The simulation of super-Eddington accretion spectrum

Hui-Hsuan Chung, 106025504

Scientific Motivation

Super-Eddington accretion is a key mechanism to support how to power extremely high luminosity of compact sources in nearby universe. Ultraluminous X-ray sources (ULXs) are non-nuclear sources with X-ray luminosity $\geq 10^{39}\ erg\ s^{-1}$. While this luminosity limit corresponds to the Eddington limit for a $\sim 10\ M_{\odot}$ black hole, the physical nature of ULXs is not clear for most of the cases. The mechanisms powering ULXs include stellar mass black holes with super-Eddington accretion, intermediate-mass black holes, high-mass X-ray binaries, supernova remnants, and accreting pulsars.

By simulating the super-Eddington accretion spectrum and exploring their best spectral fitting, we could have a closer look at the properties of the observed ULXs and determine whether they are super-Eddington accreting and their spectral states., especially when conventional spectral fitting cannot tell insufficient information due to lack of X-ray photon counts received.

The ULX will be studied in this work is the ULX in M83 galaxy. This source has been studied by XMM-Newton and reported as an super-Eddington accretion candidate (Soria et al. 2015). However, the quality of Chandra data is not as good as of XMM-Newton. From the spectral fitting I did with the Chandra analysis software, the derived spectral parameters have relatively large upper and lower bounds. In this work, I will perform how to simulate super-Eddington accretion spectrum based on slim disk model by adding different physical parameters. Moreover, I will estimate the mean squared error (MSE) of Chandra data and each simulated spectrum and further determine the best fit of the super-Eddington fitting.

Slim Disk Model

One important spectral features of an super-Eddington accreting flows is photon trapping, happening when matters and photons are frequently interact in optically thick flow regions. The intense interactions will prevent the radiation energy release from the disk and in turn the photons will be trapped in the accretion flow. Finally the trapped photons will fall onto the central black hole with the accretion gas. This phenomenon can be described by slim disk model assuming some approximations to describe super-Eddington accretion spectrum. The slim disk model is based on the extended blackbody or multi-color disk model. The slim disk spectra can be obtained by integrating the blackbody spectrum at each radius which will have different temperature, where we will assume the effective temperature profile. The emergent spectra can be obtained as following: $S_{\nu} = \frac{cosi}{D^2}\int_{r_{in}}^{r_{out}} B_{\nu}[T_{eff}]2\pi r dr$, while we have the temperature profile $T_{eff} = T_{eff,in}(\frac{r}{r_{in}})^p$ and p is the temperature dependence setting as a free parameter. Later we change the variables and we have $S_{\nu} = \frac{cos i\ 4\pi\ hr_{in}^2}{D^2c^2p}(\frac{k_BT_{eff,in}}{h\nu})^{\frac{2}{p}}\int_{x_{in}}^{x_{out}}\frac{x^{2/p-1}}{e^x-1}dx$, while $x_{in} = \frac{h\nu}{k_BT_{eff,in}}$ and $x_{out} = \frac{h\nu}{k_BT_{eff,in}}(\frac{r_{out}}{r_{in}})^p$ .In the function,$i, D, c, h, r_{in}, r_{out}, k_B, T_{eff,in}, \nu$ corresponds to the inclination angle, distance between the source and the observer, speed of light, Planck constant, the innermost radius of the accretion disk, the outermost radius of the accretion disk, Boltzmann constant, the effective temperature at the innermost radius and frequency. The physical derivation is depend on the context of Kato et al. 2008.

Methodology

I require trapezoidal method to simulate the spectra of the slim disk model by calculating the integral term. Substituting different parameters such as the parameter p and the effective temperature, I can generate the super-Eddington accretion spectra for black holes with different physical properties. After that, I will compare the Chandra data of the ULX in M83 galaxy with the simulated spectra. Further I will determine the best fit among these spectra used on the MSE value. The MSE is calculated by this formula $\frac{1}{n}\sum_{i=1}^{n}(observed\ value_i - simulated\ value_i)^2$ where n is the numbers of data points.

Spectrum Examination

I plot the specific flux as a function of frequency in order to examine whether the generated spectra is accurate. From the spectral formula, the value of p in temperature profile will bring different slopes to the spectra in the middle-frequency domain. The S_{ν}

for slim disk (p=0.5) and standard disk (p=0.75) is proportional to ν^{-1} and $\nu^{\frac{1}{3}}$, respectively. Figure 1 shows that for slim disk or standard disk model, the simulated spectra is consistent with the predicted profile.

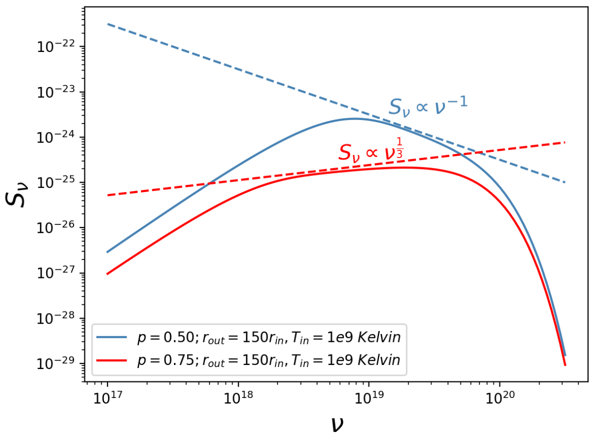


Figure 1: specific flux density as a function of frequency for slim and standard disk model when p is equal to 0.5 and 0.75 by setting outermost radius and innermost temperature.

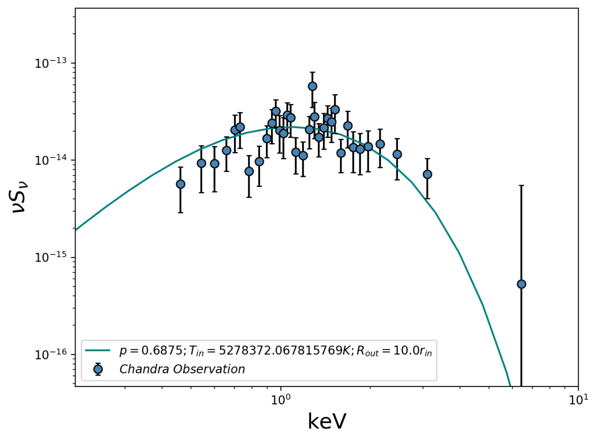


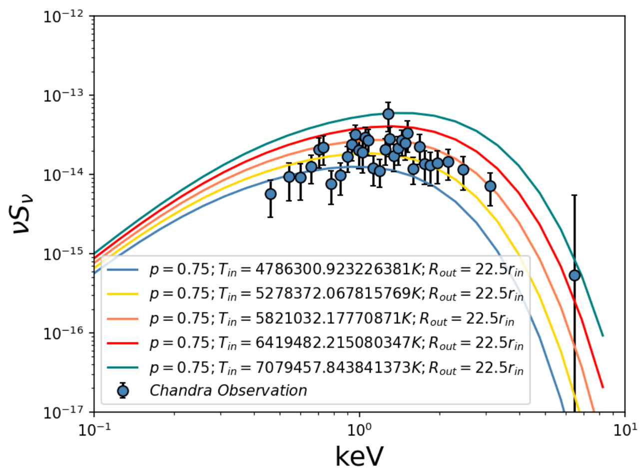
Figure 2: The best fitted spectra as a function as energy (keV) with a minimal MSE=7.628253674900715e-29 among all the simulated models.

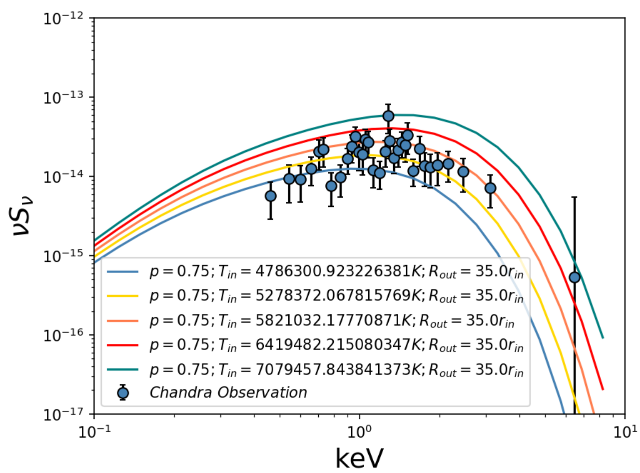
Result

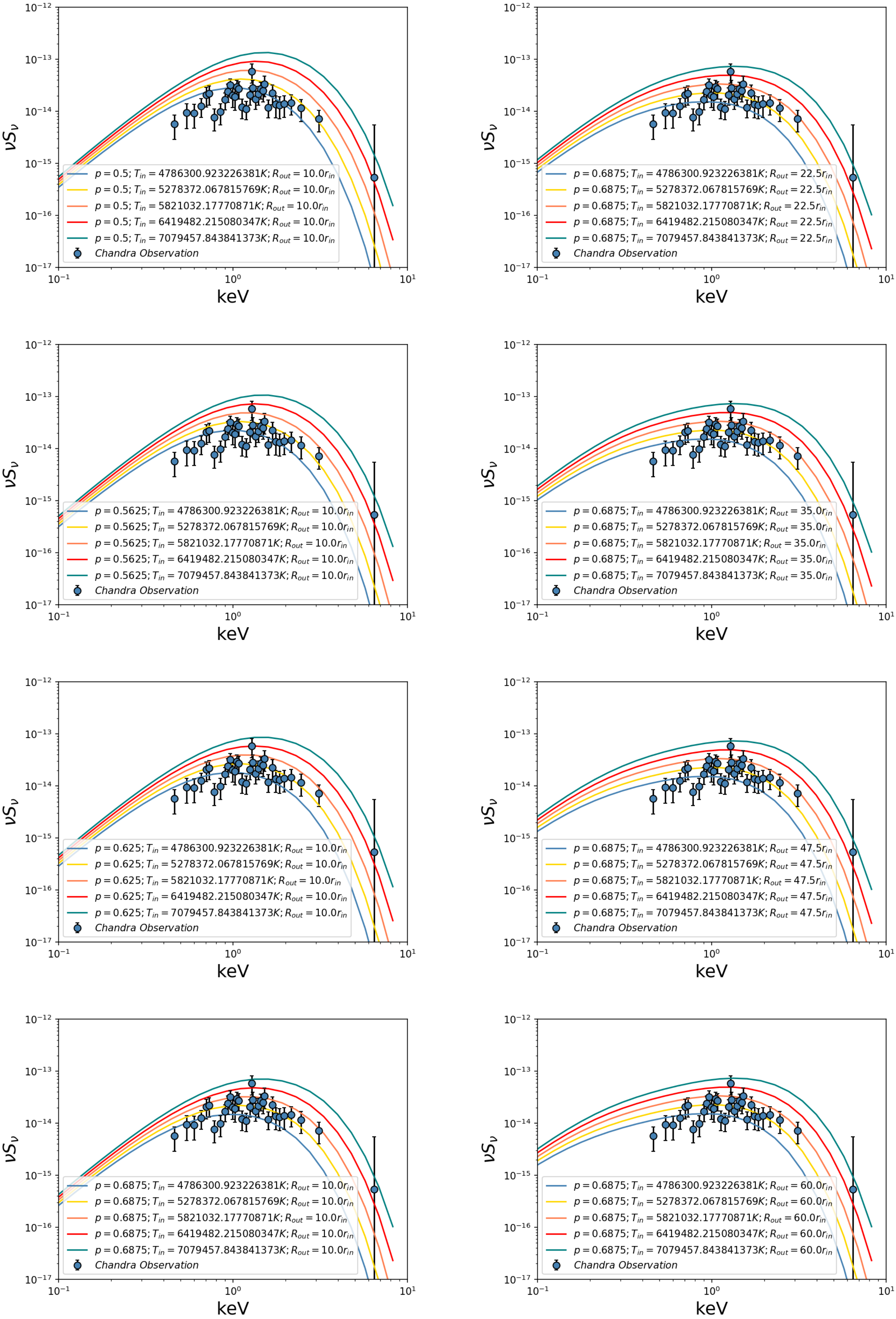
I simulate 5*5*5=125 spectra by setting p values ranging from 0.5 to 0.75, innermost temperature T_{in} ranging from 4786300 to 6419482, and the outermost radius R_{out} ranging from 10 to 60 times innermost radius R_{in}. Among all these simulated spectra, the minimal MSE value happens when p=0.6875, T_{in}=5278372 and R_{out}=10 R_{in}(Fig. 2). The physical parameters implies that the ULX may between the standard disk state and the super-Eddington action state since the p value is between 0.75 and 0.5. Moreover, the innermost disk temperature is relatively lower compared with super-Eddington accretion disk temperature high than 10^7 Kelvin. However, there is no other high spectral-resolution X-ray telescope observing this target around 2011 Dec., when this Chandra data have taken. It is difficult to confirm the exact physical states as this event only collected 370 counts while some well-studied ULXs have several thousand ounce to perform better spectral analysis. On the other hand, due to the relatively poor sensitivity at harder X-ray band compared with XMM-Newton (Soria et al. 2015), the data points near 6.4 keV might be unreliable, leading to an unreliable spectral shape when I adopt the spectral data for this analysis. Therefore, adopting this method to study other ULXs is required to examine the accuracy of my method. However, in this case, fitting the observational data with simulated spectra could provide another window if there is not accessible qualified spectral data. In the future, it would be helpful to adopt much more spectral models for spectral energy distribution fitting.

Figure Appendix

These are part of the simulated spectra according to different different p values, innermost temperature and the outermost radius.







νS_ν
keV
$p=0.5; T_{in}=4786300.923226381K; R_{out}=10.0r_{in}$
$p=0.5; T_{in}=5278372.067815769K; R_{out}=10.0r_{in}$
$p=0.5; T_{in}=5821032.17770871K; R_{out}=10.0r_{in}$
$p=0.5; T_{in}=6419482.215080347K; R_{out}=10.0r_{in}$
$p=0.5; T_{in}=7079457.843841373K; R_{out}=10.0r_{in}$
Chandra Observation
νS_ν
keV
$p=0.6875; T_{in}=4786300.923226381K; R_{out}=22.5r_{in}$
$p=0.6875; T_{in}=5278372.067815769K; R_{out}=22.5r_{in}$
$p=0.6875; T_{in}=5821032.17770871K; R_{out}=22.5r_{in}$
$p=0.6875; T_{in}=6419482.215080347K; R_{out}=22.5r_{in}$
$p=0.6875; T_{in}=7079457.843841373K; R_{out}=22.5r_{in}$
Chandra Observation
νS_ν
keV
$p=0.5625; T_{in}=4786300.923226381K; R_{out}=10.0r_{in}$
$p=0.5625; T_{in}=5278372.067815769K; R_{out}=10.0r_{in}$
$p=0.5625; T_{in}=5821032.17770871K; R_{out}=10.0r_{in}$
$p=0.5625; T_{in}=6419482.215080347K; R_{out}=10.0r_{in}$
$p=0.5625; T_{in}=7079457.843841373K; R_{out}=10.0r_{in}$
Chandra Observation
νS_ν
keV
$p=0.6875; T_{in}=4786300.923226381K; R_{out}=35.0r_{in}$
$p=0.6875; T_{in}=5278372.067815769K; R_{out}=35.0r_{in}$
$p=0.6875; T_{in}=5821032.17770871K; R_{out}=35.0r_{in}$
$p=0.6875; T_{in}=6419482.215080347K; R_{out}=35.0r_{in}$
$p=0.6875; T_{in}=7079457.843841373K; R_{out}=35.0r_{in}$
Chandra Observation
νS_ν
keV
$p=0.625; T_{in}=4786300.923226381K; R_{out}=10.0r_{in}$
$p=0.625; T_{in}=5278372.067815769K; R_{out}=10.0r_{in}$
$p=0.625; T_{in}=5821032.17770871K; R_{out}=10.0r_{in}$
$p=0.625; T_{in}=6419482.215080347K; R_{out}=10.0r_{in}$
$p=0.625; T_{in}=7079457.843841373K; R_{out}=10.0r_{in}$
Chandra Observation
νS_ν
keV
$p=0.6875; T_{in}=4786300.923226381K; R_{out}=47.5r_{in}$
$p=0.6875; T_{in}=5278372.067815769K; R_{out}=47.5r_{in}$
$p=0.6875; T_{in}=5821032.17770871K; R_{out}=47.5r_{in}$
$p=0.6875; T_{in}=6419482.215080347K; R_{out}=47.5r_{in}$
$p=0.6875; T_{in}=7079457.843841373K; R_{out}=47.5r_{in}$
Chandra Observation
νS_ν
keV
$p=0.6875; T_{in}=4786300.923226381K; R_{out}=10.0r_{in}$
$p=0.6875; T_{in}=5278372.067815769K; R_{out}=10.0r_{in}$
$p=0.6875; T_{in}=5821032.17770871K; R_{out}=10.0r_{in}$
$p=0.6875; T_{in}=6419482.215080347K; R_{out}=10.0r_{in}$
$p=0.6875; T_{in}=7079457.843841373K; R_{out}=10.0r_{in}$
Chandra Observation
νS_ν
keV
$p=0.6875; T_{in}=4786300.923226381K; R_{out}=60.0r_{in}$
$p=0.6875; T_{in}=5278372.067815769K; R_{out}=60.0r_{in}$
$p=0.6875; T_{in}=5821032.17770871K; R_{out}=60.0r_{in}$
$p=0.6875; T_{in}=6419482.215080347K; R_{out}=60.0r_{in}$
$p=0.6875; T_{in}=7079457.843841373K; R_{out}=60.0r_{in}$
Chandra Observation

Reference

Kitaki T., Mineshige S., Ohsuga K., Kawashima T., 2017, PASJ, 69, 92 (https://arxiv.org/pdf/1709.01531.pdf)

Kato, S., Fukue, J., Mineshige, S., 2008, Black-Hole Accretion disks – To wards a New Paradigm (Kyoto: Kyoto University Press)

Soria R., Kuntz K. D., Long K. S., Blair W. P., Plucinsky P. P., Winkler P. F., 2015, ApJ, 799, 140 Straub O. et al., 2011, A&A, 5

Code Reference

You could also refer to my code spec.py for color syntax!

```python
#!/usr/bin/env python3

import numpy as np
import matplotlib.pyplot as plt
import math


#-----------------------------------------------------------------------------------------------
# universal constants and units
h            = 6.626068e-27             # Plank constant (erg/s)
c            = 3e10                     # Spped of light (cm/s)
k_B          = 1.38064852e-16           # Boltzmann constant (erg/Kelvin)
G            = 6.6738e-8                # Gravitational constant
Mpc_to_cm    = 3.08567758e24            # Mpc/cm
M_sun        = 1.9891e33                # Solar mass (g)
pi           = math.pi                  # Basically pi
M_p          = 1.6726e-24               # Proton mass (g)
sig_T        = 1.640e-16

M            = 15*M_sun                 # Blackhole mass
L_edd        = 4*pi*G*M*M_p*c/sig_T     # Eddington luminosity
eta          = 0.1                      # Energy conversion efficiency
M_dot        = 1e19*(0.1/eta)*(M/M_sun) # Mass accretion rate

r_g          = 2.*G*M/c**2              # Schwarzschild radius
r_in         = 1.83*r_g                 # Innermost radius
sigma        = 5.6704e-5                # Stefan-Boltzmann constant (erg cm-2 s-1 K-4)

# The temperature (Kelvin) at the inner radius (cm)
# T_eff_in     = (3.*G*M_dot*M/(8*pi*r_in**3*sigma))**0.25

p_value      = 0.63                     # Temperature dependence
distance     = 4.03*Mpc_to_cm           # Distance from M83 to the observer
deg_to_rad   = pi/180.                  # convert degree to radian
intervals    = 10000.                   # the step numbers used in the integration
step         = 100                      # step numbers for frequency range
#-----------------------------------------------------------------------------------------------



#------------------------------------------------------------------
# load Chandra data: ULX in M83 galaxy
energy, f, f_err = np.loadtxt('plot_flux.dat', unpack='True')
#------------------------------------------------------------------



#------------------------------------------------------------------------------------------------------------------------------------
# Slim disk model
def slim(inclination, nu, distance, p_value, pi, planck_const, k_B, r_in, r_out, T_eff_in):
    x_in   = (h*nu)/(k_B*T_eff_in)
    x_out  = (h*nu)/(k_B*T_eff_in)*(r_out/r_in)**p_value
#------------------------------------------------------------------
    # perform trapsoidal intergration
    def trap(x_in, x_out, intervals):
        #----------------------------------------------
        def func(x):
            return x**(2./p_value - 1.)/(math.exp(x) - 1.)
        #----------------------------------------------
        h            = (x_out - x_in)/intervals
        sum_area     = 0.0

        while x_in < x_out:
            part     = h*(func(x_in) + func(x_in + h))/2.
            sum_area = sum_area + part
            x_in     = x_in + h
        integration  = sum_area
        return integration
#------------------------------------------------------------------
    trap   = trap(x_in, x_out, intervals)
    S_nu   = (math.cos(inclination)*4.*pi*h/c**2./distance**2.)*r_in**2./p_value*(k_B*T_eff_in/h/nu)**(2./p_value)*nu**3.*trap
    return S_nu
#------------------------------------------------------------------------------------------------------------------------------------



#----------------------------------------------------------------------------------------------------------
# Calculate S_nu at different nu
def plot_spec(inclination, color, label, r_out, p_value, T_eff_in):
    S_nu    = []

    for i in nu:
        S_nu_slim = slim(inclination, i, distance, p_value, pi, h, k_B, r_in, r_out, T_eff_in)
        S_nu.append(S_nu_slim)
    plt.plot(nu, S_nu, color=color, label=label)
#----------------------------------------------------------------------------------------------------------



#------------------------------------------------------------------------------------------------------------------------------------------------
# Check the frequencey dependence for p=0.5 and 0.75
def spec_dependence():

    plot_spec(0 *deg_to_rad, 'steelblue',  r'$p = 0.50; r_{out}=150r_{in}, T_{in}=1e9\ Kelvin$', 150.*r_g, 0.50, 1e9)
    plot_spec(0 *deg_to_rad, 'red',        r'$p = 0.75; r_{out}=150r_{in}, T_{in}=1e9\ Kelvin$', 150.*r_g, 0.75, 1e9)

    plt.plot(nu, (nu**-1)*(10**-4.5),    color='steelblue', linestyle='--')
    plt.plot(nu, (nu**(1/3))*(10**-30.95), color='red',       linestyle='--')

    plt.annotate(r'$S_{\nu}\propto \nu^{-1}$',          xy=(1.4*10**19, 3.26*10**-24), color='steelblue', weight='bold', size=15)
    plt.annotate(r'$S_{\nu}\propto \nu^{\frac{1}{3}}$', xy=(5.8*10**18, 3.12*10**-25), color='red',       weight='bold', size=15)

    plt.legend(loc='lower left')
    plt.xlabel(r'$\nu$', fontsize=18)
```

```python
    plt.ylabel(r'$S_{\nu}$', fontsize=18)
    plt.xscale('log')
    plt.yscale('log')
    plt.show()

# Spectrum Examination
def run_spec_check():
    # set up frequency range
    nu = np.logspace(17, 20.5, step)
    spec_dependence()

#run_spec_check()
#------------------------------------------------------------------------------------------------------------------------------------


#--------------------------------------------------------------------------------
# Set up frequence range for simulated spectrum
step = 100
nu   = np.logspace(16.0, 18.3, step)

# Convert nu to keV
keV  = (nu*h/(1.6022e-9))
#--------------------------------------------------------------------------------


#----------------------------------------------------------------------------------------------------
# Calculate nu*S_nu at different keV based on the function slim()
def plot_fit(inclination, color, label, r_out, p_value, T_eff_in):
    nu_S_nu = []

    for i in nu:
        S_nu_slim = slim(inclination, i, distance, p_value, pi, h, k_B, r_in, r_out, T_eff_in)
        nu_S_nu.append(S_nu_slim*i)
    plt.plot(keV, nu_S_nu, color=color, label=label)

    n = len(energy)
    MSE = []

    for i in range(n):
        nu_chandra = energy[i]*(1.6022e-9)/h
        f_model    = slim(0*deg_to_rad, nu_chandra, distance, p_value, pi, h, k_B, r_in, r_out, T_eff_in)
        MSE.append((f[i] - f_model*nu_chandra)**2)

    MSE_val = sum(MSE)/n

    # select small MSE value
    if MSE_val < 1e-28:
        print(MSE_val)

#----------------------------------------------------------------------------------------------------


#------------------------------------------------------------------------------------------------------
Temp  = np.logspace(6.68, 6.85, 5)
p     = np.linspace(0.5,  0.75, 5)
R_out = np.linspace(10.,   60.,  5)
color = ['steelblue','gold','coral','red','teal']
index = range(len(Temp))

print(energy)
def fit(index_p, index_r_out):
    for k in index:
        label = r'$p=$' + str(p[index_p]) + r'$; T_{in}=$' + str(Temp[k]) + r'$ K; R_{out}=$' + str(R_out[index_r_out]) + r'$ r_{in}$'
        plot_fit(0*deg_to_rad, color[k],  label, R_out[index_r_out]*r_g, p[index_p], Temp[k])

    # plot Chandra data
    kwargs = dict(ecolor='k', color='k', capsize=2, ms=7)
    plt.errorbar(energy, f, yerr=f_err, fmt='o', mfc='steelblue', **kwargs, label=r'$Chandra\ Observation$')

    plt.legend(loc='lower left')
    plt.xlabel(r'keV', fontsize=18)
    plt.ylabel(r'$\nu S_{\nu}$', fontsize=18)
    plt.xlim(0.1, 10.0)
    plt.ylim(1e-17, 1e-12)

    plt.xscale('log')
    plt.yscale('log')

def run_fit():
    for i in index:
        for j in index:
            print(i, j)
            fit(i, j)
            #plt.savefig('fig/'+str(i)+'_'+str(j)+'.png', dpi=150)
            plt.show()

# Run the fitting
#run_fit()

# Plot best fit
def best_fit():
    label = r'$p=$' + str(p[2]) + r'$; T_{in}=$' + str(Temp[1]) + r'$ K; R_{out}=$' + str(R_out[0]) + r'$ r_{in}$'
    plot_fit(0*deg_to_rad, 'teal',  label, R_out[0]*r_g, p[2], Temp[1])

    # plot Chandra data
    kwargs = dict(ecolor='k', color='k', capsize=2, ms=7)
    plt.errorbar(energy, f, yerr=f_err, fmt='o', mfc='steelblue', **kwargs, label=r'$Chandra\ Observation$')

    plt.legend(loc='lower left')
    plt.xlabel(r'keV', fontsize=18)
    plt.ylabel(r'$\nu S_{\nu}$', fontsize=18)
    plt.xlim(0.1, 10.0)
    plt.ylim(1e-17, 1e-12)
    plt.xscale('log')
    plt.yscale('log')
    plt.show()
# Plot the best fitted simulated spectra (set up the index by myself)
#best_fit()
#-----------------------------------------------------------------------------------------------------


#------------------------------------------------------------------------------------------------------
# set up the temperature range based on Chandra spectral fitting
inclination=0.0
cos_theta = math.cos(inclination*deg_to_rad)
temp = [5990719.989183, 4867531.939266, 7113908.0391]
norm =  0.0401643   #[0.0401643, 0.0728194]
D10 = 4.03*10**3/10
Rin_obs = (norm/cos_theta)**0.5*D10*10**5
#------------------------------------------------------------------------------------------------------
```