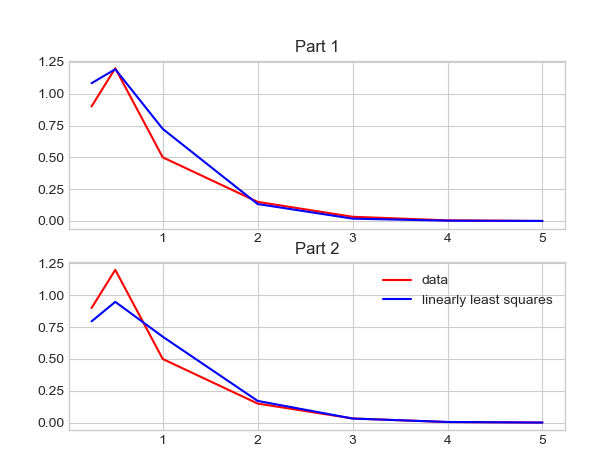
שאלה 4  
נחפש משוואה לינארית:  
נעביר את הנקודות למרחב החדש:  
ולכן:   
נבחין כי ולכן:  
   
  
נציב בחזרה בשתנים המקוריים:  
*נציב בחזרה במודל:*

שאלה 5

נפשט את הביטוי:

נשנה את שמות המשתנים:  
קיבלנו משוואה לינארית כך ש .

באמצעות curve fitting נמצא פולינום עבור המשוואה החדשה. במאצעותו נמצא את ערכי A ו-B, ונוכל לחשב את a, b המקוריים, ונקבל מודל מהתצורה המבוקשת עבור הנקודות הנתונות.  
  
ב. נבחין כי עבור שינוי מזערי בקלט קיבלנו מודל שונה משמעותית. הסיבה לכך היא כשכעברנו מהמודל המבוקש למודל לינארי, ביצענו טרנספורמציה לא לינארית. השינוי במרחב החדש זעיר, אך לאחר הטרנספורמציה אין הבטחה כי הוא ישאר כך (כמו במקרה זה).



הנקודות במודל (סעיף א'):

[[0.25, 1.0823244811875883], [0.5, 1.1922540979475191], [1.0, 0.723371696532291], [2.0, 0.13314255684980986], [3.0, 0.018379493415505987], [4.0, 0.002255265408434436], [5.0, 0.0002594376828445686]]

הנקודות במודל (סעיף ב'):  
[[0.25, 0.7952159183145904], [0.5, 0.9486097889100176], [1.0, 0.6749361348059603], [2.0, 0.17083735574557127], [3.0, 0.032431293065820384], [4.0, 0.005472593198137467], [5.0, 0.0008657516826664173]]

קוד:  
import numpy as np  
import math  
import matplotlib.pyplot as plt  
plt.style.use('seaborn-whitegrid')  
  
  
def solve(x\_vector, y\_vector):  
 new\_x = x\_vector  
 # new\_y = log (y / x)  
 new\_y = np.divide(y\_vector, x\_vector)  
 new\_y = [math.log(y) for y in new\_y]  
  
 curve = np.polyfit(new\_x, new\_y, 1) # Ax + B  
  
 b = -curve[0] # b = -A => b = -A  
 a = math.exp(curve[1]) # ln a = B => e^B = a  
  
 # y = a \* x \* math.exp(-b\*x)  
 y\_2 = lambda x: a \* x \* math.exp(-b \* x)  
  
 return y\_2  
  
  
x\_measure = np.array([0.25, 0.5, 1, 2, 3, 4, 5])  
y\_measure1 = np.array([0.9, 1.2, 0.5, 0.15, 0.033, 0.005, 0.0001])  
y\_measure2 = np.array([0.9, 1.2, 0.5, 0.15, 0.033, 0.005, 0.001])  
  
line1 = solve(x\_measure, y\_measure1)  
line2 = solve(x\_measure, y\_measure2)  
  
line1\_y = np.array([line1(x) for x in x\_measure])  
line2\_y = np.array([line2(x) for x in x\_measure])  
  
print('part 1')  
print([list(a) for a in zip(x\_measure, line1\_y)])  
  
print('part 2')  
print([list(a) for a in zip(x\_measure, line2\_y)])  
  
  
f, axarr = plt.subplots(2, 1)  
  
axarr[0].plot(x\_measure, y\_measure1, color='red', label="data")  
axarr[1].plot(x\_measure, y\_measure2, color='red', label="data")  
  
axarr[0].set\_title("Part 1")  
axarr[0].plot(x\_measure, line1\_y, color='blue', label="curve fitting")  
  
axarr[1].set\_title("Part 2")  
axarr[1].plot(x\_measure, line2\_y, color='blue', label="curve fitting")  
  
  
plt.legend()  
plt.show()