```python
# Import Modules
from numpy import arctan, pi, arange, sqrt, log, exp
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from scipy.optimize import fsolve

# Define Constants
T_rh = 1e5
g_high = 500
g_low = 50
g_low_s = 200
T_0 = 100
T_width = 3
M_pl = 2.4e18

# ============================
# Plotting g(T) Analytic Form
# ============================

# Define analytic g(T) functions provided in the document
g_star = lambda T: g_low + ((g_high - g_low) / pi) * (arctan((T - T_0) / T_width) + pi / 2)
g_star_s = lambda T: g_low_s + ((g_high - g_low_s) / pi) * (arctan((T - T_0) / T_width) + pi / 2)

# Define T range to plot for
T_evals = arange(0, T_rh)
T_evals_zoomed = arange(T_0 - 10, T_0 + 100)

# Plot g(T)
plt.loglog(T_evals, g_star(T_evals), 'r-', label=r'$g_{\star}$')
plt.loglog(T_evals, g_star_s(T_evals), 'b-', label=r'$g_{\star S}$')
plt.hlines(g_low, min(T_evals), max(T_evals), linestyles='dashed', colors='k', linewidth=0.5)
plt.hlines(g_high, min(T_evals), max(T_evals), linestyles='dashed', colors='k', linewidth=0.5)
plt.hlines(g_low_s, min(T_evals), max(T_evals), linestyles='dashed', colors='k', linewidth=0.5)
plt.title(r'$g(T)$ Transition')
plt.ylabel(r'$g$')
plt.xlabel(r'$T$ (GeV)')
plt.legend(loc='best')
plt.savefig('g_transition_analytical', dpi=300)
plt.show()


# ==========================================
# Plotting T(a) Analytic and Approximate Form
# ==========================================


def T(a):
    def f(T, a):
        return T_rh * (g_high ** (1 / 3)) * (g_star_s(T) ** (-1 / 3)) / a

    func = lambda T: T - f(T, exp(a))
    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_piecewise(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
```

```python
def a_dot(t, a):
    g = g_star(T(log(a)))
    Temp = T(log(a))
    return (pi / (3 * M_pl * sqrt(10))) * sqrt(g) * (Temp ** 2) * a * exp(t)


# Define reheat and critical time
t_rh = ((3 * M_pl * sqrt(10)) / (2 * pi)) * 1 / (T_rh ** 2 * sqrt(g_high))
t0 = ((3 * M_pl * sqrt(10)) / (2 * pi)) * 1 / (T_0 ** 2 * sqrt(0.5 * (g_low + g_high)))

# Define t space
t_evals = arange(log(t_rh), log(1e12 * t_rh), 0.01)

# Solve FRW and plot solution
soln = odeint(a_dot, y0=1, t=t_evals, tfirst=True)
plt.loglog(exp(t_evals), soln, 'k--', label=r'Analytic', linewidth=1)

# Solve for piecewise solution and plot
a_soln = [a_piecewise(t_val) for t_val in t_evals]
plt.loglog(exp(t_evals), a_soln, 'b-', label=r'Piecewise Approx.', linewidth=1.25)
plt.vlines(t0, min(a_soln), max(a_soln), linestyles='dashed', colors='k', linewidth=0.5)
plt.title(r'$a(t)$ Transition')
plt.ylabel(r'$a$')
plt.xlabel(r'$t$')
plt.legend(loc='best')
plt.savefig('a_t_transition', dpi=300)
plt.show()

print("Ratio at t=1e12T_rh: ", soln[-1] / a_soln[-1])

# ==========================================
# Plotting T(t) Analytic and Approximate Form
# ==========================================

T_t_soln = [T(log(a_val)) for a_val in soln]
T_t_approx = [T(log(a_piecewise(t))) for t in t_evals]
plt.loglog(exp(t_evals), T_t_soln, 'k--', label=r'Analytic', linewidth=1.5)
plt.loglog(exp(t_evals), T_t_approx, 'b-', label=r'Approximate', linewidth=1.5)
plt.title(r'$T(t)$ Transition')
plt.ylabel(r'$T$')
plt.xlabel(r'$t$')
plt.legend(loc='best')
plt.savefig('T_t_function', dpi=300)
plt.show()

```