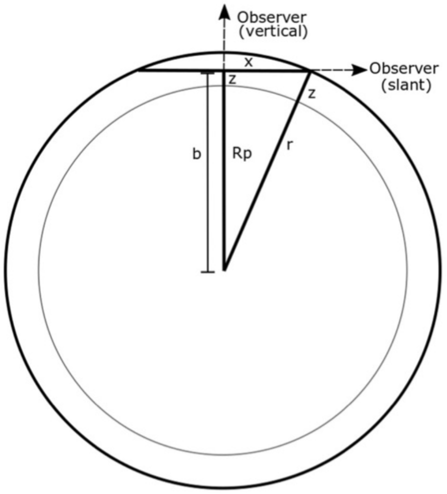
HW 4

1. In this problem, we will study the optical depth of a planet's atmosphere using two viewing geometries (see figure below). The vertical viewing geometry is looking straight down through the planet's atmosphere to the surface. The slant viewing geometry is looking through one side of the atmosphere to the other, without reaching the surface.

A planet's scale height, H, is defined as $H = \frac{kT}{mg}$, where m is the mean molecular mass of the molecules in the planet's atmosphere, k is the Stefan-Boltzmann constant and T is the atmospheric temperature. Although most of these quantities are functions of altitude, season, and latitude, it is sufficient for our purposes to approximate H as constant.



a) Use the hydrostatic balance ($dP/dz = -\rho g$), the ideal gas law, and the definition of H to show that a planet's atmospheric density as a function of altitude is given by $\rho(z) = \rho_0 e^{-z/H}$, where ρ_0 is the surface density and z is the altitude above the surface. You may treat g as constant.

$H = \frac{kT}{mg} = cst$, $\frac{dP}{dz} = -\rho g$, $Pm = \rho kT$

m — mean molecular mass of molecules

so $\frac{d}{dz}\left(\frac{\rho kT}{m}\right) = -\rho g$

$\frac{1}{\rho}\frac{d\rho}{dz} = -g\left(\frac{m}{kT}\right)$

$\frac{1}{\rho}d\rho = -\cancel{g}\cdot\frac{1}{H\cancel{g}}dz$

$\int_{\rho_0}^{\rho(z)} \frac{1}{\rho}d\rho = -\frac{1}{H}\int_0^z dz$

$\ln(\rho(z)) - \ln(\rho_0) = \frac{-z}{H}$

$\rho(z) = \rho_0 e^{-z/H}$

b) Use the definition of $d\tau$ to show that the vertical optical depth through a planet's atmosphere is given by $\tau_v = \int_0^\infty \rho_0 \kappa\, e^{-z/H} dz$, assuming that the atmosphere extends to infinity and taking the opacity, κ, as constant.

$d\tau = -\kappa\rho\, dz$

* with τ measured from top to the surface

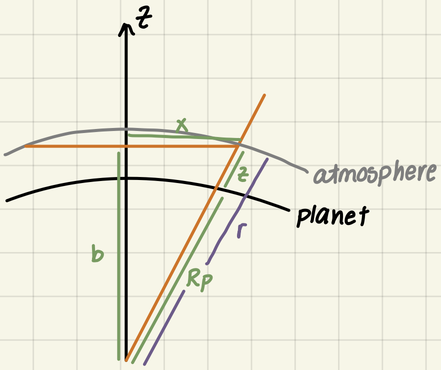
$\int_{\tau_{0,v}}^{\tau_{f,v}} d\tau = \int_0^\infty -\kappa\rho_0 e^{-z/H} dz$

$0 - \tau_v = -\int_0^\infty \kappa\rho_0 e^{-z/H} dz$

$\tau_{top} = 0$

$\tau_v = \int_0^\infty \kappa\rho_0 e^{-z/H} dz$

c) Show that the slant optical depth is $\tau_s = \int_{-\infty}^{\infty} \rho_0 \kappa e^{-(\sqrt{x^2+b^2}-R_P)/H} dx$, where b is the impact parameter, or the lowest altitude reached by a ray passing through the atmosphere, and R_P is the planet's radius.



so $z = r - R_P$

$= \sqrt{b^2+x^2} - R_P$

$$\Rightarrow \tau_s = \int_0^{\infty} \rho_0 \kappa e^{-z/H} dz$$

$$\tau_s = \int_{-\infty}^{\infty} \rho_0 \kappa e^{-(\sqrt{b^2+x^2}-R_P)/H} dx$$

d) In the Python script OpticalDepth.py, the function `vertical` calculates τ_v by integrating from 0 to an upper limit high above the planet's surface, where there is essentially no atmosphere. Write a similar function, `slant`, which calculates τ_s at impact parameter b, keeping the same upper limit. Set $b = R_P$ and use your function to calculate the ratio $\frac{\tau_s}{\tau_v}$ for Earth. Use the κ, ρ_0 and R_P defined in the Python file.

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as si

kappa = 1.e-4          #opacity in m^2/kg
rho0 = 1.225          #surface density in kg/m^3
H = 8e3              #scale height in m
Rp = 6371.e3          #Earth radius in m
limit = 1.e7           #limit of integration in m; should theoretically be infinity but there is essentially no atmosphere at this altitude


# Calculates the vertical optical depth between altitudes 'lower' and 'upper'
def vertical(lower=0, upper=limit):         #calculates the vertical optical depth, integrating from lower to upper
    def integrand(z):                             #function we want to integrate
        return np.exp(-z/H)
    tau = rho0*kappa*si.quad(integrand, lower, upper)[0]        #integrates from far above the surface to
    return tau

def slant(b=Rp, lower=-1e7, upper=limit):
    def integrand(x):
        return np.exp(-(np.sqrt(b**2 + x**2)-Rp)/H)
    tau = rho0*kappa*si.quad(integrand, lower, upper)[0]
    return tau

ratio = slant()/vertical()
print(ratio)
```

→ Ratio = 70.77

e) The atmosphere transitions from transparent to opaque at $\tau \approx 2/3$, so the photons we receive come from the altitude at which this transition occurs. Find the lower limit of integration for the transition (to the nearest 100 m), and the corresponding altitude, for both viewing geometries by incrementing the lower limit of integration.

Use the difference in your answers to explain why stars appear brighter in the centre than near the edges (limb darkening).

```python
# e)
def altitude(type, step=10):
    lower = 0

    while True:
        # calculate tau for the current altitude
        tau = vertical(lower=lower) if type=='vertical' else slant(lower=lower)
        # tau at lower=0 is large, and decreasing as lower increases altitude
        # so stop when we cross 2/3
        if tau <= 2/3:
            break
        # increment altitude
        lower += step

    return lower, tau

lower_vertical, tau_vertical = altitude('vertical')
print(f'Altitude for vertical optical depth = {tau_vertical}: {lower_vertical} m')
lower_slant, tau_slant = altitude('slant')
print(f'Altitude for slant optical depth = {tau_slant}: {lower_slant} m')
print('Difference in altitude: ', lower_slant-lower_vertical)
```

```
Altitude for vertical optical depth = 0.6660085922041606: 3090 m
Altitude for slant optical depth = 0.6666076644731158: 529250 m
Difference in altitude:  526160
```

looking at the center, we are able to see further/deeper in the atmosphere (vertical altitude for $\tau = 2/3$ is lower) so we are probing a brighter/hotter region of the planet. the slant altitude is much higher which means we are probing a cooler/dimmer region.

f) Considering the slant optical depth through the entire atmosphere, find (to the nearest 100 m) the impact parameter at which the atmosphere is transparent $\tau_s = 2/3$.

```python
# f)
def find_impact_parameter(step=10):
    b = Rp
    while True:
        tau = slant(b)
        if tau <= 2/3:
            break
        b += step
    return b, tau

b, tau = find_impact_parameter()
print(f'Impact parameter at tau = {tau}: {b} m')
```

```
Impact parameter at tau = 0.6659098242929855: 6408190.0 m
```

2. A classic HII region is created around a hot star that is embedded in a cloud of hydrogen gas. The radius of the HII region, or *Stromgren Sphere*, R_{ss} depends on the number of ionizing photons emitted by the star every second, and the density and temperature of the surrounding gas.

$$R_{ss} = (3/4\pi\alpha)^{1/3}\ N_{uv}^{1/3}\ n_p^{-2/3}$$

Table 15.2. Rates of H-ionizing photons for main sequence stars.

Spectral type	Photons/s ($\times 10^{48}$)
O5	51
O6	17.4
O7	7.2
O8	3.9
O9	2.1
B0	0.43
B1	0.0033

Table 15.2 from Kutner's "Astronomy: A Physical Perspective"

The table to the left gives N_{uv}, the rate at which photons capable of ionizing hydrogen are emitted, for stars ranging in spectral type from O5 to B1.

a. Calculate the wavelength of a photon in nm that is just capable of ionizing H from the ground state.

b. What are the temperatures of stars with spectral types O5 and B1? What is the ratio of the rates at which they emit photons that can ionize H I?

c. Find the size of the HII region ('*Stromgren Sphere*') surrounding an O5 and a B1 star in units of cm and AU assuming they are embedded in a gas cloud with a density of $n = 10^4$ cm^{-3} and T = 10000 K, so the temperature dependent coefficient $\alpha = 2 \times 10^{-13}$ cm^3/s.

d. What is the ratio of their *Stromgren Radii*?

a) we know that we need 13.6 eV to ionize H so we can get λ from $E = hc/\lambda$

$\Rightarrow \lambda = \dfrac{hc}{13.6\,eV} = 91.16$ nm

b) from links in previous assignment. $T_{O5} = 39810.7$ K and $T_{B1} = 22387.2$ K

Ratio $= 51/0.0033 = 15454.54$

c)

```
O5: 8.475211086774894e+17 cm, 56653.28688916221 AU
B1: 3.402502437289813e+16 cm, 2274.432397579448 AU
```

d) $\dfrac{R_{SS,O5}}{R_{SS,B1}} = 24.909$

code for this question:

```python
import numpy as np
import astropy.units as u
import scipy.constants as sc

h = sc.h * u.J * u.s
c = sc.c * u.m / u.s
E = 13.6 * u.eV

# a)
lam = h * c / E
print(lam.to(u.nm))

# b)
print(51/0.0033)

# c)
def Rss(N, n, alpha):
    return (3/(4*np.pi*alpha))**(1/3) * N**(1/3) * n**(-2/3)

n = 1e4 * u.cm**(-3)
alpha = 2e-13 * u.cm**3 / u.s
N_05 = 51e48 * u.s**(-1)
N_B1 = 0.0033e48 * u.s**(-1)

Rss_05 = Rss(N_05, n, alpha)
Rss_B1 = Rss(N_B1, n, alpha)
print(f'05: {Rss_05}, {Rss_05.to(u.au)}\nB1: {Rss_B1}, {Rss_B1.to(u.au)}')

# d)
print(f'Ratio: {Rss_05/Rss_B1}')
```