```python
# Import modules

import matplotlib.pyplot as plt
from numpy import inf, exp, sqrt, linspace, array, pi
import numpy as np
from scipy.integrate import quad


def N_eq(x):
    '''
    Calculates the equilibrium co-moving number density for the WIMP particle

    Parameters
    ----------
    x - Defined to be x=m_x/T

    Returns
    -------
    N_eq : float/array : Equilibrium co-moving number density

    '''
    A = 45 / (4 * pi ** 4)
    g_x = 4
    g_s = 106.75

    const = A * g_x / g_s

    if isinstance(x, np.float64) or type(x) == float:
        # If x is a float and N_eq is wished to be determined at a point
        return const * quad(lambda a: a ** 2 / (exp(sqrt(a ** 2 + x ** 2)) - 1), 0, inf, epsabs=inf)[0]
    else:
        # If x is an array and N_eq is wished to be determined at multiple points
        return array(
            [const * quad(lambda a: a ** 2 / (exp(sqrt(a ** 2 + x_ ** 2)) - 1), 0, inf, epsabs=inf)[0] for x_
             in
             x])


# Set-up x space
x_min = 0.1
x_max = 50
x = linspace(x_min, x_max, 200)

# Plot Solution
plt.loglog(x, N_eq(x), 'k--', linewidth=0.5, label=r'$N_{eq}(x)$')
plt.title(r'$N_x^{eq}(x)$')
plt.ylabel(r'$N_x^{eq}$')
plt.xlabel(r'$x$')
plt.xlim(x_min, x_max)
plt.ylim(1e-10, 1e-1)
plt.savefig('N_eq_plot', dpi=300)
plt.show()

# Print Critical Values
x_vals = [0.1, 1., 10.]
```

```
for x in x_vals:
    print("N_eq({}) = {}".format(x, "{:.3e}".format(N_eq(x))))
```