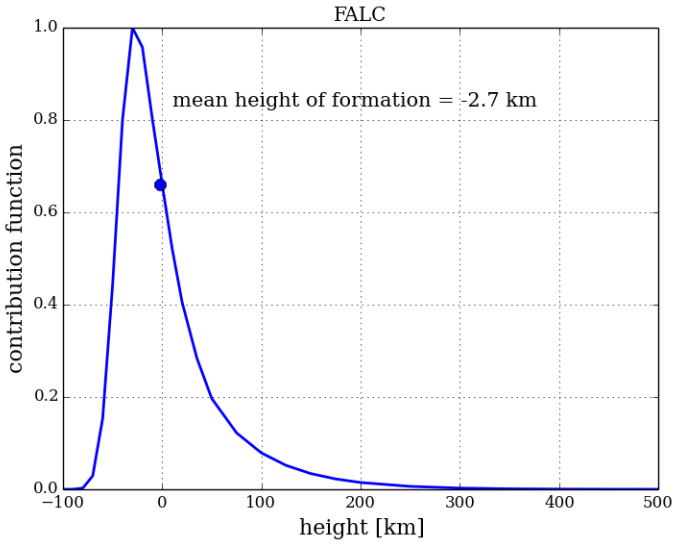


Figure 21: Peak-normalized contribution function plotted against height. Overplotted as a point is the mean height of formation. For $\lambda = 500$ nm.

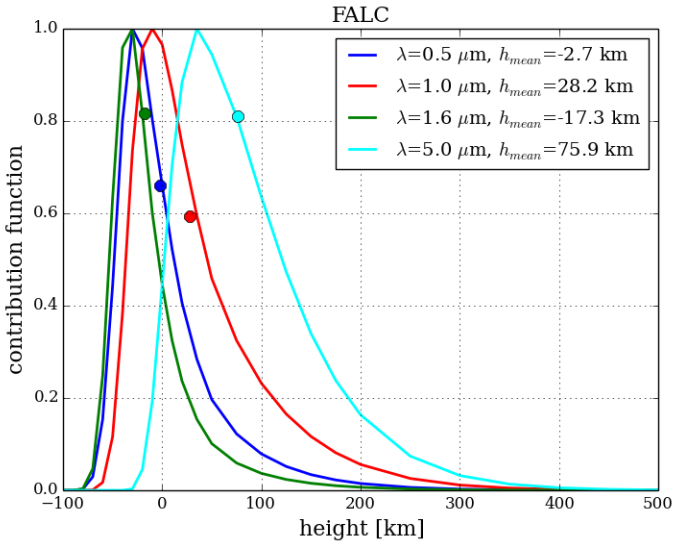


Figure 22: Peak-normalized contribution function plotted against height for different wavelengths λ. Overplotted as a point is the mean height of formation.

2.5 Disk-center intensity

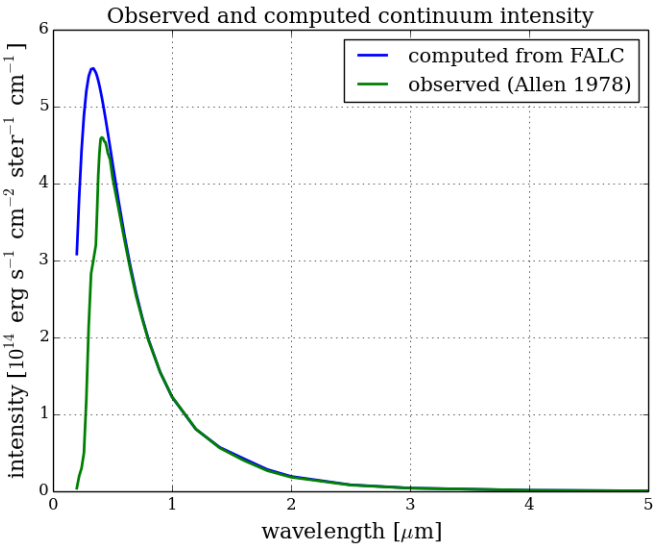


Figure 23: Disk-center intensity plotted against wavelength. Overplotted is the observed solar continuum in `solspect.dat`.

2.6 Limb darkening

We can modify out code for disk-center intensity to give the intensity that emerges under an angle $\mu = \cos\theta$.

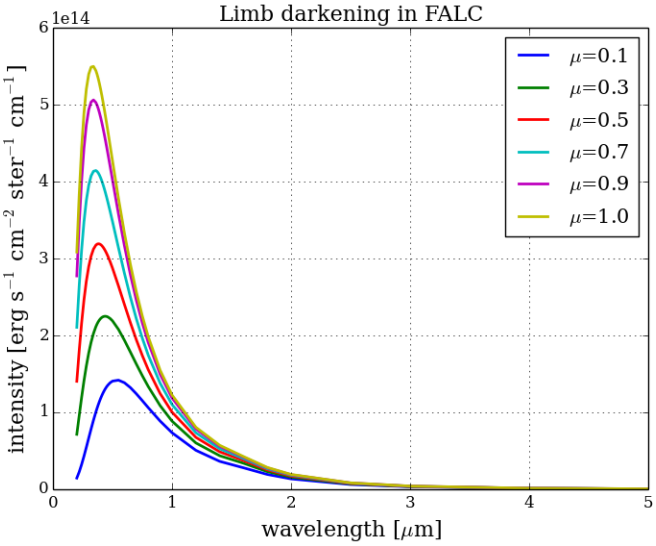


Figure 24: Intensity evaluated for multiple values of μ, plotted against wavelength. We see that for lower values of μ, the intensity peaks drop down quite a lot.

Was not successfull in creating the plots for the next part of this subsection, and thus there are no more plots here.

2.7 Flux integration

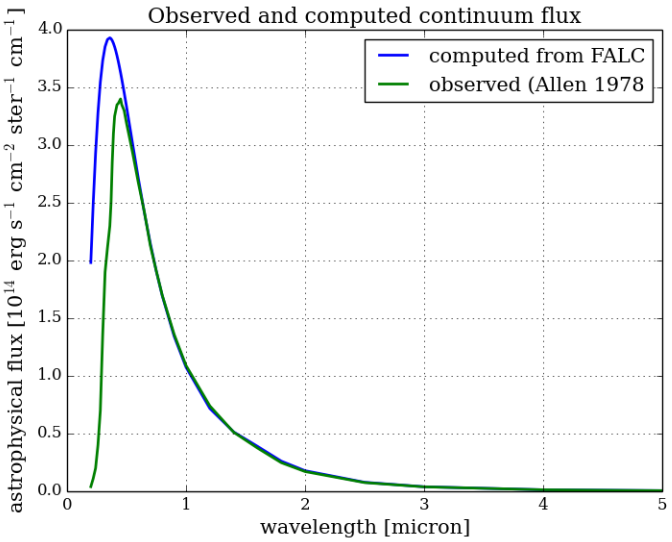


Figure 25: Emergent solar flux plotted against wavelength. Overplotted is the measured flux from `solspect.dat`. We see that the figure is very similar to the one for the emergent intensity, which is maybe not very surprising given how similar in shape the four spectral distributions we plotted earlier were.

3 Spectral lines from the solar atmosphere

3.1 Observed Na D line profiles and Na D wavelengths

We start by reading in the data from the data file

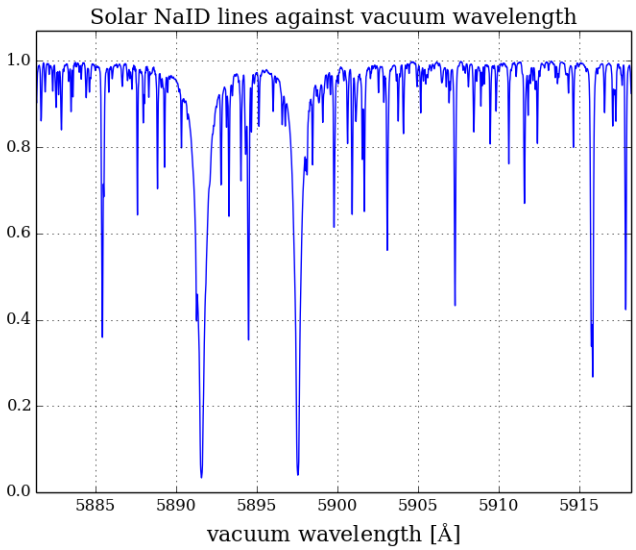


Figure 26: Solar disk-center intensity plotted against vacuum wavelengths. We find the location of the minimas to be in $\lambda_1 = 5891.58$ Å, and $\lambda_2 = 5897.55$ Å.

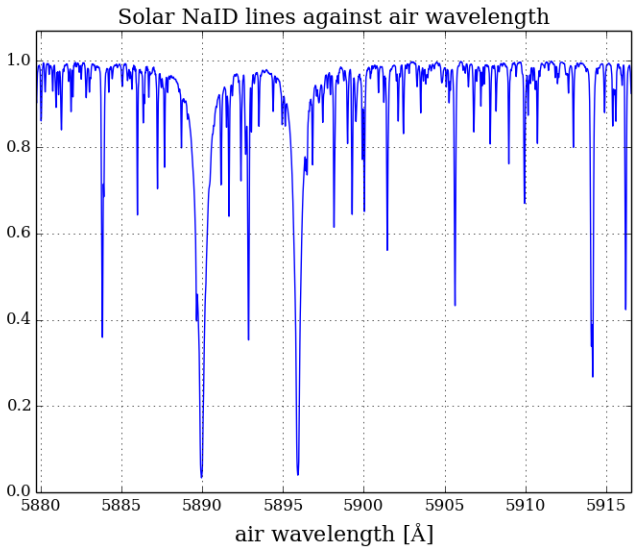
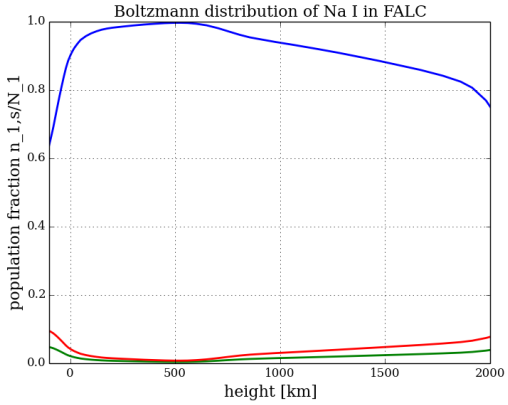


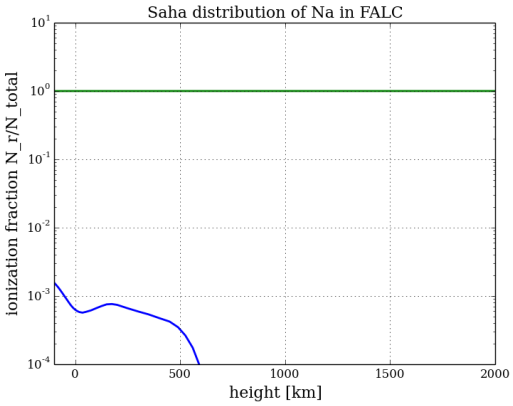
Figure 27: Solar disk-center intensity plotted against air wavelengths. We find the location of the minimas to be in $\lambda_1 = 5889.95$ Å, and $\lambda_2 = 5895.92$ Å. We observe that the minimas are very close to the expected location of the lines.

3.2 Implementing LTE line formation

We test to see if our implementation of the Saha-Boltzmann distributions return reasonable results.



(a) Boltzmann distribution for different excitation levels s.



(b) Saha distribution for different ionization levels r.

Figure 28: The panels show that the Saha-Boltzmann functions for Na gives the results we would expect, based on the figure shown to us.

3.3 Computed Na D_1 line profile

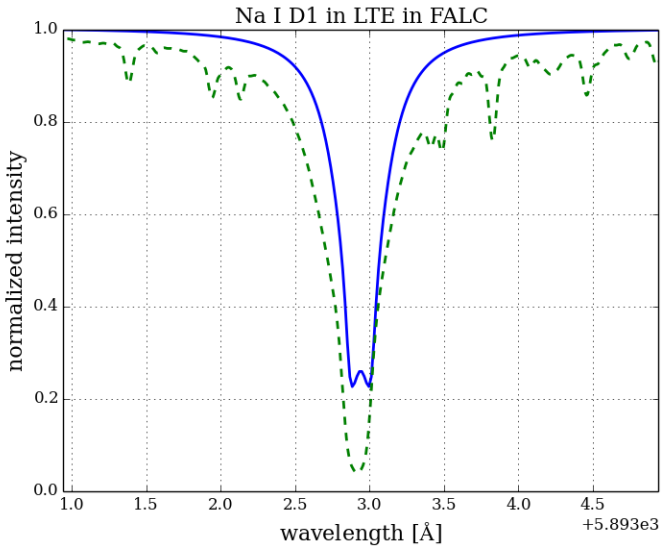


Figure 29: The final computed Na D_1 line profile plotted against wavelength. Overplotted in dashed lines is the observed Na D line profile we plotted earlier. We see that the computed line follows roughly the same shape as the observed line profile. One of the differences, is that our line does not go as deep, and that at the center, line-center reversal occurs. Our model is a fairly decent model for photospheric lines, but one of the key assumptions of our model was LTE. Since the computed line profile reproduces the core poorly, we can assume that the assumption of LTE breaks down for the core.

Following tradition, we add a collisional enhancement factor E to the γ_{vdW}, in order to attempt obtaining a better fit of the line wings.

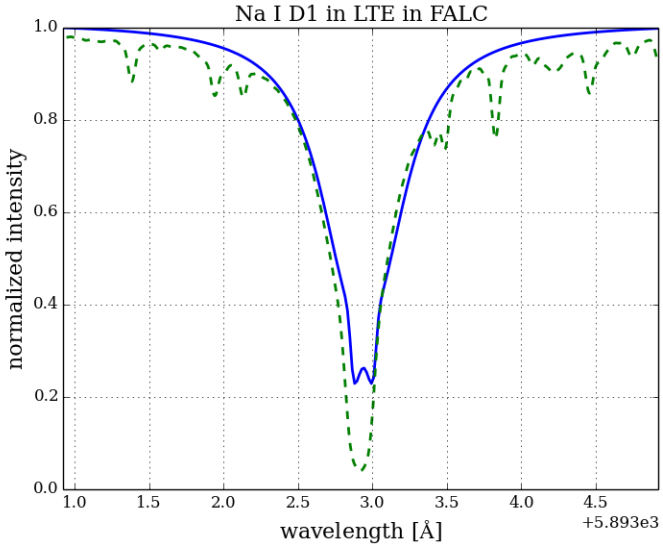


Figure 30: Computed line profile plotted against wavelength, with the observed line profile overplotted. Here we have after experimenting set the fudge factor E to $E = 3$ in order to get a better fit. As we see, it fits the observed data better than without the factor.

4 Appendix - Python code

All code used can be found in the GitHub repository located at
https://github.com/simennb/ast4310/tree/master/exercise2/src
The python-code is also appended below

4.1 General tools and functions for all programs

```
from pylab import *
from matplotlib import rc
rc('font',**{'family':'serif'})
import numpy as np
import astropy as ap
from astropy import units as u
from scipy import special

# Useful constants and other things
class SSB:
    def __init__(self):
        self.m_H = ap.constants.m_p.cgs   # mass of hydrogen/proton [g]
        self.m_He = 3.97*self.m_H         # mass of helium [g]
        self.m_e = ap.constants.m_e.cgs   # mass of electron [g]
```

```
    self.h = ap.constants.h.cgs       # planck's constant [erg s]
    self.k_B = ap.constants.k_B.cgs   # Boltzmann's constant [erg/K]
    self.k_eV = self.k_B.to('eV/K')   # Boltzmann's constant [eV/K]
    self.c = ap.constants.c.cgs       # speed of light [cm/s]
    self.g_E = ap.constants.g0.cgs    # gravitation acc. on earth [cm/s^2]
    self.sigma_T = 6.648e-25*u.cm**2  # Thompson e scattering cross section
    self.hc = (self.h*self.c).to('eV AA') # in order to get photon energy

    # Initializing some things for Na I lines
    self.g_rs = [2.,2.,4.] # ground state, 3pS1/2, 3pS3/2 level
    self.chi_ion = [5.139, 47.29, 71.64] # ionization energies in eV
    self.chi_level = zeros(3) # chi[0] = 0 cause shared ground state
    self.chi_level[1] = self.hc.value/5895.94   # energy of NaI D1 [eV]
    self.chi_level[2] = self.hc.value/5889.97   # energy of NaI D2 [eV]
    self.wav0 = 5895.94

##############################################################################
##############################################################################
# Functions for reading from file

def read_falc(self):
    h,tau5,colm,temp,vturb,nhyd,nprot,nel,ptot,pgasptot,dens = np.loadtxt('../data/falc.dat',unpa
    # give each an astropy unit
    h *= u.km
    colm  *= u.g*u.cm**(-2)
    temp  *= u.K
    vturb *= u.km/u.s
    nhyd  *= u.cm**(-3)
    nprot *= u.cm**(-3)
    nel   *= u.cm**(-3)
    ptot  *= u.dyn*u.cm**(-2)
    dens  *= u.g*u.cm**(-3)

    return h, tau5, colm, temp, vturb, nhyd, nprot, nel, ptot, pgasptot, dens

def read_earth(self):
    h,logP,temp,logrho,logN = np.loadtxt('../data/earth.dat',unpack=True)
    # giving each an astropy unit
    h *= u.km
    temp *= u.K

    return h, logP, temp, logrho, logN

def read_solspect(self):
    wav, F, F_c, I, I_c = np.loadtxt('../data/solspect.dat',unpack=True)
    wav *= u.micron
```

```
    F   *= u.erg/(u.cm**2*u.s)
    F_c *= u.erg/(u.cm**2*u.s) #1 # 10**10 erg cm**-2 s**-1
    I   *= u.erg/(u.cm**2*u.s*u.micron) # neglecting steradian
    I_c *= u.erg/(u.cm**2*u.s*u.micron)

    # for some reason not allowed to do in same operation as units
    # Scaling back from 1e10 erg to erg
    F   *= 1e10
    F_c *= 1e10
    I   *= 1e10
    I_c *= 1e10

    return wav, F, F_c, I, I_c

def read_NaID(self):
    sigma,lines= np.loadtxt('../data/int_nad.dat',usecols=(0,2),unpack=True)

    sigma /= u.cm
    wav_vac = 1./sigma

    return sigma, wav_vac, lines

##############################################################################
##############################################################################
# Functions for calculating various things

def planck(self, temp, wav):
    # making sure wavelength is in cgs
    wav = wav.to('cm')
    h = self.h
    k_B = self.k_B
    c = self.c

    return 2*h*c**2/wav**5*1./(exp(h*c/(wav*k_B*temp))-1)

def brightness_temp(self, I_c, wav):
    # making sure wavelength is in cgs
    wav = wav.to('cm')
    h = self.h
    k_B = self.k_B
    c = self.c

    return h*c/(wav*k_B)/log(2*h*c**2/(I_c*wav**5)+1)

def exthmin(self, 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
'''
wave = (wav.to('AA')).value # for simplicity
# other parameters
theta = 5040./temp.value
elpress = eldens*self.k_B*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*wave +2.64243E-6*wave**2
           -4.40524E-10*wave**3 +3.23992E-14*wave**4
           -1.39568E-18*wave**5 +2.78701E-23*wave**6)
sigmabf *= 1e-18  # cm^2 per H-min ion
if size(wave) > 1:
    sigmabf[wave > 16444] = 0 # H-min ionization limit at lambda=1.6444 micron
elif (size(wave) == 1):
    if wave > 16444:
        sigmabf = 0

# Astropy unit, adding because simplicity
sigmabf *= u.cm**2

# 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(-self.h*self.c/(wav*1e-8*self.k_B*temp))) # correct stimulated
kappabf = kappabf.value/u.cm  # cause astropy trouble

# check Gray's Saha-Boltzmann with AFYC (edition 1999) p168
#logratio=-0.1761-np.log10(elpress.value)+np.log10(2.)+2.5*np.log10(temp.value)-theta*0.754
#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(wave)
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)
    kappaff = kappaff.value/u.cm

    return kappabf+kappaff

def vactoair(self,wav_vac):
    return 0.99972683*wav_vac + 0.0107-196.25/wav_vac

################################################################################
################################################################################
# Integrations

def optical_depth(self, h, temp, eldens,n_neutral, wav=500*u.nm):
    '''
    calculates the optical depth integral
    '''
    tau = np.zeros(len(h), dtype=float) # initializing tau array
    ext = self.exthmin(wav, temp, eldens)
    # add Thompson scattering
    ext = ext*n_neutral.value + (self.sigma_T*eldens)

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

    return tau

def emergent_intensity(self,h,tau5,temp,nhyd,nprot,nel, wl=0.5*u.micron,mu=1.0):
    ext = np.zeros(len(tau5))/u.cm
    tau = np.zeros(len(tau5))
    integrand = np.zeros(len(tau5))*u.erg/(u.cm**3*u.s)
    contfunc = np.zeros(len(tau5))*u.erg/(u.cm**4*u.s)
    intt = 0.0*integrand.unit
    hint = 0.0*h.unit*integrand.unit

    for i in range(1, len(tau5)):
        ext[i] = (self.exthmin(wl, temp[i], nel[i])*(nhyd[i]-nprot[i]).value
                  + self.sigma_T*nel[i])
        tau[i] = tau[i-1] + 0.5 * (ext[i] + ext[i-1]) * (h[i-1]-h[i])
        integrand[i] = (self.planck(temp[i],wl)*np.exp(-tau[i]/mu))
        intt += 0.5*(integrand[i]+integrand[i-1])*(tau[i]-tau[i-1])/mu
        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
```

```
    return intt, contfunc, hmean

def flux_integration(self,h,tau5,temp,nhyd,nprot,nel, wav):
    #remove units
    h = h.value; temp = temp.value; nhyd = nhyd.value
    nprot = nprot.value; nel = nel.value; wav = wav.value

    # 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] = (self.exthmin(wl*u.micron, temp[i]*u.K, nel[i]/u.cm**(-3)).value*(nhyd[i
                          + self.sigma_T.value*nel[i])
                tau[i] = (tau[i-1] + 0.5 * (ext[i] + ext[i-1]) *
                          (h[i-1]-h[i])*1E5)

                integrand[i] = self.planck(temp[i]*u.K,wl*u.micron).value*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}

    return fluxspec

##########################################################################
##########################################################################
# NaID model

def partfunc_Na(self,temp):
    theta = 5040./temp
    c0 =  0.30955
```

```
    c1 = -0.17778
    c2 =  1.10594
    c3 = -2.42847
    c4 =  1.70721
    logtheta = log10(theta)
    logU = c0+c1*logtheta+c2*logtheta**2+c3*logtheta**3+c4*logtheta**4

    U = zeros(3)
    U[0] = 10**logU
    U[1] = 1.  # Allen 1976
    U[2] = 6.  # Allen 1976

    return U

def saha_Na(self, temp, eldens,rstage):
    h = self.h.value
    k_erg = self.k_B.value
    k_eV  = self.k_eV.value
    m_e = self.m_e.value

    U = self.partfunc_Na(temp)
    U = np.append(U,2)
    sahaconst = 2./eldens*(2*pi*m_e*k_erg*temp/h**2)**1.5

    nstage = zeros(len(self.chi_ion)+1)
    nstage[0] = 1
    for r in range(len(self.chi_ion)):
        nstage[r+1] = nstage[r]*sahaconst*U[r+1]/U[r]*exp(-self.chi_ion[r]/(k_eV*temp))

    ntotal = sum(nstage)
    nstagerel = nstage/ntotal

    return nstagerel[rstage-1]

def boltz_Na(self, temp, s):
    k_eV = self.k_eV.value
    U = self.partfunc_Na(temp)

    relnrs =  self.g_rs[s-1]/U[0]*exp(-self.chi_level[s-1]/(k_eV*temp))
    return relnrs

def sahaboltz_Na(self, temp, eldens, r, s):
    return self.saha_Na(temp,eldens,r)*self.boltz_Na(temp,s)

def dopplerwidth(self, wav, temp, vmicro, atmass):
    c = self.c.value
```

```
    k_B = self.k_B.value


    return wav/c*sqrt(2*k_B*temp/atmass + vmicro**2*1e10)

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

    return V

def gammavdw(self, 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 = self.rsq(s)
    rsq_1 = self.rsq(1) # lower level D1 and D2 lines is ground state s=1
    loggvdw = (6.33 + 0.4*log10(rsq_u - rsq_1)
               + log10(pgas) - 0.7 * log10(temp))

    return 10**loggvdw

def rsq(self, 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
    E_ionization = self.chi_ion[0]
    E_n = self.chi_level
    Z = 1.                 # ionization stage, neutral Na: NaI
    Rydberg = 13.6         # Rydberg constant [eV]
    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

def line_broadening(self, wav, temp, pgas, vmicro, atmass, s,E=1.0):
    c = self.c.value
    dopplerwidth = self.dopplerwidth(wav, temp, vmicro, atmass)
    gammavdw = self.gammavdw(temp, pgas, s)
    a_voigt = wav**2 / (4*pi*c) * E*gammavdw / dopplerwidth
    v_voigt = (wav-self.wav0*1e-8)/dopplerwidth
```

```
    voigt_NaD = self.voigt(a_voigt,abs(v_voigt))
    voigt_NaD /= dopplerwidth
    return dopplerwidth, voigt_NaD

def NaD1_ext(self, wav, temp, eldens, nhyd, vmicro, pgas):
    # Wavelength given in [cm]
    # not using astropy in order to increase speed, hopefully
    # Functions called are modified to simplify and only work for NaI D1
    m_e = self.m_e.value
    c = self.c.value
    hc = (self.h*c).value
    k_B = self.k_B.value

    m_Na = 22.99*1.6605e-24  # natrium mass [g]
    A_Na = 1.8e-6   # sodium abundance (Allen 1976)
    f_D1 = 0.318    # oscillator strength for NaI D1
    f_D2 = 0.631    # oscillator strength for NaI D2
    e = 4.803204e-10  # electron charge in cgs [statcoloumb]
    # Natrium I D1 line
    r = 1
    s = 2
    bl = 1.0
    bu = 1.0

    # Function calls
    dopplerwidth, H = self.line_broadening(wav,temp,pgas,vmicro,m_Na,s,3)

    alpha1 = sqrt(pi)*e**2/(m_e*c)*wav**2/c*bl
    alpha2 = self.sahaboltz_Na(temp, eldens, r, 1)
    alpha3 = nhyd*A_Na*f_D1
    alpha4 = H
    alpha5 = 1. - bu/bl*exp(-hc/(wav*k_B*temp))
    alpha = alpha1*alpha2*alpha3*alpha4*alpha5

    return alpha

def emergent_intensity_Na(self,h,tau5,temp,nhyd,nprot,nel,wl,vmicro,pgas):
    # New almost identical function since removed units in Na
    # calculations as astropy seems to really slow things down

    # SSB 2.4 page 16 emergent intensity, contribution function and mean height of formation in H
    sigma_Thomson = 6.648E-25 # Thomson cross-section [cm^2]

    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] = (self.exthmin(wl*1e4*u.AA, temp[i]*u.K, nel[i]*u.cm**(-3)).value*(nhyd[i]-nprot[
                      + sigma_Thomson*nel[i]
                      + self.NaD1_ext(wl*1e-4,temp[i-1],nel[i-1],nhyd[i-1],vmicro[i-1],pgas[i-1]))[redacted]
            tau[i] = tau[i-1] + 0.5 * (ext[i] + ext[i-1]) * (h[i-1]-h[i])*1E5
            integrand[i] = self.planck(temp[i]*u.K,wl*u.micron).value*np.exp(-tau[i])[redacted]
            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

        return intt
```

4.2 FALC plotting

```
from ssb import *
# np, ap, u and pylab *

# some plotting parameters
path = '../figures/falc/' # folder where figures are stored
fs = 16  # font size

ssb = SSB()

h,tau5,colm,temp,vturb,nhyd,nprot,nel,ptot,pgasptot,dens = ssb.read_falc()

nhel = 0.1*nhyd


##############################################################################
# Temperature stratification
'''
figure()
plot(h, temp,lw=2)
grid('on')
xlabel('height [km]',size=fs)
ylabel('temperature [K]',size=fs)
title('Temperature stratification of the FALC model',size=fs)
ylim([0,20000])
savefig(path+'falc_h_temp.png')
'''
```

```
################################################################################
# Plot total pressure against column mass, linearly and logarithmically
'''
figure()
plot(colm,ptot,lw=2)
xlabel(r'column mass [g cm$^{-2}$]',size=fs)
ylabel(r'total pressure [dyn cm$^{-2}$]',size=fs)
grid('on')
savefig(path+'falc_colm_ptot.png')
yscale('log')
savefig(path+'falc_colm_ptot_ylog.png')
'''

################################################################################
# Ratio of hydrogen mass density versus height
'''
figure()
plot(h,nhyd*ssb.m_H/dens,label='Hydrogen',lw=2)
plot(h,nhel*ssb.m_He/dens,label='Helium',lw=2)
#plot(h,(nhel*ssb.m_He+nhyd*ssb.m_H)/dens)
xlabel('height [km]',size=fs)
ylabel('ratio of mass density',size=fs)
title('Ratio of mass density vs total mass density',size=fs)
legend(loc='best',fontsize=14)
grid('on')
savefig(path+'falc_h_dens.png')

# Fraction of metal
print 'Fraction of metal in the model mix = %.3e'% mean(1-(nhyd*ssb.m_H+nhel*ssb.m_He)/dens)
'''

################################################################################
# Plot column mass against height
'''
figure()
plot(h,colm,lw=2)
xlabel('height [km]',size=fs)
ylabel('column mass [g cm$^{-2}$]',size=fs)
title('Column mass against height',size=fs)
grid('on')
savefig(path+'falc_h_colm_lin.png')
yscale('log')
savefig(path+'falc_h_colm_ylog.png')
'''

################################################################################
```

```
# Density scale height
'''
height = 2017*u.km
rho0 = dens[h==0]
rhoN = dens[h==height]
H_p = height/(log(rho0/rhoN))
print 'Density scale height = %.2f km\n'%H_p.value

# Plot gas density against height
figure()
plot(h,dens,lw=2,label='FALC')
plot(h,rho0*exp(-h/H_p),'--',lw=2,label=r'$H_{\rho}$ = %.1f km'%H_p.value)
xlabel('height [km]',size=fs)
ylabel('gas density [g cm$^{-3}$]',size=fs)
title('Gas density against height',size=fs)
grid('on')
legend(loc='best',fontsize=14)
savefig(path+'falc_h_dens2.png')
yscale('log')
savefig(path+'falc_h_dens3.png')
'''

################################################################################
# Gas pressure plotted against height
'''
for i in range(3): # because multiple plots
    pgas = pgasptot*ptot
    p_str = 'n_H + n_e'
    if i==0:
        prod = (nhyd + nel)*ssb.k_B*temp
        yl = r'gas pressure [dyn cm$^{-2}$]'
    elif i==1:
        prod = (nhyd + nel)*ssb.k_B*temp
        prod /= ptot
        pgas /= ptot
        yl = r'relative gas pressure'
    elif i==2:
        prod = (nhyd + nel + nhel)*ssb.k_B*temp
        prod /= ptot
        pgas /= ptot
        p_str = 'n_H + n_e + n_{He}'
        yl = r'relative gas pressure'
    figure()
    plot(h,pgas,lw=2,label='Gas pressure')
    plot(h,prod,lw=2,label=r'$(%s)kT$'%p_str)
    xlabel(r'height [km]',size=fs)
```

```
    ylabel(yl,size=fs)
    yscale('log')
    xscale('log')
    if i!=0:
        ylim([0.5,1.5])
    title('Plot of gas pressure against height',size=fs)
    legend(loc='best',fontsize=14)
    grid('on')
    savefig(path+'falc_h_pgas_%d.png'%i)
'''

##############################################################################
# Plot of total hydrogen density against height
'''
figure()
plot(h,nhyd,lw=2,label='hydrogen')
plot(h,nel,lw=2,label='all electrons')
plot(h,nprot,lw=2,label='protons')
plot(h,(nel-nprot),lw=2,label='electrons not from H')

xlabel('height [km]',size=fs)
ylabel(r'density [cm$^{-3}$]',size=fs)
legend(loc='best',fontsize=14)
grid('on')
savefig(path+'falc_h_n.png')
yscale('log')
savefig(path+'falc_h_n_log.png')
'''

##############################################################################
# Plot of the hydrogen ionization fraction, log
'''
figure()
plot(h,nprot/nhyd,lw=2)
xlabel('height [km]',size=fs)
ylabel('fraction of ionized hydrogen',size=fs)
title('Plot of fraction of ionized hydrogen',size=fs)
yscale('log')
grid('on')
savefig(path+'falc_h_Hion.png')
'''

##############################################################################
# do final part of 1.2
'''
# at deepest part of the atmosphere
```

```
print 'At h = %.3f :'%(h[-1]).value
print ' n_hydrogen = %.3e'%(nhyd[-1]).value
print ' n_photon   = %.3e'%(20*temp[-1]**3).value

# at highest part
print '\nAt h = %.3f :'%(h[0]).value
print ' n_hydrogen = %.3e'%(nhyd[0]).value
temp_eff = 5770*u.K
print ' n_photon   = %.3e'%(20*temp_eff**3/(2*pi)).value
'''

show()
```

4.3 Earth plotting

```
from ssb import *
# np, ap, u and pylab *

# some plotting parameters
path = '../figures/earth/' # folder where figures are stored
fs = 16  # font size

ssb = SSB()

h,logP,temp,logrho,logN = ssb.read_earth()


##############################################################################
# Plot temperature against height
'''
plot(h,temp,lw=2)
xlabel('height [km]',size=fs)
ylabel('temperature [K]',size=fs)
title('Earth atmosphere',size=fs)
grid('on')
savefig(path+'earth_h_temp.png')
'''

##############################################################################
# Particle density against height
'''
figure()
plot(h,10**(logN),lw=2)
xlabel('height [km]',size=fs)
ylabel(r'particle density [cm$^{-3}$]',size=fs)
title('Earth atmosphere',size=fs)
```

```
grid('on')
yscale('log')
savefig(path+'earth_h_N.png')
'''

##############################################################################
# Gas density against height
'''
figure()
plot(h,10**(logrho),lw=2)
xlabel('height [km]',size=fs)
ylabel('gas density [g cm$^{-3}$]',size=fs)
title('Earth atmosphere',size=fs)
grid('on')
yscale('log')
savefig(path+'earth_h_rho.png')
'''

##############################################################################
# Pressure against height
'''
figure()
plot(h,10**logP,lw=2)
xlabel('height [km]',size=fs)
ylabel(r'pressure [dyn/cm$^2$]',size=fs)
title('Earth atmosphere',size=fs)
grid('on')
yscale('log')
savefig(path+'earth_h_P.png')
'''

##############################################################################
# Pressure and density stratifications together in normalized units
'''
P = 10**(logP)
N = 10**(logN)

P /= max(P)
N /= max(N)

figure()
plot(h,P,lw=2,label='pressure')
plot(h,N,lw=2,label='density')
xlabel('height [km]',size=fs)
ylabel('pressure and density in normalized units',size=fs)
title('Earth atmosphere',size=fs)
```

```
grid('on')
savefig(path+'earth_h_PN.png')
'''

##############################################################################
# Mean molecular weight against height
'''
mu = 10**(logrho)/(10**(logN)*ssb.m_H)
figure()
plot(h,mu,lw=2)
xlabel('height [km]',size=fs)
ylabel('mean molecular weight',size=fs)
title('Earth atmosphere',size=fs)
grid('on')
savefig(path+'earth_h_mu.png')
'''

##############################################################################
# Estimate the density scale height of the lower terrestrial atmosphere
'''
rho = 10**logrho
rho1 = rho[0]/e
index = argmin(abs(rho-rho1))
height = h[index]-h[0]
print 'Density scale height = %.2f km'%height.value

# Mount Everest
h_ME = 8.848*u.km
rho_func = lambda h: rho[0]*exp(-h/height)
everest = argmin(abs(h-h_ME))
#print 'Everest to ground ratio = %.3f' %(rho[everest]/rho[0])
print 'Everest to ground ratio = %.3f' %(rho_func(h_ME)/rho[0]) # using function
print 'It is %.3f times harder to breath at Mount Everest than at the surface level\n'%(rho[0]/rho_fu
'''

##############################################################################
# Compare the terrestrial parameter values to the solar ones, at the base
# of each atmosphere. Ratio of the particle densities at h=0 in both.
'''
h_sun,tau5,colm_sun,temp_sun,vturb,nhyd,nprot,nel,ptot,pgasptot,dens = ssb.read_falc()
nhel = 0.1*nhyd
n_sun = nhyd+nel+nhel
print 'Ratio N_earth/N_sun = %.3f'%(10**logN[0]/n_sun[h_sun==0]).value
'''

##############################################################################
```

```
# Estimate atmospheric column mass at the Earth's surface and compare to the Sun
'''
colm = (10**logP*u.dyn*u.cm**(-2)/ssb.g_E).decompose().cgs
print '\nAtmospheric column mass at Earth surface = %.3f g cm^-2'%colm[h==0].value
print 'Column mass at base of stellar photosphere= %.3f g cm^-2'%colm_sun[h_sun==0].value
'''

##############################################################################
# Irradiance
'''
temp_eff = 5770*u.K
n_photon   = (20*temp_eff**3/(2*pi))#.value
D = ap.constants.au.cgs
R = ap.constants.R_sun.cgs
N_p_E = pi*R**2/D**2*n_photon

print '\nSunshine proton density = %.3e'%N_p_E.value
print 'Particle density in air = %.3e'%10**logN[h==0]
print 'Local thermal photon prod = %.3e'%(20*temp[h==0]**3).value
'''

show()
```

4.4 Exercise 2.1

```
from ssb import *

path = '../figures/task2/'
fs = 16

ssb = SSB()
wav,F, F_c, I, I_c = ssb.read_solspect()

##############################################################################
# Plot the four spectral distributions together in one figure lambda [0,2]
'''
wave = wav.value
figure()
plot(wav[wave<2.1],F[wave<2.1],lw=2,label=r'$F_{\lambda}$')
plot(wav[wave<2.1],F_c[wave<2.1],lw=2,label=r"$F_{\lambda}'$")
plot(wav[wave<2.1],I[wave<2.1],lw=2,label=r'$I_{\lambda}$')
plot(wav[wave<2.1],I_c[wave<2.1],lw=2,label=r"$I_{\lambda}'$")

xlabel('wavelength [micron]',size=fs)
```

```
ylabel(r'spectral distributions',size=fs)
title(r'Spectral distributions per $\Delta\lambda$',size=fs)
legend(loc='best',fontsize=15)
grid('on')
savefig(path+'solspect_lambda_spec_dist.png')
print 'max(I_c) = %.3e, at lambda = %.3f\n'%(max(I_c.value),wave[argmax(I_c.value)])
'''

###############################################################################
# Convert spectral distributions into values per frequency bandwidth
'''
nu = (ssb.c/wav).to('Hz')
F_nu  = F/nu*wav
F_cnu = F_c/nu*wav
I_nu  = I/nu*wav
I_cnu = I_c/nu*wav

figure()
plot(wav[wave<2.1],F_nu[wave<2.1],lw=2,label=r'$F_{\nu}$')
plot(wav[wave<2.1],F_cnu[wave<2.1],lw=2,label=r"$F_{\nu}'$")
plot(wav[wave<2.1],I_nu[wave<2.1],lw=2,label=r'$I_{\nu}$')
plot(wav[wave<2.1],I_cnu[wave<2.1],lw=2,label=r"$I_{\nu}'$")

xlabel('frequency [Hz]',size=fs)
ylabel(r'spectral distributions',size=fs)
title(r'Spectral distributions per $\Delta\nu$',size=fs)
legend(loc='best',fontsize=15)
grid('on')
savefig(path+'solspect_lambda_spec_dist_nu.png')
print 'max(I_c) = %.3e, at nu = %.3f\n'%(max(I_cnu.value),wave[argmax(I_cnu.value)])
'''

###############################################################################
# Trying to fit a Planck function  to the solar continuum intensity
'''
temp = linspace(4000,8000,200)*u.K
min_i = 0
delta = sum(abs(I_c[wave<2.1])**2) # starting value for finding optimal temp
planck_best = 0
for i in range(len(temp)):
    planck = ssb.planck(temp[i],wav[wave<2.1])
    #planck *= 1e-10   # scaling in units of 1e10 erg

    delta_current = sum(abs(planck-I_c[wave<2.1])**2)
    if delta_current < delta:
        min_i = i
```

```
        delta = delta_current
        planck_best = planck
print 'Best fit temperature, T[%d] = %.3f\n' %(min_i, temp[min_i].value)

figure()
plot(wav[wave<2.1],I_c[wave<2.1],lw=2,label=r"$I_{\lambda}'$")
plot(wav[wave<2.1],planck_best.to(I_c.unit),lw=2,label=r'Planck, T = %.1f K'%temp[min_i].value)

xlabel('wavelength [micron]',size=fs)
ylabel('spectral distributions',size=fs)
title(r'$B_{\lambda}$ fitted to the solar continuum intensity',size=fs)
legend(loc='best',fontsize=15)
grid('on')
savefig(path+'solspect_h_planck.png')
'''

###############################################################################
# Inverting the Planck function
'''
T_b = ssb.brightness_temp(I_c,wav)
figure()
plot(wav[wave<2.1],T_b[wave<2.1],lw=2,label='brightness temperature')
xlabel('wavelength [micron]',size=fs)
ylabel('brightness temperature [K]',size=fs)
title('Brightness temperature versus wavelength',size=fs)
grid('on')
savefig(path+'solspect_h_Tb.png')
'''

show()
```

4.5 Rest of exercise 2

```
from ssb import *
# np, ap, u and pylab *

# some plotting parameters
path = '../figures/task2/'
fs = 16  # font size

ssb = SSB()

h,tau5,colm,temp,vturb,nhyd,nprot,nel,ptot,pgasptot,dens = ssb.read_falc()
nhel = 0.1*nhyd
wav,F, F_c, I, I_c = ssb.read_solspect()
wave = wav.value
```

```
################################################################################
'''
temp_b = ssb.brightness_temp(I_c,wav)
kappa = ssb.exthmin(wav, temp_b, nel[h==0])

plot(wav[wave<2.1],kappa[wave<2.1],lw=2)
xlabel('wavelength [micron]',size=fs)
ylabel(r'H$^-$ extinction [dyn cm$^{-2}$]',size=fs)
title('Extinction profile versus wavelength',size=fs)
grid('on')
#ylim([1.0e-25,1.8e-25])
savefig(path+'exthmin_lam_kappa.png')
'''

################################################################################
# height dependent extinction
'''
kappa = ssb.exthmin(wav[wave==0.5],temp,nel)
# Make it not per neutral hydrogen
n_neutral = nhyd-nprot
kappa *= n_neutral

figure()
plot(h,kappa,lw=2)
xlabel('height [km]',size=fs)
ylabel('extinction ',size=fs)
title(r'Height dependent extinction, $\lambda=0.5\mu m$',size=fs)
yscale('log')
grid('on')
xlim([-100,2100])
savefig(path+'exthmin_2.png')
'''

################################################################################
# Thompson scattering
'''
kappa2 = kappa.value + (ssb.sigma_T*nel).value

figure()
plot(h,kappa,lw=2,label=r'H$^-$')
plot(h,ssb.sigma_T*nel,lw=2,ls='--',label='Thompson')
plot(h,kappa2,lw=2,label='total')
xlabel('height [km]',size=fs)
ylabel('extinction ',size=fs)
title(r'Height dependent extinction, $\lambda=0.5\mu m$',size=fs)
```

```
legend(loc='best',fontsize=16)
yscale('log')
xlim([-100,2100])
ylim([1e-18,1e-5])
grid('on')
savefig(path+'exthmin_3.png')
'''

##############################################################################
# Integration
'''
tau = ssb.optical_depth(h, temp, nel, n_neutral)

figure()
plot(h,tau5,lw=2,label=r'$\tau_{500}$ from FALC')
plot(h,tau,lw=2,label=r'$\tau_{500}$ from H$^-$ and Thompson extinction')
xlabel('height [km]',size=fs)
ylabel('opacity',size=fs)
title('FALC',size=fs)
legend(loc='best',fontsize=14)
yscale('log')
grid('on')
savefig(path+'exthmin_h_tau.png')
'''

##############################################################################
# Emergent intensity and height of formation
'''
wl = 0.5*u.micron
intt, contfunc, hmean = ssb.emergent_intensity(h,tau5,temp,nhyd,nprot,nel,wl)

print ('computed continuum intensity wl =%g : %g erg s-1 cm-2 ster-1 cm-1'%(wl.value, intt.value))
w = np.where(wav.value == wl.value)
print ('observed continuum intensity wav=%g : %g erg s-1 cm-2 ster-1 cm-1'%(wav[w].value, I_c[w].valu
'''

##############################################################################
# Plot one wl
'''
figure()
contfunc /= max(contfunc.value)
plot(h,contfunc,lw=2)
plot([hmean.value], [contfunc[argmin(abs((h-hmean).value))].value], marker='o',ms=9,color='blue')
xlabel('height [km]',size=fs)
ylabel('contribution function',size=fs)
title('FALC')
```

```
grid('on')
xlim([-100,500])
annotate('mean height of formation = %.1f km'%hmean.value, (10,0.83),size=15)
savefig(path+'emergent_intensity_1.png')
'''

##############################################################################
# Plot multiple wl
'''
cl = ['blue','red','green','cyan']
figure()

for wl,i in zip([0.5, 1.0, 1.6, 5.0],range(4)):
    intt, contfunc, hmean = ssb.emergent_intensity(h,tau5,temp,nhyd,nprot,nel,wl*u.micron)
    contfunc /= max(contfunc.value)
    plot(h, contfunc/max(contfunc.value),lw=2,color=cl[i],label=r'$\lambda$=%.1f $\mu$m, $h_{mean}$=%
    plot([hmean.value], [contfunc[argmin(abs((h-hmean).value))].value], marker='o',ms=9,color=cl[i])

xlabel('height [km]',size=fs)
ylabel('contribution function',size=fs)
title('FALC',size=fs)
legend(loc='best',fontsize=14)
xlim([-100,500])
grid('on')
savefig(path+'emergent_intensity_2.png')
'''


##############################################################################
# Disk-center intensity
'''
em_int = np.zeros(len(wave))
for i in range(len(wave)):
    intt, contfunc, hmean = ssb.emergent_intensity(h,tau5,temp,nhyd,nprot,nel,wav[i])
    em_int[i] = intt.value

figure()
plot(wav,em_int/1e14,lw=2,label='computed from FALC')
plot(wav,I_c/1e10,lw=2,label='observed (Allen 1978)')
xlabel(r'wavelength [$\mu$m]',size=fs)
ylabel(r'intensity [$10^{14}$ erg s$^{-1}$ cm$^{-2}$ ster$^{-1}$ cm$^{-1}$]',size=fs)
title('Observed and computed continuum intensity',size=fs)
legend(loc='best',fontsize=15)
grid('on')
savefig(path+'disc_center.png')
'''
```

```
##############################################################################
# Limb darkening
'''
figure()
mu = [0.1, 0.3, 0.5, 0.7, 0.9, 1.0]
int_mu = np.zeros(len(wave))
for i in np.arange(len(mu)):
    for j in np.arange(len(wave)):
        intt, contfunc, hmean = ssb.emergent_intensity(h,tau5,temp,nhyd,nprot,nel,wav[j],mu[i])
        int_mu[j] = intt.value

    plot(wav,int_mu,lw=2,label=r'$\mu$=%.1f'%mu[i])

xlabel(r'wavelength [$\mu$m]',size=fs)
ylabel(r'intensity [erg s$^{-1}$ cm$^{-2}$ ster$^{-1}$ cm$^{-1}$]',size=fs)
title('Limb darkening in FALC',size=fs)
legend(loc='best',fontsize=15)
grid('on')
savefig(path+'limb_darkening_1.png')
'''

##############################################################################
# Plot 2
'''
figure()
wav = array([0.2, 0.5, 1.6, 2.5, 5.0])*u.micron
sin_theta = h/ap.constants.R_sun
theta = arcsin(sin_theta)
mu = cos(theta)

int_mu = np.zeros(len(mu))
for i in range(len(wav.value)):
    intt, contfunc, hmean = ssb.emergent_intensity(h,tau5,temp,nhyd,nprot,nel,wav[i],mu)
    int_mu = intt.value

    plot(wav,int_mu,lw=2,label=r'$\lambda$=%.1f $\mu$m'%wav[i].value)

xlabel(r'$\sin(\theta) = r/R_{\odot}$',size=fs)
ylabel(r'intensity [erg s$^{-1}$ cm$^{-2}$ ster$^{-1}$ cm$^{-1}$]',size=fs)
title('Limb darkening in FALC',size=fs)
legend(loc='best',fontsize=15)
grid('on')
savefig(path+'limb_darkening_2.png')
'''
```

```
################################################################################
# Flux integration
'''
fluxspec = ssb.flux_integration(h,tau5,temp,nhyd,nprot,nel, wav)

figure()
plot(wav,fluxspec*1e-14,lw=2,label='computed from FALC')
plot(wav,F_c*1e-10,lw=2,label='observed (Allen 1978')
xlabel('wavelength [micron]',size=15)
ylabel(r'astrophysical flux [10$^{14}$ erg s$^{-1}$ cm$^{-2}$ ster$^{-1}$ cm$^{-1}$]',size=15)
title('Observed and computed continuum flux',size=fs)
grid('on')
legend(loc='best',fontsize=15)
savefig(path+'flux.png')
'''

show()
```

4.6 Exercise 3

```
from ssb import *

# some plotting parameters
path = '../figures/task3/'
fs = 16  # font size

ssb = SSB()

h,tau5,colm,temp,vturb,nhyd,nprot,nel,ptot,pgasptot,dens = ssb.read_falc()
nhel = 0.1*nhyd
wav,F, F_c, I, I_c = ssb.read_solspect()
wave = wav.value

sigma, wav_vac, lines = ssb.read_NaID()

################################################################################
# Plot solar NaID lines against vacuum wavelength at various dispersions
wav_vac = wav_vac.to('AA')
'''
figure()
plot(wav_vac,lines)#,lw=2)
xlabel(r'vacuum wavelength [$\AA$]',size=fs)
#ylabel(r'wavenumber $\sigma$ [cm$^{-1}$]',size=fs)
title('Solar NaID lines against vacuum wavelength',size=fs)
grid('on')
xlim([min(wav_vac.value),max(wav_vac.value)])
```

```
ylim([0.0,1.07])
savefig(path+'NaID_wav_vac_sigma.png')
'''

# Find the minimas for the NaID lines
ind = argmin(abs(5895-wav_vac.value))
min1 = wav_vac[argmin(lines[0:ind])]
min2 = wav_vac[ind-1+argmin(lines[ind-1:])]
print min2, min1
print ssb.vactoair(min2.value), ssb.vactoair(min1.value)

# Plot NaID lines against air wavelength
wav_air = ssb.vactoair(wav_vac.value)
wav_air *= u.AA

'''
figure()
plot(wav_air,lines)#,lw=2)
xlabel(r'air wavelength [$\AA$]',size=fs)
#ylabel(r'wavenumber $\sigma$ [cm$^{-1}$]',size=fs)
title('Solar NaID lines against air wavelength',size=fs)
grid('on')
xlim([min(wav_air.value),max(wav_air.value)])
ylim([0.0,1.07])
savefig(path+'NaID_wav_air_sigma.png')

show()
'''

##############################################################################
# Check Saha-Boltzmann functions    WORKS
'''
saha = zeros(len(temp))
boltz = zeros(len(temp))
figure()
for s in range(1,4):
    for i in range(len(temp)):
        boltz[i] = ssb.boltz_Na(temp[i].value, s)
    plot(h,boltz,lw=2)
title('Boltzmann distribution of Na I in FALC',size=fs)
xlabel('height [km]',size=fs)
ylabel('population fraction n_1,s/N_1',size=fs)
xlim([-100,2000])
grid('on')
savefig(path+'boltzmann.png')
figure()
```

```
for r in range(1,3):
    for i in range(len(temp)):
        saha[i]  = ssb.saha_Na(temp[i].value, nel[i].value,r)
    plot(h,saha,lw=2)
title('Saha distribution of Na in FALC',size=fs)
xlabel('height [km]',size=fs)
ylabel('ionization fraction N_r/N_total',size=fs)
xlim([-100,2000])
yscale('log')
ylim([1e-4,1e1])
grid('on')
savefig(path+'saha.png')
'''

################################################################################
# NaI D1 line D:

lambda0 = ssb.vactoair(min1.value)#5895.94
wav = linspace(lambda0-2,lambda0+2,250)
I = zeros(len(wav))

h = h.value
temp = temp.value
eldens = nel.value
nhyd = nhyd.value
vmicro = (vturb).value
pgas = (pgasptot*ptot).value

for i in range(len(wav)):
    I[i] = ssb.emergent_intensity_Na(h,tau5,temp,nhyd,nprot.value,eldens,wav[i]*1e-4,vmicro,pgas)

figure()
plot(wav,I/max(I),lw=2)
plot(wav_air,lines/max(lines),'--',lw=2)
xlim([wav[0],wav[-1]])
xlabel(r'wavelength [$\AA$]',size=fs)
ylabel('normalized intensity',size=fs)
title('Na I D1 in LTE in FALC',size=fs)
grid('on')
savefig(path+'Na_D1.png')

show()
```