In this programme I will calculate the neutron star mass and radius by solving the Tolman-Oppenheimer-Volkoff equations :

$$\frac{dp}{dr} = -\frac{Gm(r)\epsilon(r)}{r^2c^2}\left(1+\frac{p(r)}{\epsilon(r)}\right)\left(1+\frac{4\pi r^3 p(r)}{m(r)c^2}\right)\left(1-\frac{2Gm(r)}{rc^2}\right)^{-1}$$

$$\frac{dm}{dr} = \frac{4\pi r^2\epsilon(r)}{c^2}$$

To solve this equation we will write the rhs of this TOV Equations in rescale units : energy density and pressure are in units of GeV/fm^3.We write mass in units of solar mass $M_{solar} = 2\times10^{30}$ kg and radius in units of Schwarzschild radius $R_s = \frac{2GM_{solar}}{c^2} = 2.95$ km and a quantity α is defined as $\alpha = \frac{M_{solar}c^2}{R_s^3} = 41.325$ in units of $\frac{GeV}{fm^3}$

Then the formula will take the form :

$$\frac{dp}{dx} = -\frac{0.5\times m\left(1+\frac{p}{\epsilon}\right)\left(1+\frac{4\pi p}{\alpha m}x^3\right)}{(x^2-mx)}$$

$$\frac{dm}{dx} = \frac{4\pi x^2\epsilon}{\alpha}$$

Here m is mass in units of solar mass and x is the radial distance in units of Schwarzchild radius.

Here we will use a simple equation of state for a non-interacting gas of neutrons.The equation of state is specified through a simple numerical fitting formula :

$\epsilon(p) = a_{nr}p^{\frac{3}{5}} + a_r p$ where a_{nr} and a_r are constants we have to choose properly.

In [1]:
```
import numpy as np
import scipy.integrate as spi
import matplotlib.pyplot as plt
import math
```

Here we have defined a function called " $tovrhs$ " that takes two parameters : " $initial$ " and " x ". This function calculates the right-hand side (RHS) of the Tolman-Oppenheimer-Volkoff (TOV) equations.Assuming $'initial'$ is a list like object we have assigned first element of this list to the " p " variable and second element of the list to the " m " variable. Then we have called a function " eos " with " p " variable as an argument and assigned the returned value to the variable " ϵ ".It calculates the energy density based on the given pressure.Then we have assigned the components of rhs of the " TOV " equations by " one " and " two " and the calculated values of " one " and " two " are stored in a list " f " which is returned as the output of the " $tovrhs$ " function.

In [2]:
```
def tovrhs(initial,x):#This function takes two parameters "initial" and "x" that represent the inputs.
    p=initial[0] #Here the first element in the initial list is assigned to variable "pres"
    m=initial[1] #Here the second element in the initial list is assigned to variable "pres"
    α=41.325 #alfa= (M_solar*c**2/R_s**3) in units of GeV/fm^3
    ϵ=eos(p) #Here an equation of state is called as a function of pressure and is assigned to the energy density
    one=-0.5*ϵ*m*(1+(p/ϵ))*(1+(4*np.pi/α)*(p/m)*x**3)/(x**2-m*x) #RHS of the TOV equation dp/dr
    two=4*np.pi*x**2*ϵ/α # RHS of the TOV Equation dm/dr
    f=[one,two] #it stores the calculated values from the previous lines into a list f.
    return f
```

Below we have defined a function eos which has taken a parameter p representing pressure. Then we have assigned the values to the coefficients a_{nr} and a_r.These values are coefficients used in a fit to the non-interacting neutron matter pressure. Then a conditional statement is used to handle the case where given pressure is non-negative.It sets a small positive pressure value 10^{-8} GeV/fm^3 to avoid undefined behaviour that could occur when using non-positive value. The last line calculated the equation of state value and the calculated value is returned as the output of the function.

In summary the function eos has taken a pressure value as an input . If pressure is non-negative it sets a very small value if necessary,Then it calculated the equation of state using the given pressure and returned the result.

In [3]:
```
def eos(p):
    anr=4.015 #These are basically coefficients used in the fit to the non-interacting neutron matter pressure.
    ar=2.137
    if p<=0: # It is used to handle the case where given pressure is non-negative.It sets a small positive pressure value to avoid undefined behaviour that could occur when using non-positive value.
        p=1.e-8
    return anr*p**(3/5)+ar*p
```

In below a function $tsolve$ is called with two arguments $p0$ and $xfinal$ representing central pressure and final value of the independent variable respectively.

Then the function eos is called with central pressure as an argument and the returned value is assigned to the variable ϵ.It calculates the energy density based on the given central preesure.

dx represents the step size in the independent variable x and is used for numerical integration.

Then the initial conditions for the TOV equations are written in a form of tupple assigned to the variable $initial$ containing the central pressure $p0$ and initial mass calculated using the formula $\frac{4}{3}\pi\epsilon d(x^3)/\alpha$.

Now an array x is created which represents the values of the independent variable over which the differential equations will be solved.

Then the differential equations are solved using $spi.odeint()$.It uses the " $tovrhs$ " as the ODE system,the initial conditions from " $initial$ " and the array of values " X ".

Variables "rstar","mstar","count" are initialized with value 0. They are used to store values related to some physical quantities.By setting it zero initially,it ensures that variables are initialized before any calculations or updates are performed." $rstar$ " and " $mstar$ " represent final value of radial distance or radius and mass respectively i.e., the radial distance and at last iteration of the loop.

Next, three empty lists " m "," r "," p " are created to store updated values of " $rstar$ "," $mstar$ " and pressure(" i ") during the each loop iteration.

Now,a loop is started that iterates over the values in the first column $([:,0])$ of the " tov_sol " array.Each value represents a pressure at a particular iteration.

Then if the pressure " i " is greater than 10^{-7},the " $count$ " variable is incremented by 1, the values of " $rstar$ " and " $mstar$ " are updated and then appended to the the " r " and " m " list respectively.Finally, it appends the current value of pressure to the " p " list.

" $count$ " serves as a counter variable that keeps track of the number of iterations or steps in which the pressure " i " is greater than $1.e-7$. It is used to access and store specific values related to the $mass(mstar)$ from the tov_sol array.

Then fuction $tsolve$ returns the final values of the radial distance " $(rstar)$ "and mass " $(mstar)$ " as well as the lists " r "," m ", and " p ", which contain the updated values of the radial distance, mass, and pressure at each iteration of the loop as output of it as a tuple.

F is the resulting tuple contains all these quantities.

In [55]:
```
def tsolve(p0,xfinal): # This function takes two parameters "pcent" that raprents central pressure and "xfinal" is final value of independent parameters.
    α=41.325

    ϵ=eos(p0)

    dx=0.00025 #This represents the step size in independent variables.

    initial=p0,4*np.pi*ϵ*dx**3/(3*α) #This line caculates the initial conditions of differential equations.It assigns a tuple to the variable "initial" containing the "central pressure(pcent)" and "initia
    X=np.arange(dx,xfinal,dx) #This line generates an array using np.arange()

    tov_sol=spi.odeint(tovrhs,initial,X) #It solves the ordinary differential equations using the "tovrhs" function,initial conditions "initial" and array of values "x".

    rstar=0 #We initialize "rstar","mstar","count" with value 0. They are used to store values related to some physical quantities.By setting it zero initially,it ensures that variables are initialized bef
    mstar=0
    count=0

    m=[] # These are the three empty lists are created to store updated values of rstar,mstar and i
    r=[]
    p=[]
    k=[]
    for i in tov_sol[:,0]: #This starts a loop that iterates over the values in the first column ([:, 0]) of the tov_sol array.Each value represent the pressure.

        if i>1.e-7:
            count=count+1
            rstar=rstar+2.95*dx
            r.append(rstar) #It appends the current value of rstar to the list r.It stores the values of the radial distance.
            mstar=tov_sol[count,1] #This line assigns the value from tov_sol array at index "[count,1]" to the variable "mstar".It represents the mass at thecurrent iteration.
            m.append(mstar) #It appends the current value of mstar to the list m.It stores the values of the mass.
            p.append(i)#It appends the current value of "i" to the list w.It stores the values of the pressure.
            k.append(i*(1.6*10**36))
    return rstar,mstar,r,m,p,k #Finally the values of "rstar","mstar","r","m","w" are returned as a tuple.

F=tsolve(0.15,10) #It assigns the returned tuple from "tsolve" to the variable F,i.e.,F becomes a tuple containing the values returned by the "tsolve" function.
```

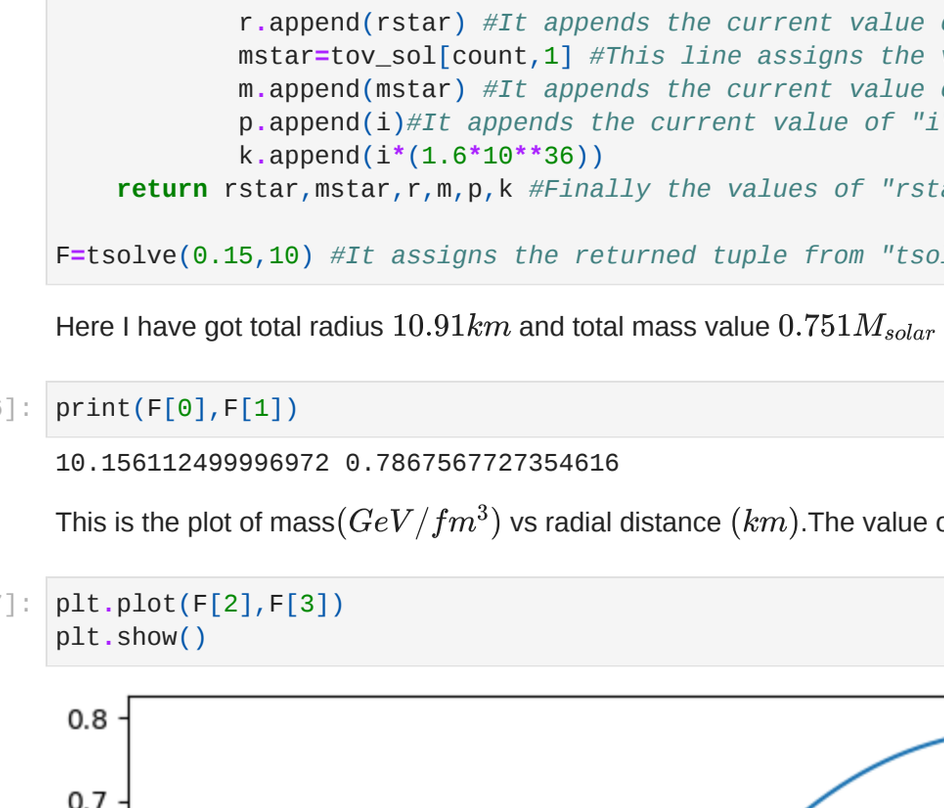
Here I have got total radius $10.91km$ and total mass value $0.751M_{solar}$

In [56]:
```
print(F[0],F[1])
```

```
10.156112499996972 0.7867567727354616
```

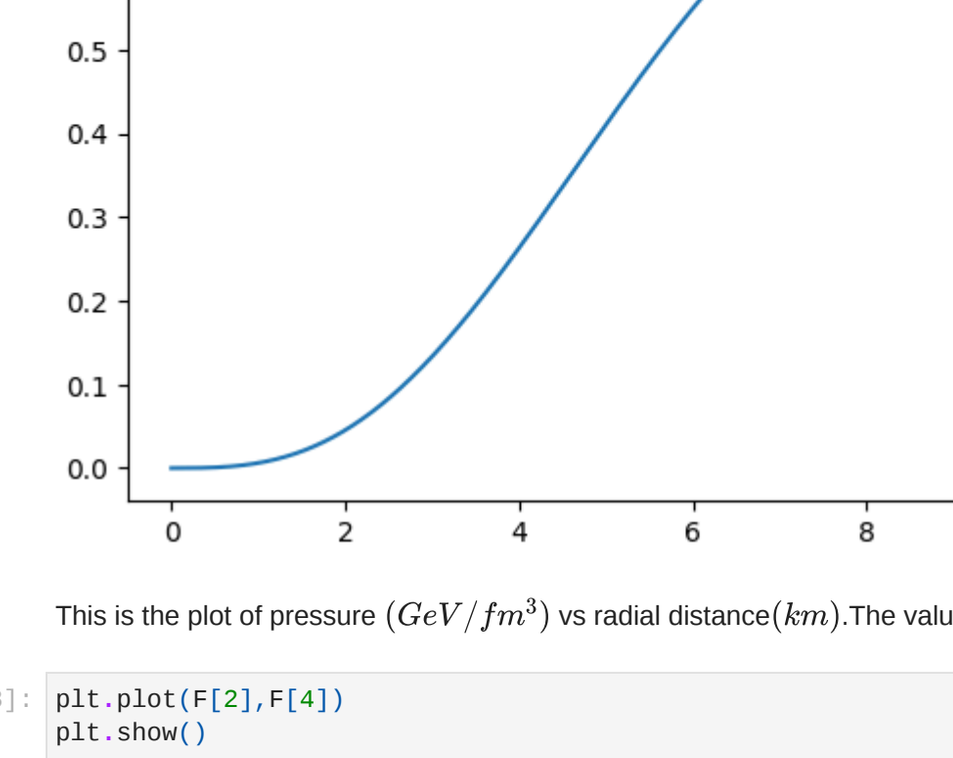
This is the plot of mass(GeV/fm^3) vs radial distance (km).The value of radial distance where the mass saturates gives us total radius value.

In [57]:
```
plt.plot(F[2],F[3])
plt.show()
```

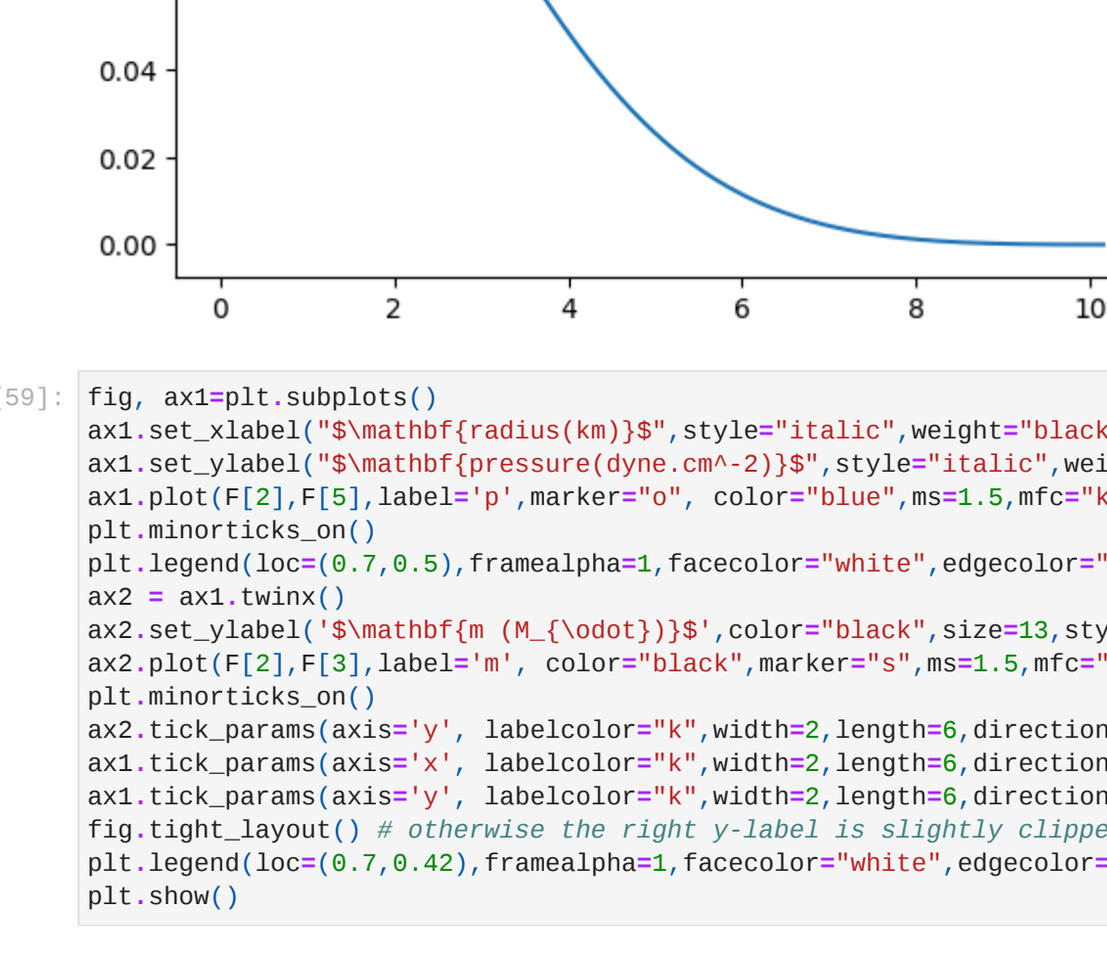


This is the plot of pressure (GeV/fm^3) vs radial distance(km).The value of radial distance where pressure tends to zero gives us outer radius of the star.

In [58]:
```
plt.plot(F[2],F[4])
plt.show()
```



In [59]:
```
fig, ax1=plt.subplots()
ax1.set_xlabel("$\mathbf{radius(km)}$",style="italic",weight="black",labelpad=4,size=13)
ax1.set_ylabel("$\mathbf{pressure(dyne.cm^-2)}$",style="italic",weight="black",labelpad=4,size=13)
ax1.plot(F[2],F[5],label='p',marker="o", color="blue",ms=1.5,mfc="k",ls="solid",lw=1)
plt.minorticks_on()
plt.legend(loc=(0.7,0.5),framealpha=1,facecolor="white",edgecolor="black",fontsize=12,markerfirst=False)
ax2 = ax1.twinx()
ax2.set_ylabel('$\mathbf{m (M_{\odot})}$',color="black",size=13,style="italic",weight="black",labelpad=4)
ax2.plot(F[2],F[3],label='m', color="black",marker="s",ms=1.5,mfc="k",ls="solid",lw=1)
plt.minorticks_on()
ax2.tick_params(axis='y', labelcolor="k",width=2,length=6,direction="out",pad=10,right=True)
ax1.tick_params(axis='x', labelcolor="k",width=2,length=6,direction="out",pad=10)
ax1.tick_params(axis='y', labelcolor="k",width=2,length=6,direction="out",pad=10)
fig.tight_layout() # otherwise the right y-label is slightly clipped,
plt.legend(loc=(0.7,0.42),framealpha=1,facecolor="white",edgecolor="black",fontsize=12,markerfirst=False)
plt.show()
```



A function "M_R" is called that take two parameters " $pmin$ " and " $pmax$ " representing minimum and maximum value of pressure.

" $imax$ " represents maximum number of iterations in the subsequent loop.

Here three numpy arrays " $p0$ "," M "," R " of desired size("imax") are created and their all elements are initially set to zero.These arrays will store the values of the pc,radius,mass during the iteration of the loop.

Then a loop is started which iterates over the range of " $imax$ " which is from 0 to 29 $(30$iterations).In each iteration it calculates the value of " $p0[i]$ " based on the index " i ".

Then the radius and mass is calculated at index " i " by calling the " $tsolve$ " function with " $p0[i]$ " as the central pressure and 10 as the " $xfinal$ " value. The " $tsolve$ " function returns a tuple containing $(rstar, mstar, r, m, p)$.Since the radius and mass is stored at index 0 and 1 respectively within the tuple, $[0]$ and $[1]$ is used to access radius and mass value .The calculated radius and mass are then assigned to " $R[i]$ " and " $M[i]$ ". Here " $tsolve$ " function is utilized to calculate the radius and mass for each " $p0[i]$ " value within a loop.The resulting values are stored in the R and M arrays, respectively, at the corresponding indices i.

Next,the function will return these three arrays $(p0, R, M)$ as a tuple.Each array contains the values calculated during the loop and order of the arrays in the tuple mathches the order in which they are listed in the return statement.

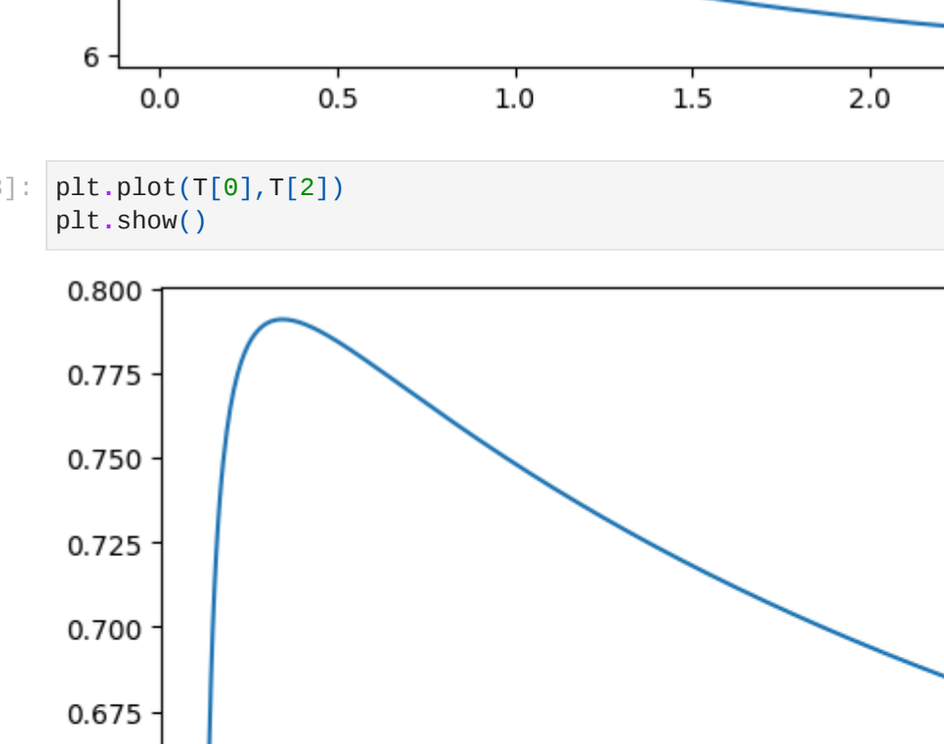
The returned tuple is assigned to the variable " T " from "M_R-function".T contain three arrays : $T[0]$ will be the "p0" array, $T[1]$ will be the "R" array, and $T[2]$ will be the "M" array.

In [60]:
```
def M_R(pmin,pmax):
    imax=500 # It represents the maximum number of iterations in the subsequent loop.
    p0=np.zeros(imax) #Here a numpy array pc of desired size("imax")  is created and its all elements are initially set to zero.This array will store the values of pc during the iteration of the loop.
    M=np.zeros(imax)
    R=np.zeros(imax)
    k0=np.zeros(imax)
    for i in range(imax):
        p0[i]=pmin + (pmax-pmin)*(i)/500 # in each iteration this line calculates the value of pc[i] based on the index i.
        k0[i]=(pmin + (((pmax-pmin)*(i))/500))*(1.6*(10**36))#in dyne.cm^-2

        R[i]=tsolve(p0[i],10)[0]
        # This line calculates the radius at index i by calling the tsolve function with pc[i] as the central pressure and 10 as the xfinal value. The tsolve function returns a tuple containing (rstar, mst
        M[i]=tsolve(p0[i],10)[1]
        #  These lines utilize the tsolve function to calculate the radius and mass for each pc[i] value within a loop. The resulting values are stored in the radius and mass arrays, respectively, at the c
    return p0,R,M,k0 #The function will return these three arrays as a tuple.Each array contains the values calculated during the loop and order of the arrays in the tuple mathches the order in which they

T=M_R(0.01,2.5) # This is the resulting tuple containing three arrays: T[0] will be the "pc" array, T[1] will be the "radius" array, and T[2] will be the "mass" array.
```

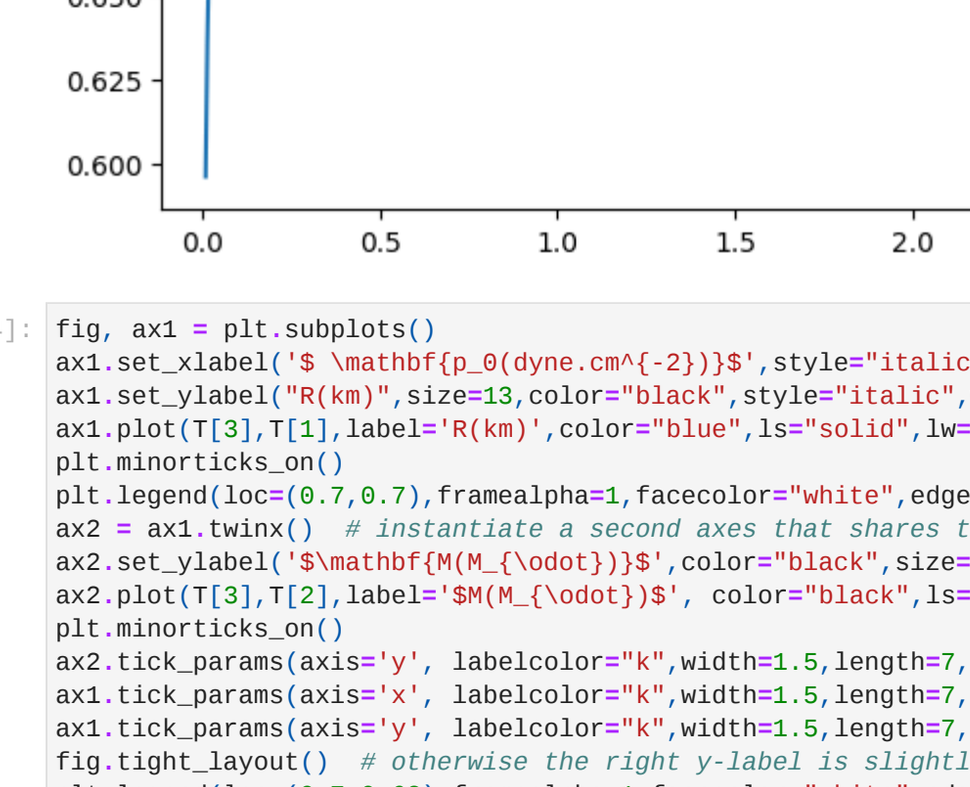
In [61]:
```
print (T[0])
print(T[1])
print(T[2])
```

```
[0.01    0.01498 0.01996 0.02494 0.02992 0.0349  0.03988 0.04486 0.04984
 0.05482 0.0598  0.06478 0.06976 0.07474 0.07972 0.0847  0.08968 0.09466
 0.09964 0.10462 0.1096  0.11458 0.11956 0.12454 0.12952 0.1345  0.13948
 0.14446 0.14944 0.15442 0.1594  0.16438 0.16936 0.17434 0.17932 0.1843
 0.18928 0.19426 0.19924 0.20422 0.2092  0.21418 0.21916 0.22414 0.22912
 0.2341  0.23908 0.24406 0.24904 0.25402 0.259   0.26398 0.26896 0.27394
 0.27892 0.2839  0.28888 0.29386 0.29884 0.30382 0.3088  0.31378 0.31876
 0.32374 0.32872 0.3337  0.33868 0.34366 0.34864 0.35362 0.3586  0.36358
 0.36856 0.37354 0.37852 0.3835  0.38848 0.39346 0.39844 0.40342 0.4084
 0.41338 0.41836 0.42334 0.42832 0.4333  0.43828 0.44326 0.44824 0.45322
 0.4582  0.46318 0.46816 0.47314 0.47812 0.4831  0.48808 0.49306 0.49804
 0.50302 0.508   0.51298 0.51796 0.52294 0.52792 0.5329  0.53788 0.54286
 0.54784 0.55282 0.5578  0.56278 0.56776 0.57274 0.57772 0.5827  0.58768
 0.59266 0.59764 0.60262 0.6076  0.61258 0.61756 0.62254 0.62752 0.6325
 0.63748 0.64246 0.64744 0.65242 0.6574  0.66238 0.66736 0.67234 0.67732
 0.6823  0.68728 0.69226 0.69724 0.70222 0.7072  0.71218 0.71716 0.72214
 0.72712 0.7321  0.73708 0.74206 0.74704 0.75202 0.757   0.76198 0.76696
 0.77194 0.77692 0.7819  0.78688 0.79186 0.79684 0.80182 0.8068  0.81178
 0.81676 0.82174 0.82672 0.8317  0.83668 0.84166 0.84664 0.85162 0.8566
 0.86158 0.86656 0.87154 0.87652 0.8815  0.88648 0.89146 0.89644 0.90142
 0.9064  0.91138 0.91636 0.92134 0.92632 0.9313  0.93628 0.94126 0.94624
 0.95122 0.9562  0.96118 0.96616 0.97114 0.97612 0.9811  0.98608 0.99106
 0.99604 1.00102 1.006   1.01098 1.01596 1.02094 1.02592 1.0309  1.03588
 1.04086 1.04584 1.05082 1.0558  1.06078 1.06576 1.07074 1.07572 1.0807
 1.08568 1.09066 1.09564 1.10062 1.1056  1.11058 1.11556 1.12054 1.12552
 1.1305  1.13548 1.14046 1.14544 1.15042 1.1554  1.16038 1.16536 1.17034
 1.17532 1.1803  1.18528 1.19026 1.19524 1.20022 1.2052  1.21018 1.21516
 1.22014 1.22512 1.2301  1.23508 1.24006 1.24504 1.25002 1.255   1.25998
 1.26496 1.26994 1.27492 1.2799  1.28488 1.28986 1.29484 1.29982 1.3048
 1.30978 1.31476 1.31974 1.32472 1.3297  1.33468 1.33966 1.34464 1.34962
 1.3546  1.35958 1.36456 1.36954 1.37452 1.3795  1.38448 1.38946 1.39444
 1.39942 1.4044  1.40938 1.41436 1.41934 1.42432 1.4293  1.43428 1.43926
 1.44424 1.44922 1.4542  1.45918 1.46416 1.46914 1.47412 1.4791  1.48408
 1.48906 1.49404 1.49902 1.504   1.50898 1.51396 1.51894 1.52392 1.5289
 1.53388 1.53886 1.54384 1.54882 1.5538  1.55878 1.56376 1.56874 1.57372
 1.5787  1.58368 1.58866 1.59364 1.59862 1.6036  1.60858 1.61356 1.61854
 1.62352 1.6285  1.63348 1.63846 1.64344 1.64842 1.6534  1.65838 1.66336
 1.66834 1.67332 1.6783  1.68328 1.68826 1.69324 1.69822 1.7032  1.70818
 1.71316 1.71814 1.72312 1.7281  1.73308 1.73806 1.74304 1.74802 1.753
 1.75798 1.76296 1.76794 1.77292 1.7779  1.78288 1.78786 1.79284 1.79782
 1.8028  1.80778 1.81276 1.81774 1.82272 1.8277  1.83268 1.83766 1.84264
 1.84762 1.8526  1.85758 1.86256 1.86754 1.87252 1.8775  1.88248 1.88746
 1.89244 1.89742 1.9024  1.90738 1.91236 1.91734 1.92232 1.9273  1.93228
 1.93726 1.94224 1.94722 1.9522  1.95718 1.96216 1.96714 1.97212 1.9771
 1.98208 1.98706 1.99204 1.99702 2.002   2.00698 2.01196 2.01694 2.02192
 2.0269  2.03188 2.03686 2.04184 2.04682 2.0518  2.05678 2.06176 2.06674
 2.07172 2.0767  2.08168 2.08666 2.09164 2.09662 2.1016  2.10658 2.11156
 2.11654 2.12152 2.1265  2.13148 2.13646 2.14144 2.14642 2.1514  2.15638
 2.16136 2.16634 2.17132 2.1763  2.18128 2.18626 2.19124 2.19622 2.2012
 2.20618 2.21116 2.21614 2.22112 2.2261  2.23108 2.23606 2.24104 2.24602
 2.251   2.25598 2.26096 2.26594 2.27092 2.2759  2.28088 2.28586 2.29084
 2.29582 2.3008  2.30578 2.31076 2.31574 2.32072 2.3257  2.33068 2.33566
 2.34064 2.34562 2.3506  2.35558 2.36056 2.36554 2.37052 2.3755  2.38048
 2.38546 2.39044 2.39542 2.4004  2.40538 2.41036 2.41534 2.42032 2.4253
 2.43028 2.43526 2.44024 2.44522 2.4502  2.45518 2.46016 2.46514 2.47012
 2.4751  2.48008 2.48506 2.49004 2.49502]
[15.2640375 14.455     13.887125  13.452     13.099475  12.804475
 12.550775  12.32805   12.1304    11.9534    11.7918875 11.645125
 11.509425  11.3833125 11.2667875 11.1576375 11.055125  10.9585125
 10.8670625 10.780775  10.6989125 10.621475  10.5469875 10.4761875
 10.4076    10.3427    10.2800125 10.220275  10.1620125 10.1059625
 10.052125  10.0005     9.95035    9.901675   9.854475   9.80875
  9.7637625  9.7209875  9.67895    9.6383875  9.5985625  9.5602125
  9.5226     9.485725   9.450325   9.4156625  9.381      9.3478125
  9.3153625  9.28365    9.252675   9.2224375  9.1922     9.1634375
  9.1354125  9.1073875  9.0793625  9.0528125  9.0262625  9.00045
  8.975375   8.9503     8.9259625  8.901625   8.878025   8.854425
  8.8323     8.8094375  8.7873125  8.765925   8.7445375  8.72315
  8.7025     8.68185    8.6619375  8.642025   8.62285    8.603675
  8.5845     8.5660625  8.547625   8.5291875  8.5114875  8.4937875
  8.476825   8.459125   8.4421625  8.4259375  8.4097125  8.39275
  8.3772625  8.3610375  8.34555    8.3300625  8.314575   8.299825
  8.2843375  8.2695875  8.255575   8.240825   8.2268125  8.2128
  8.1987875  8.184775   8.1715     8.1574875  8.1442125  8.1309375
  8.1184     8.105125   8.0925875  8.08005    8.0675125  8.054975
  8.043175   8.0306375  8.0188375  8.0070375  7.9952375  7.9834375
  7.972375   7.960575   7.9495125  7.93845    7.9273875  7.916325
  7.9052625  7.8942     7.883875   7.87355    7.8624875  7.8521625
  7.8418375  7.8315125  7.821925   7.8116     7.8020125  7.7916875
  7.7821     7.7725125  7.762925   7.7533375  7.74375    7.7341625
  7.7253125  7.715725   7.706875   7.6972875  7.6884375  7.6795875
  7.6707375  7.6618875  7.6530375  7.644925   7.636075   7.6279625
  7.6191125  7.611      7.60215    7.5940375  7.585925   7.5778125
  7.5697     7.5615875  7.553475   7.5461     7.5379875  7.529875
  7.5225     7.5143875  7.5070125  7.4996375  7.4922625  7.4848875
  7.476775   7.4701375  7.462025   7.4553875  7.4480125  7.4406375
  7.434      7.426625   7.41925    7.4126125  7.405975   7.3986
  7.3919625  7.385325   7.37795    7.3713125  7.364675   7.3580375
  7.3514     7.3455     7.3388625  7.332225   7.3255875  7.3196875
  7.31305    7.3064125  7.3005125  7.293875   7.287975   7.282075
  7.2754375  7.2695375  7.2636375  7.2577375  7.2518375  7.2459375
  7.2400375  7.2341375  7.2282375  7.2223375  7.2164375  7.2105375
  7.205375   7.199475   7.193575   7.1884125  7.1825125  7.1766125
  7.17145    7.16555    7.1603875  7.155225   7.149325   7.1441625
  7.139      7.1338375  7.1279375  7.122775   7.1176125  7.11245
  7.1072875  7.102125   7.0969625  7.0918     7.0866375  7.0822125
  7.07705    7.0718875  7.066725   7.0623     7.0571375  7.051975
  7.04755    7.0423875  7.0379625  7.0328     7.028375   7.0232125
  7.0187875  7.013625   7.0092     7.004775   6.9996125  6.9951875
  6.9907625  6.9863375  6.9819125  6.9774875  6.9730625  6.9679
  6.963475   6.95905    6.954625   6.9502     6.945775   6.9420875
  6.9376625  6.9332375  6.9288125  6.9243875  6.9207     6.916275
  6.91185    6.907425   6.9037375  6.8993125  6.8948875  6.8912
  6.886775   6.8830875  6.8786625  6.874975   6.87055    6.8668625
  6.8624375  6.85875    6.8550625  6.8506375  6.84695    6.8432625
  6.8388375  6.83515    6.8314625  6.827775   6.82335    6.8196625
  6.815975   6.8122875  6.8086     6.8049125  6.801225   6.7975375
  6.79385    6.7901625  6.786475   6.7827875  6.7791     6.7754125
  6.771725   6.7680375  6.76435    6.7606625  6.7577125  6.754025
  6.7503375  6.74665    6.7429625  6.7400125  6.736325   6.7326375
  6.7296875  6.726      6.7223125  6.7193625  6.715675   6.712725
  6.7090375  6.70535    6.7024     6.6987125  6.6957625  6.692075
  6.689125   6.686175   6.6824875  6.6795375  6.67585    6.6729
  6.66995    6.6662625  6.6633125  6.6603625  6.656675   6.653725
  6.650775   6.647825   6.6441375  6.6411875  6.6382375  6.6352875
  6.6323375  6.62865    6.6257     6.62275    6.6198     6.61685
  6.6139     6.61095    6.608      6.60505    6.6021     6.59915
  6.5962     6.59325    6.5903     6.58735    6.5844     6.58145
  6.5785     6.57555    6.5726     6.56965    6.5674375  6.5644875
  6.5615375  6.5585875  6.5556375  6.5526875  6.550475   6.547525
  6.544575   6.541625   6.5394125  6.5364625  6.5335125  6.5313
  6.52835    6.5254     6.5231875  6.5202375  6.5172875  6.515075
  6.512125   6.5099125  6.5069625  6.50475    6.5018     6.49885
  6.4966375  6.4936875  6.491475   6.488525   6.4863125  6.4833625
  6.48115    6.4789375  6.4759875  6.473775   6.470825   6.4686125
  6.4664     6.46345    6.4612375  6.4582875  6.456075   6.4538625
  6.45165    6.4487     6.4464875  6.444275   6.441325   6.4391125
  6.4369     6.4346875  6.432475   6.429525   6.4273125  6.4251
  6.4228875  6.420675   6.417725   6.4155125  6.4133     6.4110875
  6.408875   6.4066625  6.40445    6.4022375  6.400025   6.397075
  6.3948625  6.39265    6.3904375  6.388225   6.3860125  6.3838
  6.3815875  6.379375   6.3771625  6.37495    6.3727375  6.3712625
  6.36905    6.3668375  6.364625   6.3624125  6.3602     6.3579875
  6.355775   6.3535625  6.3520875  6.349875   6.3476625  6.34545
  6.3432375  6.3417625  6.33955    6.3373375  6.335125   6.33365
  6.3314375  6.329225   6.3270125  6.3255375  6.323325   6.3211125
  6.3196375  6.317425   6.3152125  6.3137375  6.311525   6.3093125
  6.3078375  6.305625   6.30415    6.3019375  6.299725   6.29825
  6.2960375  6.2945625  6.29235    6.290875   6.2886625  6.2871875
  6.284975   6.2835   ]
[0.59636208 0.63651156 0.66366762 0.68366703 0.69916406 0.71158798
 0.72178966 0.73032162 0.73756137 0.74377518 0.74915856 0.75385294
 0.75797787 0.76161809 0.76484741 0.7677193  0.77028418 0.77257843
 0.77463716 0.7764863  0.77814926 0.77964663 0.78099523 0.78221074
 0.78330411 0.78428804 0.78517498 0.78596952 0.78668139 0.78731703
 0.78788435 0.78838962 0.78883289 0.78922555 0.78956633 0.78986087
 0.79011192 0.79032403 0.79049902 0.79063986 0.79074846 0.79082736
 0.79087824 0.79090297 0.79090358 0.79088135 0.79083748 0.790774
 0.79069167 0.79059165 0.79047329 0.79034303 0.79019422 0.79003562
 0.78986185 0.7896761  0.78947884 0.78927096 0.78905275 0.78882495
 0.78858846 0.78834294 0.7880891  0.78782765 0.78755961 0.78728437
 0.78700325 0.78671548 0.78642208 0.78612379 0.78581958 0.78551087
 0.78519773 0.78488008 0.78455857 0.78423322 0.78390463 0.78357258
 0.78323708 0.78289843 0.78255739 0.78221359 0.78186741 0.78151833
 0.78116791 0.78081467 0.78045993 0.7801034  0.77974495 0.77938522
 0.77902387 0.77866072 0.77829677 0.7779316  0.7775654  0.77719814
 0.77682983 0.77646093 0.77609137 0.77572087 0.77534978 0.7749784
 0.7746063  0.7742335  0.7738607  0.77348728 0.77311405 0.77274013
 0.77236635 0.77199207 0.77161789 0.77124365 0.77086943 0.77049535
 0.77012102 0.76974699 0.76937309 0.76899937 0.76862583 0.76825248
 0.76787986 0.76750679 0.76713449 0.76676235 0.76639041 0.76601906
 0.76564803 0.76527741 0.76490712 0.76453749 0.76416808 0.76379936
 0.76343096 0.76306316 0.76269592 0.76232915 0.76196305 0.76159776
 0.76123276 0.76086837 0.76050455 0.7601411  0.759779   0.75941686
 0.75905531 0.75869462 0.75833476 0.75797584 0.75761684 0.75725893
 0.7569016  0.7565451  0.75618931 0.75583428 0.75547984 0.75512617
 0.75477315 0.75442093 0.75406943 0.75371849 0.75336848 0.75301935
 0.75267082 0.75232304 0.75197599 0.75162982 0.75128426 0.75093931
 0.75059536 0.75025215 0.74990952 0.74956801 0.7492272  0.74888708
 0.7485476  0.7482099  0.74787212 0.74753459 0.74719827 0.74686176
 0.74652868 0.74619398 0.74586092 0.74552858 0.74519701 0.74486599
 0.74453568 0.74420604 0.74387762 0.74354982 0.74322337 0.74289674
 0.74257146 0.74224665 0.74192352 0.74159985 0.74127731 0.74095623
 0.7406352  0.74031554 0.73999633 0.73967768 0.73935907 0.73904309
 0.73872613 0.73841053 0.73809645 0.73778256 0.73746932 0.73715693
 0.73684534 0.73653442 0.7362242  0.73591484 0.73560607 0.73529807
 0.73499096 0.73468459 0.73437869 0.73407373 0.73376878 0.73346518
 0.73316238 0.73286022 0.73255883 0.73225823 0.73195822 0.73165904
 0.73136057 0.73106281 0.73076566 0.7304693  0.73017373 0.72987879
 0.72958456 0.72929088 0.72899866 0.72870668 0.72841444 0.72812376
 0.72783367 0.72754433 0.72725571 0.72696764 0.72668027 0.72639372
 0.72610783 0.7258225  0.72553794 0.72525393 0.72497071 0.72468803
 0.72440613 0.72412477 0.72384417 0.72356407 0.72328476 0.72300605
 0.72272807 0.72245072 0.72217406 0.72189812 0.72162263 0.72134793
 0.72107351 0.72080002 0.72052714 0.72025489 0.71998327 0.71971236
 0.71944198 0.71917222 0.71890308 0.71863456 0.71836673 0.71809943
 0.71783273 0.71756665 0.71730127 0.71703639 0.71677211 0.71650853
 0.71624545 0.71598305 0.71572118 0.71545995 0.71519923 0.71493921
 0.71467966 0.7144208  0.7141628  0.71390469 0.71364756 0.71339101
 0.71313492 0.71287951 0.71262467 0.71237039 0.7121166  0.71186344
 0.71161086 0.7113588  0.71110733 0.71085641 0.71060604 0.71035623
 0.71010697 0.7098582  0.70961059 0.70936246 0.70911536 0.70886901
 0.70862279 0.70837728 0.70813234 0.70788798 0.70764441 0.70740082
 0.70715825 0.70691571 0.70667399 0.70643283 0.70619225 0.70595191
 0.70571232 0.70547316 0.7052345  0.70499646 0.70475883 0.70452218
 0.70428516 0.70404906 0.70381353 0.70357844 0.70334391 0.70311019
 0.70287632 0.70264325 0.70241106 0.70217903 0.70194728 0.70171596
 0.7014854  0.70125519 0.70102561 0.7007965  0.70056837 0.70033969
 0.70011202 0.69988479 0.69965804 0.6994318  0.69920606 0.6989807
 0.69875627 0.69853151 0.69830747 0.69808414 0.69786135 0.69763886
 0.69741687 0.69719529 0.69697417 0.69675361 0.69653344 0.69631376
 0.69609438 0.69587562 0.69565725 0.69543936 0.69522193 0.69500505
 0.69478854 0.6945724  0.69435678 0.69414159 0.69392684 0.69371266
 0.69349883 0.69328543 0.69307241 0.69285993 0.69264794 0.69243624
 0.69222485 0.69201425 0.691804   0.69159411 0.69138464 0.69117571
 0.69096711 0.69075894 0.6905517  0.69034399 0.69013716 0.68993082
 0.68972482 0.68951932 0.68931414 0.68910953 0.68890523 0.6887014
 0.688498   0.68829499 0.68809247 0.68789027 0.68768866 0.68748734
 0.68728654 0.68708615 0.68688614 0.68668662 0.6864874  0.68628866
 0.68609077 0.68589242 0.68569503 0.68549794 0.68530139 0.68510516
 0.68490945 0.68471404 0.68451917 0.68432469 0.68413063 0.68393685
 0.68374371 0.6835509  0.68335849 0.68316647 0.68297498 0.68278386
 0.68259318 0.68240291 0.682213   0.68202365 0.68183464 0.68164608
 0.68145796 0.68127024 0.68108299 0.6808962  0.6807097  0.68052369
 0.68033811 0.68015289 0.67996832 0.67978408 0.67960027 0.67941691
 0.679234   0.67905149 0.67886943 0.67868777 0.67850684 0.67832624
 0.67814567 0.67796587 0.67778649 0.67760753 0.67742897 0.67725104
 0.67707352 0.67689628 0.67671973 0.67654356 0.6763678  0.67619256
 0.67601775 0.67584351 0.67566963 0.67549627 0.67532338 0.67515102
 0.67497901 0.67480766 0.67463675 0.67446638 0.67429637 0.67412688
 0.67395807 0.67378958 0.67362164 0.67345431 0.67328737 0.67312114
 0.67295519 0.67278977 0.67262509 0.67246079 0.67229705 0.6721339
 0.67197125 0.67180917 0.67164771 0.67148684 0.67132643 0.6711666
 0.67100736 0.67084883]
```

In [62]:
```
plt.plot(T[0],T[1])
plt.show()
```

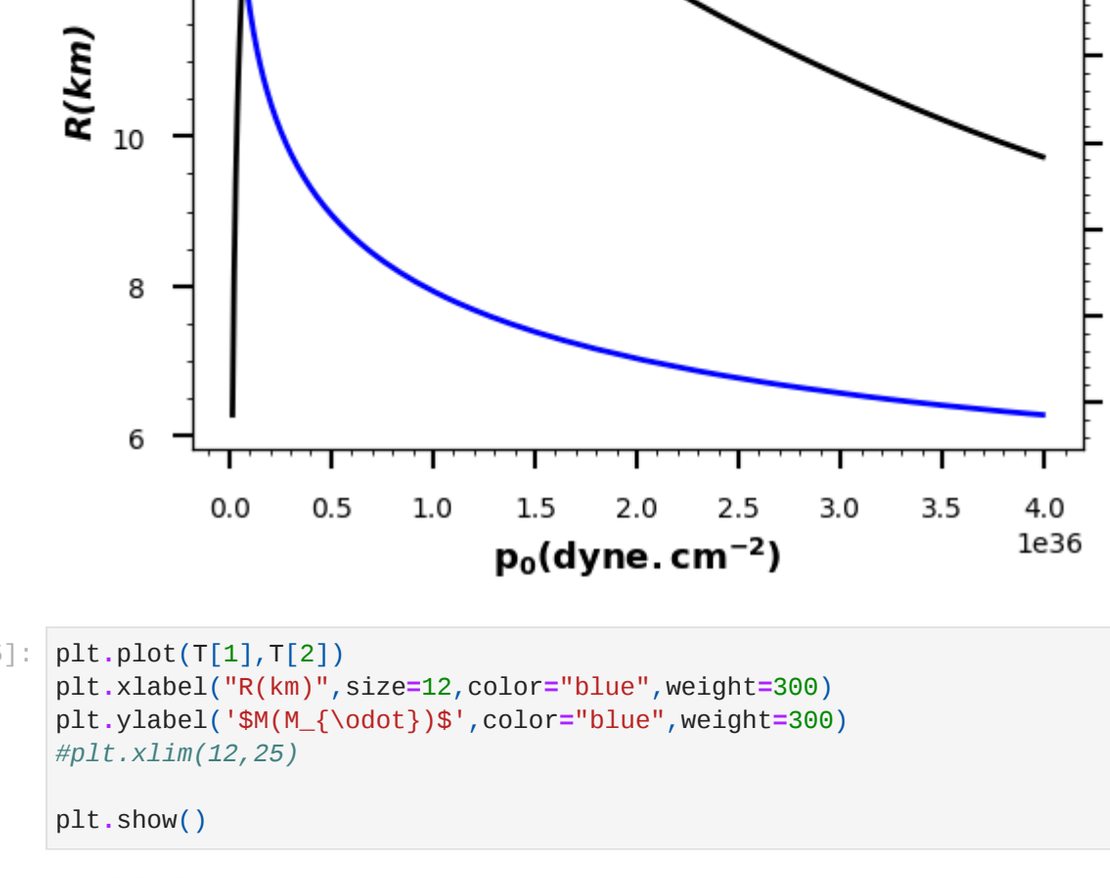


In [63]:
```
plt.plot(T[0],T[2])
plt.show()
```

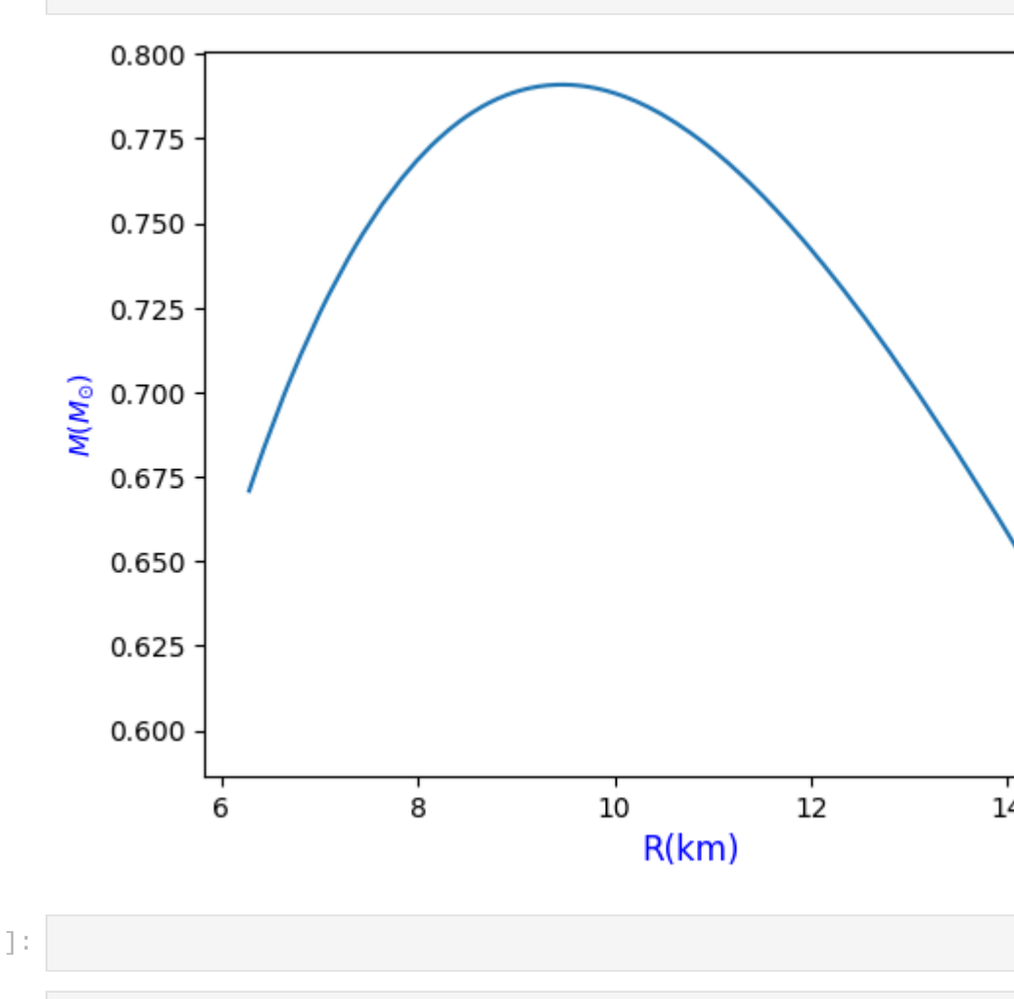


In [64]:
```
fig, ax1 = plt.subplots()
ax1.set_xlabel('$ \mathbf{p_0(dyne.cm^{-2})}$',style="italic",weight="black",labelpad=4,size=13)
ax1.set_ylabel("R(km)",size=13,color="black",style="italic",weight="black",labelpad=4)
ax1.plot(T[3],T[1],label='R(km)',color="blue",ls="solid",lw=2)
plt.minorticks_on()
plt.legend(loc=(0.7,0.7),framealpha=1,facecolor="white",edgecolor="black",fontsize=12,markerfirst=False)
ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis
ax2.set_ylabel('$\mathbf{M(M_{\odot})}$',color="black",size=13,style="italic",weight="black",labelpad=4)  # we already handled the x-label with ax1
ax2.plot(T[3],T[2],label='$M(M_{\odot})$', color="black",ls="solid",lw=2)
plt.minorticks_on()
ax2.tick_params(axis='y', labelcolor="k",width=1.5,length=7,direction="out",pad=10,right=True)
ax1.tick_params(axis='x', labelcolor="k",width=1.5,length=7,direction="out",pad=10)
ax1.tick_params(axis='y', labelcolor="k",width=1.5,length=7,direction="out",pad=10)
fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.legend(loc=(0.7,0.62),framealpha=1,facecolor="white",edgecolor="black",fontsize=12,markerfirst=False)
#plt.title("Calculated masses and radii for white dwarfs using the relativistic polytropic EoS.",style="italic",color="black",pad=15,size=18,weight="black")
max_mass_index = np.argmax(T[2])
pressure_at_max_mass = T[3][max_mass_index]
radius_at_max_mass = T[1][max_mass_index]

print("Pressure at maximum mass:", pressure_at_max_mass)
print("Maximum Mass Value:", T[2][max_mass_index])
print("Radius at maximum mass:", radius_at_max_mass)
plt.show()
```

```
Pressure at maximum mass: 3.665920000000001e+35
Maximum Mass Value: 0.7909035815435332
Radius at maximum mass: 9.450324999997342
```



In [65]:
```
plt.plot(T[1],T[2])
plt.xlabel("R(km)",size=12,color="blue",weight=300)
plt.ylabel('$M(M_{\odot})$',size=12,color="blue",weight=300)
#plt.xlim(12,25)

plt.show()
```



In [ ]:

In [ ]:

In [ ]: