solution

November 8, 2020

1 Coupled Linear ODE (RK-4 Method)

Computational Astrophysics - ESA614 / ESA414 Shivam Kumaran SC17b122

1.1 Importing Libraries

[22]:
```
from rk4 import rk4
import numpy as np
from matplotlib import pyplot as plt
plt.rcdefaults()
plt.style.use('seaborn-ticks')
#plt.style.use('dark_background')
plt.style.use('seaborn-dark-palette')
#plt.style.use('bmh')
plt.rcParams.update({'font.size': 12})
```

2 RK4

[23]:
```
def rk4_cp(x , x_0 , y_0 , z_0, fy , fz , tol=1e-5):

    def rk4_next_val(f,x_0 , y_0 , h):
        f0 = f(x_0,y_0)
        f1 = f(x_0+h/2 ,  y_0+(h/2)*f0)
        f2 = f(x_0+h/2 , y_0+(h/2)*f1)
        f3 = f(x_0+h , y_0+h*f2)
        y_next = y_0+(h/6)*(f0+2*f1+2*f2+f3)
        return y_next

    def calc(h):
        n = int(abs((x_0-x)/h))
        x_next,y_next ,z_next = x_0, y_0 , z_0
        for i in range(n):
            y_next = rk4_next_val(fy(z_next) , x_next , y_next , h)
            z_next = rk4_next_val(fz(y_next) , x_next , z_next , h)
            x_next += h
        return y_next , z_next
```

```
    if(abs((x-x_0))<1e-14):
        return (y_0 , z_0)
    else:
        h = (x-x_0)/2
        prev = calc(h)
        h = h/2
        nxt = calc(h)
        err1 = abs((prev[0]-nxt[0])/(prev[0]))
        err2 = abs((prev[1]-nxt[1])/(prev[1]))
        i = 0

        while(err1>tol or err2>tol):
            i+=1
            h = h/2
            prev = nxt
            nxt = calc(h)
            err1 = abs((prev[0]-nxt[0])/(prev[0]))
            err2 = abs((prev[1]-nxt[1])/(prev[1]))
        return nxt
```

2.1 Defining Constants

all in CGS Units

[2]:
```
kb = 1.38e-16
me = 9.1e-28
mp = 1.6e-24
c = 2.99792458e10
rg = 3e6
```

2.2 Defining Functions

2.2.1 Coulomb Coupling Γ_{ep}

[3]:
```
def gamma(n , tp, te ):
    val = (3.2e-12)*(kb/mp)*(n**2)*(tp-te)*(((me)/(te**3))**0.5)
    return val
```

2.2.2 Bremsstrahlung cooling Λ_e

[4]:
```
def lmd(n,te):
    val = (1.4e-27)*(n**2)*(te**0.5)
    return val
```

2.2.3 Number Density

[5]:
```
def nx(m_dot ,  x):
    denom = 2*np.pi*mp*(rg**2)*c*(x**(3/2))
    val = m_dot/denom
    return val
```

2.2.4 Derivative of T_p w.r.t x $\frac{dT_p}{dx}$

[6]:
```
def f_tp_wrap(m_dot):
    def f_tp(te):
        def f_in(x, tp):
            n = nx(m_dot,x)
            g = gamma(n , tp , te)
            term1 = ((4*np.pi*mp*rg**3)/(3*kb*m_dot))*g*(x**2)
            term2 = tp*((3*x-4)/(3*x*(x-1)))
            #print(term1,term2)
            val = term1 - term2
            return val
        return f_in
    return f_tp
```

2.2.5 Derivative of T_e w.r.t x $\frac{dT_e}{dx}$

[7]:
```
def f_te_wrap(m_dot):
    def f_te(tp):
        def f_in(x, te):
            #print(te)
            n = nx(m_dot,x)
            g = gamma(n , tp , te)
            l = lmd(n,te)
            term1 = ((4*np.pi*mp*rg**3)/(3*kb*m_dot))*(g-l)*(x**2)
            term2 = te*((3*x-4)/(3*x*(x-1)))
            #print(term1 , term2)
            val = -term2 -term1
            return val
        return f_in
    return f_te
```

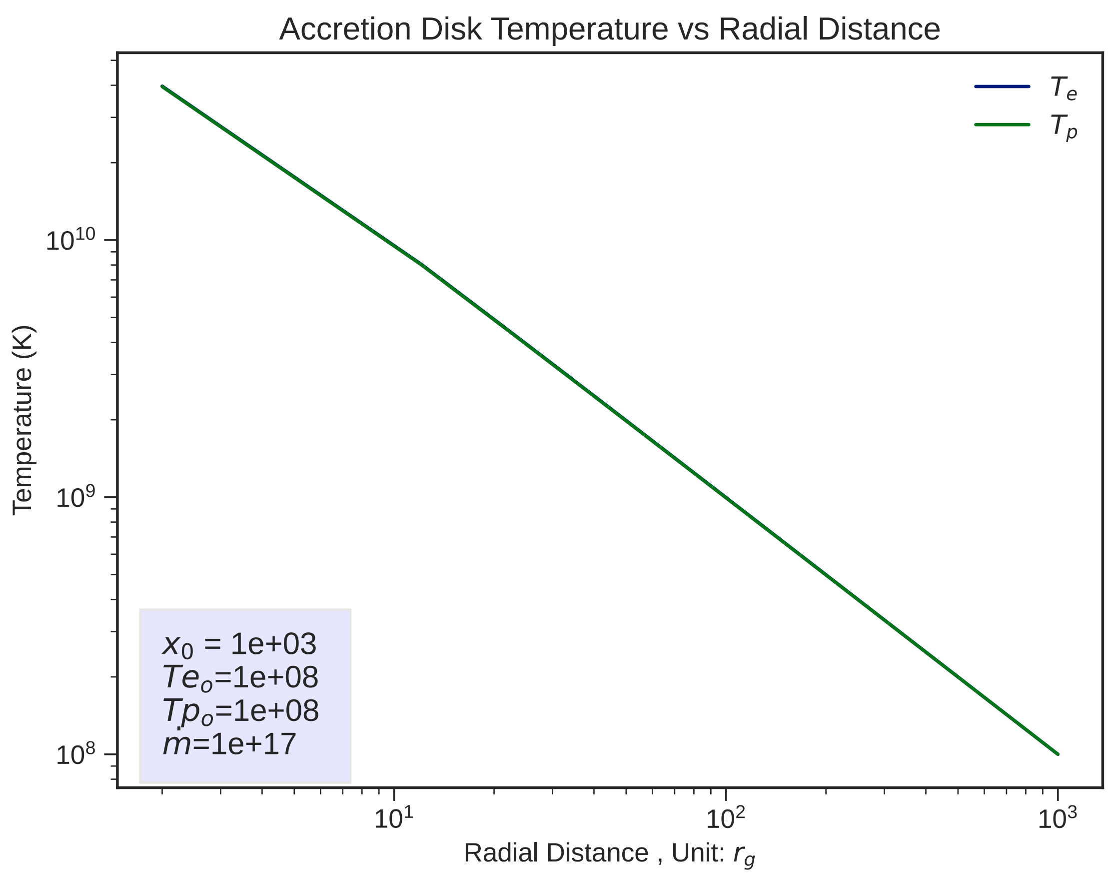
2.3 Problem (c)

Initial Conditions $> x_0 = 10^3 \; Te_0 = 10^8 K \; Tp_0 = 10^8 K \; \dot{m} = 10^{17} gm/cc$

[8]:
```
te_0 = 1e8
tp_0 = 1e8
x_0 = 1e3
m_dot_0 = 1e17
```

```python
x_range = np.linspace(2,1e3 , 100)
t_q1 = []
for x in x_range:
    temp = rk4_cp(float(x),x_0,te_0,tp_0,f_tp_wrap(m_dot_0),f_te_wrap(m_dot_0)␣
↪, tol=1e-5)
    #print('x:{:.2f} ,  T_p: {:.2f} , T_e : {:.2f}'.format(x,temp[0],temp[1]))
    t_q1.append(list(temp))
#data = rk4_cp(x , x_0 , te_0 , tp_0 , f_tp_wrap(m_dot_0) , f_te_wrap(m_dot_0))
```

[18]:
```python
te_0 = 1e8
tp_0 = 1e8
x_0 = 1e3
m_dot_0 = 1e17
t_q1 = np.asarray(t_q1)
te_q1 = t_q1[:,0]
tp_q1 = t_q1[:,1]
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111)
ax.loglog(x_range, te_q1)
ax.loglog(x_range,tp_q1)
ax.set_xlabel('Radial Distance , Unit: $r_g$')
ax.set_ylabel('Temperature (K)')
ax.set_title('Accretion Disk Temperature vs Radial Distance')
ax.text(2,1e8 , '$x_0$ = {:.0e}\n$Te_o$={:.0e}\n$Tp_o$={:.0e} \n$\dot m$={}'.
↪format(x_0,te_0,tp_0,m_dot_0) , bbox = {'facecolor':'blue' , 'alpha':0.1,␣
↪'pad':10 } , fontsize = 14)
plt.legend(['$T_e$','$T_p$'])
plt.show()
```

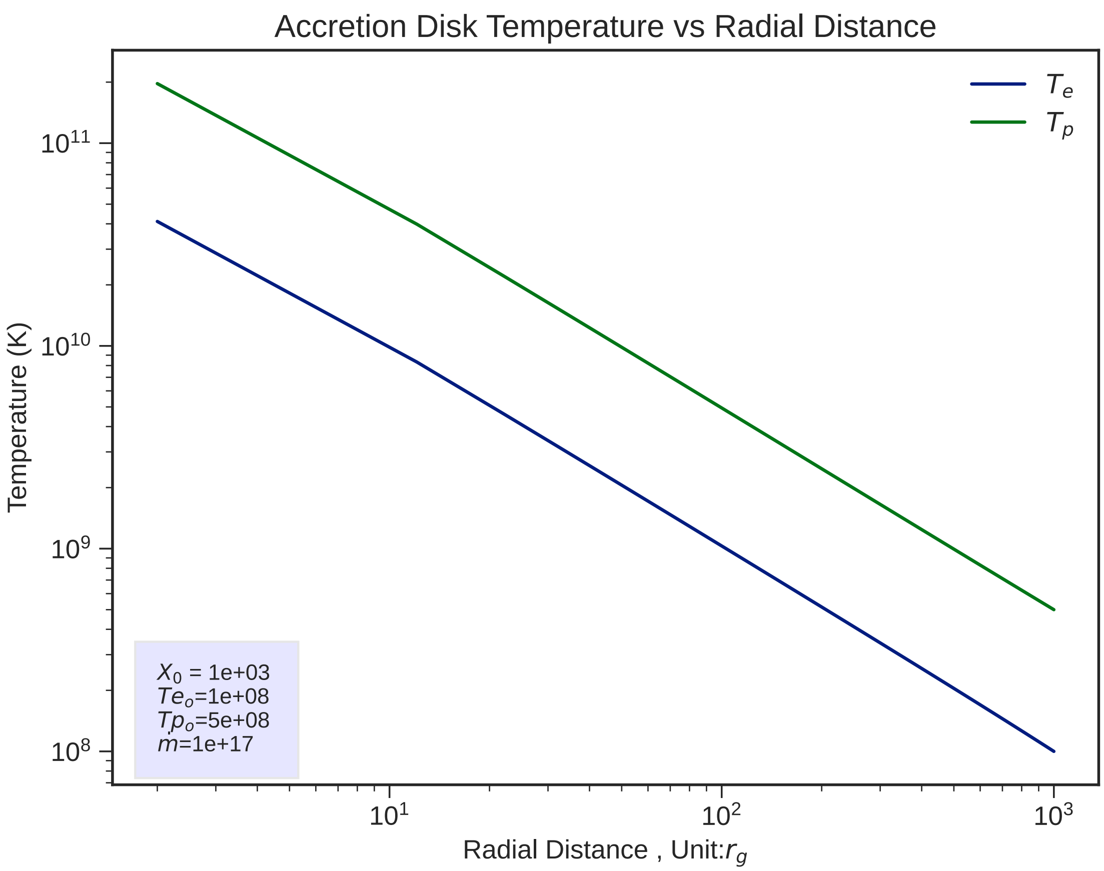


2.4 Problem (c)

Initial Conditions $> x_0 = 10^3 \; Te_0 = 10^8 K \; Tp_0 = 5 \times 10^8 K \; \dot{m} = 10^{17} gm/cc$

[10]:
```
te_0 = 1e8
tp_0 = 5e8
x_0 = 1e3
m_dot_0 = 1e17
x_range = np.linspace(2,1e3 , 100)
t_q2 = []
#print('X , $T_p$ , ')
for x in x_range:
    temp = rk4_cp(x,x_0,te_0,tp_0,f_tp_wrap(m_dot_0),f_te_wrap(m_dot_0) , tol =
 ↪1e-5)
    #print('x:{:.2f} ,  T_p: {:.2f} , T_e : {:.2f}'.format(x,temp[0],temp[1]))
    t_q2.append(list(temp))
#data = rk4_cp(x , x_0 , te_0 , tp_0 , f_tp_wrap(m_dot_0) , f_te_wrap(m_dot_0))
```

[19]:
```
te_0 = 1e8
tp_0 = 5e8
x_0 = 1e3
m_dot_0 = 1e17
t_q2 = np.asarray(t_q2)
te_q2 = t_q2[:,0]
tp_q2 = t_q2[:,1]
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111)
ax.loglog(x_range, te_q2)
ax.loglog(x_range,tp_q2)
ax.set_xlabel('Radial Distance , Unit:$r_g$')
ax.set_ylabel('Temperature (K)')
ax.set_title('Accretion Disk Temperature vs Radial Distance')
ax.text(2,1e8 , '$X_0$ = {:.0e}\n$Te_o$={:.0e}\n$Tp_o$={:.0e} \n$\dot m$={}'.
↪format(x_0,te_0,tp_0,m_dot_0) , bbox = {'facecolor':'blue' , 'alpha':0.1,␣
↪'pad':10 } , fontsize = 10)
plt.legend(['$T_e$','$T_p$'])
plt.show()
```



2.5 Problem (c)

Initial Conditions

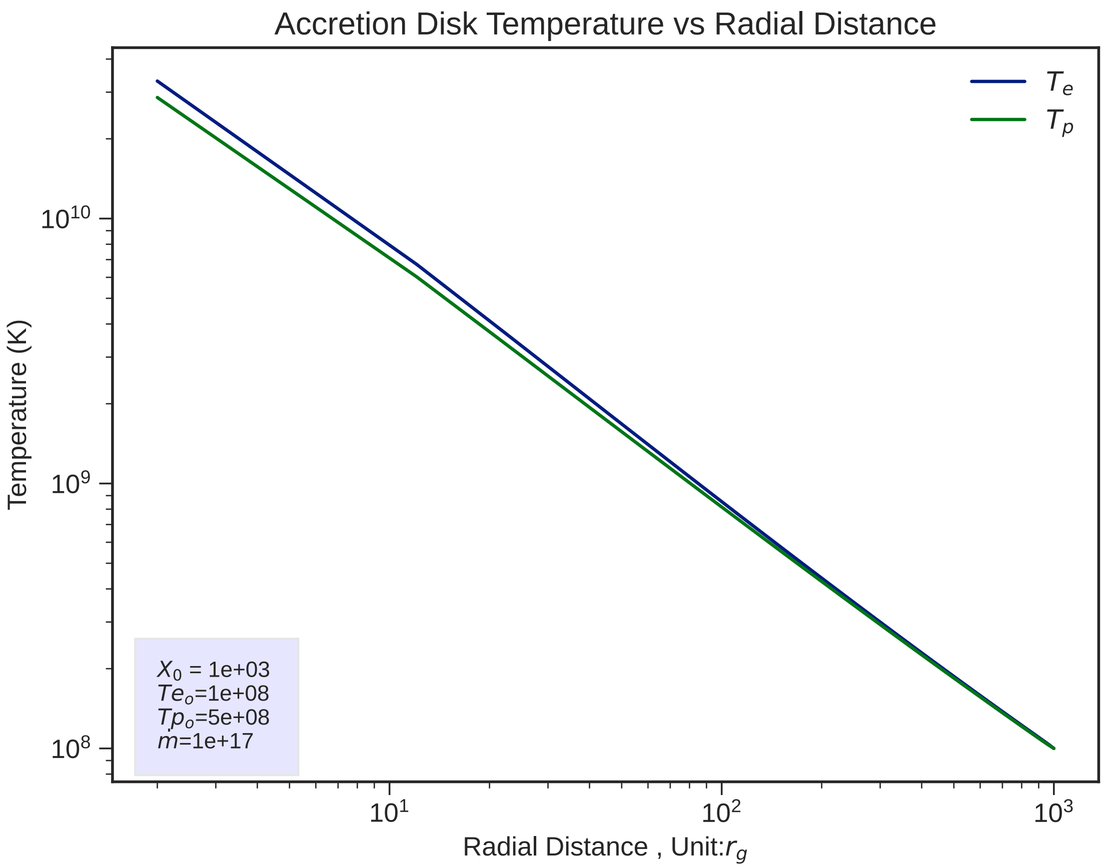
$$x_0 = 10^3\ ,\ Te_0 = 10^8 K\ ,\ Tp_0 = 10^8 K\ ,\ \dot{m} = 10^{19} gm/cc$$

[12]:
```
te_0 = 1e8
tp_0 = 1e8
x_0 = 1e3
m_dot_0 = 1e19
x_range = np.linspace(2,1e3 , 100)
t_q3 = []

for x in x_range:
    temp = rk4_cp(float(x),x_0,te_0,tp_0,f_tp_wrap(m_dot_0),f_te_wrap(m_dot_0)␣
 ↪, tol = 1e-5)
    #print('x:{:.2f} ,  T_p: {:.2f} , T_e : {:.2f}'.format(x,temp[0],temp[1]))
    t_q3.append(list(temp))

#data = rk4_cp(2.0 , x_0 , te_0 , tp_0 , f_tp_wrap(m_dot_0) ,␣
 ↪f_te_wrap(m_dot_0) , tol=1e-2)
#print(data)
```

[20]:
```
t_q3 = np.asarray(t_q3)
te_q3 = t_q3[:,0]
tp_q3 = t_q3[:,1]
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111)
ax.loglog(x_range, te_q3)
ax.loglog(x_range,tp_q3)
ax.set_xlabel('Radial Distance , Unit:$r_g$')
ax.set_ylabel('Temperature (K)')
ax.set_title('Accretion Disk Temperature vs Radial Distance')
ax.text(2,1e8 , '$X_0$ = {:.0e}\n$Te_o$={:.0e}\n$Tp_o$={:.0e} \n$\dot m$={}'.
 ↪format(x_0,te_0,tp_0,m_dot_0) , bbox = {'facecolor':'blue' , 'alpha':0.1,␣
 ↪'pad':10 } , fontsize = 10)
plt.legend(['$T_e$','$T_p$'])
plt.show()
```


Accretion Disk Temperature vs Radial Distance
T_e
T_p
X_0 = 1e+03
Te_o=1e+08
Tp_o=5e+08
$\dot{m}$=1e+17
10^{10}
10^9
10^8
Temperature (K)
10^1
10^2
10^3
Radial Distance , Unit:r_g