Quantum Information Project

Dimitrios Psarras[1]

[1] Presenting author, dpsarras@auth.gr

June 28, 2022

ARISTOTLE UNIVERSITY OF THESSALONIKI

Project

Dimitrios Psarras

Table of Contents

Functions

Functions

```python
################################################################################
# FUNCTIONS

# Lane emden function
def lane_emden(t, x, g):
    theta, z = x
    return [z,-(2/t)*z-(theta**(1/(g-1)))]

# create loading bar during the execution of the program
def loadbar(iterations,total,prefix='',suffix='',decimals=1,length=100,fill='\u25AE'):
    percentage = ('{0:.'+str(decimals)+'f}').format(100*(iterations/total))
    filled = int(length*iterations//total)
    bar = fill * filled + '_' * (length-filled)
    print(f'\r{prefix} |{bar}| {percentage}% {suffix}',end='\r')
    if iterations == total:
        print()
```

Main Code

Constants and Initialization

```python
##############################################################################
# Initialize arrays and constant variables and arrays

# Constants
gamma = np.linspace(1.2, 1.9, 169)          # gamma values
k_min_coef = np.array([0.95, 1.00, 1.05])   # coefficients of k_min for fig 3.
coef_calc = 1                               # coefficient for which figures 1 and 2 are drawn
t_init = 10e-4                              # inital value of xi for solving the differential equation
t_end = 30                                  # final value of xi for solving the differential equation
M_coef = 200                                # Mass coefficient to scale fig 2.
d = 10000                                   # number of values returned from solve_ivp and simpson
pg = np.linspace(1.2, 1.9, 8)               # values that will be plotted in figure of lane emden solutions

# Initialization of empty arrays
S = np.zeros(len(gamma))                    # array where the configurational entropy values will be stored
S3 = np.zeros((3,len(gamma)))               # array where the configurational entropy times a^(3) will be stored
M = np.zeros(len(gamma))                    # array where the mass values will be stored

# Initial values
w0 = [1.,0.]                                # boundary conditions with values of theta and the derivative of theta
```

Initialization of Lane Emden figure and figure1


```python
##############################################################################################
# Initialize the figures

# Figure of the solutions of lane emden
plt.figure(0)
plt.title("Lane Emden")
plt.ylabel(r"$\theta(\xi)$")
plt.xlabel(r"$\xi$")
plt.ylim([-1.3, 1.1])
plt.xlim([0, 10])
plt.grid()

# Figure 1 Normalized modal fraction f(|k|) for sample values of polytropic index gamma (figure 1
    of paper)
plt.figure(1)
plt.tick_params(
    bottom=False, top=True,
    left=True, right=True)
plt.title("Fig 1.")
plt.ylabel(r"$\bar f(|k|)$")
plt.xlabel(r"$k/\sqrt{4\pi G/K\rho_0^{\gamma-2}}$")
plt.xlim([0, 1.5])
plt.ylim([0, 1.1])
plt.grid()
```

Initialization of figures 2 and 3


```python
# Figure 2 Configurational entropy times \rho^(-1) (continuous line) and mass (dotted line) versus polytropic
# index gamma (figure 2 of paper)
plt.figure(2)
plt.title("Fig 2.")
plt.xlabel(r"$\gamma$")
plt.xlim([1.25, 1.7])
plt.ylim([0.4, 1.3])
plt.grid()

# Figure 3 Configurational entropy versus polytropic index gamma for polytropes. We display results for several
# choices of cutoff for k_min (figure 5 of paper)
plt.figure(3)
plt.title("Fig 3.")
plt.xlabel(r"$\gamma$")
plt.ylabel(r"$Sa^{3}$")
plt.xlim([1.25, 1.75])
plt.ylim([4.6, 5.6])
plt.axvline(x=4/3,color='k',ls='--')
plt.text(4/3, 4.63, "4/3", rotation=0)
plt.axvline(x=5/3,color='k',ls='--')
plt.text(5/3, 4.63, "5/3", rotation=0)
plt.grid()
```

Solving Lane Emden


```
################################################################################
# Main for loop to compute the solutions of lane emden and the integrals

# iniate loading bar
loadbar(0,len(gamma),prefix='Progress:',suffix='Complete',decimals=2,length=40)

# FOR LOOP
# Loops over all values of polytropic index gamma
for ind,g in enumerate(gamma):

    # SOLVING THE DIFFERANTIAL EQUATION
    # xi arrays hols the values of xi for the evaluation of theta
    # solve_ivp is used to solve the differential equation with method Runge-Kutta of 4th and 5th order
    # values of the solutions for theta and xi are stored indside the arrays the and ti_n
    xi = np.linspace(t_init , t_end , d)
    sol = inte.solve_ivp(fun=lane_emden,t_span=(t_init,t_end),y0=w0, method='RK45',args=(g,),t_eval=xi)
    the = sol.y
    ti_n = sol.t
```

Plot Lane Emden solutions


```python
    # theta(xi)==0 correspodens to rho(R)==0 hence negative values of theta have no physical meaning
    # in the next three line we find the index of positive theta and we store the desired values of theta and xi
    # inside the arrays x_n for theta and ti for xi
    indexx, = np.where(the[0] >= 0)
    x_n = the[0][indexx]
    ti = ti_n[indexx]

    # with this if statement we chooce for which polytropic index gamma the solution of lane-emden will be plotted
    if g in pg:
        plt.figure(0)
        plt.plot(ti_n,the[0],label=f'\u03B3 = {g:.2f}')
```

Calculation of Normalized modal fraction f


```python
# in order to plot figure 3 the calculations have to be executed for three different values of kappa_min=ak
# in array min_k the three values of kappa_min are stored
min_k = (np.pi/ti[-1])/k_min_coef

# Calculation for all values of k_min to compute the product Sa^(3) for each k_min

# values of kappa for which the integral is computed
max_k = 100*min_k
k = np.linspace(min_k,max_k,len(ti))

# using simpson function the h_min is computed for the value k_min
y_min = (x_n**(1/(g-1)))*np.sin(np.multiply.outer(min_k, ti))*ti
h_min = (inte.simpson(y_min, ti, dx=0.001,axis=1)*(1/min_k))**2

# For all values of kappa the h(ak) is computed
y_data = (x_n**(1/(g-1)))*np.sin(np.multiply.outer(k,ti))*ti
h = (inte.simpson(y_data, ti, dx=0.001,axis=2)*(1/k))**2

# now that both h(kappa) and h(kappa_min) have been computed the Normalized modal fraction f(|k|)
f = (h/h_min)
```

Calculation of Configurational entropy and Mass


```
    # using simpson the Mass divided by a constant factor is computed
    m = (x_n**(1/(g-1)))*ti**2
    M[ind] = (4*np.pi*((g/(g-1))**((3/2)))*inte.simpson(m,ti,dx=0.001))/M_coef

    # finally Sa^(3)/((g/(g-1))**(-(3/2))) is computed only for pi/(1.00R) and Sa^(3) for each
      k_min
    s = f*np.log(f)*(k)**2
    S3[:,ind] = -4*np.pi*inte.simpson(s,k,dx=0.0001,axis=0)
    S[ind] = ((g/(g-1))**(-(3/2)))*S3[1,ind]
```

Plot Figure 1

```
    # here only for value pi/(1.00R) the figure 1 is plotted only for smaller than 1 f(|k|)
    # also in figure 1 using plt.scatter the points for k_min are added
    plt.figure(1)
    index, = np.where(f[:,coef_calc] <= 1)
    if g==1.2:
        plt.scatter(k[index[0],coef_calc]/np.sqrt(g/(g-1)),f[index[0],coef_calc],marker='v',color='r')
        plt.plot(k[index[0]:,coef_calc]/np.sqrt(g/(g-1)),f[index[0]:,coef_calc],'--',color='r',label=f'\u03B3 = {g}')
    elif g==1.4:
        plt.scatter(k[index[0],coef_calc]/np.sqrt(g/(g-1)),f[index[0],coef_calc],marker='v',color='g')
        plt.plot(k[index[0]:,coef_calc]/np.sqrt(g/(g-1)),f[index[0]:,coef_calc],color='k',label=f'\u03B3 = {g}')
    elif g==1.7:
        plt.scatter(k[index[0],coef_calc]/np.sqrt(g/(g-1)),f[index[0],coef_calc],marker='v',color='c')
        plt.plot(k[index[0]:,coef_calc]/np.sqrt(g/(g-1)),f[index[0]:,coef_calc],'-.',color='c',label=f'\u03B3 = {g}')

    # update loading bar
    loadbar(ind+1,len(gamma),prefix='Progress:',suffix='Complete',decimals=2,length=40)
```

Completion of figures 1 and 2

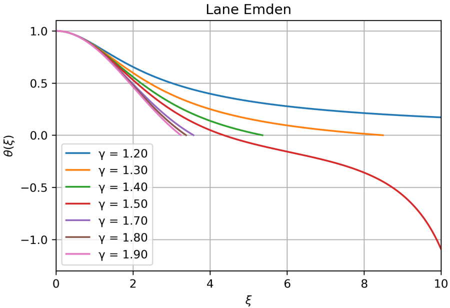

```python
################################################################################
# Finalize and print the figures with legends

# added legend in plot
plt.figure(0)
plt.legend()

# added legend in plot
plt.figure(1)
plt.legend()

# plot Mass and configurational entropy S and add legends
plt.figure(2)
plt.plot(gamma,S,color='r',label=r'$S\rho_0^{-1}/\left( \left( \frac{K}{4\pi G}\right)^{-\frac{3}{2}} \rho_c^{2-\frac{3}{2}\gamma} \right)$')
plt.plot(gamma,M,'--',color='g',label=r'$M/\left( 200\left( \frac{K}{4\pi G}\right)^{\frac{3}{2}} \rho_c^{\frac{3}{2}\gamma -2} \right)$')
plt.legend()
```

Completion of figure 3

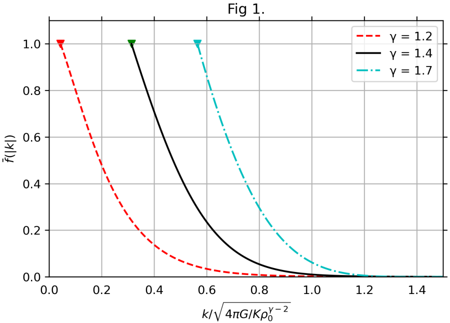

```python
# plot configurational entropy S times a^(3), points for max and min and legends
plt.figure(3)
plt.scatter(gamma[np.argmax(S3[0])],np.amax(S3[0]),marker='v',color='r')
plt.scatter(gamma[np.argmax(S3[1])],np.amax(S3[1]),marker='v',color='r')
plt.scatter(gamma[np.argmax(S3[2])],np.amax(S3[2]),marker='v',color='r')
plt.scatter(gamma[np.argmin(S3[0])],np.amin(S3[0]),marker='o',color='b')
plt.scatter(gamma[np.argmin(S3[1])],np.amin(S3[1]),marker='o',color='b')
plt.scatter(gamma[np.argmin(S3[2])],np.amin(S3[2]),marker='o',color='b')
plt.plot(gamma,S3[0],'--',color='b',label=r'$\pi/(0.95R)$')
plt.plot(gamma,S3[1],color='k',label=r'$\pi/(1.00R)$')
plt.plot(gamma,S3[2],'-.',color='r',label=r'$\pi/(1.05R)$')
plt.legend()

################################################################################
# Stop timing and printed

print(f'Execution Time: {datetime.now() - start}')
```

Figures

Solutions of Lane Emden and Figure 1



(a) Lane Emden Solutions



(b) Normalized modal fraction for sample values of the polytropic index γ. From left to right, $\gamma = 1.2$, $\gamma = 1.4$, and $\gamma = 1.7$.

Project

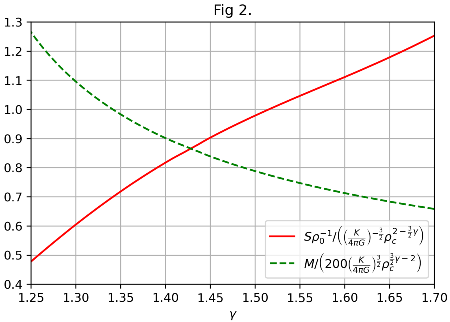
Dimitrios Psarras

Functions

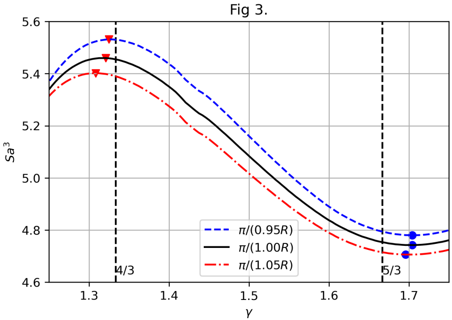
Main Code

Figures

Figures 2 and 3



(c) Configurational entropy times ρ_0^{-1} (continuous line) and mass (dotted line) versus polytropic index γ for $\rho_0 = \rho_c$.



(d) Configurational entropy versus polytropic index γ for polytropes. We display results for several choices of cutoff for k_{min}.