

```

1 # Import modules
2
3 import matplotlib.pyplot as plt
4 import numpy as np
5 from q2_b import g_star, g_star_s
6
7 # Define a, t ranges
8 a_range = np.arange(np.log(1e11), np.log(1e19), 0.0075)
9 t_range = np.arange(np.log(1e12), np.log(1e26), 0.0075)
10
11 # Define Constants
12 T_rh = 1e14
13 M_pl = 2.4e18
14 g_high = g_star(T_rh)
15 t_rh = ((3 * M_pl * np.sqrt(10)) / (2 * np.pi)) * 1 / (T_rh ** 2 * np.sqrt(g_high))
16
17 # Define T thresholds
18 T_thres = [100, 30, 15, 1, 0.2, 0.05, 0.00025] # GeV
19
20 # Define a thresholds
21 a_thres = [T_rh / T_crit for T_crit in T_thres]
22
23 # Data structure to help later
24 g_before_after = []
25 for T in T_thres:
26     g_before_after.append([g_star_s(T * 1000 + 10), g_star_s(T * 1000 - 10)])
27 g_star_before_after = []
28 for T in T_thres:
29     g_star_before_after.append([g_star(T * 1000 + 10), g_star(T * 1000 - 10)])
30
31
32 def T_approx(a):
33     T_a_prop_const = T_rh * g_high ** (1 / 3)
34     if np.exp(a) < a_thres[0]:
35         return T_a_prop_const * (g_before_after[0][0] ** (-1 / 3)) / np.exp(a)
36     elif a_thres[0] ≤ np.exp(a) < a_thres[1]:
37         return T_a_prop_const * (g_before_after[0][1] ** (-1 / 3)) / np.exp(a)
38     elif a_thres[1] ≤ np.exp(a) < a_thres[2]:
39         return T_a_prop_const * (g_before_after[1][1] ** (-1 / 3)) / np.exp(a)
40     elif a_thres[2] ≤ np.exp(a) < a_thres[3]:
41         return T_a_prop_const * (g_before_after[2][1] ** (-1 / 3)) / np.exp(a)
42     elif a_thres[3] ≤ np.exp(a) < a_thres[4]:
43         return T_a_prop_const * (g_before_after[3][1] ** (-1 / 3)) / np.exp(a)
44     elif a_thres[4] ≤ np.exp(a) < a_thres[5]:
45         return T_a_prop_const * (g_before_after[4][1] ** (-1 / 3)) / np.exp(a)
46     elif a_thres[5] ≤ np.exp(a) < a_thres[6]:
47         return T_a_prop_const * (g_before_after[5][1] ** (-1 / 3)) / np.exp(a)
48     elif a_thres[6] ≤ np.exp(a):
49         return T_a_prop_const * (g_before_after[6][1] ** (-1 / 3)) / np.exp(a)
50
51
52 def T_approx_exp(a):
53     T_a_prop_const = T_rh * g_high ** (1 / 3)
54     if a < a_thres[0]:
55         return T_a_prop_const * (g_before_after[0][0] ** (-1 / 3)) / a
56     elif a_thres[0] ≤ a < a_thres[1]:

```

```

57     return T_a_prop_const * (g_before_after[0][1] ** (-1 / 3)) / a
58 elif a_thres[1] ≤ a < a_thres[2]:
59     return T_a_prop_const * (g_before_after[1][1] ** (-1 / 3)) / a
60 elif a_thres[2] ≤ a < a_thres[3]:
61     return T_a_prop_const * (g_before_after[2][1] ** (-1 / 3)) / a
62 elif a_thres[3] ≤ a < a_thres[4]:
63     return T_a_prop_const * (g_before_after[3][1] ** (-1 / 3)) / a
64 elif a_thres[4] ≤ a < a_thres[5]:
65     return T_a_prop_const * (g_before_after[4][1] ** (-1 / 3)) / a
66 elif a_thres[5] ≤ a < a_thres[6]:
67     return T_a_prop_const * (g_before_after[5][1] ** (-1 / 3)) / a
68 elif a_thres[6] ≤ a:
69     return T_a_prop_const * (g_before_after[6][1] ** (-1 / 3)) / a
70
71
72 # Plot T(a)
73 T_approx_array = [T_approx(a_val) for a_val in a_range]
74 plt.loglog(np.exp(a_range), T_approx_array, 'b-', label=r'Piecewise Approx.',
75           linewidth=1)
76 for a in a_thres:
77     plt.vlines(a, min(T_approx_array), max(T_approx_array), linestyle='dashed',
78             colors='k', linewidth=0.5)
79 plt.title(r'$T(a)$ Transition')
80 plt.ylabel(r'$T$ (GeV)')
81 plt.xlabel(r'$a$')
82 plt.legend(loc='best')
83 plt.savefig('T_a_transition_universe', dpi=300)
84 plt.show()
85
86 # Calculate t0 depending on the thresholds
87 def calculatet0(T_thres, i):
88     return ((3 * M_pl * np.sqrt(10)) / (2 * np.pi)) * 1 / (
89         T_thres ** 2 * np.sqrt(0.5 * (g_before_after[i][0] + g_before_after[i
90 ][1])))
91
92 t0 = []
93 for i in range(len(T_thres)):
94     t0.append(calculatet0(T_thres[i], i))
95
96 a_t_prop_const = (1 / np.sqrt(t_rh)) * np.sqrt(g_star(T_rh) ** (1 / 6))
97
98 def a_pieewise(t):
99     a_t_prop_const = (1 / np.sqrt(t_rh)) * np.sqrt(g_star(T_rh) ** (1 / 6))
100     if np.exp(t) < t0[0]:
101         return a_t_prop_const * np.sqrt(
102             (g_star_before_after[0][0] ** (1 / 2)) / (g_before_after[0][0] ** (2
103 / 3))) * np.sqrt(
104             np.exp(t))
105     elif t0[0] ≤ np.exp(t) < t0[1]:
106         return a_t_prop_const * np.sqrt(
107             (g_star_before_after[0][1] ** (1 / 2)) / (g_before_after[0][1] ** (2
108 / 3))) * np.sqrt(
109             np.exp(t))
110     elif t0[1] ≤ np.exp(t) < t0[2]:

```

```

109         return a_t_prop_const * np.sqrt(
110             (g_star_before_after[1][1] ** (1 / 2)) / (g_before_after[1][1] ** (2
111 / 3))) * np.sqrt(
112             np.exp(t))
113     elif t0[2] ≤ np.exp(t) < t0[3]:
114         return a_t_prop_const * np.sqrt(
115             (g_star_before_after[2][1] ** (1 / 2)) / (g_before_after[2][1] ** (2
116 / 3))) * np.sqrt(
117             np.exp(t))
118     elif t0[3] ≤ np.exp(t) < t0[4]:
119         return a_t_prop_const * np.sqrt(
120             (g_star_before_after[3][1] ** (1 / 2)) / (g_before_after[3][1] ** (2
121 / 3))) * np.sqrt(
122             np.exp(t))
123     elif t0[4] ≤ np.exp(t) < t0[5]:
124         return a_t_prop_const * np.sqrt(
125             (g_star_before_after[4][1] ** (1 / 2)) / (g_before_after[4][1] ** (2
126 / 3))) * np.sqrt(
127             np.exp(t))
128     elif t0[5] ≤ np.exp(t) < t0[6]:
129         return a_t_prop_const * np.sqrt(
130             (g_star_before_after[5][1] ** (1 / 2)) / (g_before_after[5][1] ** (2
131 / 3))) * np.sqrt(
132             np.exp(t))
133
134 # Plot a(t)
135 a_approx_array = [a_pieewise(t_val) for t_val in t_range]
136 plt.loglog(np.exp(t_range), a_approx_array, 'b-', label=r'Piecewise Approx.',
137             linewidth=1)
138 for t0_val in t0:
139     plt.vlines(t0_val, min(a_approx_array), max(a_approx_array), linestyle='
140 dashed', colors='k', linewidth=0.5)
141 plt.title(r'$a(t)$ Transition')
142 plt.ylabel(r'$a$')
143 plt.xlabel(r'$t$ (1/GeV)')
144 plt.legend(loc='best')
145 plt.savefig('a_t_transition_universe', dpi=300)
146 plt.show()
147
148 # Plot T(t)
149 T_t_array = [T_approx_exp(a_val) for a_val in a_approx_array]
150 plt.loglog(np.exp(t_range), T_t_array, 'b-', label=r'Piecewise Approx.',
151             linewidth=1)
152 for t0_val in t0:
153     plt.vlines(t0_val, min(T_t_array), max(T_t_array), linestyle='dashed',
154                 colors='k', linewidth=0.5)
155 plt.title(r'$T(t)$ Transition')
156 plt.ylabel(r'$T$ (GeV)')
157 plt.xlabel(r'$t$ (1/GeV)')
158 plt.legend(loc='best')
159 plt.savefig('T_t_transition_universe', dpi=300)

```

```
156 plt.show()  
157
```