# CW1: Unknown Signal Report

Marcus Patel

## The Task

We were given files each containing a multiple of 20 data points, then we would reconstruct the signal of the points and calculate residual error to see how close the signal is to the data points. We want our model to be able to predict new data points accurately, so we would want our model to minimise this resulting error as this reflects its performance.

## Least Squares Regression (LSR)

The Least Squares Regression Line is the line that makes the vertical distance from the data points to the regression line as small as possible. The parameters of the line can be used in our reconstructed signal as the line will be as close as possible to our data points. For a linear line, the total vertical distance of our line from data points is $R(a,b) = \sum_{i=0}^{n}(y_i - (a + bx_i))^2 = \|\mathbf{y} - X\mathbf{a}\|^2$ which would be our residual error, our aim is to minimise this value. To make calculations easier, we will represent it using matrices and vectors, where n the represents number of data points and

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \mathbf{a} = \begin{pmatrix} a \\ b \end{pmatrix}.$$

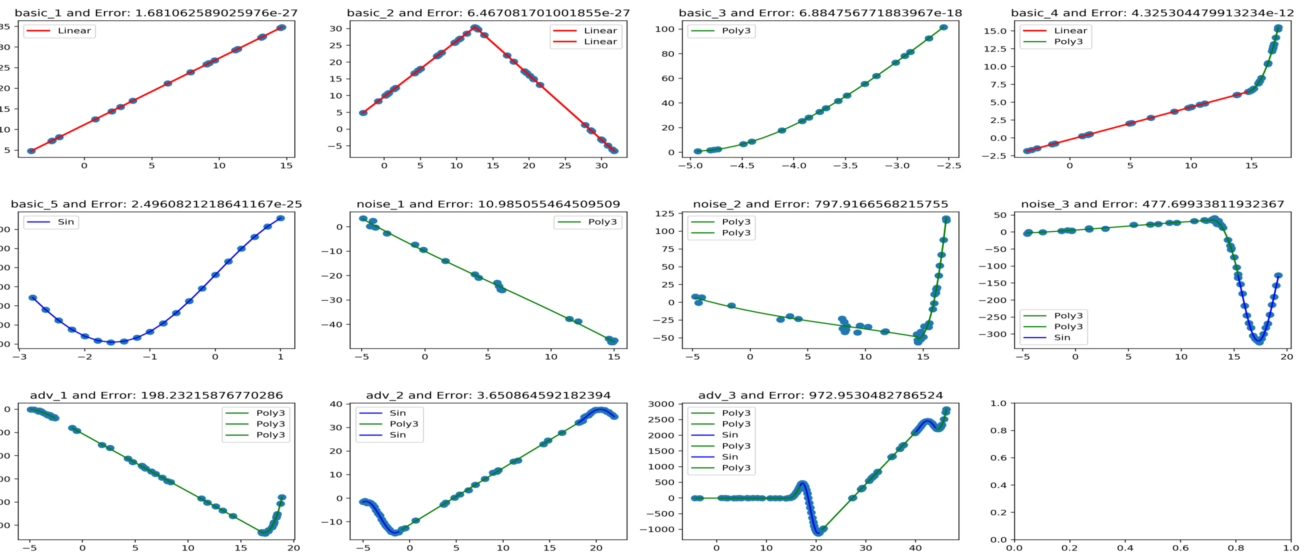
We need to solve $\|\mathbf{y} - X\mathbf{a}\|^2 = 0$ in order to find the parameters $\mathbf{a}$. As we want the vector length to be 0, we can re-arrange as such $X\mathbf{a} = \mathbf{y}$. We need to make $\mathbf{a}$ the subject but we can't multiply the inverse of X to both sides as it is not a square matrix, so we multiple both side by the transpose $X^T$, so that we have a square matrix multiplying $\mathbf{a}$. As we know have a square matrix, we can solve $\mathbf{a}$ by simply multiplying both sides by $(X^TX)^{-1}$. Finally, we have our solution $\mathbf{a} = (X^TX)^{-1}X^T\mathbf{y}$ by having used least squares regression to find our parameters $a$ and $b$.

However, the best fitted line may not be linear, the signal could be exponential, quadratic, etc. As we are solving in matrices, we can easily extend our method of least squares so that we can fit other line types. For a polynomial of degree $p+1$, $y_i = a_0 + a_1x_1 + a_2x_2^2 ... + a_nx_n^p$ can be solved as above where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^p \\ 1 & x_2 & x_2^2 & ... & x_2^p \\ \vdots & \vdots & \vdots & ... & \vdots \\ 1 & x_n & x_n^2 & ... & x_n^p \end{bmatrix}, \quad \mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}.$$

Same as before, $\mathbf{a} = (X^TX)^{-1}X^T\mathbf{y}$ in order to get parameters of the polynomial line. To fit a line that follows the Sin curve, we have $y_i = a_0 + a_1 sin(x_i)$, and then define matrices as similar to above, this can also be replicated for an exponential curve by replacing sin. Our model will use this method of Least Squares to determine which type of line best fits each segment of data points so that we can reconstruct the signal with minimal error.

## Results



The figures show all the plots for the basic, noise and advance data points, with their respective file and the function used to estimate that segment.

By having a look at all the basic graphs, we can see that residual error is close to 0. Initially, the functions included to determine type of line segment were Linear, all Polynomials of degree up to 4, Sine and exponential. We can assume that our unknown function is the Sine function as it reconstructs the signal for basic5 perfectly and the polynomial function I had chosen was of degree 3 as the model determined it fitted best for basic3. So I removed the other functions as they were not a best fit from any of the line segments for the basic files.

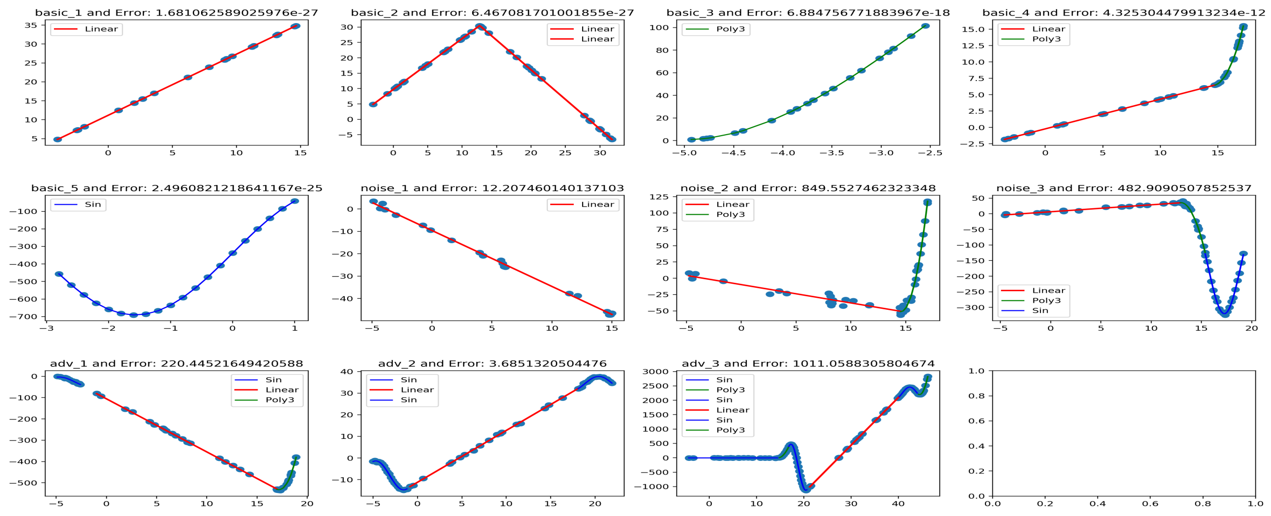
## Overfitting

Now we will have a look at the noise and advance files, these are different from the basic graphs as the points noise (randomness in data) and as a result we do not have a negligible residual error. From the plots we can see that the model is predicting the training data points highly accurately as the reconstructed signal follows the data points. However, it chooses the polynomial functions when the data points clearly show a linear relationship. This is because the polynomial function results in the smallest residual error.

This is evidence of overfitting in our model. This is a problem as the model won't generalize well from our training data to unseen data. We have overfitting here unlike in the basic plots as these signals have more noise, and our model's parameters are memorising this noise so when it makes predictions on new data, instead of making a more general prediction, it is making predictions based on the noise from the training data. For example, choosing the Polynomial function to decrease error further even though it clearly shows a linear relationship.

As a result, it will be less accurate overall at predicting new data, as it won't replicate noise. We don't care about performance on training data as we already know the values. Therefore, to perform better I included cross validation so that I can generalize the new data.

# Cross Validation



**How cross validation can fix overfitting**

Cross Validation uses the training data to generate “unseen” data by splitting into two set, training and test data, so that the model uses the training set to calculate parameters and we test those parameters by looking at the residual error on the test set. Before, our model was always choosing the Polynomial function when there was linear relationship as it could accurately predict the noise of the training data but by using cross validation, when the model chooses the Polynomial function on half of the dataset, it tests it parameters on the test dataset and as it does not have the same noise, there will be a greater residual error than if it were to choose a general linear function. This can prevent overfitting as if the more complex functions are unable to accurately predict the “unseen” data, it will choose a simpler function as this is a more general representation of the data.

My cross validation process is that I paired the x and y values together in a list of tuples, shuffled the pairs so that the train-test data points are always randomised. After they’re shuffled, we split the data in half (training and test data). Then use LSR on train data, to reconstruct the signal which we use to find residual error from the test data. I repeated this process 40 times and summed the errors for each respective function to see which function performed the best on ”unseen” data.

**Lowest Error $\neq$ Most Optimal all the time**

By including cross-validation in the model we can see the error values and functions chosen changed. The Linear function is now being chosen when there is an obvious linear line segment in the signals that have noise. We can see that cross-validation was successful in preventing overfitting. However, the residual error increased when compared to the overfitted model as it was also accurately predicting noise of the train data. But the error only went up marginally, so it is still reconstructing the signal intelligently and we now have a model that will perform better on new data. This is evidence the fitted line with the smallest error might not always be the most optimal representation of the data. Overall, the model is successful in reconstructing the signal with minimal error and generalises the data well.