Kvantiki pliroforia
Petridis Petros

June 19, 2022

[1]:

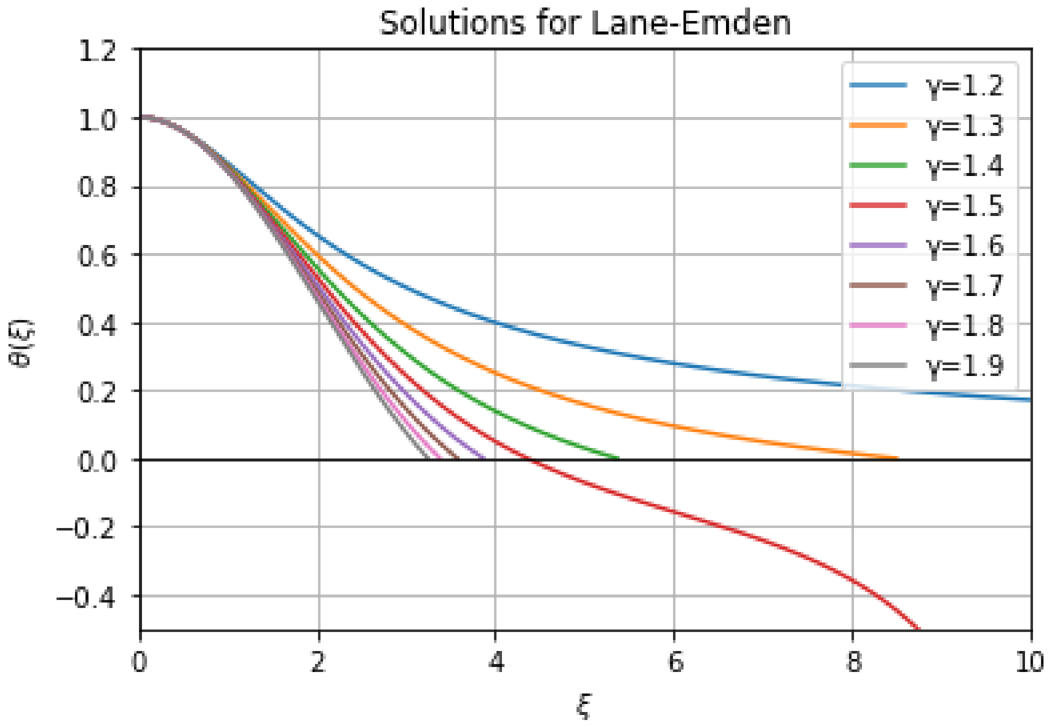
```
from scipy.integrate import odeint
from scipy.integrate import solve_ivp
from scipy.interpolate import interp1d
from scipy.integrate import quad
from scipy.integrate import simpson
from scipy.optimize import fsolve
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

w_initial=[1,0]

def model(x,y):
    yy=y[0]
    z=y[1]

    dw=[[],[]]

    dw[0]=z
    dw[1]=-2/x*z-yy**n
    return dw

x=np.linspace(10**-8,10,2000)

def hak_min(x,theta,n,wmin):
    if callable(theta):
        result=(theta(x)**n)*np.sin(wmin*x)*x
    else:
        result=(theta**n)*np.sin(wmin*x)*x
    return result

def hak(x,omega,theta,n):
    if callable(theta):
        result=(theta(x)**n)*np.sin(omega*x)*x
    else:
        result=(theta**n)*np.sin(omega*x)*x
    return result

def Mass(x,theta,n):
    if callable(theta):
        result=(theta(x)**n)*(x**2)
    else:
        result=(theta**n)*(x**2)
    return result

def S(omega,theta,n,wmin):
    if callable(theta):
        I1=quad(hak_min,xo,xr,args=(theta,n,wmin))
        paranomastis=I1[0]**2

        I2=quad(hak,xo,xr, args=(omega,theta,n))
        I2=I2.real
        arithmitis=I2[0]**2

        paragontas=(wmin/omega)**2

        item=paragontas*arithmitis/paranomastis

        result=item*np.log(item)*omega**2
    else:
        I1=simpson(hak_min(xx,theta,n,wmin),xx)
        paranomastis=I1**2

        I2=simpson(hak(xx,omega,theta,n),xx)
        arithmitis=I2**2

        paragontas=(wmin/omega)**2

        item=paragontas*arithmitis/paranomastis

        result=item*np.log(item)*omega**2
    return result
```

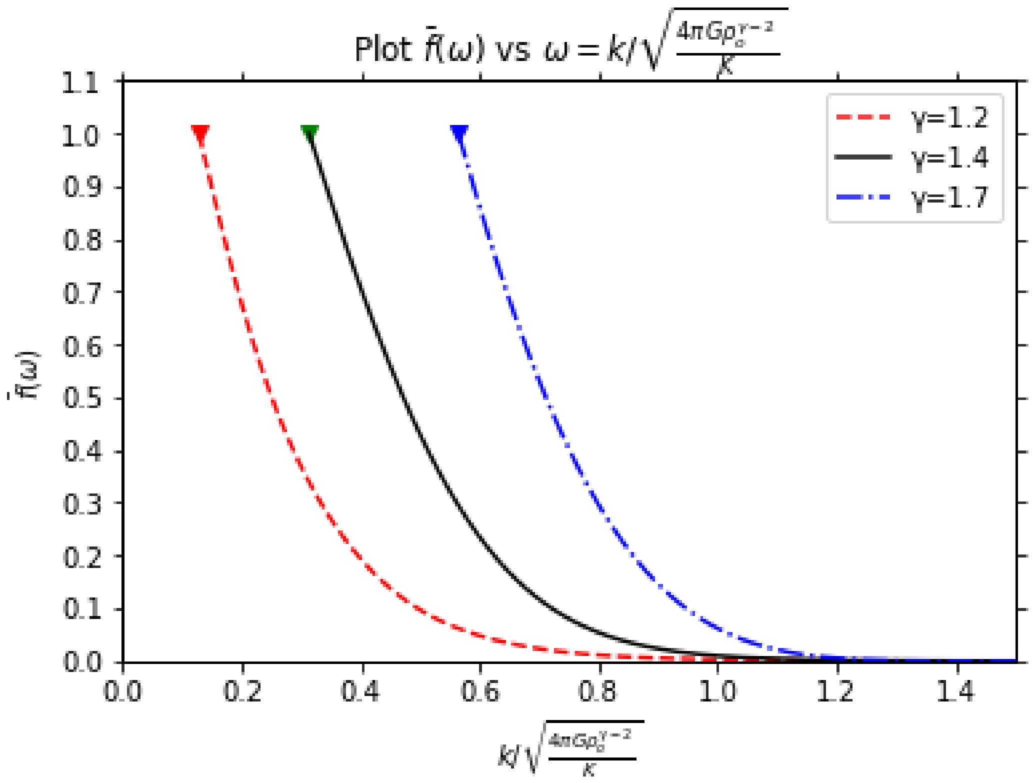
[2]:

```
#Lane Emden + Figure 1
gamma=[1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9]
LinesColor=["red","black","blue"]
LinesStyle=["dashed","solid","dashdot"]
MarkerColor=["red","green","blue"]
counter=0

fig1,ax1=plt.subplots()
fig2,ax2=plt.subplots()
```

```
for g in gamma:
    n=1/(g-1)
    CoefGamma=np.sqrt(g/(g-1))

    y=solve_ivp(fun=model,t_span=[x[0],x[-1]],y0=[1,0],t_eval=x, method='RK45')
    yy=y.y[0]

    index=np.where(~np.isnan(yy))
    xx=x[index]
    yy=yy[index]

    ax1.plot(xx,yy,label="=%.1f"%g)

    xr=xx[-1]
    wmin=np.pi/xr

    I1=simpson(hak_min(xx,yy,n,wmin),xx)
    paranomastis=I1**2

    omegas=np.arange(wmin,100,0.01)

    fakLIST=[]
    ws=[]
    if g==1.2 or g==1.4 or g==1.7:
        for w in omegas:
            paragontas=(wmin/(w))**2
            I1=simpson(hak(xx,w,yy,n),xx)
            arithmitis=I1**2

            fakLIST.append(paragontas*arithmitis/paranomastis)
            ws.append(w/CoefGamma)

        ax2.plot(ws,fakLIST,color=LinesColor[counter],linestyle=LinesStyle[counter],label="=%.1f"%g)
        ax2.scatter(ws[0],fakLIST[0],color=MarkerColor[counter],marker="v")
        ax2.legend(loc="best")
        counter=counter+1
#Lane Emden
ax1.legend(loc="best")
ax1.set_title(r"Solutions for Lane-Emden")
ax1.set_ylabel(r"$\theta(\xi)$")
ax1.set_xlabel(r"$\xi$")
ax1.grid(True)
ax1.plot(np.linspace(0,10,10),10*[0],"k",linewidth=0.8)
ax1.set_xlim([0,10])
```

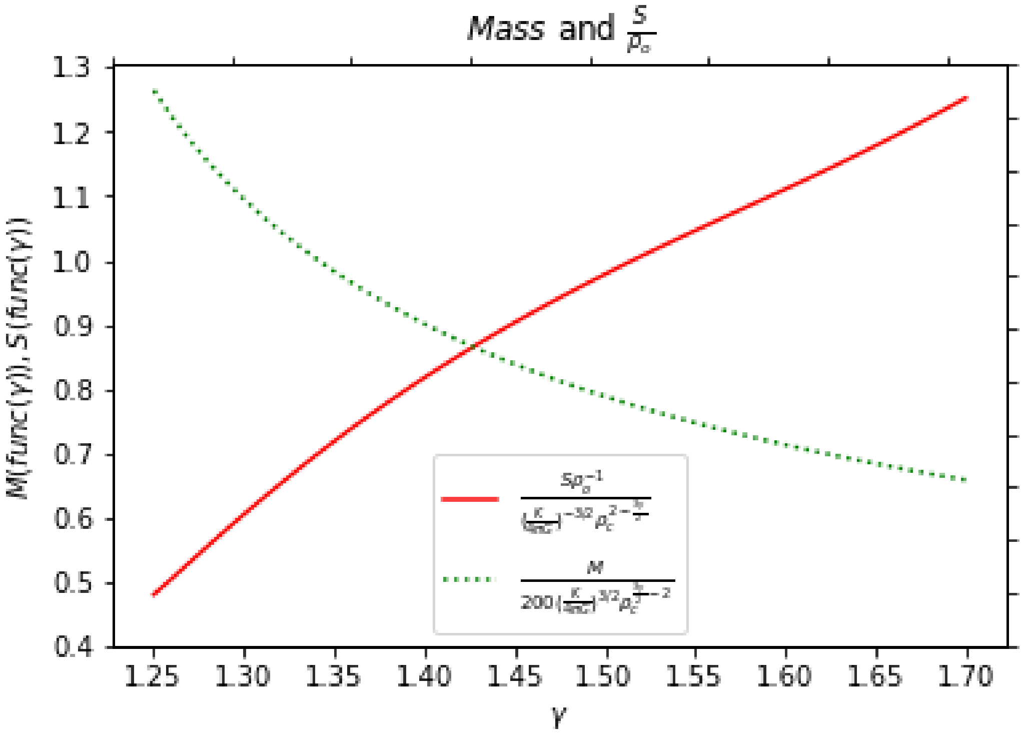
```
ax1.set_ylim([-0.5,1.2])

#Figure 1
ax2.set_xlabel(r"$k/\sqrt{\frac{4\pi G \rho_o^{\gamma-2}}{K}}$")
ax2.set_ylabel(r"$\bar{f}(\omega)$")
ax2.set_title(r"Plot "r"$\bar{f}(\omega)$" " vs " r"$ \omega=k/\sqrt{\frac{4\pi␣
 ↪G \rho_o^{\gamma-2}}{K}}$")
ax2.set_xlim([0,1.5])
ax2.set_ylim([0,1.1])
ax2.set_yticks(np.arange(0,1.2,0.1))
ax2.legend(loc="best")

ax_NEW=ax2.twinx().twiny()
ax_NEW.set_xlim([0,1.5])
ax_NEW.set_ylim([0,1.1])
ax_NEW.set_yticks(np.arange(0,1.2,0.1))
ax_NEW.set_xticklabels([])
ax_NEW.set_yticklabels([])

plt.show()
```





[3]:

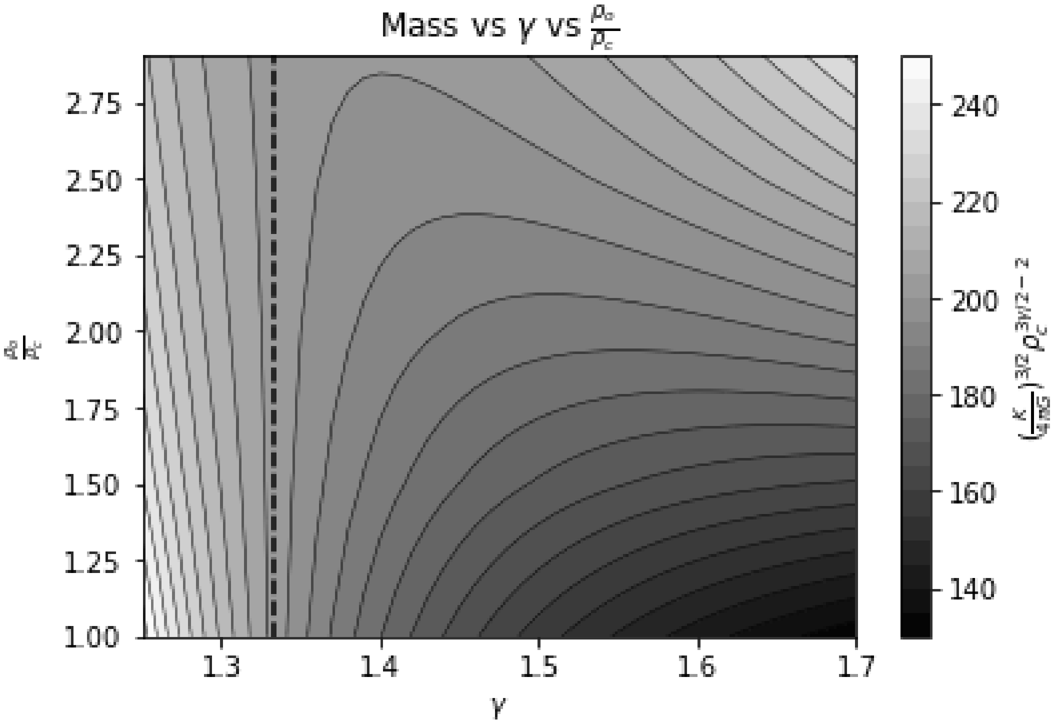
```
#Figure 2
listGamma=[]
Sro=[]
maza=[]

gamma=np.arange(1.25,1.71,0.01)
for g in gamma:
    n=1/(g-1)

    y=solve_ivp(fun=model,t_span=[x[0],x[-1]],y0=[1,0],t_eval=x, method='RK45')
    yy=y.y[0]

    index=np.where(~np.isnan(yy))
    xx=x[index]
    yy=yy[index]

    theta=interp1d(xx,yy,kind="cubic",fill_value="extrapolate")

    xo=0
    xr=fsolve(theta,6)
    xr=xr[0]
    wmin=np.pi/xr


    I=quad(S,wmin,20,args=(yy,n,wmin))
    I=I[0]
    I=-I*np.pi*4*((g-1)/g)**(3/2)

    Sro.append(I)
    listGamma.append(g)


    Imass=quad(Mass,xo,xr,args=(theta,n))
    Imass=Imass[0]
    maza.append(4*np.pi/200*(g/(g-1))**(3/2)*Imass)


fig3,ax3=plt.subplots()
ax3.plot(listGamma,Sro,color="red",\
         label=r"$\frac{S\rho_o^{-1}}{(\frac{K}{4\pi G})^{-3/
 ↪2}\rho_c^{2-\frac{3\gamma}{2}}}$")
ax3.plot(listGamma,maza,color="green",linestyle="dotted",\
         label=r"$\frac{M}{200(\frac{K}{4\pi G})^{3/
 ↪2}\rho_c^{\frac{3\gamma}{2}-2}}$")
ax3.legend()
ax3.set_xlabel(r"$\gamma$")
ax3.set_ylabel(r"$M(func(\gamma)),S(func(\gamma))$")
ax3.set_title(r"$Mass$"" and "r"$\frac{S}{\rho_o}$")
ax3.set_xticks(np.arange(1.25,1.75,0.05))
ax3.set_yticks(np.arange(0.4,1.4,0.1))

ax_NEW=ax3.twinx().twiny()
ax_NEW.set_xlim([0,1.5])
ax_NEW.set_ylim([0,1.1])
ax_NEW.set_yticks(np.arange(0,1.2,0.1))
ax_NEW.set_xticklabels([])
ax_NEW.set_yticklabels([])

plt.show()
```

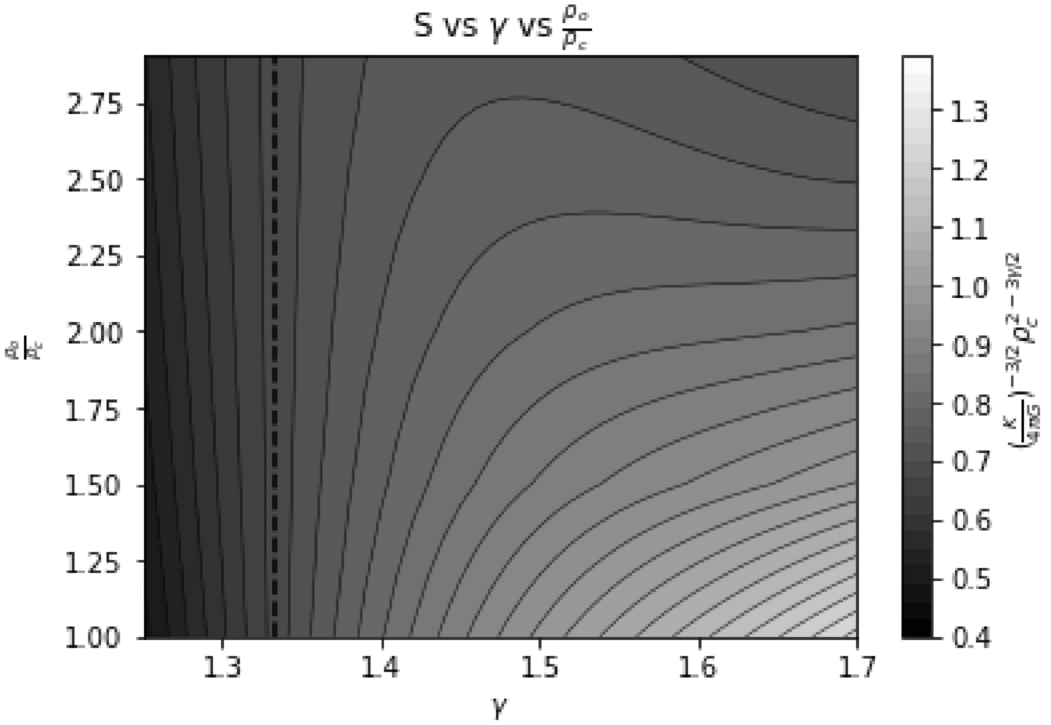


[4]:
```
RoDiaRc=np.arange(1,3.4,0.5)
gamma=np.arange(1.2,1.77,0.01)
Entropy=np.zeros((len(RoDiaRc),len(gamma)),float)
mazaCnt=np.zeros((len(RoDiaRc),len(gamma)),float)

for i in range(len(RoDiaRc)):
    for j in range(len(gamma)):
        g=gamma[j]
        n=1/(g-1)
        w0=[RoDiaRc[i]**(1/n),0]

        y=solve_ivp(fun=model,t_span=[x[0],x[-1]],y0=w0,t_eval=x, method='RK45')
        yy=y.y[0]

        index=np.where(~np.isnan(yy))
        xx=x[index]
        yy=yy[index]

        index=np.where(yy.imag==0)
        yy=yy[index]
        xx=xx[index]
        theta=interp1d(xx,yy,kind="cubic",fill_value="extrapolate")

        xo=0
        xr=fsolve(theta,6)
        xr=xr[0]
        wmin=np.pi/xr

        I=simpson(Mass(xx,yy,n),xx)
        I=4*np.pi*(np.sqrt(g/(g-1)))**3*I

        mazaCnt[i][j]=I


        rCoef=RoDiaRc[i]
        gCoef=g/(g-1)
        listY=[]
        listX=[]
        h=(40-wmin)/(1000-1)
        wtest=np.arange(wmin,40+h,h)
        for wt in wtest:
                listY.append(S(wt,yy,n,wmin))
        listX=wtest

        I=simpson(listY,listX)
        I=-I*np.pi*4./(rCoef*gCoef**(3/2))
        Entropy[i][j]=I

#Figure 3
plt.figure(4)
plt.contour(gamma,RoDiaRc,mazaCnt,levels=np.arange(130,255,5),linewidths=0.5,␣
 ↪colors='k')
plt.contourf(gamma,RoDiaRc,mazaCnt,levels=np.arange(130,255,5),cmap="gist_gray")
plt.colorbar(ticks=np.arange(140,270,20),label=r"$(\frac{K}{4\pi G})^{3/
 ↪2}\rho_c^{3\gamma/2-2}$")
plt.ylim([1,2.9])
plt.xlim([1.25,1.7])
ymin,ymax=plt.gca().get_ylim()
plt.plot([4/3,4/3],[ymin,ymax],"k--")
plt.xlabel(r"$\gamma$")
plt.ylabel(r"$\frac{\rho_o}{\rho_c}$")
plt.title("Mass vs "r"$\gamma$"" vs "r"$\frac{\rho_o}{\rho_c}$")

#Figure 4
plt.figure(5)
plt.contour(gamma,RoDiaRc,Entropy,levels=np.arange(0.4,1.4,0.03), linewidths=0.
 ↪5, colors='k')
```

```
plt.contourf(gamma,RoDiaRc,Entropy,levels=np.arange(0.4,1.4,0.
 ↪03),cmap="gist_gray")
plt.colorbar(ticks=np.arange(0.4,1.4,0.1),label=r"$(\frac{K}{4\pi G})^{-3/
 ↪2}\rho_c^{2-3\gamma/2}$")
plt.ylim([1,2.9])
plt.xlim([1.25,1.7])
ymin,ymax=plt.gca().get_ylim()
plt.plot([4/3,4/3],[ymin,ymax],"k--")
plt.xlabel(r"$\gamma$")
plt.ylabel(r"$\frac{\rho_o}{\rho_c}$")
plt.title("S vs "r"$\gamma$"" vs "r"$\frac{\rho_o}{\rho_c}$")

plt.show()
```

```
[4]: Text(0.5, 1.0, 'S vs $\\gamma$ vs $\\frac{\\rho_o}{\\rho_c}$')
```

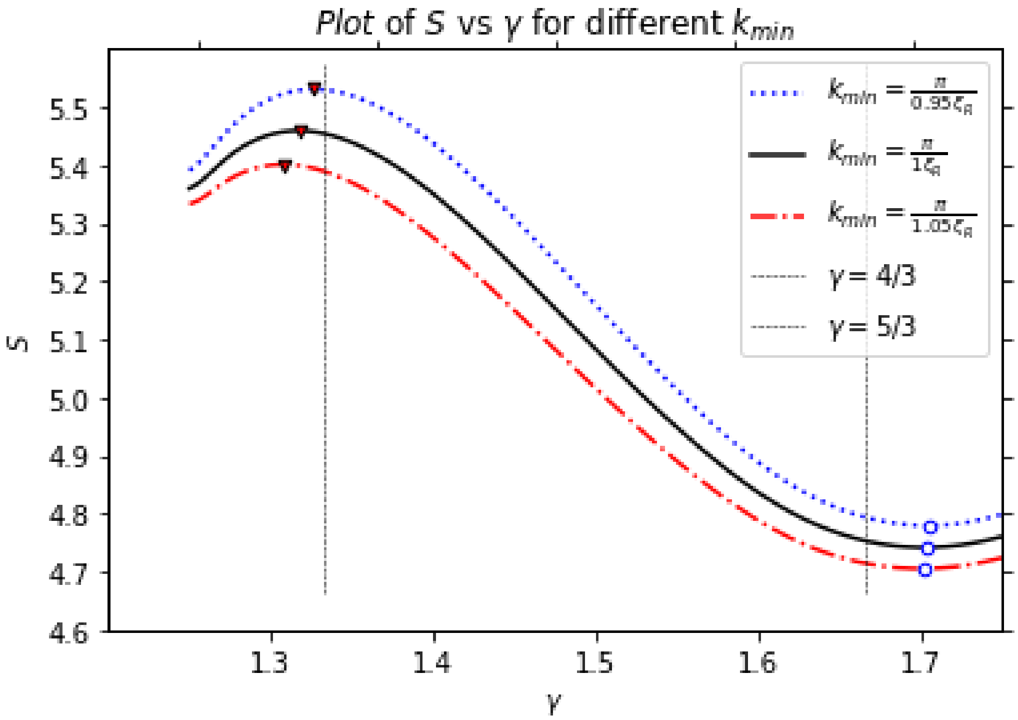




```
[6]: #Figure 5
     GammaFig5=np.arange(1.25,1.8,0.001)
     DataX=[]
     DataY=[[],[],[]]
     for g in GammaFig5:
         n=1/(g-1)

         y=solve_ivp(fun=model,t_span=[x[0],x[-1]],y0=w_initial,t_eval=x,␣
      ↪method='RK45')
         yy=y.y[0]

         index=np.where(~np.isnan(yy))
         xx=x[index]
         yy=yy[index]

         theta=interp1d(xx,yy,kind="cubic",fill_value="extrapolate")

         xo=0
         xr=fsolve(theta,6)
         xr=xr[0]

         wFIG5=[np.pi/xr/0.95,np.pi/xr,np.pi/xr/1.05]
```

```
    for i in range(len(wFIG5)):
        listY=[]

        wmin=wFIG5[i]
        h=(40-wmin)/(1000-1)
        wtest=np.arange(wmin,40+h,h)
        for wt in wtest:
                listY.append(S(wt,yy,n,wmin))

        I=simpson(listY,wtest)
        I=-I*np.pi*4
        DataY[i].append(I)
    DataX.append(g)

#Figure 5
fig4,ax4=plt.subplots()
ax4.plot(DataX,DataY[0],label=r"$k_{min}=\frac{\pi }{0.
 ↪95\xi_R}$",linestyle="dotted",color="blue")
a1=DataY[0].index(max(DataY[0]))
a2=DataY[0].index(min(DataY[0]))
ax4.
 ↪scatter(DataX[a1],DataY[0][a1],color="red",edgecolor="black",marker="v",s=20*2**0,zorder=2)
ax4.
 ↪scatter(DataX[a2],DataY[0][a2],color="white",edgecolor="blue",s=20*2**0,zorder=2)

ax4.plot(DataX,DataY[1],label=r"$k_{min}=\frac{\pi␣
 ↪}{1\xi_R}$",linestyle="solid",color="black")
a1=DataY[1].index(max(DataY[1]))
a2=DataY[1].index(min(DataY[1]))
ax4.
 ↪scatter(DataX[a1],DataY[1][a1],color="red",edgecolor="black",marker="v",s=20*2**0,zorder=2)
ax4.
 ↪scatter(DataX[a2],DataY[1][a2],color="white",edgecolor="blue",s=20*2**0,zorder=2)

ax4.plot(DataX,DataY[2],label=r"$k_{min}=\frac{\pi }{1.
 ↪05\xi_R}$",linestyle="dashdot",color="red")
a1=DataY[2].index(max(DataY[2]))
a2=DataY[2].index(min(DataY[2]))
ax4.
 ↪scatter(DataX[a1],DataY[2][a1],color="red",edgecolor="black",marker="v",s=20*2**0,zorder=2)
ax4.
 ↪scatter(DataX[a2],DataY[2][a2],color="white",edgecolor="blue",s=20*2**0,zorder=2)

axes = plt.gca()
y_min, y_max = axes.get_ylim()
ax4.plot([4/3,4/3],[y_min,y_max],"k--",label="$\gamma=4/3$",linewidth=0.5)
```

```
ax4.plot([5/3,5/3],[y_min,y_max],"k--",label="$\gamma=5/3$",linewidth=0.5)
ax4.legend()
ax4.set_xlabel(r"$\gamma$")
ax4.set_ylabel(r"$S$")
ax4.set_title(r"$Plot$"" of "r"$S$"" vs "r"$\gamma$"" for different␣
 ↪"r"$k_{min}$")
ax4.set_xlim([1.2,1.75])
ax4.set_ylim([4.6,5.6])
ax4.set_yticks(np.arange(4.6,5.6,0.1))
ax4.set_xticks(np.arange(1.3,1.8,0.1))

ax_NEW=ax4.twinx().twiny()
ax_NEW.set_xlim([1.25,1.75])
ax_NEW.set_ylim([4.6,5.6])
ax_NEW.set_yticks(np.arange(4.6,5.6,0.1))
ax_NEW.set_xticklabels([])
ax_NEW.set_yticklabels([])

plt.show()
```



[ ]: