Name: Phuc H. Lam

I. Approach:

I first compute λ, then determine the order M, finally compute $\mathbf{w}$ and MSE. Beforehand, I would split the data set into train/test sets randomly (and evenly in probability), and run on train sets only.

1. Compute λ

To find λ, I first compute the parameters $\mathbf{w}$ for all $M = 1, 2, \ldots, 20$ and when $\lambda = 1, e^{-1}, \ldots, e^{-5}$.

With each of the parameters $M, \lambda, \mathbf{w}$, I find the mean square error (MSE) of the train sets, and plot points $(-ln\lambda, \text{MSE})$ for $M = 1, 2, \ldots, 19$. An example can be seen in Figure 1 below.

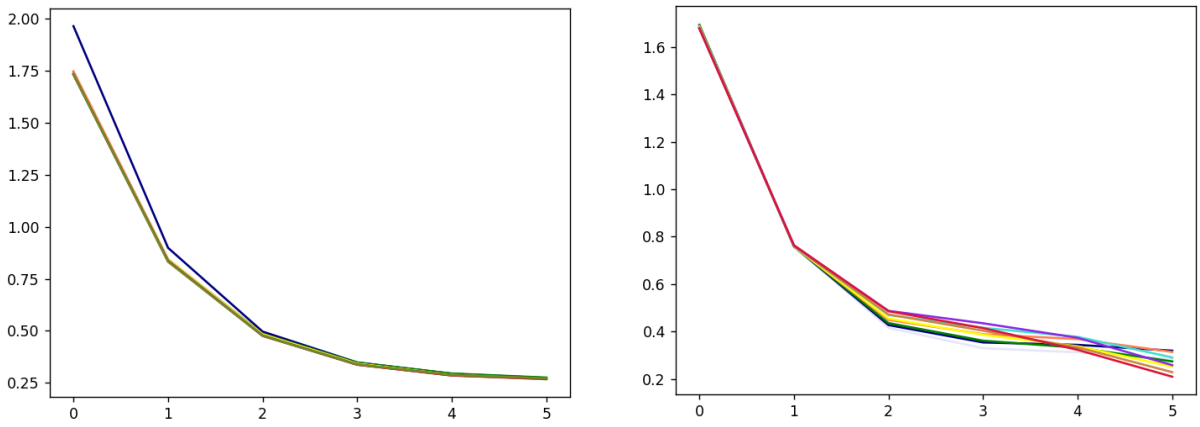

Figure 1: *plotting* $(-\ln\lambda, \text{MSE})$ *for data set A. The left picture is for* M *from* 1 *to* 9*, and the right picture is for* M *from* 10 *to* 19*.*

In the figure above, we see that the MSE drops drastically between $-\ln\lambda$ from 0 to 1 , and from 1 to 2. But from $-\ln\lambda = 2$ onwards, the MSE starts to "stabilize". This is why we can choose $-\ln\lambda = 2$, i.e. $\lambda = e^{-2}$ for data set A. A similar conclusion can be drawn for the remaining data sets.

2. Compute M

With the computed λ, we do the following:

- Partition the data set into train/test sets,
- Find the parameters $\mathbf{w}$ for each $M = 2, 3, \ldots, 19$ (using the train set),

- Find the MSE for train and test sets, and find M for which the MSE of test set is the minimum. (this is where the MSE of the test set "dips", and theoretically it gets worse afterwards).

We run the above in a loop for ρ times, and find the value of M that appears most frequently (in my code, I chose $\rho = 50$).

We can run the experiment for as many times as we want to. An example can be seen in Figure 2 below.

```
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
2
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
2
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
2
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
3
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
4
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
2
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
2
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
4
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
3
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
2
PS D:\Materials\Spring 2022\CSC 246\Project 1> python3 Project1.py
2
PS D:\Materials\Spring 2022\CSC 246\Project 1>
```

Figure 2: *A few runs to find the value of M that appears most frequently for data set A. In the above, we see that M fluctuates between $2, 3, 4$, so one of these values is the best order M. It is reasonable to choose $M = 3$ as the best order in this case.*

3. Compute $\mathbf{w}$ and MSE

Once we get M and λ, there is no need to split our data into train/test sets. Instead, we find $\mathbf{w}$ based on our whole data set, M, and λ. This is achieved using the function *find_MLE*. With the newly found $\mathbf{w}$, we can compute the MSE using the function *compute_MSE*.

(Optional) We can then scatterplot the points in the given data set and plot the best fit polynomial given $\mathbf{w}$.

II. Results and difficulties

1. Table of results (parameters are included in a separate directory)

Data set	λ	M	MSE
A	e^{-2}	3	0.5727904021045536
B	e^{-2}	4	0.39518080788551485
C	e^{-2}	5	0.8250734201567894
D	e^{-2}	19	0.428334929692502
E	e^{-2}	19	0.5424208693267885

2. Figures of data points and best fit polynomial

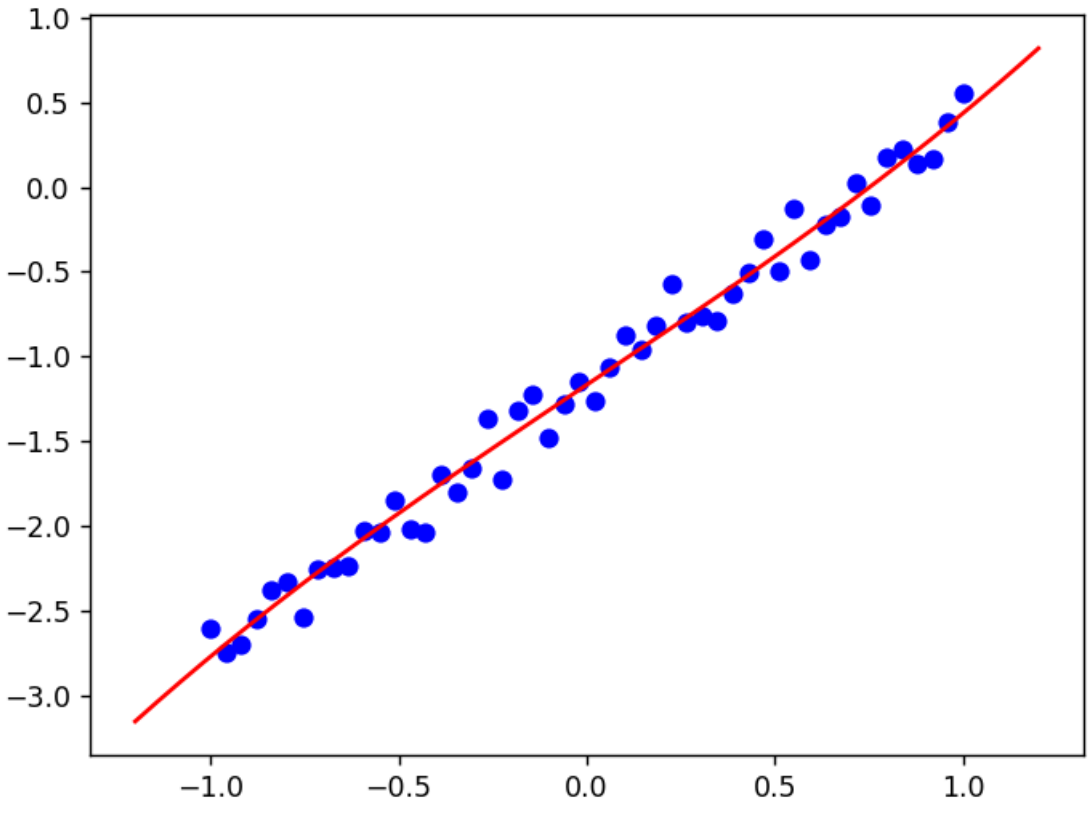

Figure 3A: *results for data set A.*

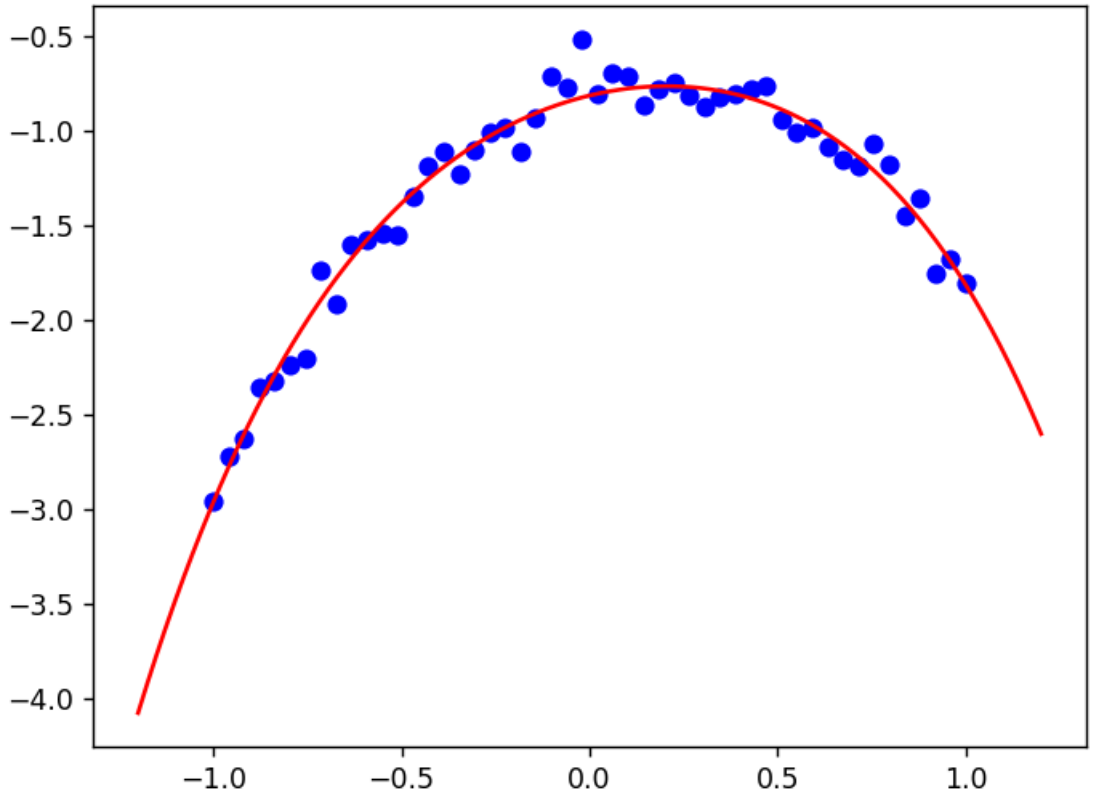

Figure 3B: *results for data set B.*

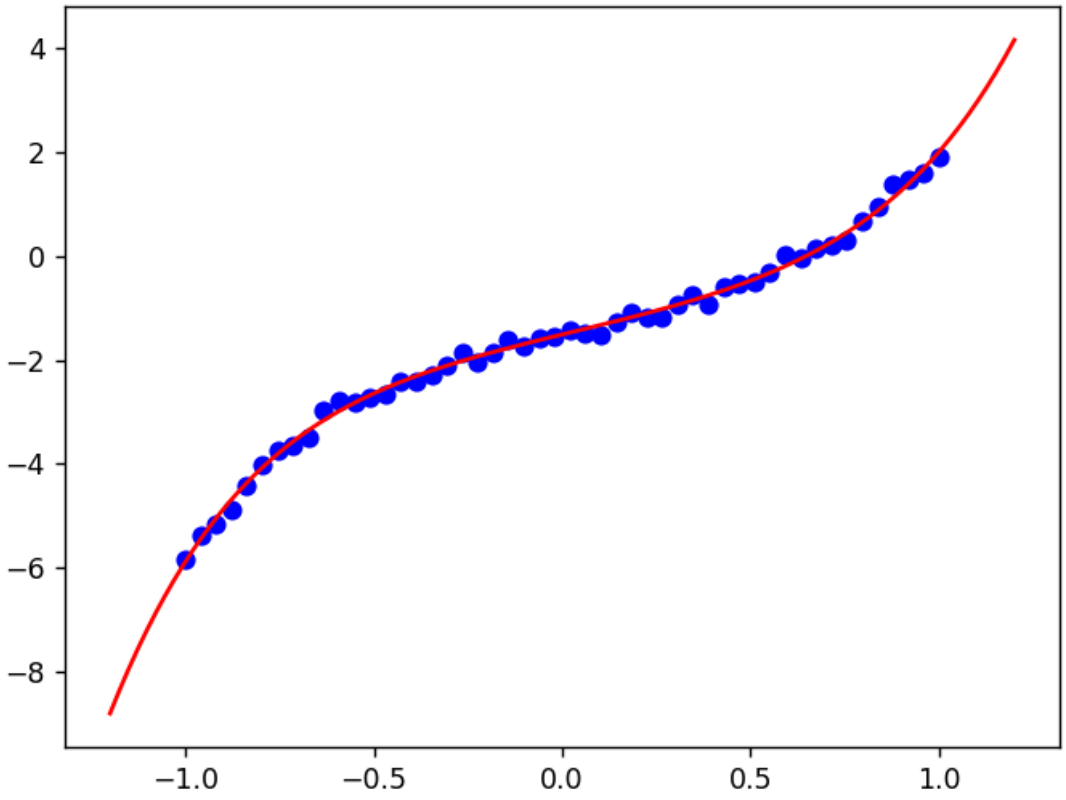

Figure 3C: *results for data set C.*

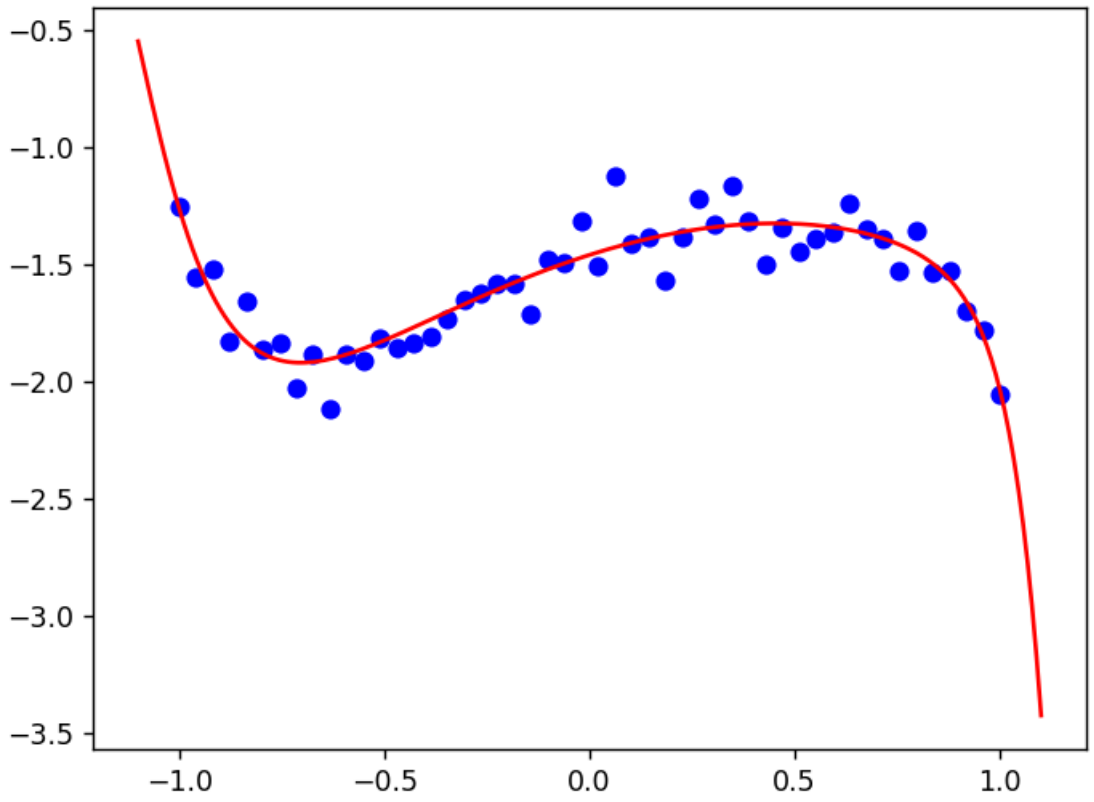

Figure 3D: *results for data set D.*

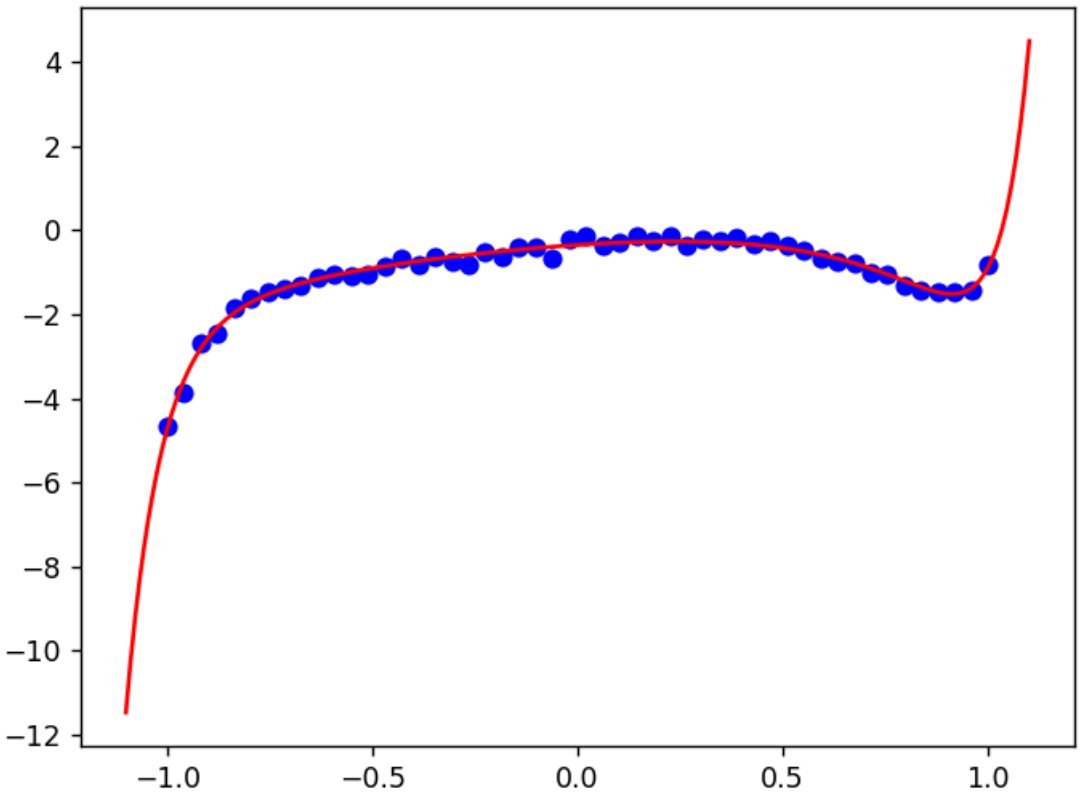

Figure 3E: *results for data set E*

3. Difficulties

A difficulty arises for data sets C and D when we find M. As we run the program, we see that there are different values of M that appears equally likely (for example, 5 and 19 for data set C, or 9 and 19 for data set D). One way to remedy this is to print the whole list of degree and see at which value it is more usually "concentrated". For example, for data set C, as we run the program over and over, we see that the value of M for which the MSE of the test set attains minimum tends to concentrate around 5 more often than 19, as in Figure 4 below.

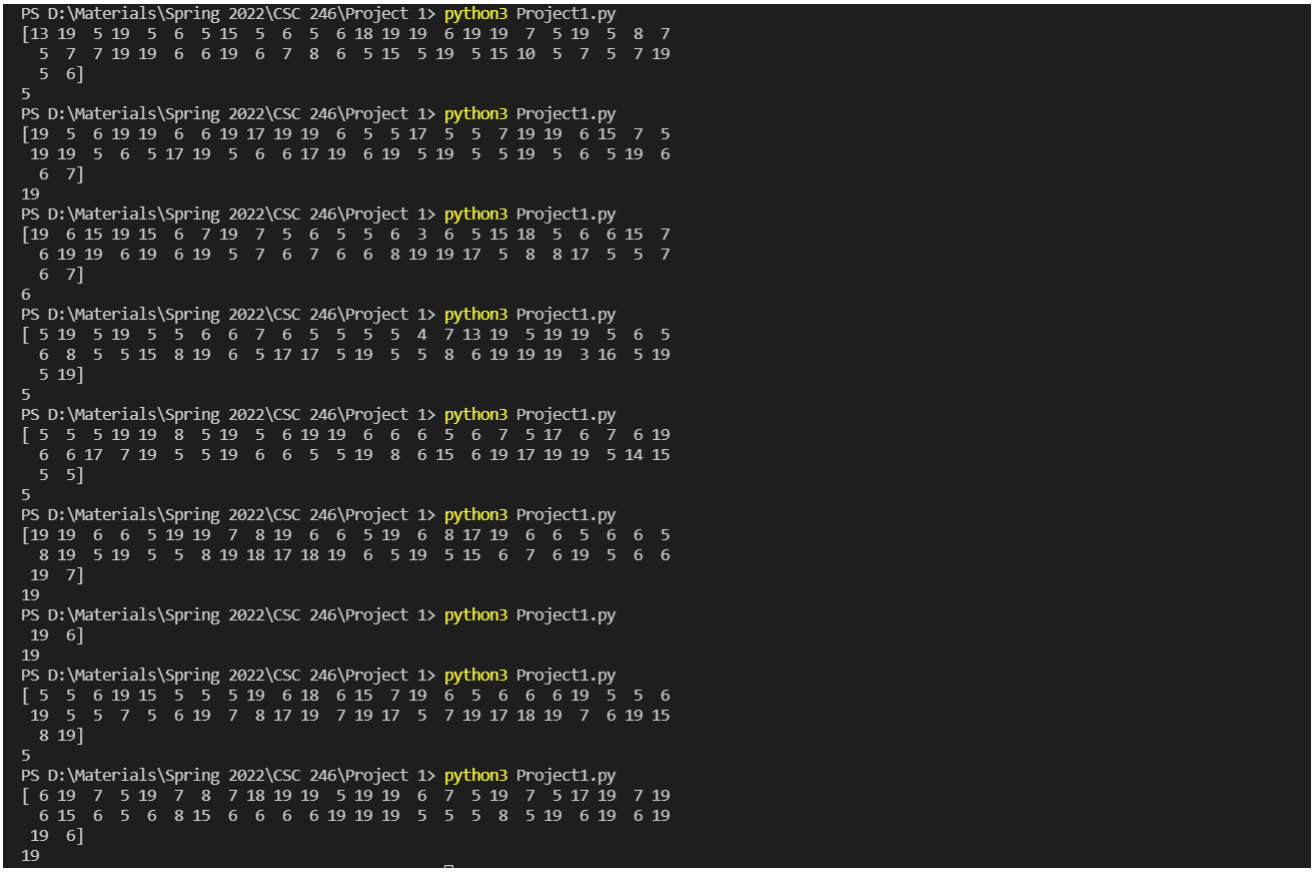

Figure 4: *list of values of M at which the MSE of the test set attains minimum.*

For data set D, the results seem much less obvious. One of the possible solutions is to create a "threshold" T. For each time we run the program (and create a loop with ρ iterations), if more than half of the list is $< T$, then the result would be 9; otherwise, the result would be 19. This depends on the way we choose the threshold T, which poses an issue, as we could not devise a somewhat subjective way to choose T.

Thus, we plot the best fit polynomial for $M = 9$ and $M = 19$, and see that they are of similar shapes, meaning that we can reasonably go with the one with a lower MSE, which is $M = 19$.