

```

1 # Import Modules
2 from numpy import arctan, pi, arange, sqrt, log, exp
3 import matplotlib.pyplot as plt
4 from scipy.integrate import odeint
5 from scipy.optimize import fsolve
6
7 # Define Constants
8 T_rh = 1e5
9 g_high = 500
10 g_low = 50
11 g_low_s = 200
12 T_0 = 100
13 T_width = 3
14 M_pl = 2.4e18
15
16 # =====
17 # Plotting g(T) Analytic Form
18 # =====
19
20 # Define analytic g(T) functions provided in the document
21 g_star = lambda T: g_low + ((g_high - g_low) / pi) * (arctan((T - T_0) / T_width)
    ) + pi / 2)
22 g_star_s = lambda T: g_low_s + ((g_high - g_low_s) / pi) * (arctan((T - T_0) /
    T_width) + pi / 2)
23
24 # Define T range to plot for
25 T_evals = arange(0, T_rh)
26 T_evals_zoomed = arange(T_0 - 10, T_0 + 100)
27
28 # Plot g(T)
29 plt.loglog(T_evals, g_star(T_evals), 'r-', label=r'$g_{\star}$')
30 plt.loglog(T_evals, g_star_s(T_evals), 'b-', label=r'$g_{\star S}$')
31 plt.hlines(g_low, min(T_evals), max(T_evals), linestyle='dashed', colors='k',
    linewidth=0.5)
32 plt.hlines(g_high, min(T_evals), max(T_evals), linestyle='dashed', colors='k',
    linewidth=0.5)
33 plt.hlines(g_low_s, min(T_evals), max(T_evals), linestyle='dashed', colors='k',
    linewidth=0.5)
34 plt.title(r'$g(T)$ Transition')
35 plt.ylabel(r'$g$')
36 plt.xlabel(r'$T$ (GeV)')
37 plt.legend(loc='best')
38 plt.savefig('g_transition_analytical', dpi=300)
39 plt.show()
40
41
42 # =====
43 # Plotting T(a) Analytic and Approximate Form
44 # =====
45
46
47 def T(a):
48     def f(T, a):
49         return T_rh * (g_high ** (1 / 3)) * (g_star_s(T) ** (-1 / 3)) / a
50
51     func = lambda T: T - f(T, exp(a))
52     return fsolve(func, T_rh)

```

```

53
54
55 def T_approx(a):
56     a_crit = T_rh / T_0
57     if exp(a) < a_crit:
58         return T_rh * (g_high ** (1 / 3)) * (g_high ** (-1 / 3)) / exp(a)
59     if exp(a) > a_crit:
60         return T_rh * (g_high ** (1 / 3)) * (g_low_s ** (-1 / 3)) / exp(a)
61
62
63 a = arange(log(1), log(1e7), step=0.01)
64 a_crit = T_rh / T_0
65
66 T_approx_array = [T_approx(a_val) for a_val in a]
67 T_analytic = [T(a_val) for a_val in a]
68
69 plt.loglog(exp(a), T_analytic, 'k--', linewidth=1, label="Analytic")
70 plt.loglog(exp(a), T_approx_array, 'b-', linewidth=1.25, label='Piecewise Approx
    .')
71 plt.title(r'$T(a)$')
72 plt.ylabel(r'$T$ GeV')
73 plt.xlabel(r'$a$')
74 plt.legend(loc='best')
75 plt.savefig('temperature', dpi=300)
76 plt.show()
77
78 plt.loglog(exp(a), T_analytic, 'k--', linewidth=1, label="Analytic")
79 plt.loglog(exp(a), T_approx_array, 'b-', linewidth=1.25, label='Piecewise Approx
    .')
80 plt.title(r'$T(a)$ (zoomed)')
81 plt.ylabel(r'$T$ GeV')
82 plt.xlabel(r'$a$')
83 plt.legend(loc='best')
84 plt.xlim(a_crit - 900, a_crit + 9000)
85 plt.ylim(T_0 - 90, T_0 + 900)
86 plt.savefig('temperature_zoomed', dpi=300)
87 plt.show()
88
89 print("Ratio at a=1e7: ", T_approx_array[-1] / T_analytic[-1])
90
91
92 # =====
93 # Plotting a(t) Analytic and Approximate Form
94 # =====
95
96 def a_pieewise(t):
97     t0 = ((3 * M_pl * sqrt(10)) / (2 * pi)) * 1 / (T_0 ** 2 * sqrt(0.5 * (g_low
    + g_high)))
98     if exp(t) < t0:
99         # return sqrt(2 * C_high) * sqrt(t)
100         return sqrt(exp(t) / t_rh)
101     if exp(t) > t0:
102         # return sqrt(2 * C_low) * sqrt(t)
103         return sqrt((g_high ** (1 / 6) * g_low ** (1 / 2)) / (g_low_s ** (2 / 3
    ))) * sqrt(exp(t) / t_rh)
104
105

```

```

106 def a_dot(t, a):
107     g = g_star(T(log(a)))
108     Temp = T(log(a))
109     return (pi / (3 * M_pl * sqrt(10))) * sqrt(g) * (Temp ** 2) * a * exp(t)
110
111
112 # Define reheat and critical time
113 t_rh = ((3 * M_pl * sqrt(10)) / (2 * pi)) * 1 / (T_rh ** 2 * sqrt(g_high))
114 t0 = ((3 * M_pl * sqrt(10)) / (2 * pi)) * 1 / (T_0 ** 2 * sqrt(0.5 * (g_low +
    g_high)))
115
116 # Define t space
117 t_evals = arange(log(t_rh), log(1e12 * t_rh), 0.01)
118
119 # Solve FRW and plot solution
120 soln = odeint(a_dot, y0=1, t=t_evals, tfirst=True)
121 plt.loglog(exp(t_evals), soln, 'k--', label=r'Analytic', linewidth=1)
122
123 # Solve for piecewise solution and plot
124 a_soln = [a_pieewise(t_val) for t_val in t_evals]
125 plt.loglog(exp(t_evals), a_soln, 'b-', label=r'Piecewise Approx.', linewidth=1.25
    )
126 plt.vlines(t0, min(a_soln), max(a_soln), linestyle='dashed', colors='k',
    linewidth=0.5)
127 plt.title(r'$a(t)$ Transition')
128 plt.ylabel(r'$a$')
129 plt.xlabel(r'$t$')
130 plt.legend(loc='best')
131 plt.savefig('a_t_transition', dpi=300)
132 plt.show()
133
134 # =====
135 # Plotting T(t) Analytic and Approximate Form
136 # =====
137
138 T_t_soln = [T(log(a_val)) for a_val in soln]
139 T_t_approx = [T(log(a_pieewise(t))) for t in t_evals]
140 plt.loglog(exp(t_evals), T_t_soln, 'k--', label=r'Analytic', linewidth=1.5)
141 plt.loglog(exp(t_evals), T_t_approx, 'b-', label=r'Approximate', linewidth=1.5)
142 plt.title(r'$T(t)$ Transition')
143 plt.ylabel(r'$T$')
144 plt.xlabel(r'$t$')
145 plt.legend(loc='best')
146 plt.savefig('T_t_function', dpi=300)
147 plt.show()
148

```