Спецпрактикум по теоретической астрофизике

Курдоякова Марина

Задание:

Вариант 3:
Рассчитать интенсивность излучения, выходящего из плоскопараллельной серой атмосферы, используя формулу:

$$I_\nu(0,\mu) = \int_0^\infty \frac{S_\nu(\tau)e^{\frac{-\tau}{\mu}}d\tau}{\mu}$$

При условии ЛТР:

$$S_\nu(\tau) = B_\nu(T(\tau)) = \frac{2h\nu^3}{c^2}\frac{1}{e^{\frac{h\nu}{kT(\tau)}}-1}$$

Функцию $T(\tau)$ взять в приближениях Шварцшильда–Шустера, Эддингтона и Чандрасекара:

$$T = T_{eff}(\tau + 1/2)^{1/4}$$
$$T = T_{eff}(3\tau/4 + 1/2)^{1/4}$$
$$T = T_{eff}(3\tau/4 + \sqrt{3}/4)^{1/4}$$

Алгоритм решения задачи:

Для решения данной задачи было сделано следующее:

1. Прописанны в программе численные константы, а так же подъинтегральные функции.

2. С помощью пакета scipy.integrate для python находились значения интегралов для интенсивностей. Ошибки при вычисления интеграла так же прописанны в результете. Если они не устраивают, то можно задать свою точность.

3. Для нахождения первого графика в задании была написанна следующая функция:
Создавался массив μ со значениями от 0.00006 до 1 с шагом n, который можно варьировать. Я остановилась на значении 300. И далее для каждого значения μ находилися соответствующие значения интенсивности. Найдя значения при $\mu = 1$, строили графики для разных значений $\nu = 3.0 * 10^{14}, \quad 5.5 * 10^{14}, \quad 1.0 * 10^{15}$ Гц.

4. Для втого графика, предложенного в задании было написано вот что:
Аналогично первой функции, создавался массив значений ν от $1.0 * 10^{13}$ до $2.0 * 10^{15}$ с шагом их разницы, деленное на n, которое я задала как 120. Далее вычислялись интенсивности для каждого из приближений для даной частоты и соответствующая функция Планка. μ в данном месте равно 1. И в конце строился их график, где ν бралось в логарифмической шкале.

Код программы:

```
import numpy as np
import math
import matplotlib.pyplot as plt
from math import sqrt, sin, pi, cos, log, exp, acos, log
from scipy.integrate import dblquad, quad, romberg

h = 6.626e-34   #Дж*сек
c = 3.0e8       #м/сек
k = 1.38e-23    #Дж*К
T_eff = 5800.0      #К
```

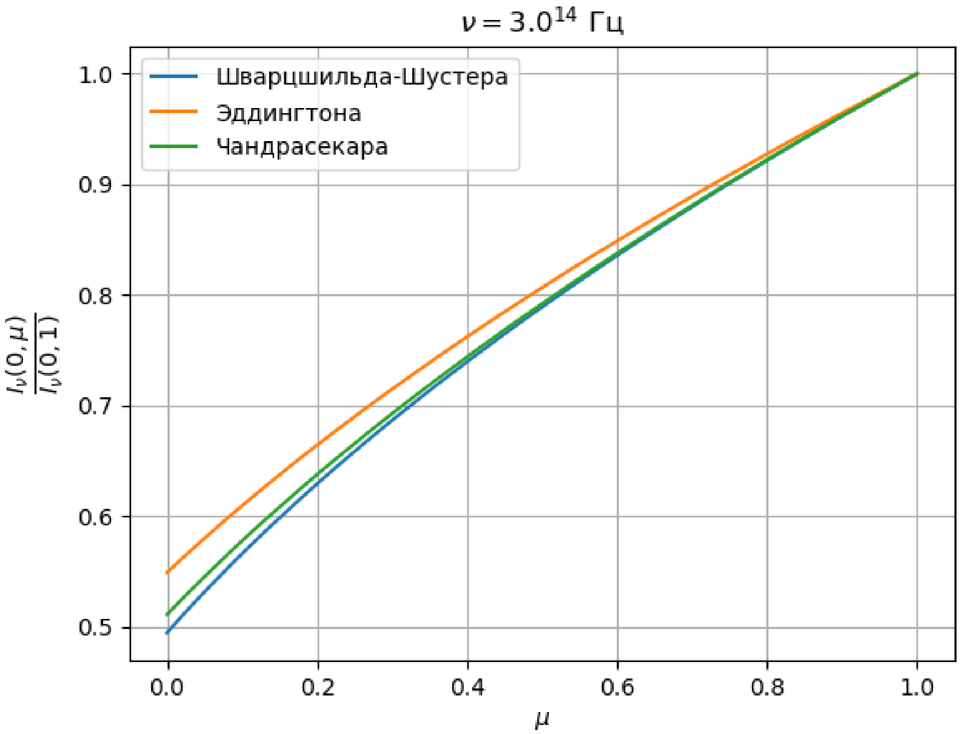
```
def solution(nu, mu):
    T_tau1 = lambda tau: T_eff*np.power(tau + 0.5, 0.25)
    T_tau2 = lambda tau: T_eff*np.power(3.0*tau/4.0 + 0.5, 0.25)
    T_tau3 = lambda tau: T_eff*np.power(3.0*tau/4.0 + sqrt(3)/4, 0.25)

    const = 2*h*(nu**3)/(c**2)
    const2 = h*nu/k

    Plank = const/(np.exp(const2/T_eff) - 1)
    S_nu = lambda tau, T, nu: const/(np.exp(const2/T(tau)) - 1)
    I_nu = lambda tau, T, nu: S_nu(tau,T,nu)*np.exp(-tau/mu)/mu

    I1 = quad(I_nu, 0, np.inf, args=(T_tau1, nu), epsabs = 1.5e-32)
    I2 = quad(I_nu, 0, np.inf, args=(T_tau2, nu), epsabs = 1.5e-32)
    I3 = quad(I_nu, 0, np.inf, args=(T_tau3, nu), epsabs = 1.5e-32)
    Eps = [I1[1], I2[1], I3[1]]
    # print(Eps)
    return I1[0], I2[0], I3[0], Plank, Eps

def first(nu):
    n = 300
    start = 0.00006
    step = (1-start)/n
    mu = [start + i*step for i in range(n+1)]

    I1_0 = np.zeros(n+1); I2_0 = np.zeros(n+1); I3_0 = np.zeros(n+1)
    I1_1 = np.zeros(n+1); I2_1 = np.zeros(n+1); I3_1 = np.zeros(n+1)
    for i in range(n+1):
     I1_0[i], I2_0[i], I3_0[i], non1, eps1 = solution(nu, mu[i])
     I1_1[i], I2_1[i], I3_1[i], non2, eps2 = solution(nu, 1)
    print('nu = ', nu)
    print('eps for ShSh = ', eps1[0])
    print('eps for Edd = ', eps1[1])
    print('eps for Shand = ',eps1[2])
    print('------------------------')
    fig, ax = plt.subplots()
    plt.plot(mu, I1_0/I1_1, label = 'Шварцшильда-Шустера')
    plt.plot(mu, I2_0/I2_1, label = 'Эддингтона')
    plt.plot(mu, I3_0/I3_1, label = 'Чандрасекара')

    plt.xlabel(r'$\mu$',fontsize=10)
    plt.ylabel(r'$\frac{I_{\nu} (0,\mu)}{I_{\nu} (0,1)}$', fontsize=15)
    ax.set_title(r'$\nu = $' + str(nu/1e14) + r'$^{14}$' + ' Гц')
    ax.legend()
    ax.grid()
    plt.show()

def second():
    n = 120
    start = 1.0e13
    step = (2.0e15-start)/n
    nu = [start + i*step for i in range(n+1)]

    I1 = np.zeros(n+1); I2 = np.zeros(n+1); I3 = np.zeros(n+1); Plank = np.zeros(n+1); eps = np.zeros((n+1
```

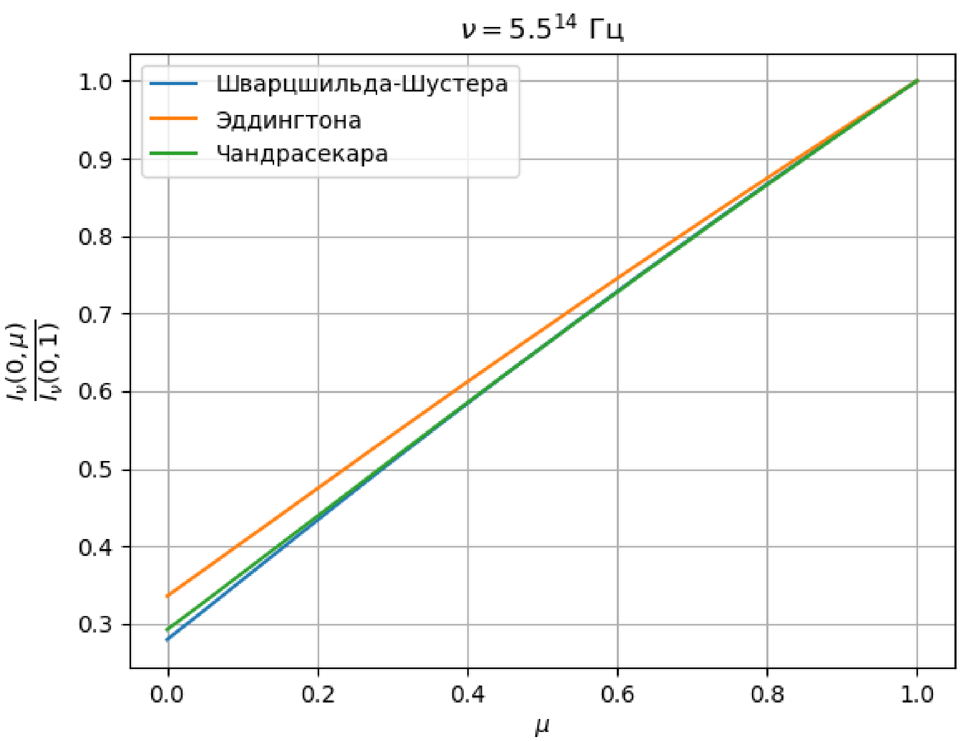
```
    for i in range(n+1):
     I1[i], I2[i], I3[i], Plank[i], eps[i] = solution(nu[i], 1)
    for i in range(n+1):
     nu[i] = log(nu[i])

    fig, ax = plt.subplots()
    plt.plot(nu, I1, label = 'Шварцшильда-Шустера')
    plt.plot(nu, I2, label = 'Эддингтона')
    plt.plot(nu, I3, label = 'Чандрасекара')
    plt.plot(nu, Plank, label = 'функция Планка')

    plt.xlabel(r'$\log (\nu) $', fontsize=12 )
    plt.ylabel(r'$I_{\nu}$', fontsize=12)
    ax.legend()
    ax.grid()
    plt.show()

def main():
    first(3.0e14)
    first(5.5e14)
    first(1.0e15)
    second()

main()
```

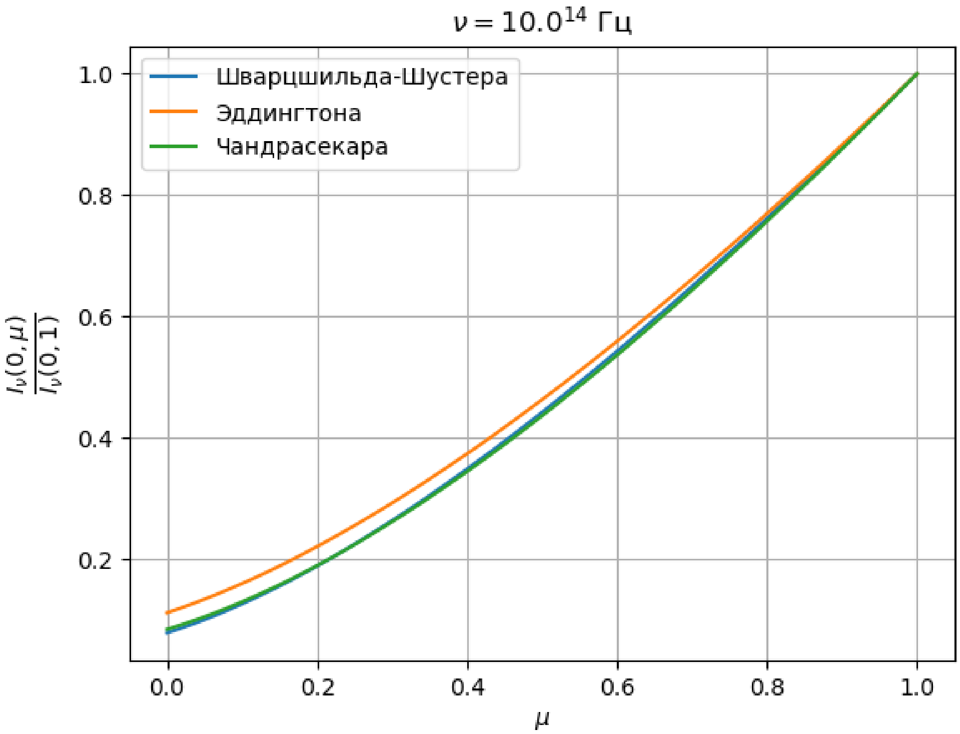
Результаты:

Давайте посмотрим на ошибки при вычислении интеграла для разных значений частоты:

```
nu =  300000000000000.0
eps for ShSh =  2.1322427443661207e-18
eps for Edd =  1.7487718980678736e-18
eps for Shand =  1.724746330440235e-18
-------------------------
nu =  550000000000000.0
eps for ShSh =  6.923264480370149e-19
eps for Edd =  7.297458787787822e-19
eps for Shand =  7.51807019521322e-19
-------------------------
nu =  1000000000000000.0
eps for ShSh =  9.325679897750694e-18
eps for Edd =  8.456926194676893e-18
eps for Shand =  8.586118924397934e-18
-------------------------
```


$\nu = 3.0^{14}$ Гц
Шварцшильда-Шустера
Эддингтона
Чандрасекара
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
$\frac{I_\nu(0,\mu)}{I_\nu(0,1)}$
μ


$\nu = 5.5^{14}$ Гц
Шварцшильда-Шустера
Эддингтона
Чандрасекара
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0
$\frac{I_\nu(0,\mu)}{I_\nu(0,1)}$
μ



И второй график, на котором видно, как соотносятся разные приближения и функция Планка: