ECE-573-Homework-1

Name: **Zekun Zhang**

NetId: **zz364**

Email: <zekunzhang.1996@gmail.com>

Note:

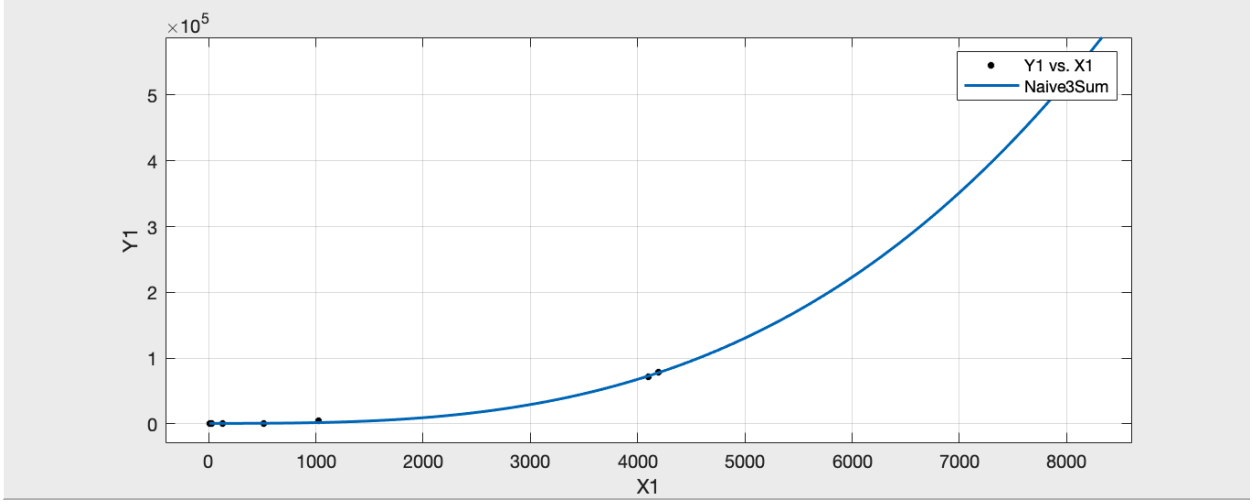
- PLEASE first read the '*README*' for data file reading part, if you want to test my code with other files.
    - For this file is written by *Markdown*, the figures can't be adjusted their sizes. I'm really sorry for that.

Q1:

1. Runtime data: (There may be some errors when your computer doesn't have enough hardware configuration—for this homework, 32 GB memory may get the data as I provided—when run 8 files together. Since there's no memory free operation until the program finished. Unless you test them separately. I'm really sorry for that.)

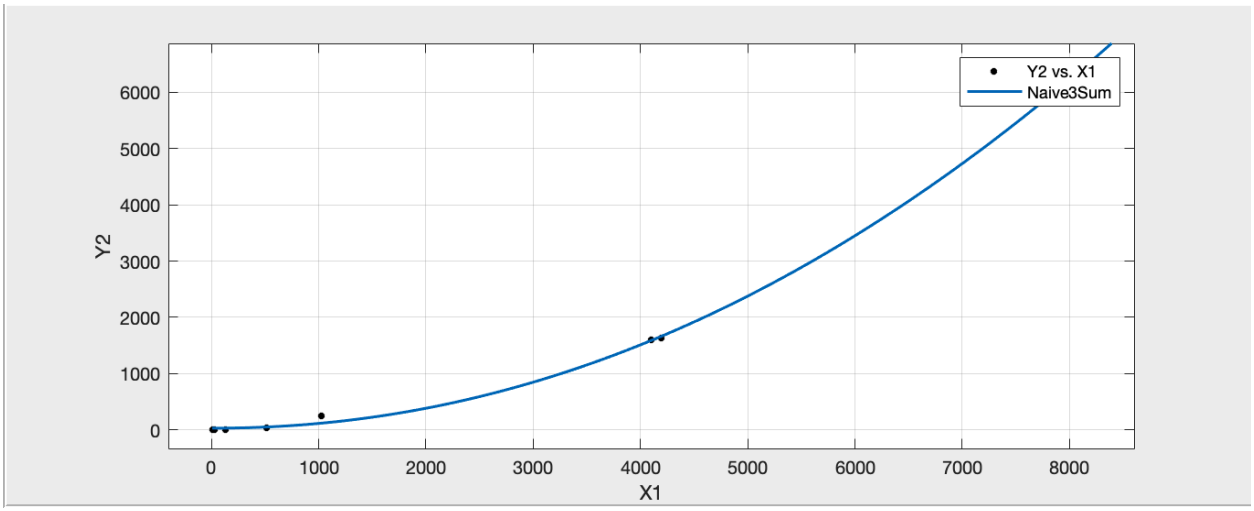
Size/Time(ms	O(N^3)	O(N^2(logN))
8	0.003	0.006
32	0.078	0.086
128	5.726	1.98
512	325.577	34.778
1024	5130.77	254.354
4096	71461.9	1598.3
4192	78771.8	1629.76
8192	559711	6537.65

2. Naive 3 Sum:



From the class, we can give a result that it's an O(N^3) algorithm since it has three nested for loop to find 3 numbers add up to 0. From the figure below, we can easily see that the result—MATLAB curve fitting tool gives out—is about x^2.959 which really fits the theoretical result.

3. Sophisticate 3 Sum:



Using the same method to analyze the other method, we can easily get that the result is x^2.065 which quite fits the result O(N(LogN)). Since (6537/1598)^(1/2) = 2.023.

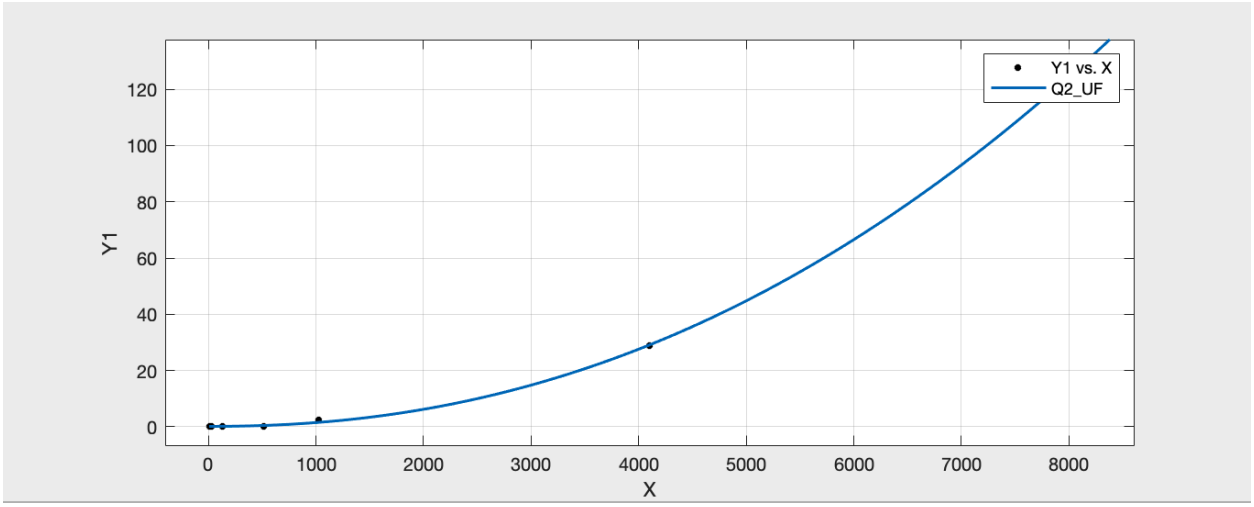
Q2:

1. Runtime:

Size/Time(ms	QuickFind	QuickUnion	WQuickUnion
8	0.013	0.004	0.002
32	0.016	0.002	0.002
128	0.021	0.005	0.006
512	0.091	0.064	0.024
1024	2.463	0.43	0.04
4096	28.939	13.073	0.168
8192	131.13	6.276	0.133

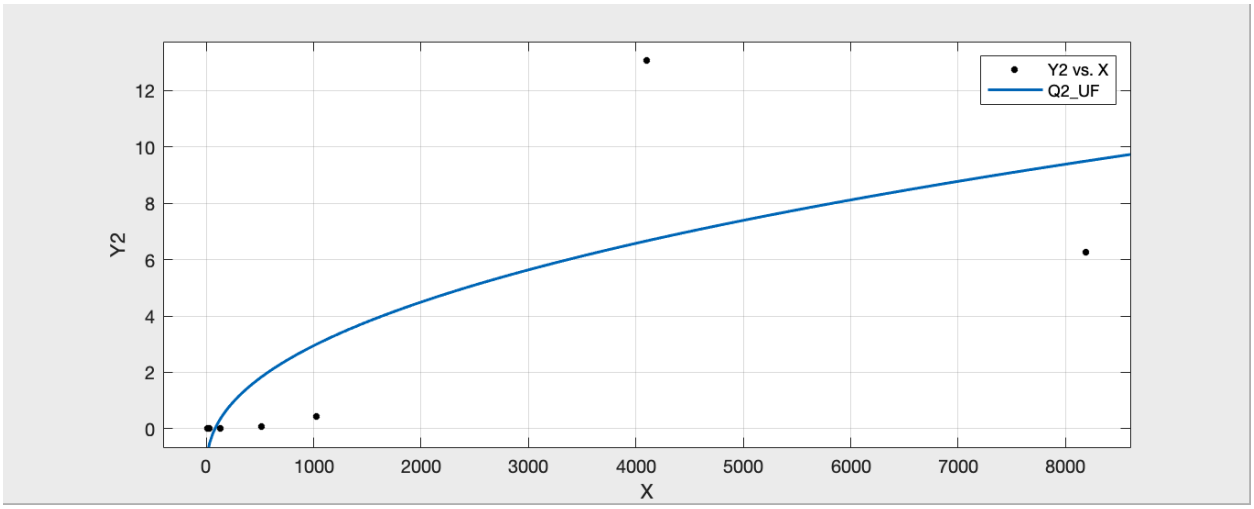
As the figure displayed, the QuickUnion algorithm is faster than the QuickFind, since it doesn't need to go through the whole array to change the root when connecting pairs. So the WeightedQuickUnion would be the fastest because there is less time to find roots.

2. QuickFind:



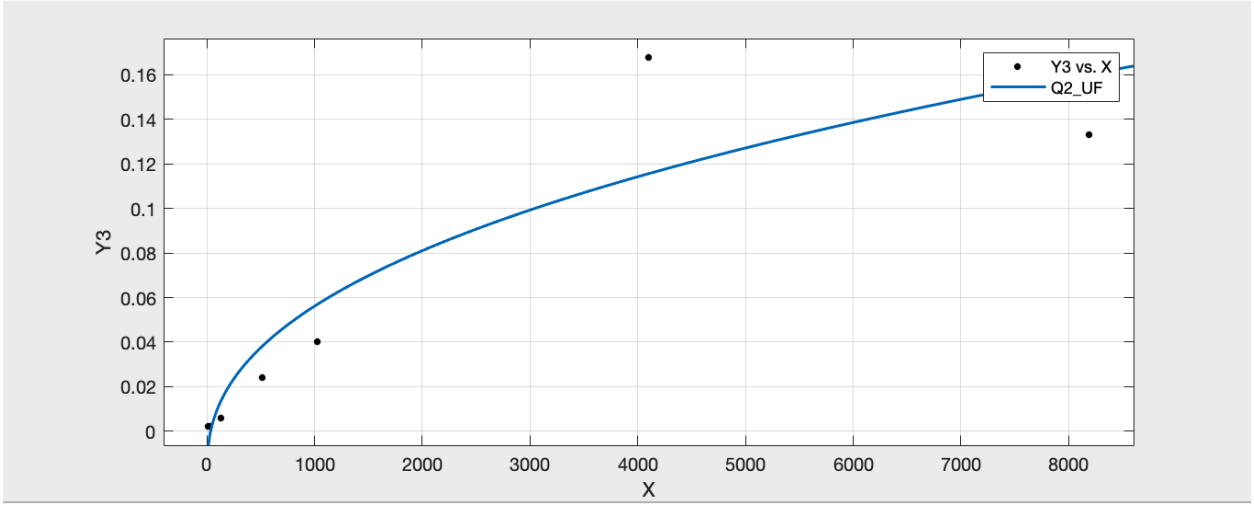
It's an algorithm, whose time complexity is O(MN). And the union() will run M times at most, which is the input, so the time complexity of it is O(MN). In the worst condition, it's O(N^2). And in the best condition, it's O(N).

3. QuickUnion:



Actually, in my view, it's the same time complexity with QuickFind. And we can also from the curves see that they are almost the same trends. These two algorithms share same time complexity and in the worst condition, it's O(N^2). And in the best condition, it's O(N).

4. WeightedQuickUnion:

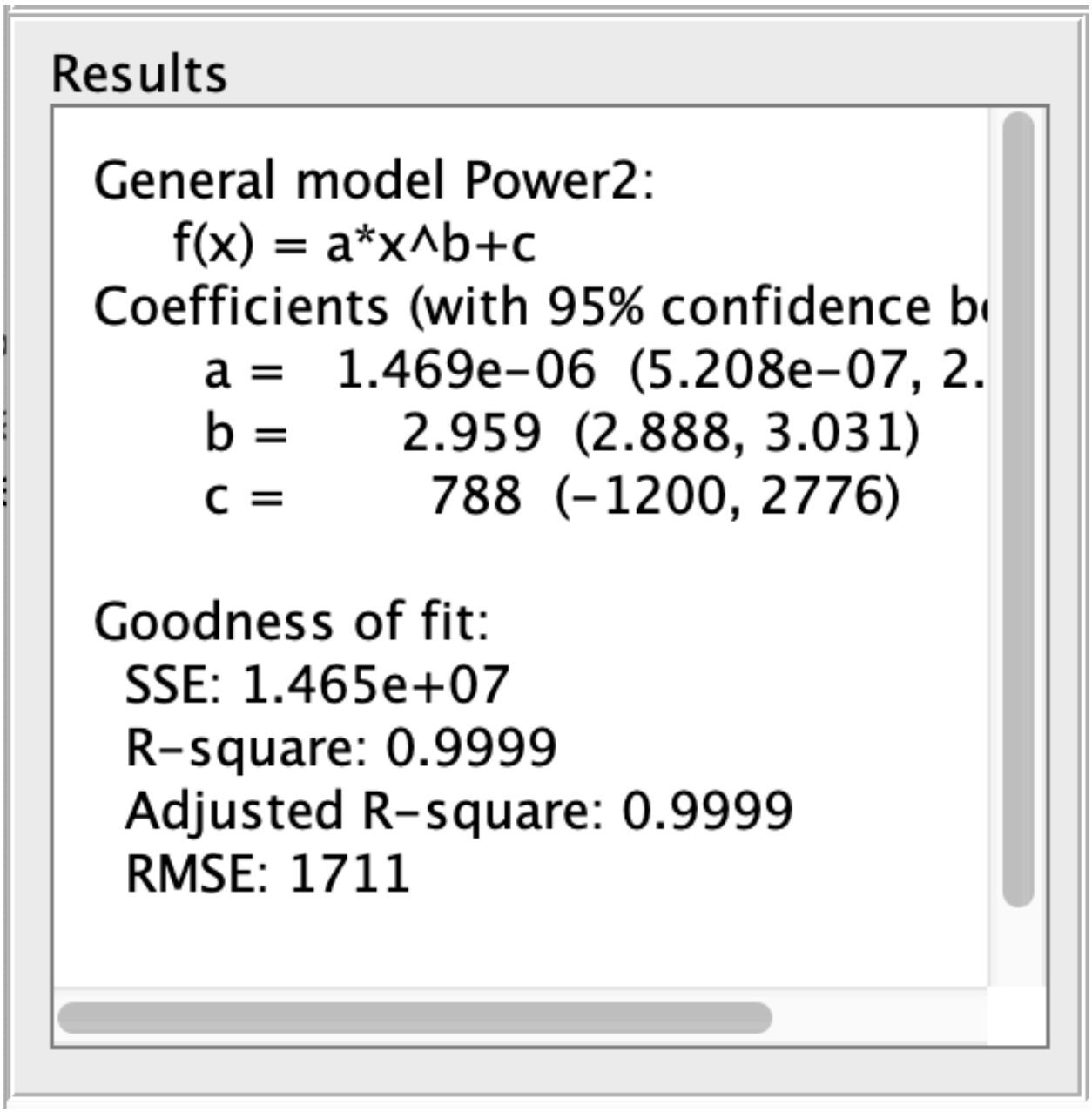


The order of this algorithm is O(logN), since there is and run at most M times is O(MlogN). The worst case is O(NlogN).

Q3:

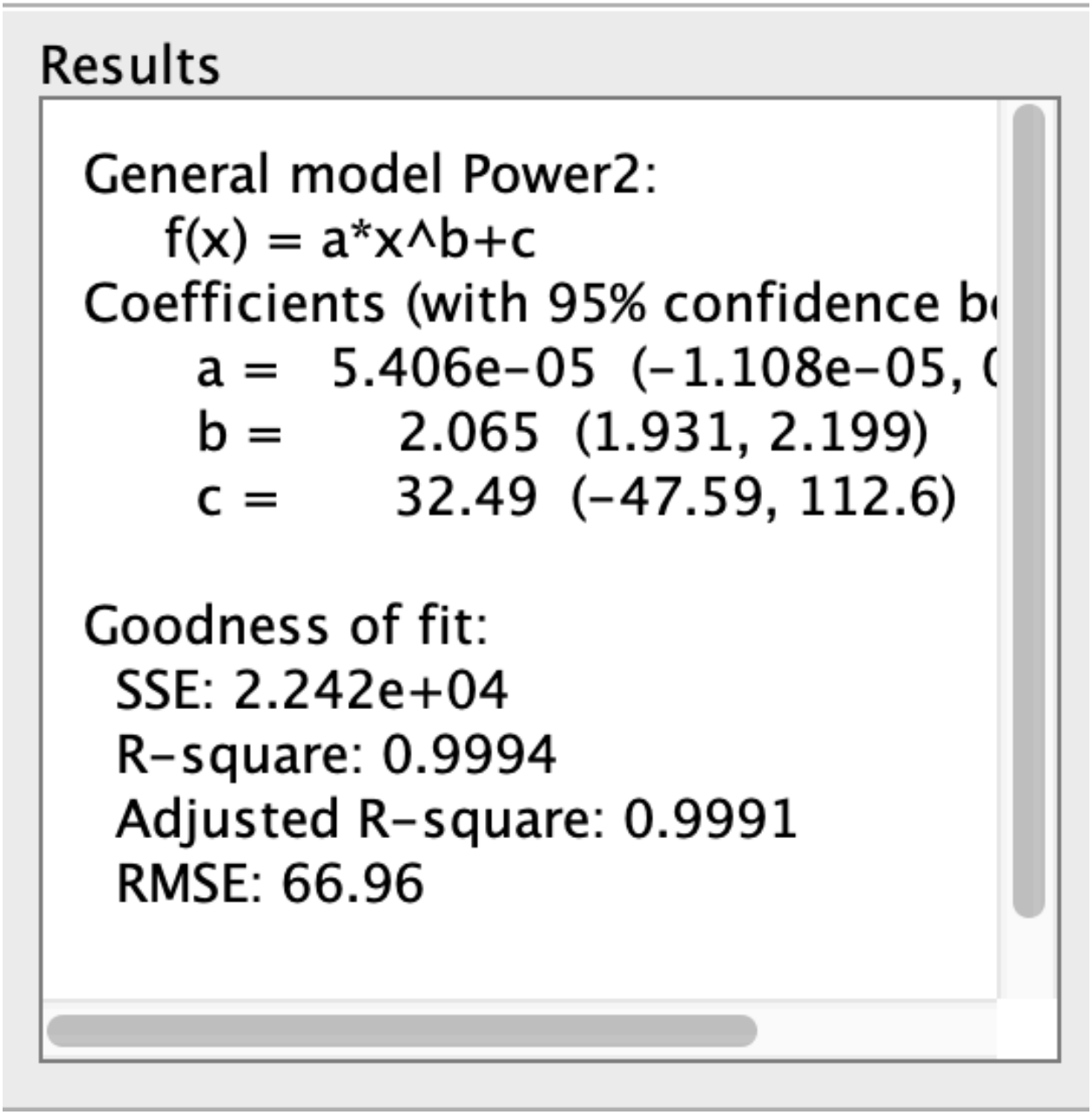
1. Suppose f(x) = 1.469xe^-06x^3, that is, F(N) = 1.469x10^-6xN^3

Assume that g(N) = N^3 and c = 1, so F(N)<cg(N), for N>Nc. We can find Nc = 1.



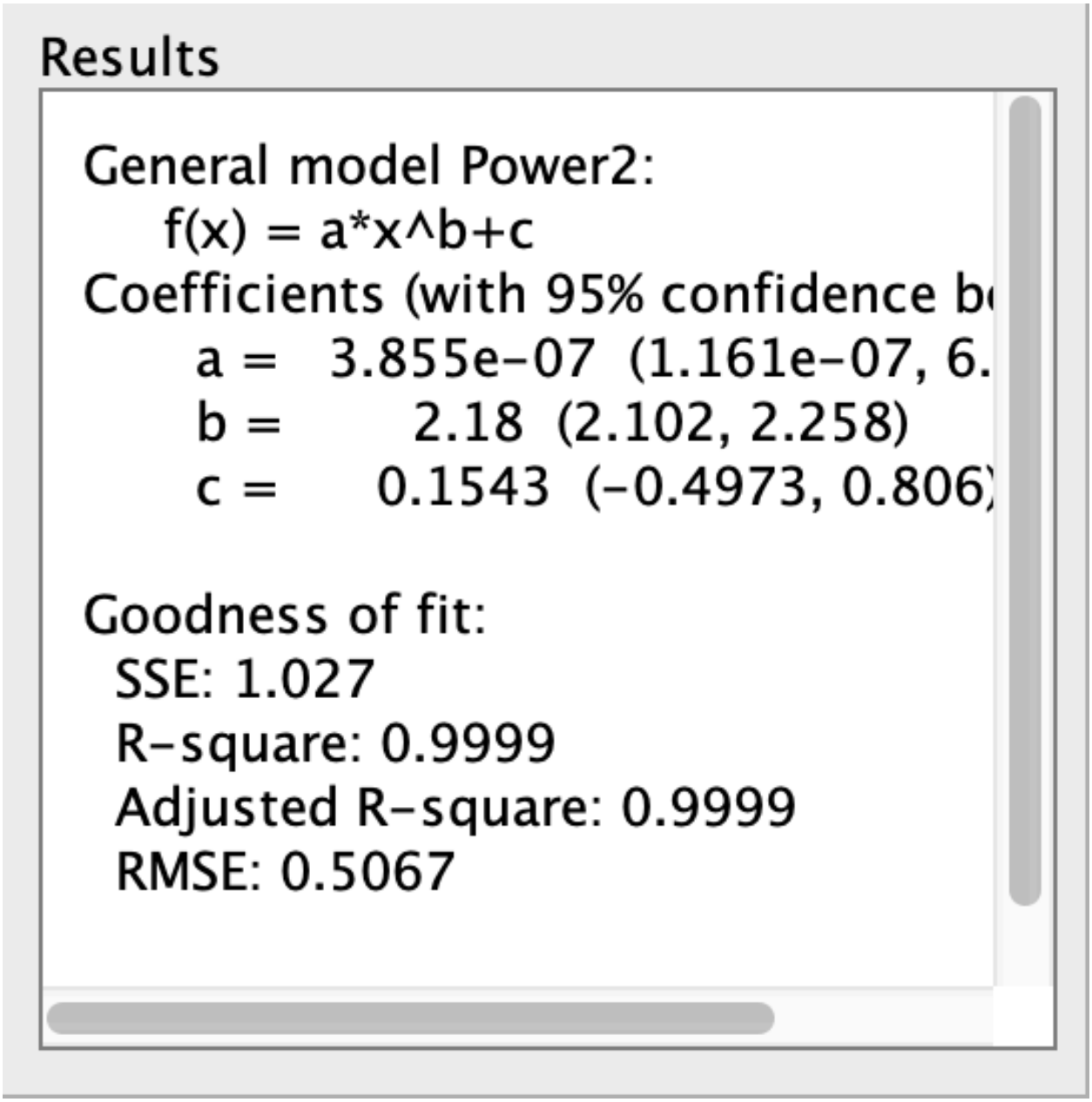
2. F(N) = 5.406x10^(-5) x N^2 x lg(N) + 32.49

Assume g(N) = N^2x lg(N), c = 2. As to get F(N)<c x g(N), where N>Nc, set Nc = 10.



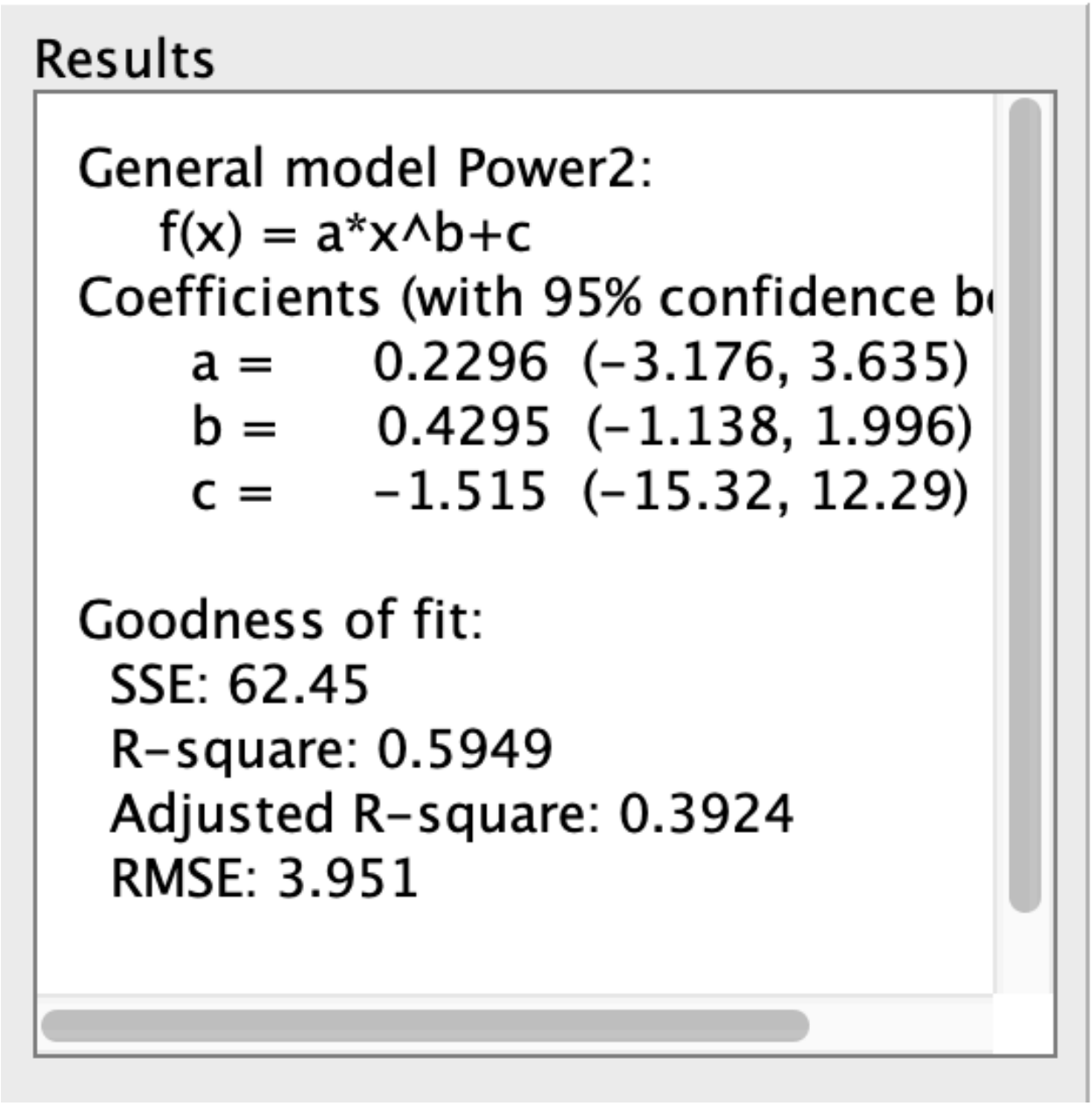
3. F(N) = 3.855 x 10^(-7) x N^2 + 0.1543.

Suppose g(N) = N^2, c = 2. Get Nc = 2.



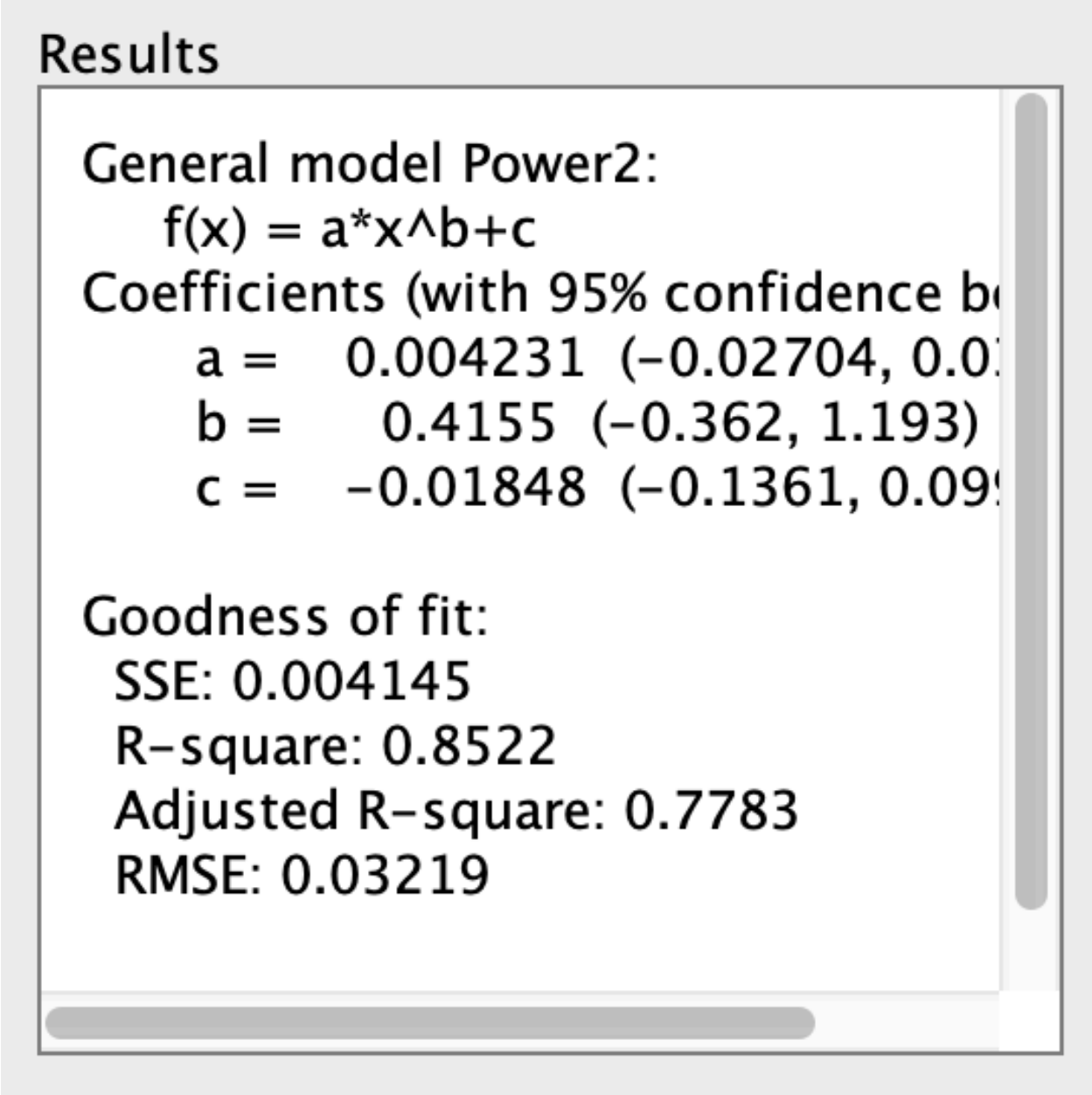
4. F(N) = 0.2296N^0.4295-1.515

Suppose g(N) = N^0.5, c = 2. Set Nc = 2.



5. F(N) = 0.004231N^0.4155 - 0.01848

Suppose g(N) = N^0.5, c = 2. Set Nc = 2.


Results
General model Power2:
f(x) = a*x^b+c
Coefficients (with 95% confidence b
a = 0.004231 (-0.02704, 0.0
b = 0.4155 (-0.362, 1.193)
c = -0.01848 (-0.1361, 0.09
Goodness of fit:
SSE: 0.004145
R-square: 0.8522
Adjusted R-square: 0.7783
RMSE: 0.03219

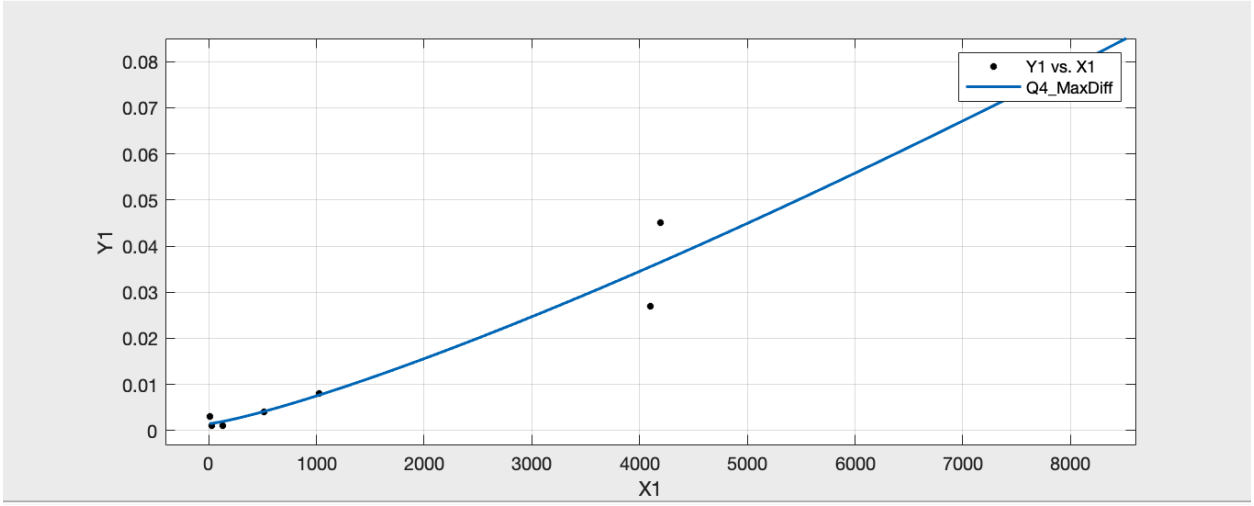
Q4:

1. Runtime:

Size	Time(ms)
8	0.003
32	0.001
128	0.001
512	0.004
1024	0.008
4096	0.027
4192	0.045
8192	0.081

I used the hw1-1 as the dataset to finish the test for this algorithm, in which I picked maximum and minimum values out and find out the maximum difference.

2. Analysis:



We can see from the curve that it's almost a linear line, which means this algorithm is O(N).

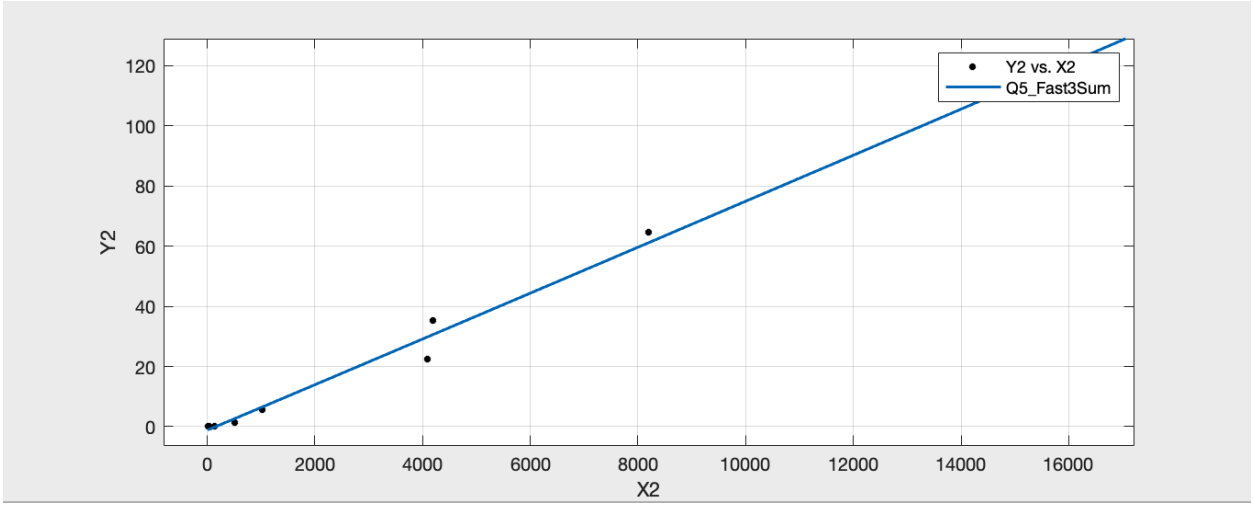
Q5:

1. Runtime:

Size	Time(ms)
8	0.005
32	0.008
128	0.113
512	1.462
1024	5.697
4096	22.401
4192	35.199
8192	64.684
16384	122.871

In this case, I also use the hw1-1 as the dataset to finish the test for this algorithm and what's more I still use more data like 16384 to test them.

2. Analysis:



However, after the curve printed out, we can only get a linear curve, I think it's because our data is always positive and when my algorithm traversal it once and find that there's no result. If we use other dataset then we can see the O(N^2) curve.