Quantum Information Project

Dimitrios Psarras[1]

[1] Presenting author, dpsarras@auth.gr

June 13, 2022

ARISTOTLE
UNIVERSITY
OF THESSALONIKI

Table of Contents

Functions

Functions


```
11 ##############################################################################
12 # FUNCTIONS
13
14 # Lane emden function
15 def lane_emden(t, x, g):
16     theta, z = x
17     return [z,-(2/t)*z-(theta**(1/(g-1)))]
18
19 # create loading bar during the execution of the program
20 def loadbar(iterations,total,prefix='',suffix='',decimals=1,length=100,fill='\u25AE'):
21     percentage = ('{0:.'+str(decimals)+'f}').format(100*(iterations/total))
22     filled = int(length*iterations//total)
23     bar = fill * filled + '-' * (length-filled)
24     print(f'\r{prefix} |{bar}| {percentage}% {suffix}',end='\r')
25     if iterations == total:
26         print()
```

Main Code

Constants and Initialization

```
40 ##############################################################################
41 # Initialize arrays and constant variables and arrays
42
43 # Constants
44 gamma = np.linspace(1.2, 1.9, 169)        # gamma values
45 k_min_coef = np.array([0.95, 1.00, 1.05]) # coefficients of k_min for fig 3.
46 coef_calc = 1                             # coefficient for which figures 1 and 2 are drawn
47 t_init = 10e-4                            # inital value of xi for solving the differentail
       equation
48 t_end = 30                                # inital value of xi for solving the differentail
       equation
49 M_coef = 200                              # Mass coefficient to scale fig 2.
50 d = 10000                                 # number of values returned from solve_ivp and simpson
51 pg = np.linspace(1.2, 1.9, 8)            # values that will be plotted in figure of lane emden
       solutions
52
53 # Initialization of empty arrays
54 S = np.zeros(len(gamma))                  # array where the configurational entropy values will be
       stored
55 S3 = np.zeros((3,len(gamma)))             # array where the configurational entropy times a^(3)
       will be stored
56 M = np.zeros(len(gamma))                  # array where the mass values will be stored
57
58 # Initial values
59 w0 = [1.,0.]                              # boundary conditions with values of theta and the
       derivative of theta
```

Initialization of Lane Emden figure and figure1

```
61 ##########################################################################
62 # Initialize the figures
63
64 # Figure of the solutions of lane emden
65 plt.figure(0)
66 plt.title("Lane Emden")
67 plt.ylabel(r"$\theta(\xi)$")
68 plt.xlabel(r"$\xi$")
69 plt.ylim([-1.3, 1.1])
70 plt.xlim([0, 10])
71 plt.grid()
72
73 # Figure 1 Normalized modal fraction f(|k|) for sample values of polytropic index gamma (figure 1
       of paper)
74 plt.figure(1)
75 plt.tick_params(
76     bottom=False, top=True,
77     left=True, right=True)
78 plt.title("Fig 1.")
79 plt.ylabel(r"$\bar f(|k|)$")
80 plt.xlabel(r"$k/\sqrt{4\pi G/K\rho_0^{\gamma-2}}$")
81 plt.xlim([0, 1.5])
82 plt.ylim([0, 1.1])
83 plt.grid()
```

Initialization of figures 2 and 3

```
85 # Figure 2 Configurational entropy times \rho^(-1) (continuous line) and mass (dotted line) versus
       polytropic
86 # index gamma (figure 2 of paper)
87 plt.figure(2)
88 plt.title("Fig 2.")
89 plt.xlabel(r"$\gamma$")
90 plt.xlim([1.25, 1.7])
91 plt.ylim([0.4, 1.3])
92 plt.grid()
93
94 # Figure 3 Configurational entropy versus polytropic index gamma for polytropes. We display results
       for several
95 # choices of cutoff for k_min (figure 5 of paper)
96 plt.figure(3)
97 plt.title("Fig 3.")
98 plt.xlabel(r"$\gamma$")
99 plt.ylabel(r"$Sa^{3}$")
100 plt.xlim([1.25, 1.75])
101 plt.ylim([4.6, 5.6])
102 plt.axvline(x=4/3,color='k',ls='--')
103 plt.text(4/3, 4.63, "4/3", rotation=0)
104 plt.axvline(x=5/3,color='k',ls='--')
105 plt.text(5/3, 4.63, "5/3", rotation=0)
106 plt.grid()
```

Solving Lane Emden


```
108 ##############################################################################
109 # Main for loop to compute the solutions of lane emden and the integrals
110
111 # iniate loading bar
112 loadbar(0,len(gamma),prefix='Progress:',suffix='Complete',decimals=2,length=40)
113
114 # FOR LOOP
115 # Loops over all values of polytropic index gamma
116 for ind,g in enumerate(gamma):
117
118     # SOLVING THE DIFFERANTIAL EQUATION
119     # xi arrays hols the values of xi for the evaluation of theta
120     # solve_ivp is used to solve the differential equation with method Runge-Kutta of 4th and 5th
          order
121     # values of the solutions for theta and xi are stored indside the arrays the and ti_n
122     xi = np.linspace(t_init, t_end, d)
123     sol = inte.solve_ivp(fun=lane_emden,t_span=(t_init,t_end),y0=w0, method='RK45',args=(g,),t_eval
          =xi)
124     the = sol.y
125     ti_n = sol.t
```

Plot Lane Emden solutions


```
127     # theta(xi)==0 correspodens to rho(R)==0 hence negative values of theta have no physical
          meaning
128     # in the next three line we find the index of positive theta and we store the desired values of
           theta and xi
129     # inside the arrays x_n for theta and ti for xi
130     indexx, = np.where(the[0] >= 0)
131     x_n = the[0][indexx]
132     ti = ti_n[indexx]
133
134     # with this if statement we chooce for which polytropic index gamma the solution of lane-emden
          will be plotted
135     if g in pg:
136         plt.figure(0)
137         plt.plot(ti_n,the[0],label=f'\u03B3 = {g:.2f}')
```

Calculation of Normalized modal fraction f

```
139     # in order to plot figure 3 the calculations have to be executed for three different values of
          kappa_min=ak
140     # in array min_k the three values of kappa_min are stored
141     min_k = (np.pi/ti[-1])/k_min_coef
142
143     # Calculation for all values of k_min to compute the product Sa^(3) for each k_min
144
145     # values of kappa for which the integral is computed
146     max_k = 100*min_k
147     k = np.linspace(min_k,max_k,len(ti))
148
149     # using simpson function the h_min is computed for the value k_min
150     y_min = (x_n**(1/(g-1)))*np.sin(np.multiply.outer(min_k,ti))*ti
151     h_min = (inte.simpson(y_min,ti,dx=0.001,axis=1)*(1/min_k))**2
152
153     # h array stores the values of h(kappa)
154     h = np.zeros(d)
155
156     # For all values of kappa the h(ak) is computed
157     y_data = (x_n**(1/(g-1)))*np.sin(np.multiply.outer(k,ti))*ti
158     h = (inte.simpson(y_data,ti,dx=0.001,axis=2)*(1/k))**2
159
160     # now that both h(kappa) and h(kappa_min) have been computed the Normalized modal fraction f(|k
          |)
161     f = (h/h_min)
```

Calculation of Configurational entropy and Mass


```
163     # using simpson the Mass divided by a constant factor is computed
164     m = (x_n**(1/(g-1)))*ti**2
165     M[ind] = (4*np.pi*((g/(g-1))**((3/2)))*inte.simpson(m,ti,dx=0.001))/M_coef
166
167     # finally Sa^(3)/(((g/(g-1))**(-(3/2))) is computed only for pi/(1.00R) and Sa^(3) for each
          k_min
168     s = f*np.log(f)*(k)**2
169     S3[:,ind] = -4*np.pi*inte.simpson(s,k,dx=0.0001,axis=0)
170     S[ind] = ((g/(g-1))**(-(3/2)))*S3[1,ind]
```

Plot Figure 1

```
172     # here only for value pi/(1.00R) the figure 1 is plotted only for smaller than 1 f(|k|)
173     # also in figure 1 using plt.scatter the points for k_min are added
174     plt.figure(1)
175     index, = np.where(f[:,coef_calc] <= 1)
176     if g==1.2:
177         plt.scatter(k[index[0],coef_calc]/np.sqrt(g/(g-1)),f[index[0],coef_calc],marker='v',color='
        r')
178         plt.plot(k[index[0]:,coef_calc]/np.sqrt(g/(g-1)),f[index[0]:,coef_calc],'--',color='r',
        label=f'\u03B3 = {g}')
179     elif g==1.4:
180         plt.scatter(k[index[0],coef_calc]/np.sqrt(g/(g-1)),f[index[0],coef_calc],marker='v',color='
        g')
181         plt.plot(k[index[0]:,coef_calc]/np.sqrt(g/(g-1)),f[index[0]:,coef_calc],color='k',label=f'\
        u03B3 = {g}')
182     elif g==1.7:
183         plt.scatter(k[index[0],coef_calc]/np.sqrt(g/(g-1)),f[index[0],coef_calc],marker='v',color='
        c')
184         plt.plot(k[index[0]:,coef_calc]/np.sqrt(g/(g-1)),f[index[0]:,coef_calc],'-.',color='c',
        label=f'\u03B3 = {g}')
185
186     # update loading bar
187     loadbar(ind+1,len(gamma),prefix='Progress:',suffix='Complete',decimals=2,length=40)
```

Completion of figures 1 and 2

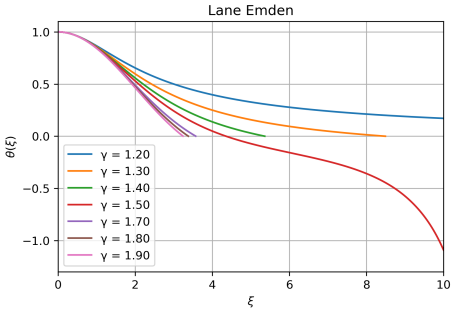

```
189 ##############################################################################
190 # Finalize and print the figures with legends
191
192 # added legend in plot
193 plt.figure(0)
194 plt.legend()
195
196 # added legend in plot
197 plt.figure(1)
198 plt.legend()
199
200 # plot Mass and configurational entropy S and add legends
201 plt.figure(2)
202 plt.plot(gamma,S,color='r',label=r'$S\rho_0^{-1}/\left( \left( \frac{K}{4\pi G}\right)^{-\frac
      {3}{2}} \rho_c^{2-\frac{3}{2}\gamma} \right)$')
203 plt.plot(gamma,M,'--',color='g',label=r'$M/\left( 200\left( \frac{K}{4\pi G}\right)^{\frac{3}{2}} \
      rho_c^{\frac{3}{2}\gamma -2} \right)$')
204 plt.legend()
```

Completion of figure 3

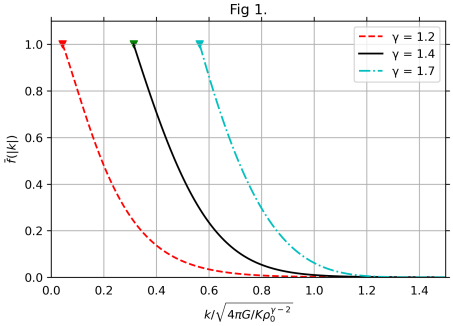

```
206 # plot configurational entropy S times a^(3), points for max and min and legends
207 plt.figure(3)
208 plt.scatter(gamma[np.argmax(S3[0])],np.amax(S3[0]),marker='v',color='r')
209 plt.scatter(gamma[np.argmax(S3[1])],np.amax(S3[1]),marker='v',color='r')
210 plt.scatter(gamma[np.argmax(S3[2])],np.amax(S3[2]),marker='v',color='r')
211 plt.scatter(gamma[np.argmin(S3[0])],np.amin(S3[0]),marker='o',color='b')
212 plt.scatter(gamma[np.argmin(S3[1])],np.amin(S3[1]),marker='o',color='b')
213 plt.scatter(gamma[np.argmin(S3[2])],np.amin(S3[2]),marker='o',color='b')
214 plt.plot(gamma,S3[0],'—',color='b',label=r'$\pi/(0.95R)$')
215 plt.plot(gamma,S3[1],color='k',label=r'$\pi/(1.00R)$')
216 plt.plot(gamma,S3[2],'—.',color='r',label=r'$\pi/(1.05R)$')
217 plt.legend()
218
219 ##########################################################################
220 # Stop timing and printed
221
222 print(f'Execution Time: {datetime.now() — start}')
```

Figures

Solutions of Lane Emden and Figure 1

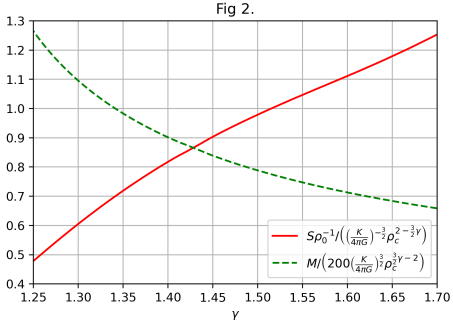


(a) Lane Emden Solutions

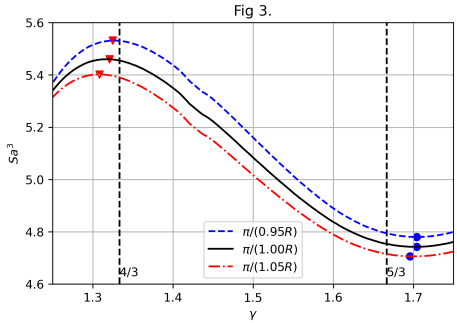


(b) Normalized modal fraction for sample values of the polytropic index γ. From left to right, $\gamma = 1.2$, $\gamma = 1.4$, and $\gamma = 1.7$.

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}.