Lab-B2

Matt Wilde

March 9, 2016

2 Continuous spectrum from the solar atmosphere

2.1 Observed solar continua

In [1]:
```
# first import everything
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import astropy.constants as const

#use the pretty LaTeX fonts
mpl.rcParams.update({'text.usetex': True})
plt.rc('font', family='serif', size=15)
#mpl.rc('axes.formatter', useoffset=False)

# plt.style.use('ggplot')
```

In [2]:
```
# define constants
h = const.h.cgs.value
c = const.c.cgs.value
k = const.k_B.cgs.value
```

In [3]:
```
! head solspect.dat
```

```
0.20  0.02  0.04  0.03  0.04
  0.22  0.07  0.11  0.14  0.20
  0.24  0.09  0.2   0.18  0.30
  0.26  0.19  0.4   0.37  0.5
  0.28  0.35  0.7   0.59  1.19
  0.30  0.76  1.36  1.21  2.15
  0.32  1.10  1.90  1.61  2.83
  0.34  1.33  2.11  1.91  3.01
  0.36  1.46  2.30  2.03  3.20
  0.37  1.57  2.50  2.33  3.62
```

- **Write IDL code to read Table 5.**

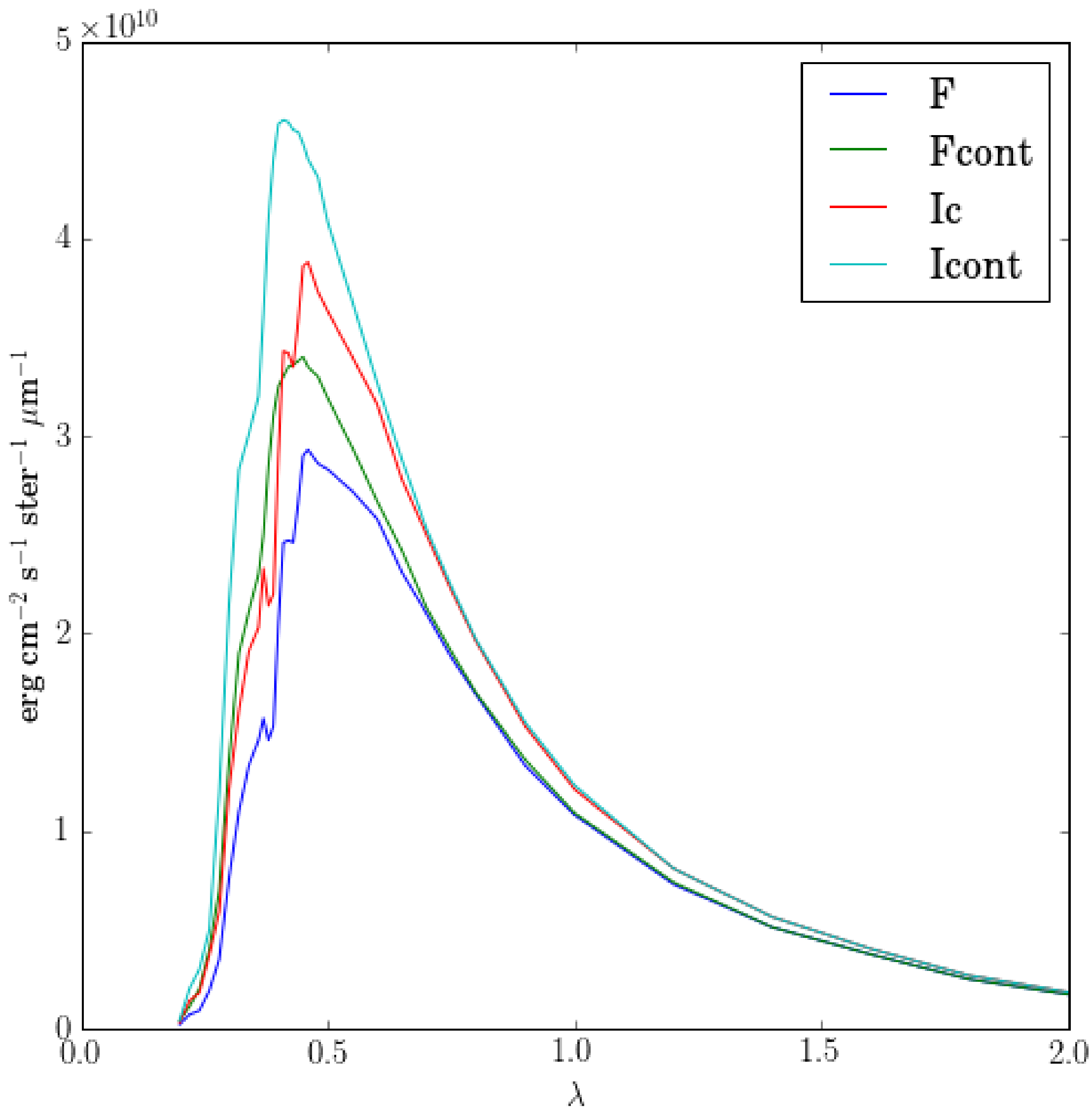
  Units are:
  wave = μm
  else: 10^{10} erg cm^{-2} s^{-1} μm^{-1} ster^{-1}

In [4]:
```
wave, F, Fcont, I, Icont = np.loadtxt('solspect.dat', unpack=True)
```

```
#convert
F = F*1e10
Fcont = Fcont*1e10
I = I*1e10
Icont = Icont*1e10

#convert to Hz
F_nu = F*1/(c/(wave)**2)*1e-4
Fcont_nu = Fcont*1/(c/(wave)**2)*1e-4
I_nu = I*1/(c/(wave)**2)*1e-4
Icont_nu = Icont*1/(c/(wave)**2)*1e-4
```

- **Plot the four spectral distributions together in one figure over the range $\lambda = 0 - 2\mu$m. Use a statement such as** `print, max(Ic) = ,max(Icont), ' at ',wav[where(Icont eq max(Icont))]` **to check that the continuum intensity reaches $Ic = 4.6 \times 10^{10}$ erg cm^{-2} s^{-1} ster^{-1} μm^{-1} at $\lambda = 0.41\mu$m.**

In [5]:
```
fig, ax = plt.subplots(figsize=(8,8))

ax.plot(wave, F, label='F')
ax.plot(wave, Fcont, label=r'Fcont$')
ax.plot(wave, I, label='Ic')
ax.plot(wave, Icont, label='Icont')

ax.set_xlim(0,2)

flux_unit = r'erg cm$^{-2}$ s$^{-1}$ ster$^{-1}$ $\mu$m$^{-1}$'
wave_unit = r'$\lambda$'
ax.set_xlabel(wave_unit)
ax.set_ylabel(flux_unit)

plt.legend()
plt.show()
```




```
In [6]: max_wave = wave[np.where(Icont == Icont.max())][0]
        max_index = np.where(Icont == Icont.max())[0]

In [7]: print 'max(Ic) = max(Icont) at {0:3} micrometers'.format(max_wave)

max(Ic) = max(Icont) at 0.41 micrometers

In [8]: print "Icont at 0.41 microns = {:.2e}".format(Icont[max_index][0])

Icont at 0.41 microns = 4.60e+10
```

- **Explain why the four distributions share the same units and discuss the differences between them.**

The astrophysical fluxes F defined by $\mathcal{F} = \pi F$ share the same units as intensity as they are the emergent intensity averaged over the disk. This flux differs from the measured flux $\mathcal{F}$ by π, where the π removes the steradians from the units.

Since $F =< I >$, F is generally lower than I because of limb darkening on the edges of the solar disk wich gets averaged into the I at the center.

The flux and intensity of the continuum are higher than the smoothed F and I, that is $Fcont > F$ and $Icont > I$ since when smoothing over the lines, the lines would pull the average down.

- **Convert these spectral distributions into values per frequency bandwidth $\Delta\nu = 1$ Hz . Plot these also against wavelength**

$$\lambda = c/\nu$$

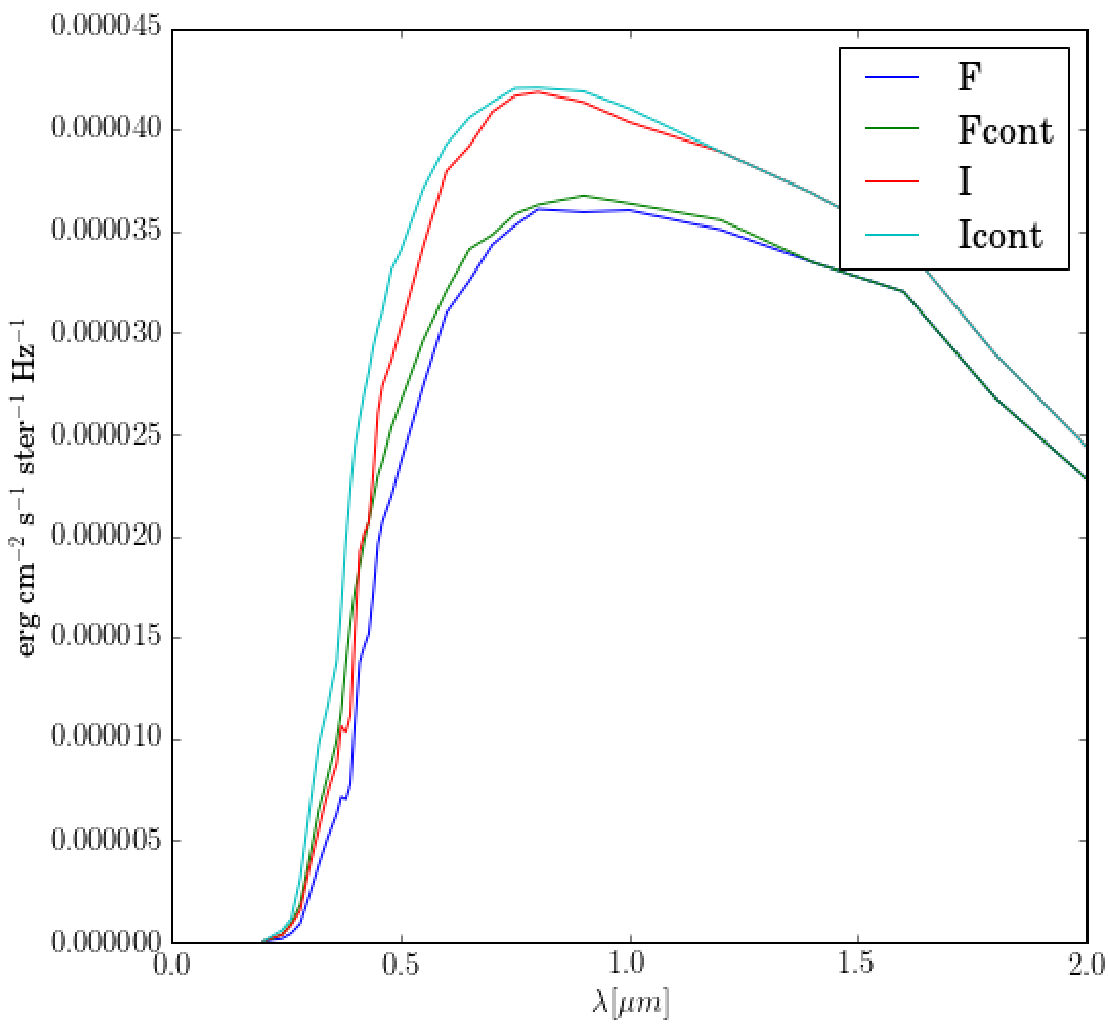
$$d\lambda = c/\nu^2 d\nu$$

$$d\nu = c/\lambda^2 d\lambda$$

$$F_\nu = F_\lambda d\lambda/d\nu$$

$$F_\nu = F_\lambda \lambda^2/c$$

In [9]:
```
fig, ax = plt.subplots(figsize=(8,8))

ax.plot(wave, F_nu, label='F')
ax.plot(wave, Fcont_nu, label='Fcont')
ax.plot(wave, I_nu, label='I')
ax.plot(wave, Icont_nu, label='Icont')

ax.set_xlim(0,2)

flux_unit = r'erg cm$^{-2}$ s$^{-1}$ ster$^{-1}$ Hz$^{-1}$'
wave_unit = r'$\lambda [\mu m]$'
ax.set_xlabel(wave_unit)
ax.set_ylabel(flux_unit)

plt.legend()
plt.show()
```



- **Check:** $I_\nu = 4.21 \times 10^{-5}$ **erg cm^{-2} s^{-1} ster^{-1} Hz^{-1} at** $\lambda = 0.8\mu$**m.**

```
In [10]: I_nu[np.where(wave == 0.8)]
```

```
Out[10]: array([  4.18422801e-05])
```

- **Write an IDL function** `planck.pro` **(or use your routine from Exercises "Stellar Spectra A: Basic Line Formation", or use mine) that computes the Planck function in the same units. Try to fit a Planck function to the solar continuum intensity. What rough temperature estimate do you get?**

Use the astropy units package to make sure my units are correct and to test it out.

```
In [11]: import astropy.units as u
         def planck(temp, wavs):
             wav = wavs*u.micrometer # to cm
             c = const.c.cgs
             h = const.h.cgs
             k = const.k_B.cgs
```

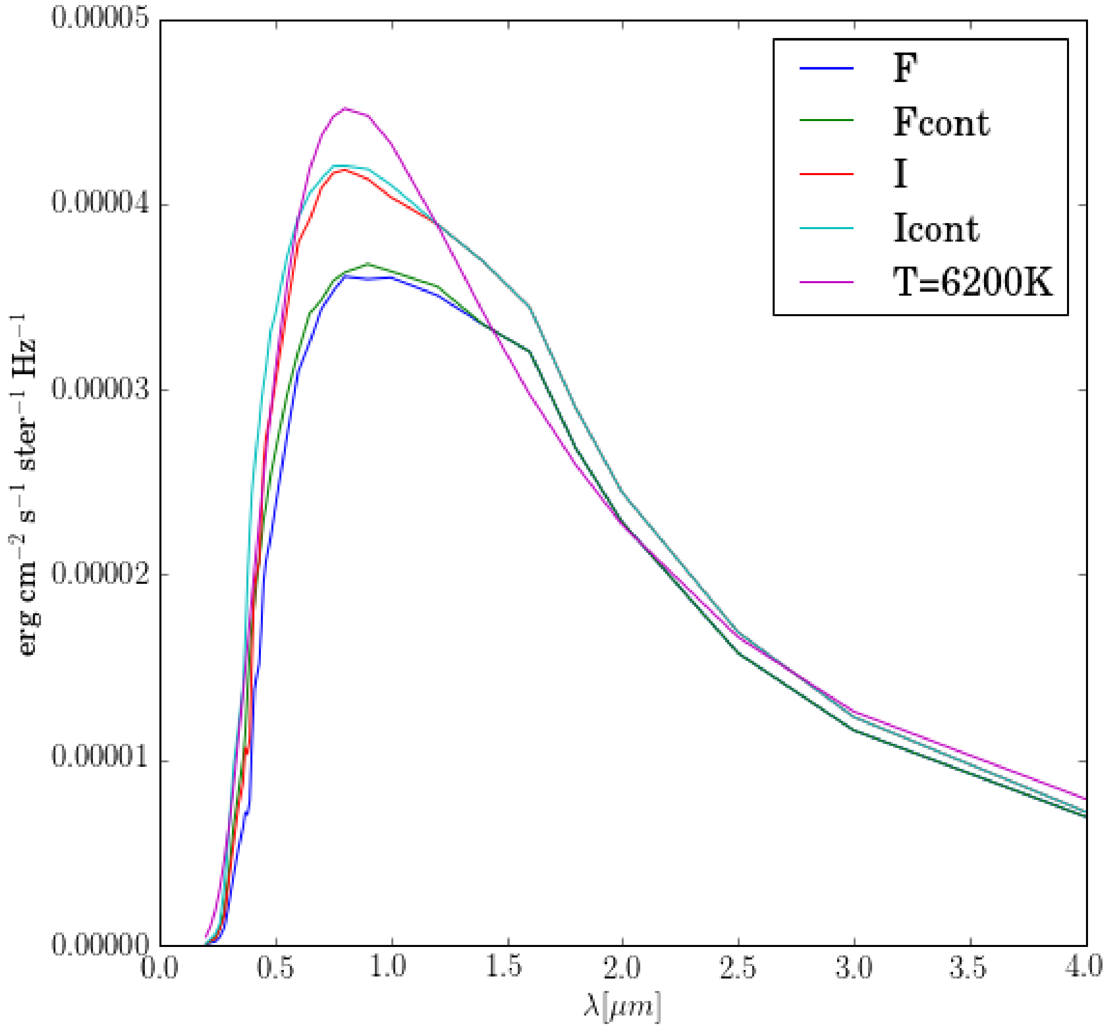
```
    temp = temp*u.K
    b = (2*h*c**2/wav**5) / (np.exp(((h*c)/(wav*k*temp))) - 1)

    return b.to(u.erg / u.cm**2 / u.s / u.Hz,
                equivalencies=u.spectral_density(wav))
```

In [12]:
```
planck(6200,0.8)
```

Out[12]:

$$4.5145277 \times 10^{-5} \ \frac{\mathrm{erg}}{\mathrm{Hz\,s\,cm^{2}}}$$

In [13]:
```
fig, ax = plt.subplots(figsize=(8,8))

ax.plot(wave, F_nu, label='F')
ax.plot(wave, Fcont_nu, label='Fcont')
ax.plot(wave, I_nu, label='I')
ax.plot(wave, Icont_nu, label='Icont')
ax.plot(wave, planck(6200, wave), label='T=6200K')

ax.set_xlim(0,4)

flux_unit = r'erg cm$^{-2}$ s$^{-1}$ ster$^{-1}$ Hz$^{-1}$'
wave_unit = r'$\lambda [\mu m]$'
ax.set_xlabel(wave_unit)
ax.set_ylabel(flux_unit)

plt.legend()
plt.show()
```



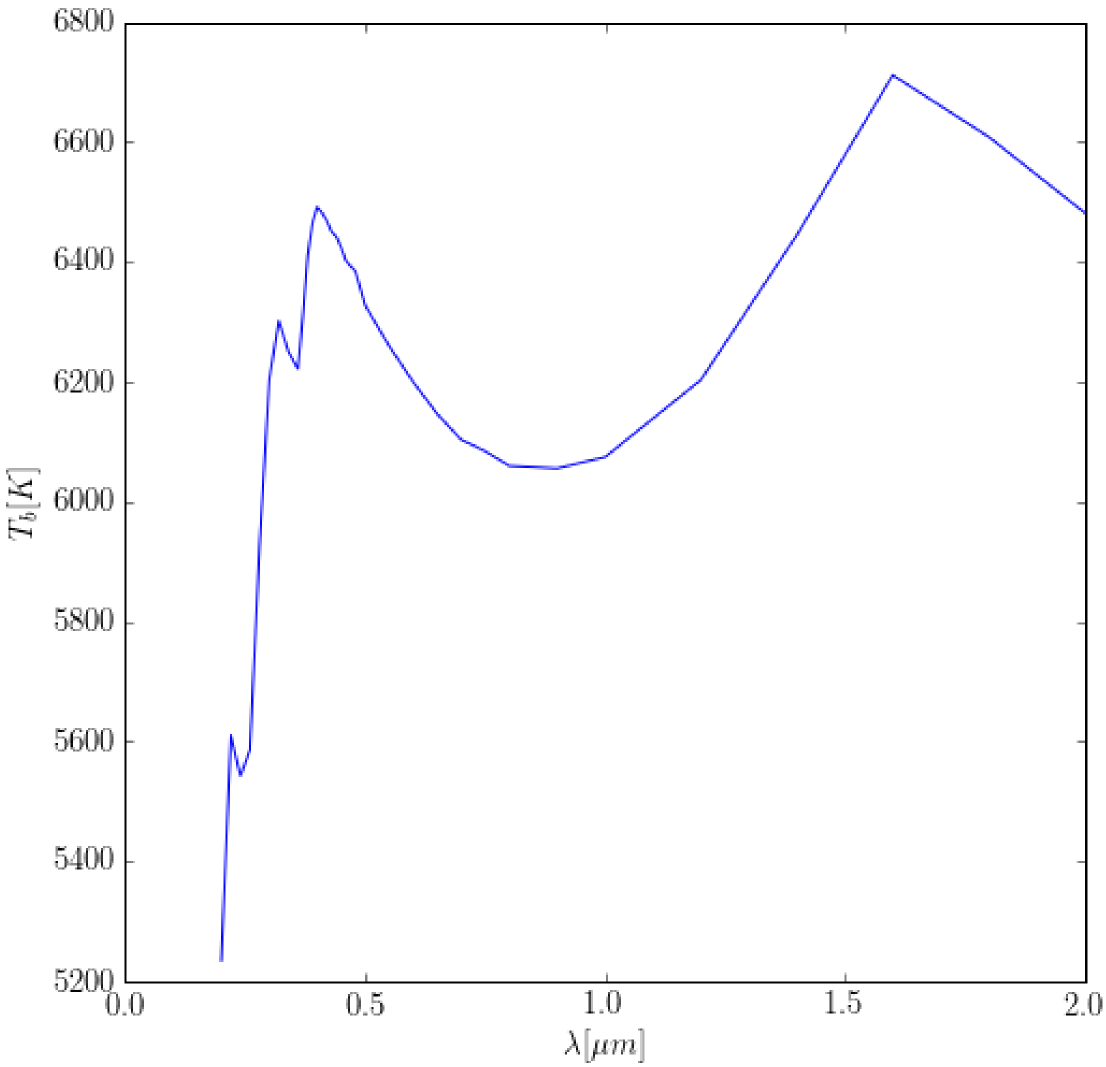
I found the temperature to be $T \approx 6200$K

- **Invert the Planck function analytically to obtain an equation which converts an intensity distribution I into brightness temperature T_b (defined by $B(T_b) = I$). Code it as an IDL function and use that to plot the brightness temperature of the solar continuum against wavelength (with the `plot,/ynozero` keyword).**

$$T_b = \frac{hc}{\lambda k \ln(\frac{2hc^2}{B\lambda^5} + 1)}$$

In [14]:
```
def inv_planck(wavs, B):
    wav = (wavs*u.micrometer).cgs # to cm
    c = const.c.cgs
    h = const.h.cgs
    k = const.k_B.cgs
    B = B*u.erg / u.cm**2 / u.s / u.micrometer
    t = h*c / (wav*k*np.log((2*h*c**2)/(B*wav**5)+1))
    return t
```

```
In [15]: fig, ax = plt.subplots(figsize=(8,8))
         ax.plot(wave, inv_planck(wave,Icont))

         ax.set_xlim(0,2)

         flux_unit = r'$T_{b} [K]$'
         wave_unit = r'$\lambda [\mu m]$'
         ax.set_xlabel(wave_unit)
         ax.set_ylabel(flux_unit)

         plt.show()
```



Discuss the shape of this curve. It peaks near $\lambda = 1.6\mu m$. What does that mean for the radiation escape at this wavelength?

The curve has two peaks, $\lambda = 1.6\mu m$ and $\lambda = 0.4\mu m$ and a local minimum around $\lambda = 0.8\mu m$.

The radiation at $\lambda = 1.6\mu m$ is escaping from an area in the atmosphere with higher temperature probably deeper in the stellar atmosphere, hence why it has a hotter T_b. Thus the extinction here must be lower than around $0.8\mu m$

2.2 Continuous extinction

bound-free absorption:

$$H^- + h\nu \rightarrow H + e^-(v)$$

free-free absorption:

$$H^- + e^-(v) + h\nu \rightarrow H^- + e^-(v)$$

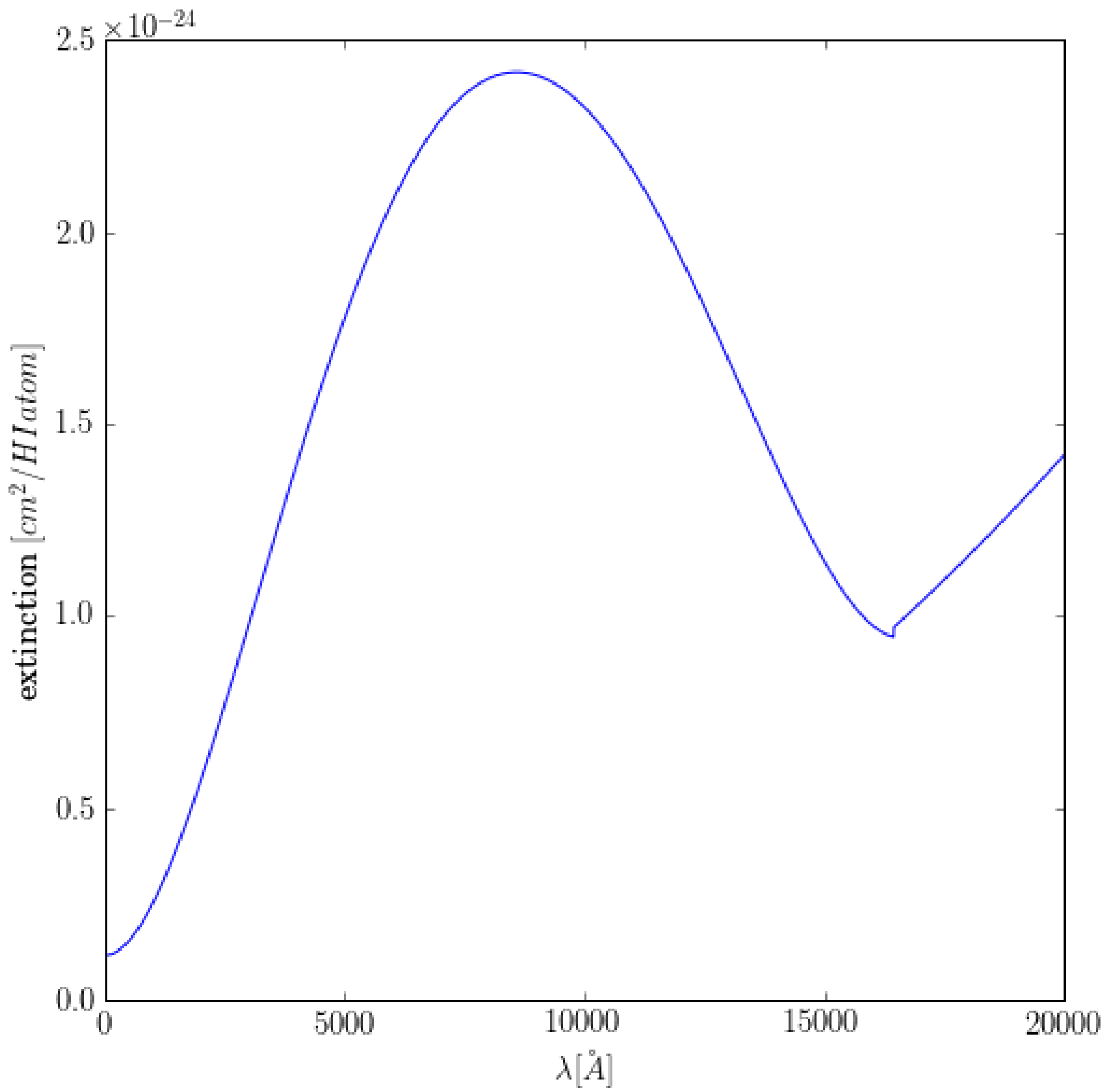
```
In [16]: def exthmin(wavs,temps,eldens):
             """
             in:  wav = wavelength [Angstrom] (number or array but then others number)
                  temp = temperature [K]  (number or array)
                  eldens = electron density [electrons cm-3] (number or array)
             out: H-minus bf+ff extinction [cm^2 per neutral hydrogen atom]
                  assuming LTE ionization H/H-min
                  NB: includes stimulated emission correction already!
             """

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

             # 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*wavs +2.64243e-6*wavs**2 -4.40524e-10*wavs**3 \
                     +3.23992e-14*wavs**4 -1.39568e-18*wavs**5 + 2.78701e-23*wavs**6
             sigmabf=sigmabf*1e-18  # cm^2 per H-min ion

             if type(wavs) == np.ndarray:
                 sigmabf[wavs > 16444] = 0. # H-min ionization limit
             else:
                 if wavs > 16444:
                     sigmabf = 0.0

             # 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*np.power(theta,2.5)*np.power(10.,(0.754*theta)) # Gray (8.12)
             kappabf=sigmabf*graysaha  # per neutral H atom
             kappabf=kappabf*(1.-np.exp(-h*c/(wavs*1.0e-8*k*temps)))
             # correct stimulated emission

             # check Gray's Saha-Boltzmann with AFYC (edition 1999) p168
             # logratio=-0.1761-alog10(elpress)+alog10(2.)+2.5*alog10(temp)-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(wavs)
             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 = 1.0e-26*elpress*np.power(10,(f0+f1*ltheta+f2*ltheta**2))   # Gray (8.13)

             return kappabf+kappaff
```

In [17]:
```
# read in the FALC data
height, tau5, colm, temp, vturb, n_Htotal, n_proton, \
    nel, ptot, pgasptot, dens = np.loadtxt('../HW4/falc.dat',skiprows=4, unpack=True)
```

- **Plot the wavelength variation of the H^- extinction for the FALC parameters at h = 0 km (see Tables 3-4) This plot reproduces the result of Chandrasekhar & Breen (1946).**

In [18]:
```
edense = nel[np.where(height == 0)]
T = temp[np.where(height == 0)]
wav = np.linspace(0.1,20000,20000)
```

In [19]:
```
fig, ax = plt.subplots(figsize=(8,8))

ax.plot(wav, exthmin(wav, T, edense))

ax.set_xlabel(r'$\lambda [\AA]$')
ax.set_ylabel(r'extinction $[cm^2/HI atom]$')
plt.show()
```



- **Compare it to Gray's version in Figure 5.**

This extinction curve of H^- has the same broad shape as Grays (figure 5) but doesnt include the bound-free HI extinctions.

- **Hydrogenic bound-free edges behave just as HI with maximum extinction at the ionization limit and decay $\sim \lambda^3$ for smaller wavelengths, as indeed shown by the HI curve in Figure 5. The H^- bound-free extinction differs strongly from this pattern. Why is it not hydrogenic although due to hydrogen?**

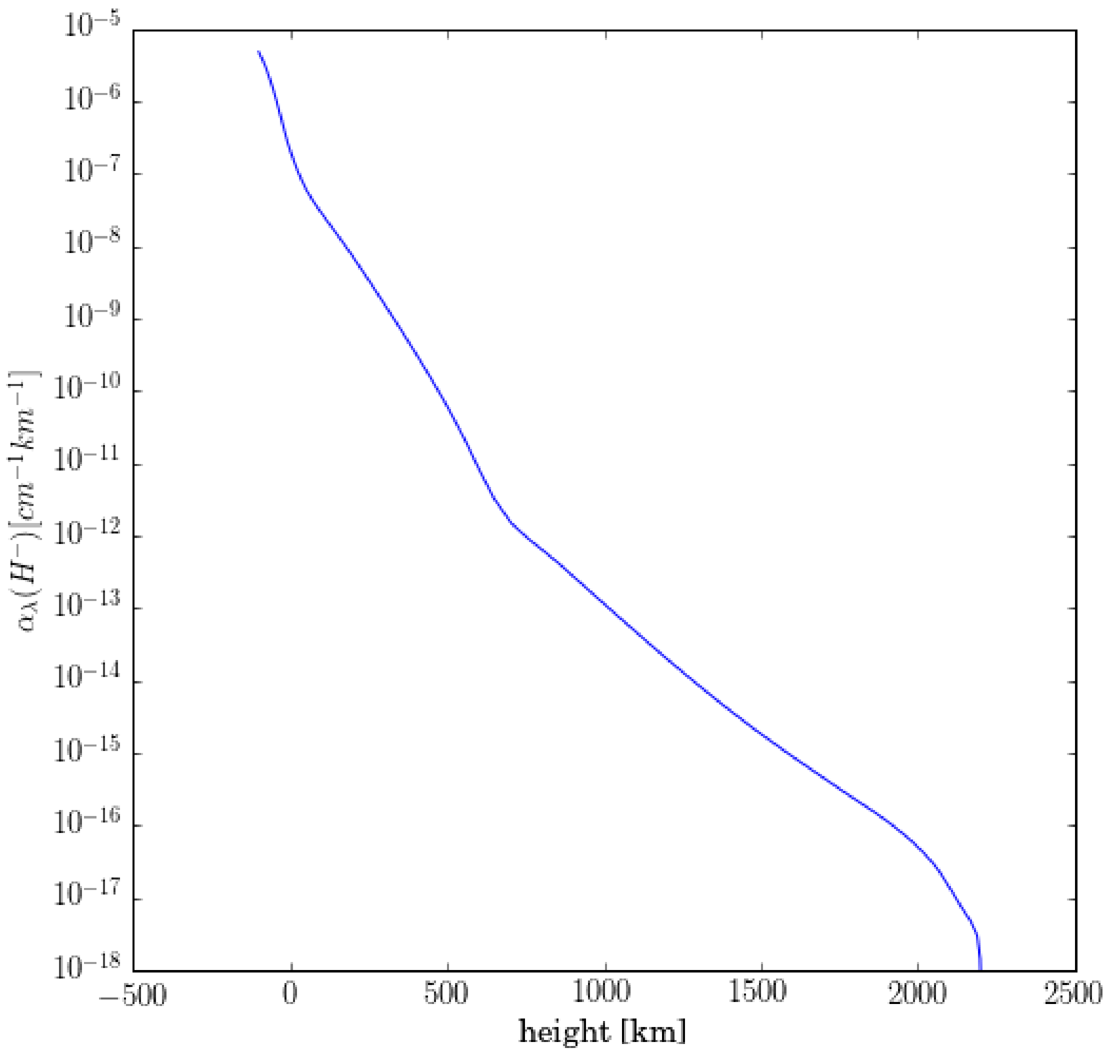
H^- extinction doesn't have this pattern as interactions with the other electron make in un-hydrogenic.

- **How should you plot this variation to make it look like the solar brightness temperature variation with wavelength? Why?**

By taking the inverse of the extinction curve it would roughly look like the brightness temperature T_b curve. This is because peaks on the brightness temperature are due to the troughs in the extinction curve, as the wavelengths there are probing deeper into the atmosphere.

- **Read in the FALC model atmosphere (see the previous exercise). Note that column nH in the FALC model of Tables 3–4 is the total hydrogen density, summing neutral atoms and free protons (and H2 molecules but those are virtually absent). This is seen by inspecting the values of nH and np at the top of the FALC table where all hydrogen is ionised. Gray's H^- extinction is measured per neutral hydrogen atom, so you have to multiply the height-dependent result from `exthmin(wav,temp,eldens)` with $n_{neutralH} \approx nH(h) - np(h)$ to obtain extinction $\alpha_\lambda(H^-)$ measured per cm path length (or crosssection per cubic cm) at every height h. Plot the variation of the H^- extinction per cm with height for $\lambda = 0.5\mu m$. This plot needs to be logarithmic in y, why?**

In [20]:
```
n_neutralH = n_Htotal - n_proton
alpha_Hminus = exthmin(0.5e4, temp, nel)*n_neutralH

fig, ax = plt.subplots(figsize=(8,8))

ax.semilogy(height, alpha_Hminus)
#ax.plot(height, alpha_Hminus)
ax.set_xlabel(r'height [km]')
ax.set_ylabel(r'$\alpha_{\lambda}(H^-) [cm^{-1}km^{-1}]$')
ax.set_ylim(1e-18, 1e-5)
plt.show()
```



- **This plot needs to be logarithmic in y, why?**

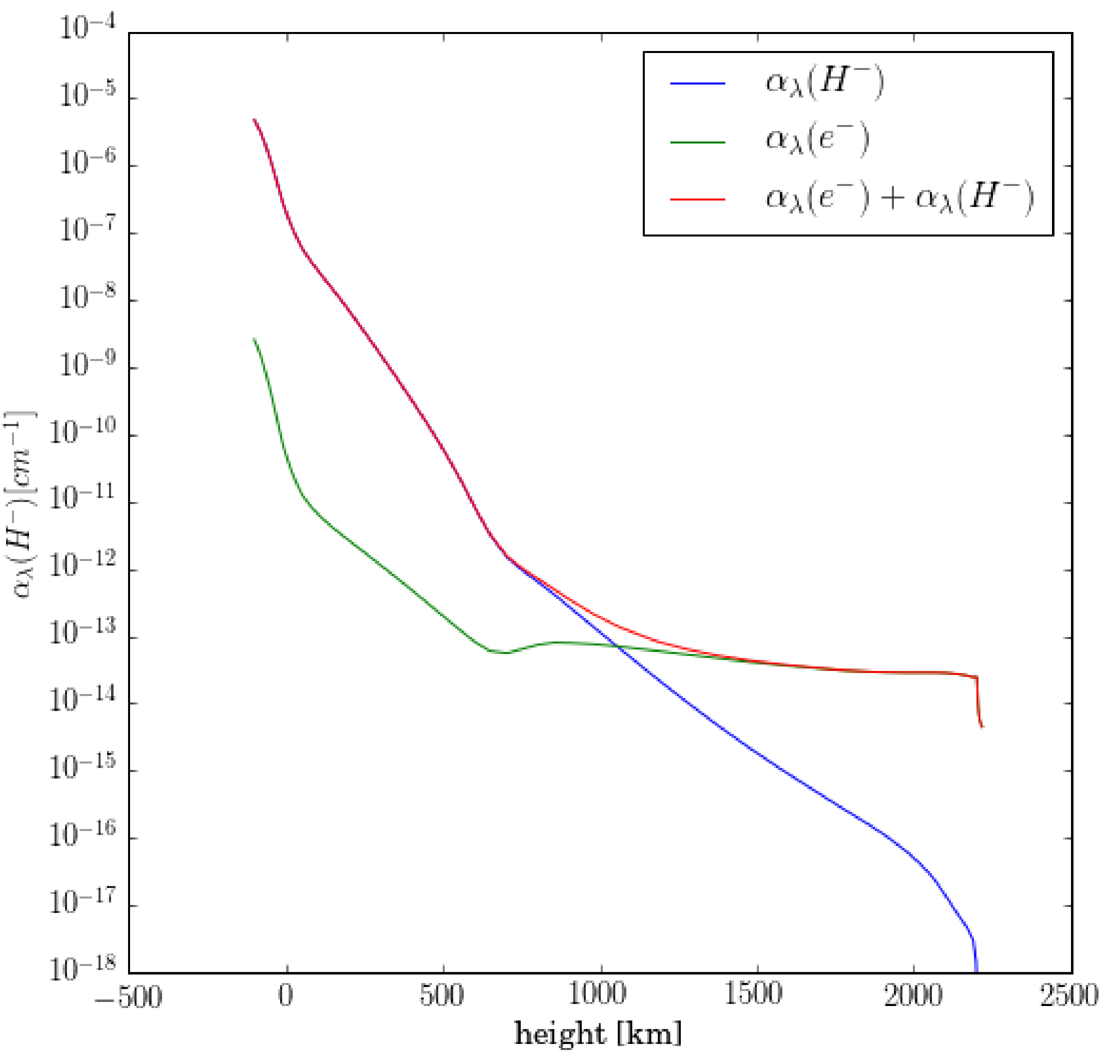
It needs to be logarithmic in y because the extinciton spans ~ 12 orders of magnitude over the 2000km.

- **Now add the Thomson scattering off free electrons to the extinction per cm. The Thomson crosssection per electron is the same at all wavelengths and is given by $\sigma_T = 6.648 \times 10^{-25} cm^2$ With which height-dependent quantity do you have to multiply this number to obtain extinction per cm? Overplot this contribution to the continuous extinction $\alpha^c_\lambda(h)$ in your graph and then overplot the total continuous extinction too. Explain the result.**

Need to multiply σ_T by $n_{electron}$ to obtain extinction per cm.

In [21]:
```
thomson = 6.648e-25
alpha_el = thomson*nel

fig, ax = plt.subplots(figsize=(8,8))

label1 = r'$\alpha_{\lambda}(H^-)$'
```

```
label2 = r'$\alpha_{\lambda}(e^-)$'
label3 = r'$\alpha_{\lambda}(e^-) + \alpha_{\lambda}(H^-)$'

ax.semilogy(height, alpha_Hminus, label=label1)
ax.semilogy(height, alpha_el, label=label2)
ax.semilogy(height, alpha_Hminus+alpha_el, label=label3)
ax.set_xlabel(r'height [km]')
ax.set_ylabel(r'$\alpha_{\lambda}(H^-) [cm^{-1}]$')
ax.set_ylim(1e-18,1e-4)
ax.legend()
plt.show()
```



- **Explain:**

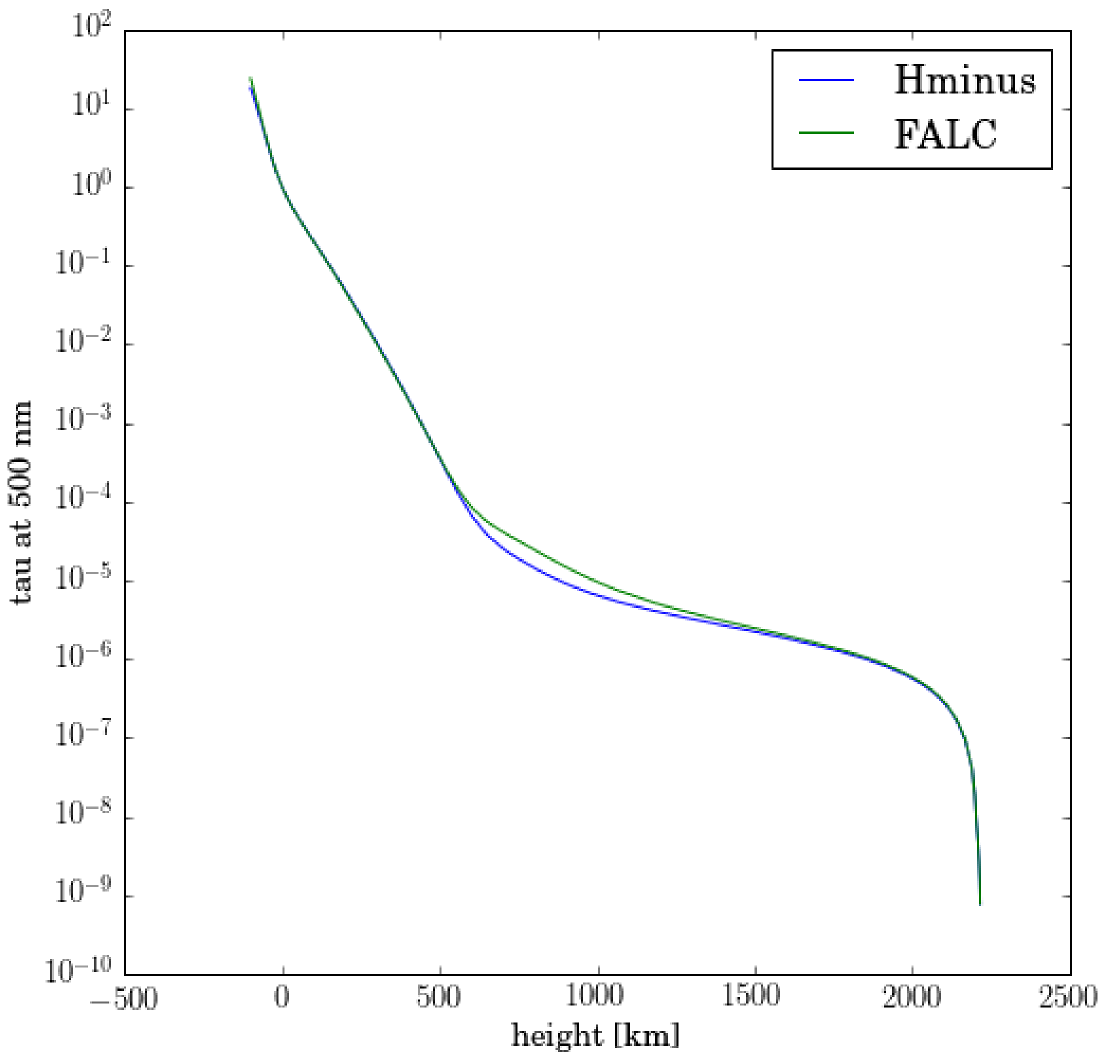
Thomson scattering dominates at depths greater than 1000km as the temperatures are high enought that H is ionized and there are more free electrons. Below 1000km, the opacity due to H^- takes over as the dominant opacity source since at this depth, Hydrogen is mostly neutral, with electrons coming from the ionization of metals.

2.3 Optical depth

- **Integrate the extinction at $\lambda = 500nm$ to obtain the τ_{500} scale and compare it graphically to the FALC τ_{500} scale. Here is my code, with `ext(ih)` denoting the continuous extinction per cm at $\lambda = 500nm$ and at height `h(ih)`:**

```
; compute and plot tau at 500 nm, compare with FALC tau5
tau=fltarr(nh)
for ih=1,nh-1 do tau[ih]=tau[ih-1]+$
0.5*(ext[ih]+ext[ih-1])*(h[ih-1]-h[ih])*1E5
plot,h,tau,/ylog,$
xtitle='height [km]',ytitle='tau at 500 nm'
oplot,h,tau5,linestyle=2
```

In [22]:
```
tau = np.zeros_like(n_Htotal)
ext = alpha_Hminus + alpha_el

for ih in range(1,len(n_Htotal)):
    tau[ih] = tau[ih-1]+0.5*(ext[ih]+ext[ih-1])*(height[ih-1]-height[ih])*1e5

fig, ax = plt.subplots(figsize=(8,8))


ax.semilogy(height, tau, label='Hminus')
ax.semilogy(height, tau5, label='FALC')
ax.set_xlabel('height [km]')
ax.set_ylabel('tau at 500 nm')
ax.legend()
plt.show()
```



2.4 Emergent intensity and height of formation

```
; emergent intensity at wavelength wl (micron)
ext=fltarr(nh)
tau=fltarr(nh)
integrand=fltarr(nh)
contfunc=fltarr(nh)
int=0.
hint=0.
for ih=1,nh-1 do begin
ext[ih]=exthmin(wl*1E4,temp[ih],nel[ih])*(nhyd[ih]-nprot[ih])+0.664E-24*nel[ih]
tau[ih]=tau[ih-1]+0.5*(ext[ih]+ext[ih-1])*(h[ih-1]-h[ih])*1E5
integrand[ih]=planck(temp[ih],wl)*exp(-tau[ih])
int=int+0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])
hint=hint+h[ih]*0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])
contfunc[ih]=integrand[ih]*ext[ih]
endfor
```

```
hmean=hint/int
```

- **The code above sits in file emergint.pro. Copy it into your IDL program and make it work setting `wl=0.5`.**

```
In [23]: def planck(temps, wavs):
             wav = wavs*u.micrometer # to cm
             c = const.c.cgs
             h = const.h.cgs
             k = const.k_B.cgs
             temp2 = temps*u.K
             b = (2*h*c**2/wav**5) / (np.exp(((h*c)/(wav*k*temp2))) - 1)
             # b.to(u.erg / u.cm**2 / u.s / u.Hz,
                             #equivalencies=u.spectral_density(0.8*u.micrometer))

             return b.to(u.erg / u.cm**2 / u.s / u.micrometer, equivalencies=u.spectral_density(wav))
```

```
In [24]: def emergent(wl):
             # emergent intensity at wavelength wl (micron)

             # redefine variables to be consistent with code
             nhyd = n_Htotal # from FALC
             nprot = n_proton
             nh = n_Htotal

             ext = np.zeros_like(nh)
             tau = np.zeros_like(nh)
             integrand = np.zeros_like(nh)
             contfunc = np.zeros_like(nh)
             ints=0.
             hint=0.

             for ih in range(1,len(n_Htotal)):
                 ext[ih] = exthmin(wl*1e4,temp[ih],nel[ih])*(nhyd[ih] - nprot[ih]) + 0.664e-24*nel[ih]
                 tau[ih] = tau[ih-1]+0.5*(ext[ih]+ext[ih-1])*(height[ih-1]-height[ih])*1e5
                 integrand[ih] = planck(temp[ih],wl).value*np.exp(-tau[ih])
                 ints = ints + 0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])
                 hint = hint + height[ih]*0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])
                 contfunc[ih] = integrand[ih]*ext[ih]

             hmean=hint/ints

             return ints, contfunc, hmean, tau
```

Make sure in correct units (micrometer)!!!

```
In [25]: wl = 0.5
         planck(temp[ih],wl)
```

Out[25]:

$$1.8725475 \times 10^{11}\ \frac{\mathrm{erg}}{\mathrm{s\,\mu m\,cm^2}}$$

```
In [26]: wl = 0.5
         intens, contfunc1, hmean1, tau1 = emergent(wl)
```

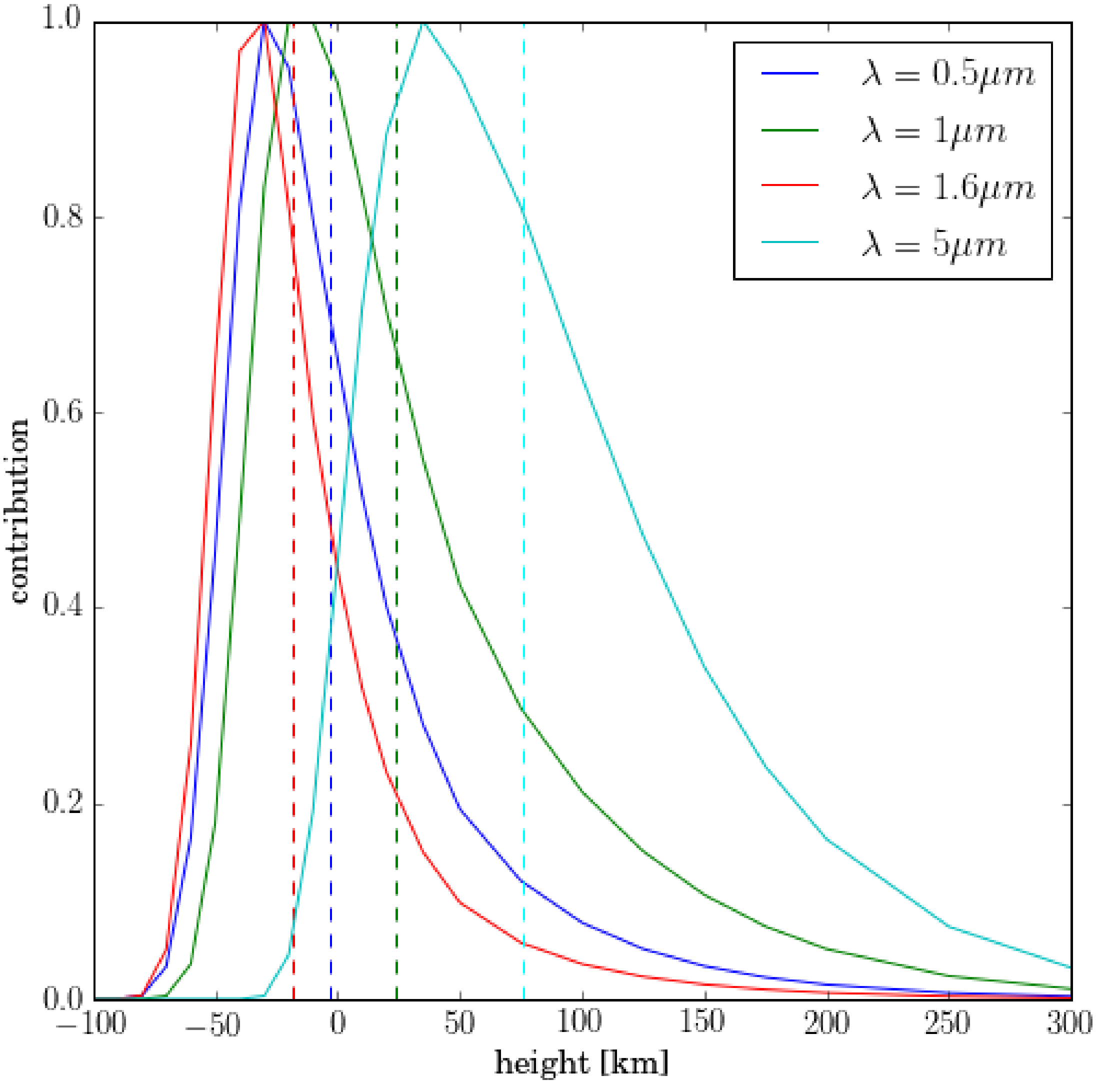
- **Compare the computed intensity at $\lambda = 500nm$ with the observed intensity, using a statement such as** `print,' observed cont int = ',Icont[where(wl eq wav)]` **to obtain the latter.**

In [27]:
```
print "computed cont int = {:.2e}".format(intens)
print "observed cont int = {:.2e}".format(Icont[wl == wave][0])
```

```
computed cont int = 4.29e+10
observed cont int = 4.08e+10
```

- **Plot the peak-normalized contribution function against height and compare its peak location with the mean height of formation.**
- **Repeat the above for $\lambda = 1\mu m$, $\lambda = 1.6\mu m$, and $\lambda = 5\mu m$.**

In [28]:
```
fig, ax = plt.subplots(figsize=(8,8))

intens_half, contfunc_half, hmean_half, tau_half = emergent(0.5)
intens1, contfunc1, hmean1, tau1 = emergent(1.0)
intens16, contfunc16, hmean16, tau16 = emergent(1.6)
intens5, contfunc5, hmean5, tau5 = emergent(5)

ax.plot(height, contfunc_half/contfunc_half.max(), label=r'$\lambda = 0.5 \mu m$')
ax.axvline(hmean_half, ls='--', c='blue')
ax.plot(height, contfunc1/contfunc1.max(), label=r'$\lambda = 1 \mu m$')
ax.axvline(hmean1, ls='--', c='green')
ax.plot(height, contfunc16/contfunc16.max(), label=r'$\lambda = 1.6 \mu m$')
ax.axvline(hmean16, ls='--', c='red')
ax.plot(height, contfunc5/contfunc5.max(), label=r'$\lambda = 5 \mu m$')
ax.axvline(hmean5, ls='--', c='cyan')


ax.set_xlim(-100,300)
ax.set_xlabel('height [km]')
ax.set_ylabel('contribution')

ax.legend()
plt.show()
```



The corresponding mean height of formation is shown as a dashed vertical line of the same color.

- **Discuss the changes of the contribution functions and their cause.** The lowest mean height of formation correspond to the 1.6 and 0.5 micrometer emission where the H^- opacity is the lowest. 1 micrometer emission corresponds a higher opacity and has a higher height of formation.

- **Check the validity of the LTE Eddington-Barbier approximation $I_\lambda \approx B_\lambda(T[\tau = 1])$ by comparing the mean heights of formation with the $\tau = 1$ locations and with the locations where $T_b = T(h)$.**

```
In [29]: def find_nearest(array,value):
             """ Brilliant idea by Jake to use
                 argmin instead of np.where
             """
             idx = (np.abs(array-value)).argmin()
             return idx
```

```
i_half = find_nearest(temp, inv_planck(0.5,intens_half).value)
i_1 = find_nearest(temp, inv_planck(1.0,intens1).value)
i_16 = find_nearest(temp, inv_planck(1.6,intens16).value)
i_5 = find_nearest(temp, inv_planck(5,intens5).value)

j_half = find_nearest(tau_half, 1.0)
j_1 = find_nearest(tau1, 1.0)
j_16 = find_nearest(tau16, 1.0)
j_5 = find_nearest(tau5, 1.0)

print  'wav mean-height h(tau=1) h(T = Tb)'
print 0.5,hmean_half, height[j_half], height[i_half]
print 1.0,hmean1, height[j_1], height[i_1]
print 1.6,hmean16, height[j_16], height[i_16]
print 5.0,hmean5, height[j_5], height[i_5]
```

```
wav mean-height h(tau=1) h(T = Tb)
0.5 -3.14001543022 0.0 1278.0
1.0 24.1089388923 10.0 20.0
1.6 -17.9440499994 -30.0 1580.0
5.0 75.9256236554 50.0 855.0
```

The Eddington-Barbier relation is only close to true for $\lambda = 1.0\mu m$ where the heights are roughly similar. Thus these photons are in LTE as opposed to $\lambda = 1.6\mu m$ where the opacity is low and thus not in LTE.

2.5 Disk-center intensity

The solar disk-center intensity spectrum can now be computed by repeating the above in a big loop over wavelength.

- **Compute the emergent continuum over the wavelength range of Table 5**
- **Compare it graphically with the observed solar continuum in Table 5 and file** `solspect.dat`**.**

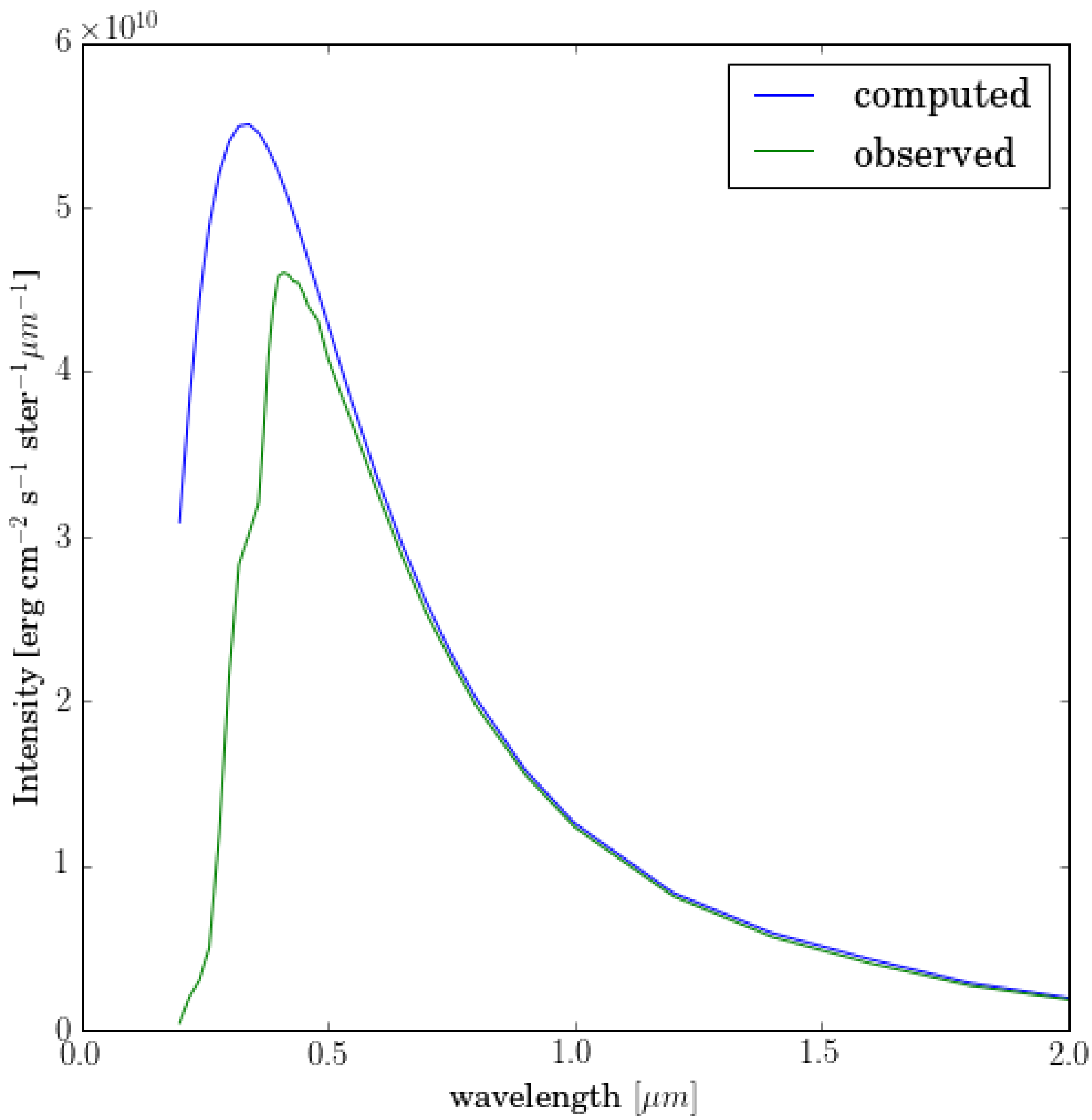
```
In [30]: em_intense_arr = np.zeros_like(wave)

         for i,w in enumerate(wave):
             em_intense_arr[i], tmp, tmp, tmp = emergent(w)
```

```
In [31]: print em_intense_arr
```

```
[  3.08409169e+10   3.82707834e+10   4.43193726e+10   4.88830410e+10
   5.20497010e+10   5.39954880e+10   5.49240946e+10   5.50346636e+10
   5.45065860e+10   5.40521926e+10   5.34937992e+10   5.28465794e+10
   5.21241935e+10   5.13388821e+10   5.05015683e+10   4.96219624e+10
   4.87086659e+10   4.77692725e+10   4.68104647e+10   4.48573119e+10
   4.28877028e+10   3.80639802e+10   3.35766767e+10   2.95369899e+10
   2.59684356e+10   2.28510893e+10   2.01456373e+10   1.57867349e+10
   1.25410265e+10   8.30442592e+09   5.87395085e+09   4.27783536e+09
   2.84074699e+09   1.93260497e+09   8.39685166e+08   4.18894223e+08
   1.37415177e+08   5.74414553e+07]
```

```
In [32]: fig, ax =  plt.subplots(figsize=(8,8))

         ax.plot(wave, em_intense_arr, label=r'computed')
```

```
ax.plot(wave, Icont, label=r'observed')

flux_unit = r'Intensity [erg cm$^{-2}$ s$^{-1}$ ster$^{-1} \mu m^{-1}$]'

ax.set_xlabel(r'wavelength $[\mu m]$')
ax.set_ylabel(flux_unit)
ax.set_xlim(0,2)
ax.legend()
plt.show()
```



2.6 Limb darkening

- **Repeat the intensity evaluation using (10) within an outer loop over μ = 0.1, 0.2,. . . , 1.0.**

In [33]:
```
# make new function to include mu
def emergent_mu(wl,mu):
```

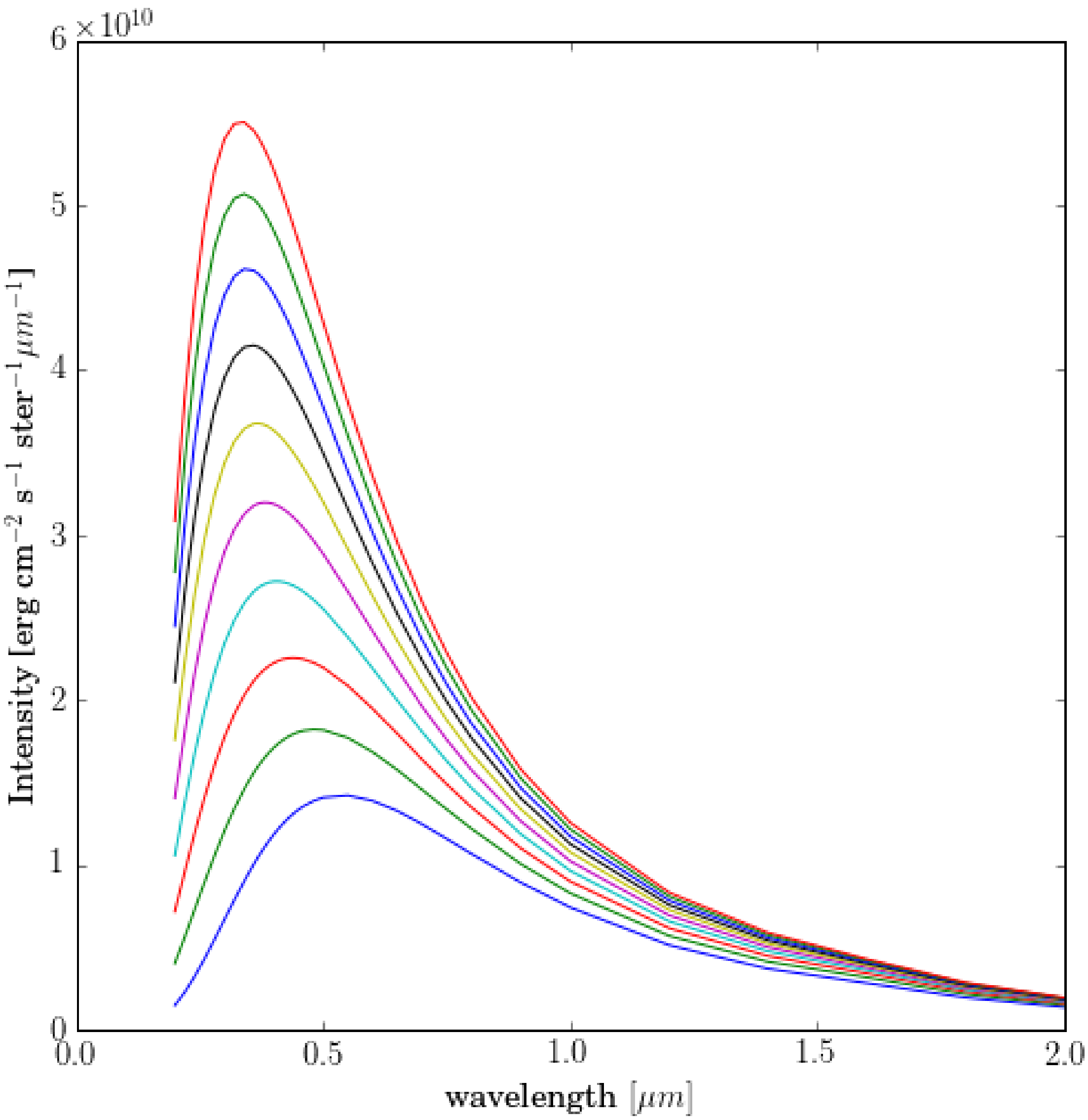
```
    # emergent intensity at wavelength wl (micron)

    # redefine variables to be consistent with code
    nhyd = n_Htotal # from FALC
    nprot = n_proton
    nh = n_Htotal

    ext = np.zeros_like(nh)
    tau = np.zeros_like(nh)
    integrand = np.zeros_like(nh)
    contfunc = np.zeros_like(nh)
    ints=0.
    hint=0.

    for ih in range(1,len(n_Htotal)):
        ext[ih] = exthmin(wl*1e4,temp[ih],nel[ih])*(nhyd[ih] - nprot[ih]) + 0.664e-24*nel[ih]
        tau[ih] = tau[ih-1]+0.5*(ext[ih]+ext[ih-1])*(height[ih-1]-height[ih])*1e5
        integrand[ih] = planck(temp[ih],wl).value*np.exp(-tau[ih]/mu)
        ints = ints + 0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])/mu
        hint = hint + height[ih]*0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])/mu
        contfunc[ih] = integrand[ih]*ext[ih]/mu

    hmean=hint/ints

    return ints, contfunc, hmean, tau
```

```
In [34]: # redefine variables to be consistent with code
         nhyd = n_Htotal # from FALC
         nprot = n_proton
         nh = n_Htotal

         def emergent_mu(wl, mu):
             # Emergent intensity at wavelength wl (micron), cos(theta) mu
             ext=np.zeros_like(nhyd)
             tau=np.zeros_like(nhyd)
             integrand=np.zeros_like(nhyd)
             contfunc=np.zeros_like(nhyd)
             intg=0.
             hint=0.
             for ih in range(1,len(nhyd)):
                 ext[ih]=exthmin(wl*1e4,temp[ih],nel[ih])*(nhyd[ih]-nprot[ih])+0.664E-24*nel[ih]
                 tau[ih]=tau[ih-1]+0.5*(ext[ih]+ext[ih-1])*(height[ih-1]-height[ih])*1e5
                 integrand[ih]=planck(temp[ih],wl).value*np.exp(-tau[ih]/mu)
                 intg=intg+0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])/mu
                 hint=hint+height[ih]*0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])
                 contfunc[ih]=integrand[ih]*ext[ih]
             hmean=hint/intg
             return intg, contfunc, hmean, tau
```

```
In [35]: mu_arr = np.arange(1,11)/10.

         I_mu_arr = np.zeros([len(wave),len(mu_arr)])

         for i,m in enumerate(mu_arr):
             for j,w in enumerate(wave):
```

```
            I_mu_arr[j,i], tmp, tmp, tmp = emergent_mu(w,m)
```

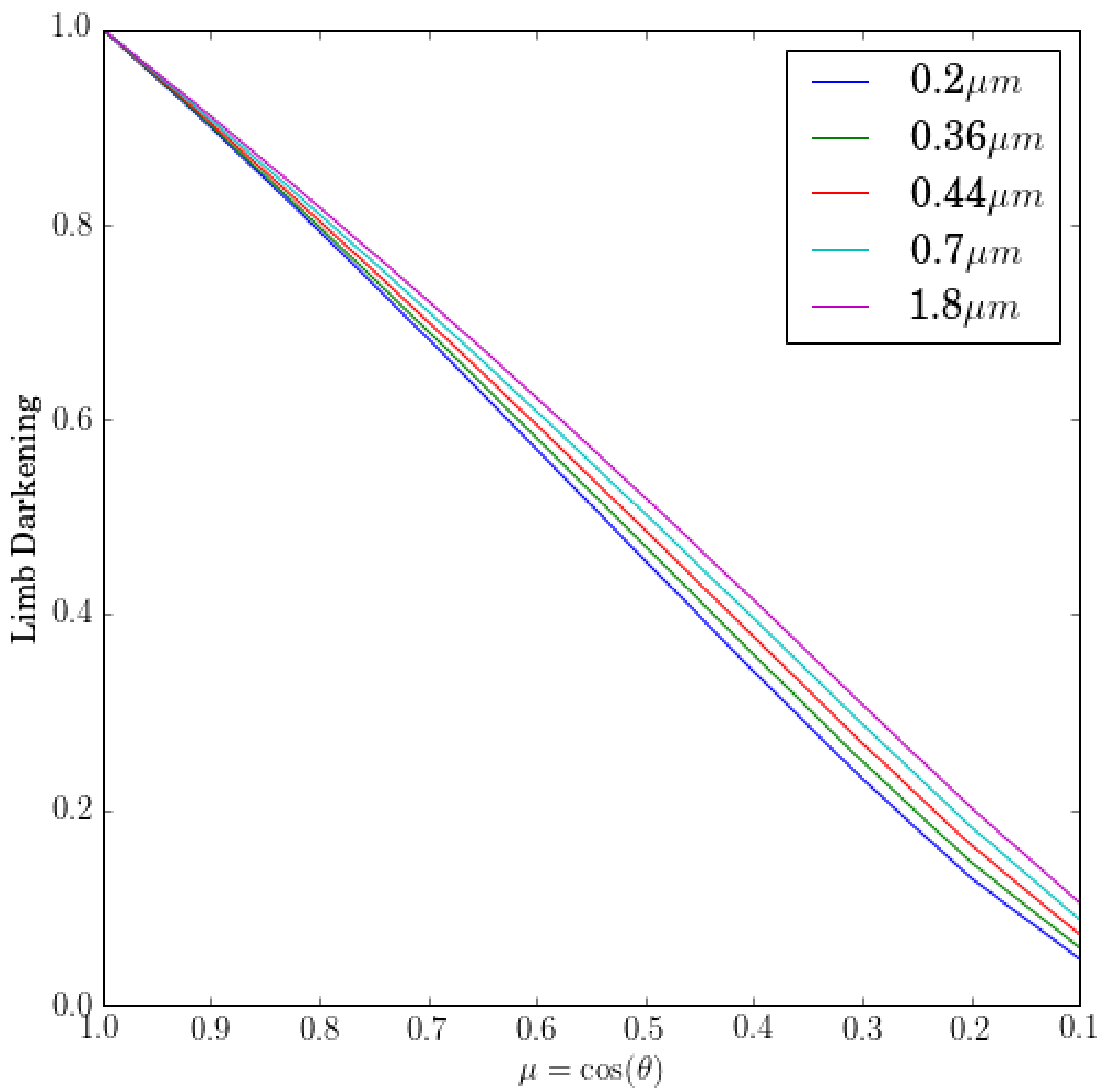
In [36]:
```
fig, ax = plt.subplots(figsize=(8,8))

ax.plot(wave, I_mu_arr[:,:])

flux_unit = r'Intensity [erg cm$^{-2}$ s$^{-1}$ ster$^{-1} \mu m^{-1}$]'

ax.set_xlabel(r'wavelength $[\mu m]$')
ax.set_ylabel(flux_unit)
ax.set_xlim(0,2)
# ax.legend()
plt.show()
```



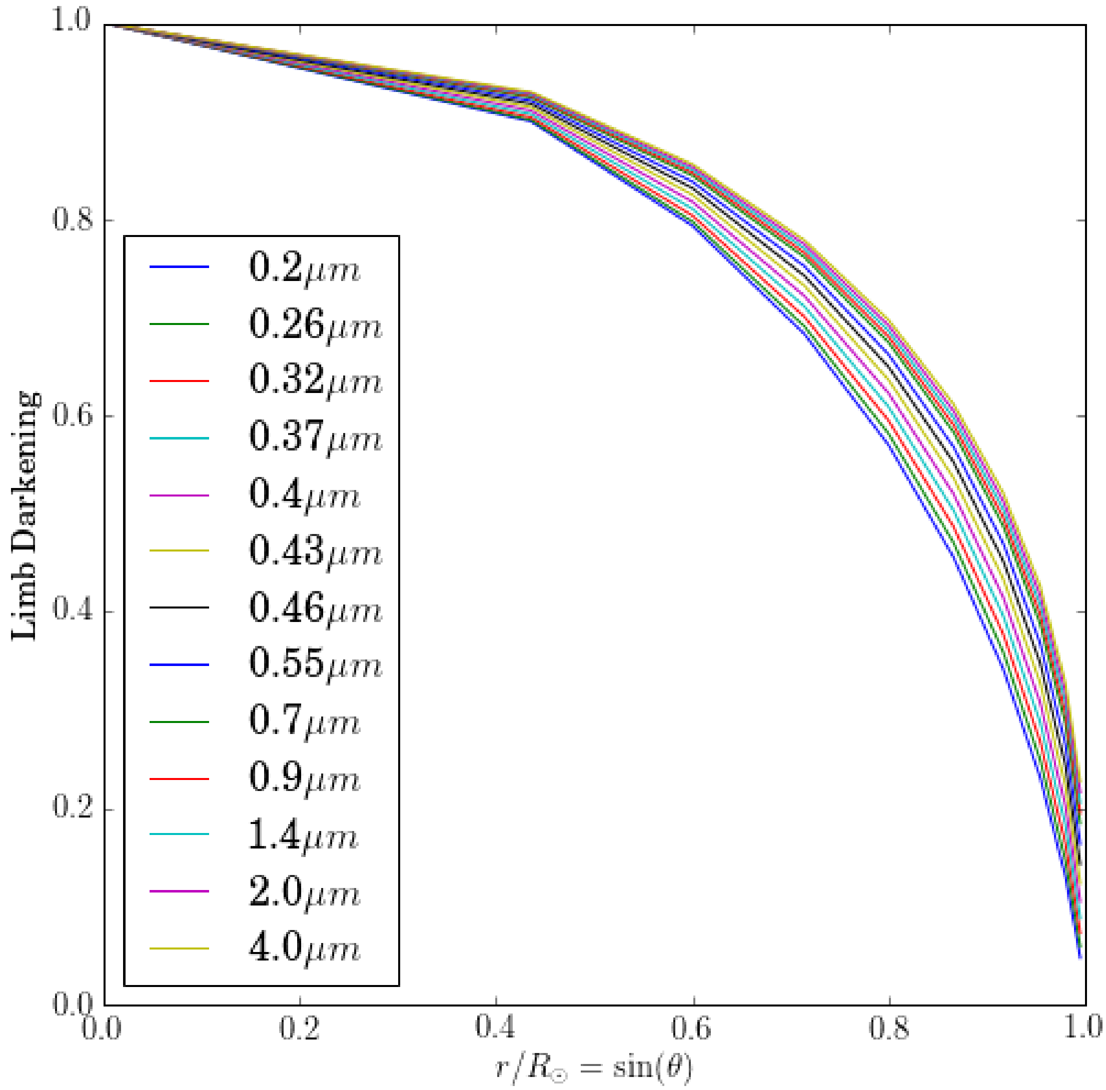
- **Plot the computed ratio $I_\lambda(0,\mu)/I_\lambda(0,1)$ at a few selected wavelengths, against μ and also**

against the radius of the apparent solar disk $r/R_\odot = sin\theta$. Explain the limb darkening and its variation with wavelength.

In [38]:
```
fig, ax = plt.subplots(figsize=(8,8))

limbs = np.zeros_like(I_mu_arr)
for i,m in enumerate(mu_arr):
    limbs[:,i] = I_mu_arr[:,i] / I_mu_arr[:,-1]

for i,w in enumerate(wave[::8]):
    lab = str(w)+r'$\mu m$'
    ax.plot(mu_arr, limbs[i,:], label=lab)

ax.set_xlabel(r"$\mu = \cos(\theta)$")
ax.set_ylabel(r"Limb Darkening")
ax.invert_xaxis()
ax.legend(loc='best')
plt.show()
```



In [39]:
```
theta = np.arccos(mu_arr)

r = np.sin(theta)

fig, ax = plt.subplots(figsize=(8,8))
for i,w in enumerate(wave[::3]):
    lab = str(w)+r'$\mu m$'
    ax.plot(r, limbs[i,:], label=lab)

ax.set_xlabel(r"$r/R_{\odot} = \sin(\theta)$")
ax.set_ylabel(r"Limb Darkening")
ax.legend(loc='best')
plt.show()
```



2.7 Flux integration

- **Compute the emergent solar flux and compare it to the observed flux in Table 5 and file `solspect.dat`. Here is my code (file `gaussflux.pro`):**

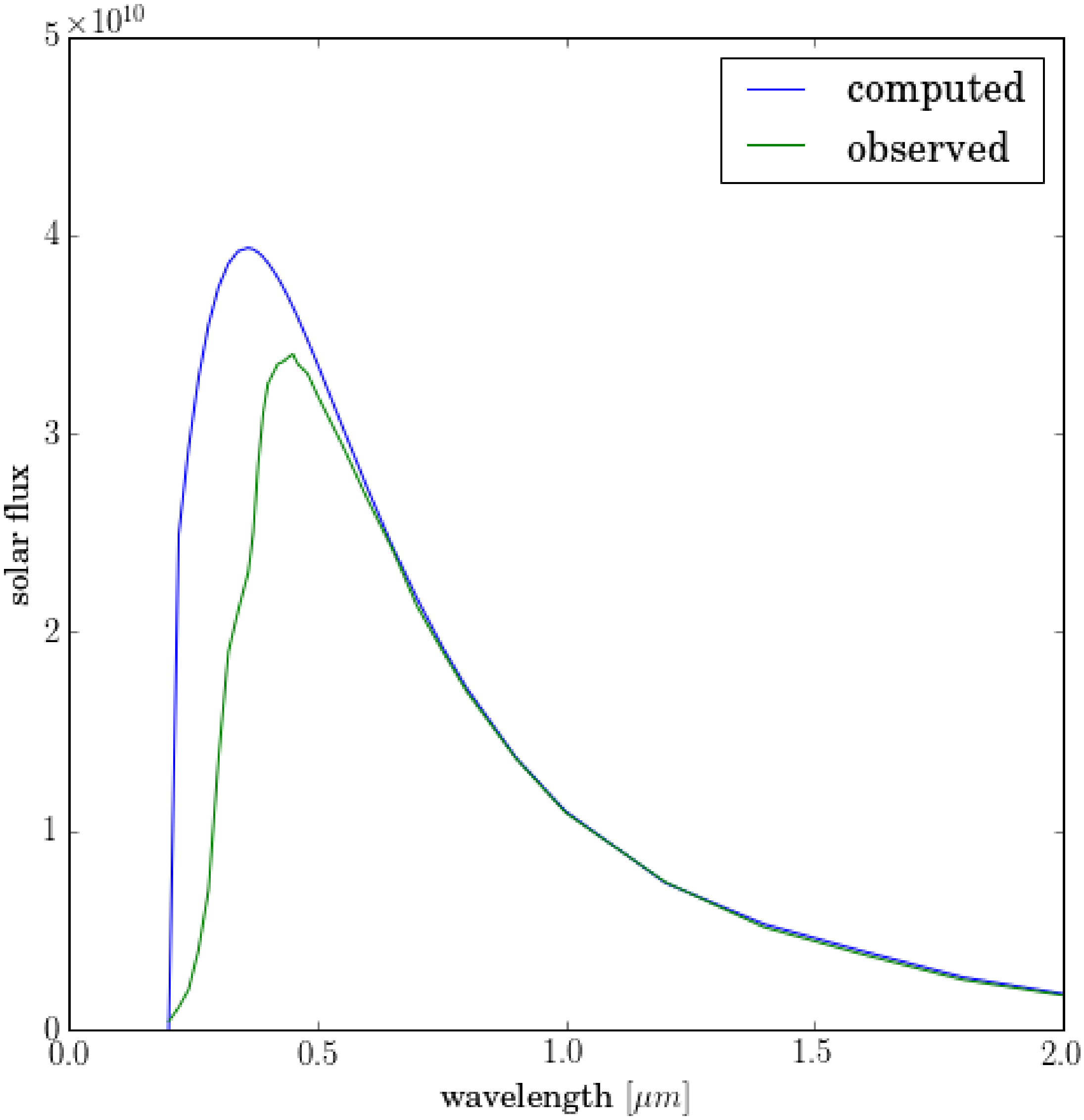
```
; ===== three-point Gaussian integration intensity -> flux
; abscissae + weights n=3 Abramowitz & Stegun page 916
xgauss=[-0.7745966692,0.0000000000,0.7745966692]
wgauss=[ 0.5555555555,0.8888888888,0.5555555555]
fluxspec=fltarr(nwav)
intmu=fltarr(3,nwav)
for imu=0,2 do begin
mu=0.5+xgauss[imu]/2. ; rescale xrange [-1,+1] to [0,1]
wg=wgauss[imu]/2. ; weights add up to 2 on [-1,+1]
for iw=0,nwav-1 do begin
wl=wav[iw]
@emergintmu.pro ; old trapezoidal integration I(0,mu)
intmu[imu,iw]=int
fluxspec[iw]=fluxspec[iw]+wg*intmu[imu,iw]*mu
endfor
endfor
fluxspec=2*fluxspec ; no !pi, AQ has flux F, not {\cal F}
plot,wav,fluxspec,xrange=[0,2],yrange=[0,5E10],xtitle='wavelength[micron]', ytitle= 'solar flux'
oplot,wav,Fcont,linestyle=2
xyouts,0.5,4E10,'computed'
xyouts,0.35,1E10,'observed'
```

```
; emergent intensity at wavelength wl (micron) and angle mu
ext=fltarr(nh)
tau=fltarr(nh)
integrand=fltarr(nh)
contfunc=fltarr(nh)
int=0.
hint=0.
for ih=1,nh-1 do begin
  ext[ih]=exthmin(wl*1E4,temp[ih],nel[ih])*(nhyd[ih]-nprot[ih])
          +0.664E-24*nel[ih]
  tau[ih]=tau[ih-1]+0.5*(ext[ih]+ext[ih-1])*(h[ih-1]-h[ih])*1E5
  integrand[ih]=planck(temp[ih],wl)*exp(-tau[ih]/mu)
  int=int+0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])/mu
  hint=hint+h[ih]*0.5*(integrand[ih]+integrand[ih-1])*(tau[ih]-tau[ih-1])/mu
  contfunc[ih]=integrand[ih]*ext[ih]/mu
endfor
hmean=hint/int
```

In [40]:
```
# ===== 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_like(wave)
intmu=np.zeros([3,len(wave)])
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(1,len(wave)):
            wl=wave[iw]
            # @emergintmu.pro # old trapezoidal integration I(0,mu)
            ints, tmp, tmp, tmp = emergent_mu(wl,mu)
            intmu[imu,iw]=ints
            fluxspec[iw]=fluxspec[iw]+wg*intmu[imu,iw]*mu


    fluxspec=2*fluxspec # no !pi, AQ has flux F, not {\cal F}
```

In [41]:
```
fig, ax = plt.subplots(figsize=(8,8))

ax.plot(wave, fluxspec, label=r'computed')
ax.set_xlim(0,2)
ax.set_ylim(0,5e10)
ax.set_xlabel(r'wavelength $[\mu m]$')
ax.set_ylabel(r'solar flux')

ax.plot(wave, Fcont, label=r'observed')

ax.legend()
plt.show()
```



5×10^{10}
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0
solar flux
wavelength $[\mu m]$
computed
observed