Priyanshu 2022CH11465 CLL361 Assignment 4

```
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

# Define the model function for nonlinear fitting (Method 1)
def model(x, b):
    return x**b  # note: a is fixed to 1

# True parameter and settings
b_true = 2.0
n_points = 100
# We use logarithmically spaced x to cover several orders of magnitude
x_data = np.logspace(0, 2, n_points)  # from 1 to 100

# Set a noise level parameter (controls heteroscedastic noise)
noise_level = 0.15  # 15% noise factor

# Function to generate a dataset and compute b1 and b2
def generate_and_fit():
    # Generate the noiseless model
    y_true = model(x_data, b_true)
    # Add multiplicative noise (heteroscedastic: error scales with the magnitude of y)
    noise = np.random.normal(loc=0.0, scale=noise_level, size=n_points)
    y_noisy = y_true * (1 + noise)

    # Method 1: Nonlinear least squares on original data.
    # Provide an initial guess for b (say b=1.5)
    popt, _ = curve_fit(model, x_data, y_noisy, p0=[1.5])
```

```
    b1 = popt[0]

    # Method 2: Linear regression on log-log data.
    # Use only positive y_noisy values for the log
    mask = y_noisy > 0
    logx = np.log(x_data[mask])
    logy = np.log(y_noisy[mask])
    # Fit a line: logy = b2 * logx + c (ignore intercept if noise is multiplicative)
    b2, _ = np.polyfit(logx, logy, 1)

    return x_data, y_noisy, b1, b2

# Loop until the relative difference condition is met
reldiff = 0
max_attempts = 1000
attempt = 0

while attempt < max_attempts:
    attempt += 1
    x_data, y_noisy, b1, b2 = generate_and_fit()
    reldiff = abs(b1 - b2) / (abs(b1) + abs(b2))
    if reldiff >= 0.05:
        break

print(f"Dataset accepted after {attempt} attempt(s)")
print(f"Method 1 (nonlinear fit) estimated b1: {b1:.4f}")
print(f"Method 2 (log-log linear fit) estimated b2: {b2:.4f}")
print(f"Relative difference: {reldiff:.4f}")

# Save the dataset to a CSV file
df = pd.DataFrame({'x': x_data, 'y': y_noisy})
```

```
df.to_csv("fitted_dataset.csv", index=False)

print("Dataset saved to 'fitted_dataset.csv'.")
```

Output:

x	y
1	0.765234
1.047616	1.170004
1.097499	1.194084
1.149757	1.449197
1.204504	1.879505
1.261857	1.668063
1.321941	1.652892
1.384886	2.056362
1.450829	2.197361
1.519911	2.777851
1.592283	2.179298
1.668101	1.69354
1.747528	3.778223
1.830738	2.397797
1.91791	2.989858
2.009233	3.698275
2.104904	4.102591
2.205131	3.587138
2.31013	5.200557
2.420128	4.979276
2.535364	6.602883
2.656088	6.205483
2.782559	7.256903
2.915053	6.146997
3.053856	10.85664
3.199267	7.780755
3.351603	12.9709
3.511192	12.85465
3.67838	13.90132
3.853529	18.10666
4.037017	16.87349
4.229243	19.09603
4.430621	20.34349
4.641589	20.83338
4.862602	23.54733
5.094138	23.81628
5.336699	31.20776
5.59081	35.16435
5.857021	37.55755
6.135907	36.2021

6.428073	47.17524
6.734151	45.86843
7.054802	45.82947
7.390722	64.48769
7.742637	70.26284
8.111308	57.18328
8.497534	65.77985
8.902151	80.1807
9.326033	75.51671
9.7701	102.703
10.23531	102.3601
10.72267	106.2021
11.23324	130.2852
11.76812	157.5292
12.32847	142.5205
12.9155	168.607
13.53048	203.0366
14.17474	180.2727
14.84968	232.4851
15.55676	255.6854
16.29751	275.5233
17.07353	358.4761
17.8865	216.5476
18.73817	219.466
19.63041	336.6514
20.56512	423.0802
21.54435	478.8513
22.5702	502.6289
23.64489	553.2851
24.77076	640.9958
25.95024	665.5803
27.18588	636.4808
28.48036	729.494
29.83647	787.0009
31.25716	934.2408
32.74549	839.6125
34.30469	1217.283
35.93814	1074.596
37.64936	1377.634
39.44206	1195.349
41.32012	1721.129
43.28761	2015.797
45.34879	1913.385
47.5081	2207.625
49.77024	2337.048
52.14008	2888.802
54.62277	2611.474
57.22368	3237.682

59.94843	2810.196
62.80291	4315.799
65.79332	4763.372
68.92612	4185.007
72.20809	5244.086
75.64633	5996.094
79.24829	6807.084
83.02176	6327.984
86.9749	7562.091
91.11628	5037.857
95.45485	8552.367
100	11489.58