CW1: An Unknown Signal

Equations for Linear Regression

The Least Squares Method was used to calculate the equation for linear regression. The vector of coefficients can be calculated using:

$$A = (X^T X)^{-1} X^T y, where$$

$$A = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y \end{bmatrix} \qquad X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

resulting in the regression line: $\hat{y} = a_0 + a_1 x_1$. We then calculate the Sum Squared Error using:

$$SSE = \sum_i (\hat{y}_i - y_i)^2$$

A similar method can be used to compute the polynomial and cubic equations by modifying the A and X matrix.

Justification for choice of Polynomial Order

In order to determine the polynomial order, the total Sum Squared Errors (SSE) for each data file were calculated, as can be seen in the following table.

File Name	Quadratic	Cubic	Quartic	Quintic
Basic_3	15.743	**2.923×10^{-18}**	1.176×10^{-14}	5.020×10^{-9}
Basic_4	7.269×10^{-3}	**2.761×10^{-13}**	1.522×10^{-4}	8.050
Adv_1	218.598	198.232	381.162	379.655
Adv_2	3.653	3.651	3.397	3.444
Adv_3	986.583	1018.635	91487.034	99497.794
Noise_1	11.850	10.985	10.537	9.687
Noise_2	809.236	797.917	727.200	821.603
Noise_3	482.214	477.699	440.869	504.115

: lowest SSE

Figure 1: Table of Total SSE (3.d.p.) for given datafiles

Basic_1, basic_2 and basic_5 files are excluded as these files do not contain any polynomial functions so would not yield any insight into the polynomial order

For all basic datafiles, the cubic polynomial outputs the minimum SSE across all polynomial orders. Notably basic_3, which contains an isolated polynomial function, fits extremely well to the cubic function:

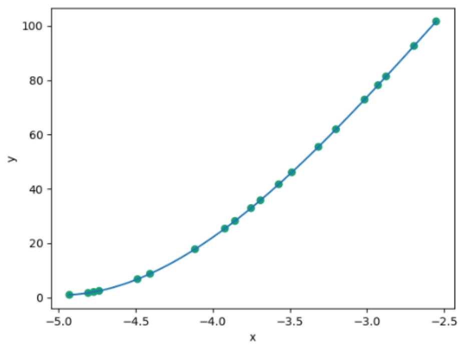


Figure 2: basic_3 plot for cubic function

We can therefore deduce that the polynomial function is likely to be cubic. To further confirm this, we can look at the other data files. We can rule out the quadratic function as it only performs marginally better in 1 datafile(adv_3). The quartic function outperforms the cubic function in 3 datafiles, but yet again, this difference is insignificant. On the other hand, both quartic and quintic clearly fit poorly in some instances. Most notably, the SSE of the quartic and quintic functions exceeds that of the cubic by over 89 and 97 times respectively in the adv_3 file. This is further exemplified in the following plots of the adv_3 datafile, where the last segment is underfitting to a linear function for orders 4 and 5.

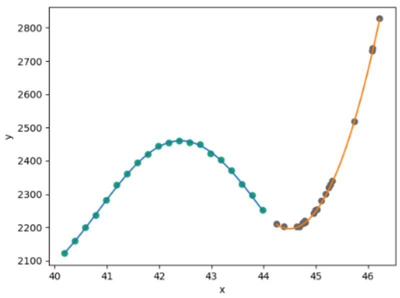


Figure 3a: last 2 segments for adv_3 for order 3

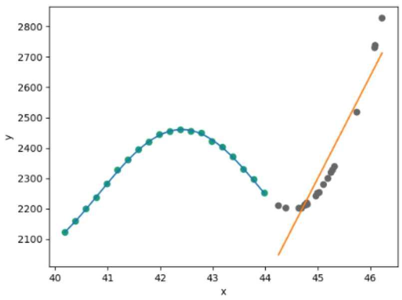


Figure 3b: last 2 segments for adv_3 for order 4

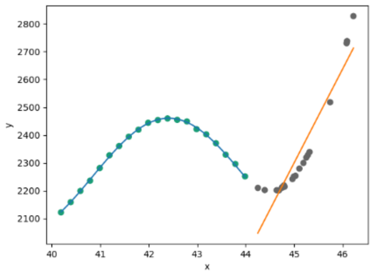


Figure 3c: last 2 segments for adv_3 for order 5

While the cubic function does not yield the lowest SSE for every file (which we can attribute to overfitting), the difference between its SSE and the lowest SSE for that file is insignificant compared to that of the other orders. We can therefore conclude that the polynomial function is a cubic function.

<u>Justification for Choice of Unknown function</u>

Prior to discovering the unknown function, all basic files outputted a total error close to 0, bar basic_5. From this, we can assume that this file contains the unknown function. We can first take a look at the polynomial orders. Polynomial order 9 gives the lowest error.

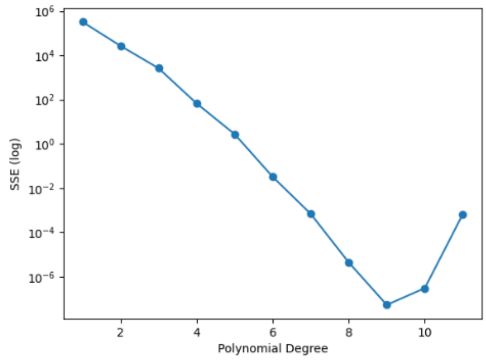
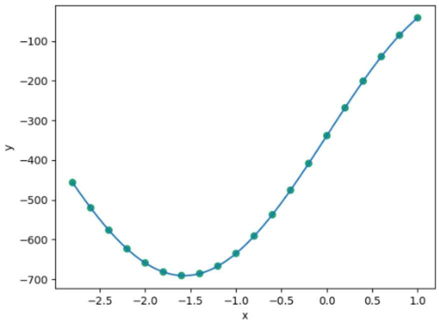


Figure 4: SSE for basic_5 of orders up to 12

However, the unknown function is likely not a polynomial, so we consider other functions by first looking at the plot. The function's fluctuations means that we can further rule out exponential functions. The oscillating pattern, coupled with a wavelength of around 2π suggests that the unknown function is likely to be a sinusoidal function. We can also see that the equilibrium is approximately 0 on the x-axis, and so a sine function would be a better fit compared to a cosine function, which is further confirmed by testing both functions on datafile basic_5 *(figure 5)*. We can deduce from both the insignificant SSE and the well-fitted plot that the unknown function is a sine function. (We can attribute the low error for polynomial order 9 to the Taylor series expansion).

Sine	Cosine
2.496×10^{-25}	661402.570

Figure 5a: Table with comparison of cosine and sine Sum Squared Errors for basic_5 datafile



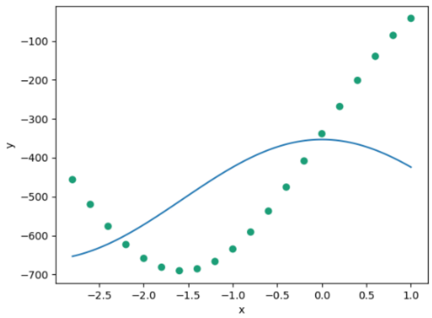


Figure 5b: basic_5 fitted to sine function

Figure 5c: basic_5 fitted to cosine function

Procedure for choosing between the 3 functions

After finding the polynomial and unknown function, we can compute the total reconstruction error by summing up the lowest SSE for each segment in the files.

File name	SSE (3.d.p.)	Function Type
basic_1	1.681×10^{-27}	linear
basic_2	6.467×10^{-27}	linear, linear
basic_3	2.923×10^{-18}	cubic
basic_4	2.761×10^{-13}	linear, cubic
basic_5	2.496×10^{-25}	sine
adv_1	198.232	cubic, cubic, cubic
adv_2	3.651	sine, cubic, sine
adv_3	1018.635	cubic, cubic, sine, cubic, sine, cubic
noise_1	10.985	cubic
noise_2	797.917	cubic, cubic
noise_3	477.699	cubic, cubic, sine

Overfitting

Figure 6: Total SSE computed by choosing function with lowest SSE for each function

However, comparing the function types and the plot, it is clear that some of these functions are overfitting. In particular, segments which are linear are instead shown to be cubic.

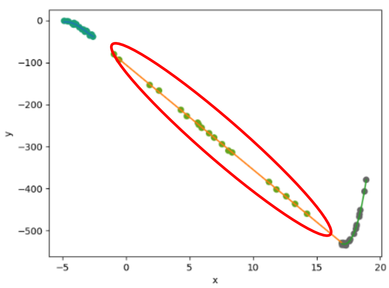


Figure 7a: adv_1 plot

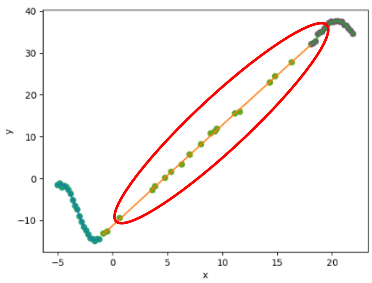


Figure 7b: adv_2 plot

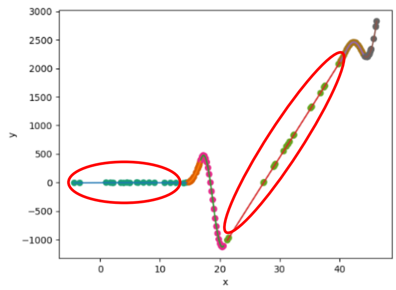


Figure 7c: adv_3 plot

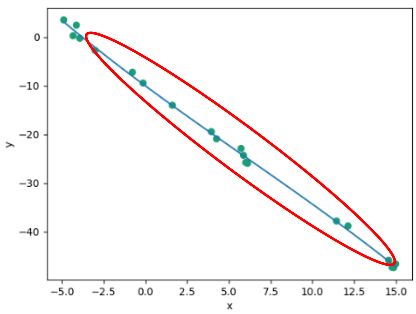


Figure 7d: noise_1 plot

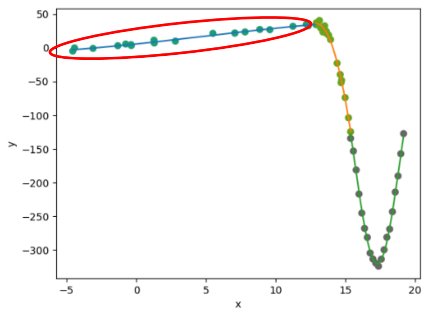


Figure 7e: noise_3 plot

In order to account for overfitting, we can use cross-validation to determine the suitable function. There are 3 different types of cross-validation to choose from – hold-out, k-fold and leave-one-out cross-validation. However, as there are only 20 datapoints for each segment, using hold-out cross-validation would lead to an overreliance on how the training and testing sets are chosen. Using k-fold and leave-one-out cross-validation (LOOCV) negates this bias but introduces a trade-off regarding efficiency. However, as the datafiles are relatively small, I believe that the benefit of more accurate results outweighs the cost of a slightly slower program. K-fold cross-validation would result in a variance in the results. As a result, I have chosen to use LOOCV at the extra cost of fitting the model 20 times, compared to k times.

File name	SSE (3 d.p.)	Function Type
basic_1	1.681×10^{-27}	linear
basic_2	6.467×10^{-27}	linear, linear
basic_3	2.923×10^{-18}	cubic
basic_4	2.761×10^{-13}	linear, cubic
basic_5	2.496×10^{-25}	sine
adv_1	199.726	cubic, linear, cubic
adv_2	3.6585	sine, linear, sine
adv_3	1056.741	sine, cubic, sine, linear, sine, cubic
noise_1	12.207	linear
noise_2	849.552	linear, cubic
noise_3	482.909	linear, cubic, sine

: SSE unchanged

Overfitting solved

Overfitting unsolved

Figure 8: Final results of total reconstruction error using leave-one-out cross-validation

This method solves the problem of overfitting with the exception of the first segment of the adv_3 file, which is fitted to a sine function. Looking at the plot generated *(figure 7c)*, a linear function should be a better fit.

We can take a look at other forms of cross-validation. Running 10-fold cross-validation 1000 times for this datafile, we obtain a variety of results:

SSE (3.d.p)	Function	Frequency
1056.741	**sine, cubic, sine, linear, sine, cubic**	**665**
1058.480	**linear, cubic, sine, linear, sine, cubic**	**266**
1044.158	sine, cubic, sine, cubic, sine, cubic	50
1045.897	linear, cubic, sine, cubic, sine, cubic	18
1031.218	cubic, cubic, sine, linear, sine, cubic	1

Overfitting solved

Overfitting unsolved

Figure 9: SSE for adv_3 by running 10-fold cross-validation 1000 times

The mode is 1056. 741 (same as LOOCV). The second most frequent result is 1058.480 (the optimal result with no overfitting), with the first segment being fitted to a linear function. K-fold cross-validation sometimes gives the optimal result but this is not always the case. In this instance, 5% of the time, k-fold cross-validation fared less well compared to LOOCV, by overfitting 2 segments. K-fold cross-validation yields sub-optimal results for other files as well. Consequently, LOOCV generally gives the most accurate results.

However, k-cross validation could potentially be more accurate if the results were to be double-checked by a human, who determines which of the outcomes is optimal. However, this would not be feasible for more complicated data where it is hard to determine the function by eye, or where the datafiles are big. Moreover, computing the datafiles numerous times is computationally expensive.