

```

1 from q1_b import *
2
3 import matplotlib.pyplot as plt
4 from numpy import sqrt, pi, log
5
6
7 def l(g_w):
8     '''
9
10    Parameters
11    -----
12    g_w : Coupling Constant
13
14    Returns
15    -----
16
17    Lambda constant in ODE Equation
18
19    '''
20    m_x = 500
21    g_s = 106.75
22
23    def sigma_v():
24        m_x = 500
25        m_w = 80.4
26
27        return (m_x ** 2 / (16 * pi ** 2)) * ((g_w) / (m_w)) ** 4
28
29    def H():
30        M_pl = 2.4e18
31        m_x = 500
32        g_s = 106.75
33
34        return ((pi / 3) * sqrt(g_s / 10) * m_x ** 2) / (M_pl)
35
36    return ((2 * pi ** 2) / 45) * g_s * m_x ** 3 * sigma_v() / H()
37
38
39 def dfX(logx, logN, g_w):
40     '''
41
42    Parameters
43    -----
44    logx : specific X point
45    logN : Specific N point
46    g_w : Coupling Constant
47
48    Returns
49    -----
50    dF/dX at a given point.
51
52    '''
53    return l(g_w) * (exp(logN) - exp(-logN + 2 * log(N_eq(exp(logx))))) * exp(-
logx)
54
55
56 def dfN(logx, logN, g_w):

```

```

57     '''
58
59     Parameters
60     -----
61     logx : specific X point
62     logN : Specific N point
63     g_w : Coupling Constant
64
65     Returns
66     -----
67     dF/dN at a given point.
68
69     '''
70     return - l(g_w) * (exp(logN) + exp(-logN + 2 * log(N_eq(exp(logx))))) * exp(-
logx)
71
72
73 def f(logx, logN, g_w):
74     '''
75
76     Parameters
77     -----
78     logx : specific X point
79     logN : Specific N point
80     g_w : Coupling Constant
81
82     Returns
83     -----
84     F(X,N) written in log transformation
85
86     '''
87     return -l(g_w) * (exp(logN) - exp(-logN + 2 * log(N_eq(exp(logx))))) * exp(-
logx)
88
89
90 def dN(logx, logN, dlogx, g_w):
91     '''
92
93     Parameters
94     -----
95     logx : specific X point
96     logN : Specific N point
97     g_w : Coupling Constant
98
99     Returns
100    -----
101    Delta N Taylor Transformation to first order
102
103    '''
104    return (f(logx, logN, g_w) * dlogx + dfX(logx, logN, g_w) * (dlogx ** 2)) / (
1 - dfN(logx, logN, g_w) * dlogx)
105
106
107 # Set-up constants
108 g_w_array = [1e-4, 1e-3, 1e-2, 1e-1, 1]
109 g_w_colors = ['b', 'g', 'r', 'c', 'm']
110

```

```

111 # Set-up x space
112 N = 1000
113 x_min = 0.1
114 x_max = 1000.
115 x_evals = np.linspace(log(x_min), log(x_max), N)
116 dlogx = x_evals[1] - x_evals[0]
117
118 # Calculate  $N_x(x)$ 
119 for i in range(len(g_w_array)):
120     log_N_x = []
121     val = log(N_eq(exp(x_min)))
122     for j in range(len(x_evals)):
123         if j == 0:
124             log_N_x.append([val])
125         else:
126             old_val = val
127             val += dN(x_evals[j], old_val, dlogx, g_w_array[i])
128             log_N_x.append([val])
129     if g_w_array[i] ≥ 1e-3:
130         print("Experimental  $N(\infty) = \{ \}$  @  $g_w = \{ \}$ ".format(exp(log_N_x[-1]),
131 g_w_array[i]))
132         plt.hlines(6e-17 * g_w_array[i] ** (-3.8), x_min, x_max, linestyle='
133 dashed', colors='{ }'.format(g_w_colors[i]),
134 linewidth=1)
135         print("Analytic  $N(\infty) = \{ \}$ ".format(6e-17 * g_w_array[i] ** (-3.8)))
136         plt.loglog(exp(x_evals), exp(log_N_x), '{ }'.format(g_w_colors[i] + '-'),
137 linewidth=1.5,
138 label=r'$g_w$ = { }'.format(g_w_array[i]))
139
140 # Plot
141 plt.loglog(x, N_eq(x), 'k--', linewidth=1, label=r'$N_{eq}(x)$')
142 plt.title(r'$N_x(x)$')
143 plt.ylabel(r'$N_x$')
144 plt.xlabel(r'$x$')
145 plt.xlim(x_min, x_max)
146 plt.legend(loc='best')
147 plt.savefig('g_w_transitions', dpi=300)
148 plt.show()
149

```