**Aims & Objectives**

**-Aims**

This report mainly focuses on using Least Square Fitting method, to evaluate a prediction model for the power consumption relation between current and resistance. The whole report contains three parts, including the introduction part telling how the program and algorithm is implemented, results part where the fitting results is compared and discussion part for model evaluation.

**-Data Initialization**

First of all, the datasets are initialized by 6 noise levels including No noise, 10 standard deviation level, 20 standard deviation level until 50 standard deviation level. To minimize the error produced in this random noise generating process, the whole data is iterated and initialized 10 times and averaged for each noise level.

**-Fitting Noisy Data**

More specifically, the clean data generated using exact accurate and precise values are the preferred values, which can only be obtained in ideal cases. The generated noisy data represents the results interfered by random errors during the measuring process.

The whole difference can be determined as following.

Eqn.1

is the measured value, is the actual value and is the occurred random error with different standard deviation under “Gaussian Distribution”. This equation tells that, the random error is unavoidable, which means the regression or fitting polynomial model should be fitted and evaluated using noisy data , rather than , as the actual value is only an ideal assumption, which cannot be obtained in real world.

**-Finding Best Fitting Model**

Therefore, all these prediction models will be evaluated using noisy data, with difference random error levels. The fitting and evaluation process will tell how the model performs under the effect of random error. The best model should be the one that fitting closest to measured values under all noisy levels.

**-Program Implementation**

Typically, the prediction models will be computed using two different methods. The one is polynomial interpolation fitting method, by a MATLAB function polyfit(). Another method is using least square fitting, where solving the coefficients from raw datasets directly by linear matrix system using mldivide() function of MATLAB.

The prediction model is defined as follows

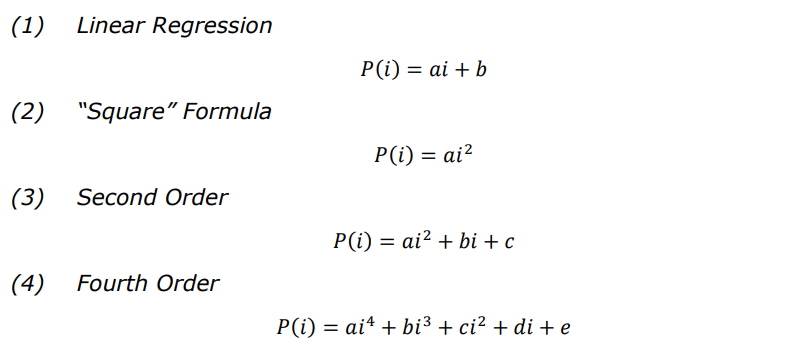


Figure 1. Prediction Models

The current data I is varying from 0 to 2, with 0.1 as an interval, total 21 points. For the power data, it is initialized as clean data by

Eqn.2

The resistance value is set to 100 Ohm. With 21 current values, the clean power data is also generated as 21 values. However, to simulate the real random error effect, the power data is mixed with 6 noise levels, whose standard deviation varies from 0 to 50, with 10 as an interval. At the same time, to minimize the random error effect of this error generation process, the error generation will be repeated by 10 times. The power data used for model fitting is the 10 times averaged output. The whole process can be represented as follows.

Eqn.3

Eqn.4

Eqn.5

Overall, the is an 6x21x10 matrix, in which the elements of a certain row represents the noise generated under a specific standard deviation condition in a particular iteration using Gaussian Distribution. The final fitting data is a 6x21 matrix, each row is the averaged output from previous 10 iteration corresponding noisy data.

Accordingly, the coefficients for each model, is initialized as a 6 x N matrix, where N is the number of coefficients in each model equation, 2 terms in linear model, 3 terms in second order model and 5 terms in fourth order model.

It should be noted that, the “Square Formula” is a special case of second order model, where the coefficients of first order term and constant number are all zero. Therefore, for polyfit() function, the square formula cannot be applied, as the function can only interpolate general second order equation, where the coefficients for the rest two terms are not zero. In contrast, for mldivide(), where elements of linear matrix system is user pre-defined, the square formula can be computed.

After all these complete models have been computed, the model predicted values using the same varying current will be compared with those pre-set pure clean data. At the same time, the fitting performance will also be compared for different noise level data, verifying the performance of modelling equations under the effect of noise.

**Results**

In this part, result of these two fitting methods will be shown, and the estimation error of each model will be calculated. At the end, a comparison between each fitting model will be presented, showing that how these two methods differ and how these differences affect the fitting resultant values.

**Estimation Error**

The estimation error is defined as following,

Eqn.6

Where the {} are the actual power values, } are the evaluated estimation power values and the N is 21.

For the actual power values, which means the data generated randomly. In other words, the actual power value used are the noisy data, which representing the data we can obtained in real measurement operation. As stated before, in the real measuring operation, the data cannot be exactly the same as the generated clean data, which can only be obtained in an ideal case. Therefore, the generated noisy data are treated as reference values, or actual power values. The errors of each method are shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Noise Fitting  Level Model | Linear | Square | Second | Fourth |
| Sd = 0 | 2.24E+04 | 1.33E-26 | 1.59E-26 | 1.33E-25 |
| Sd = 10 | 2.39E+04 | 2.85E+02 | 2.68E+02 | 2.30E+02 |
| Sd = 20 | 2.59E+04 | 1.14E+03 | 1.07E+03 | 9.19E+02 |
| Sd = 30 | 2.86E+04 | 2.56E+03 | 2.41E+03 | 2.07E+03 |
| Sd = 40 | 3.17E+04 | 4.56E+03 | 4.28E+03 | 3.68E+03 |
| Sd = 50 | 3.55E+04 | 7.12E+03 | 6.69E+03 | 5.75E+03 |

Table.1 Estimation Error of mldivide()

|  |  |  |  |
| --- | --- | --- | --- |
| Noise Fitting  Level Model | Linear | Second | Fourth |
| Sd = 0 | 2.24E+04 | 6.02E-26 | 2.35E-26 |
| Sd = 10 | 2.22E+04 | 2.03E+02 | 1.78E+02 |
| Sd = 20 | 2.23E+04 | 8.11E+02 | 7.12E+02 |
| Sd = 30 | 2.28E+04 | 1.82E+03 | 1.60E+03 |
| Sd = 40 | 2.38E+04 | 3.24E+03 | 2.85E+03 |
| Sd = 50 | 2.51E+04 | 5.07E+03 | 4.45E+03 |

Table.2 Estimation Error of polyfit()

From the perspective of error magnitude, it is clear that, the linear model results the largest error magnitude. Comparing with the rest three models, the linear regression method is clearly not the best fitting model. For the rest three models, the errors are quite close lying between e+2 to e+3 magnitude interval. It also clear that, all of these three models have the lowest error fitting the clean data, approximated to zero within the magnitude of e-26.

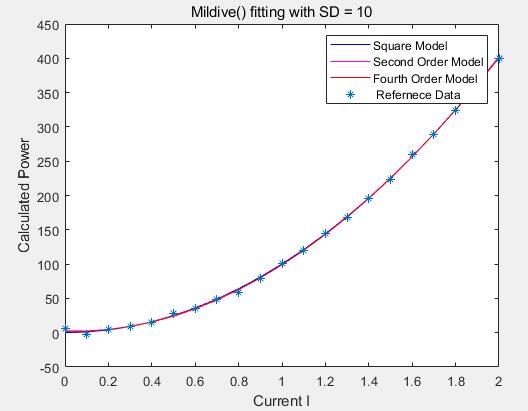
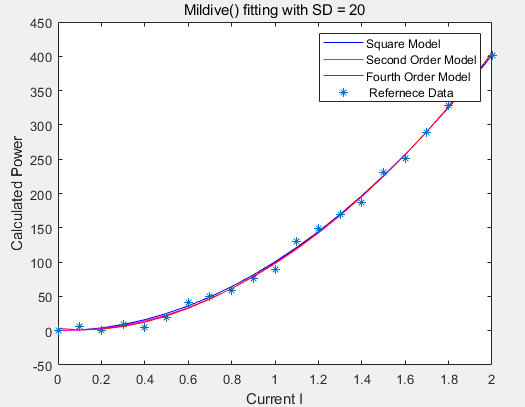
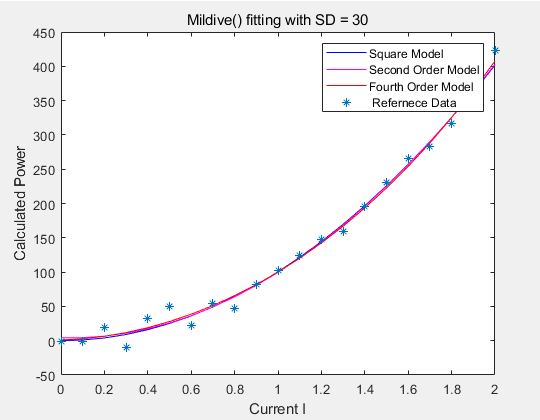
To find out how these three models fit the given data sets, the evaluated curve of each model should be compared together.

**Fitting Comparison**

**-Mldived() fitting**

As stated before, the linear model has the largest error magnitude indicating it cannot be the best fitting choice. Thus, the fitting comparison is only applied within the rest three models.

The fitting results are shown below. The first figure sets demonstrate how the mldive() fits the data under different noise level. SD indicates the standard deviation of random error.

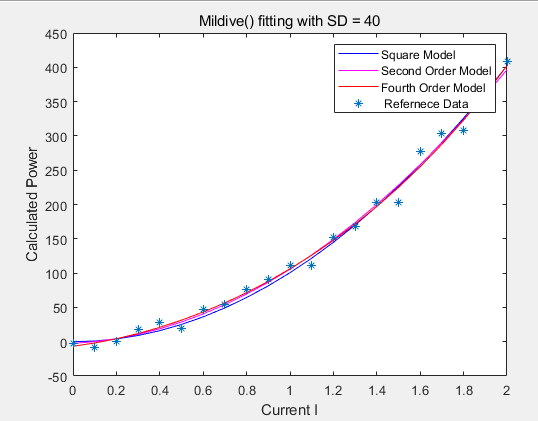
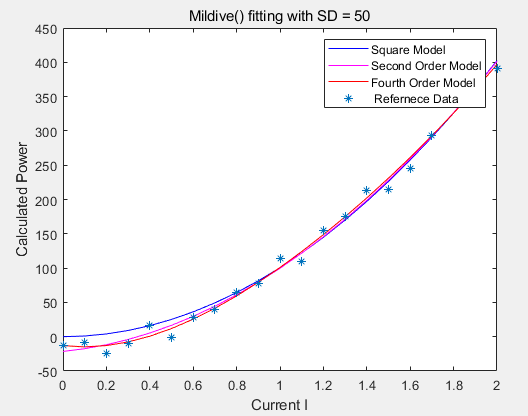
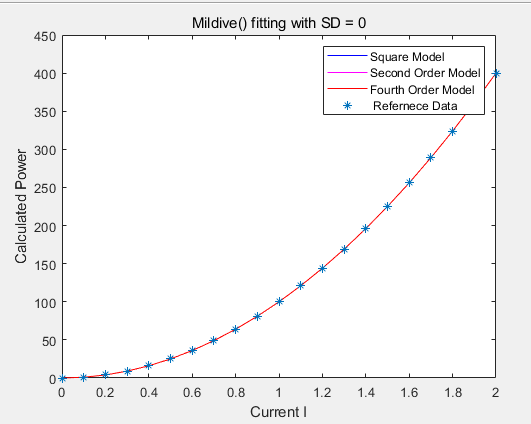
  

Figure set 1. Mldive() fitting plot

It is clear in this figure set that; the main idea of fitting is set the curve to cross data points as more as possible. With the error standard deviation increases, the data become more dispersed and the number of points that each curve crossed drops. More specifically, when no noise attached, each model can fit the reference curve perfectly with few unnoticeable and affectless errors. Compared with the maximum 50 SD figure, where the curve models are get dispersed rather than superposed situation in lower SD figures including SD=0, SD=10, SD=20 and SD=30.

The trend is that higher noise level imposes the fitting models to disperse more and tell their own model equation characteristics by these fitting curves. Focusing on the curve plots, when SD has reached 40, the second order and fourth order model starting points are not zero any more compared with previous SD levels. Only the square model always starts at zero, as the first order and constant terms are all zeros in square model. This difference is now amplified to noticeable by the increased noise dispersion.

Now, zooming into the curve equation of each model, which may tell how the estimated data varies. The following table shows the model equations of each noise level, using the poly2sym() function in MATLAB.

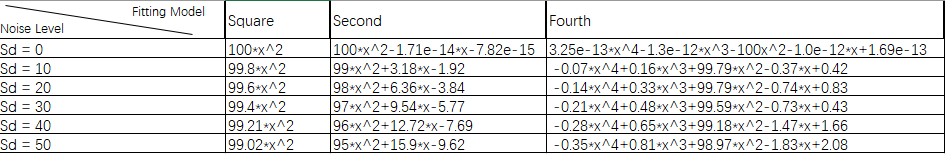


Table3. Curve Equations of mldivie() Model under different Noise Level

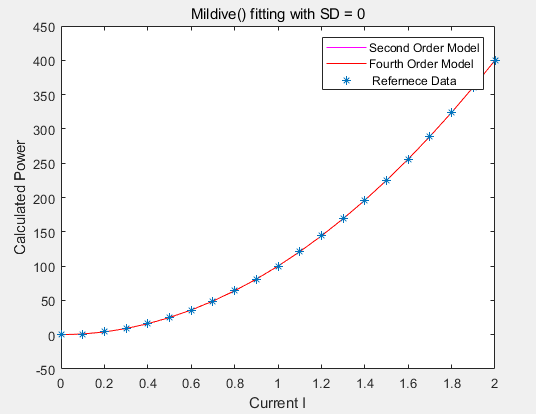
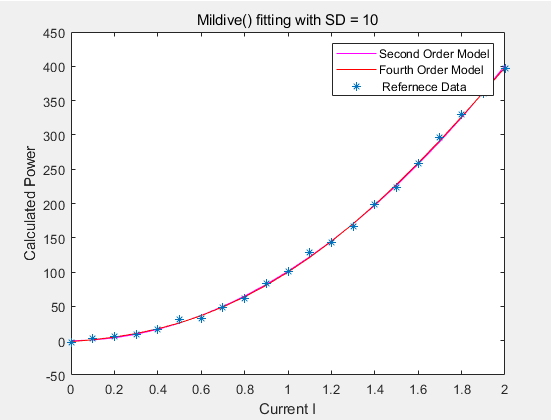
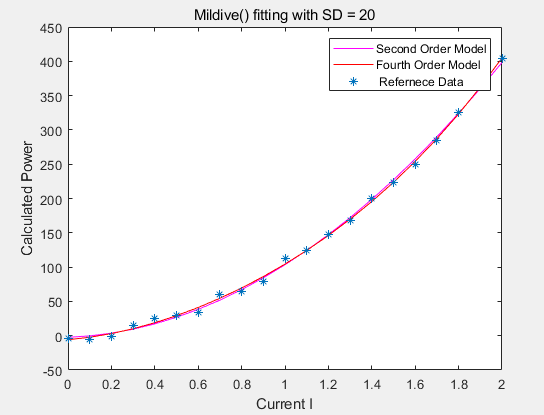
For the clean data row, sd=0, all the models except linear model are approximated to start at original point (0,0). The coefficients of fourth, third order and first order terms are approaching to zero, to reduce the contribution of higher order terms. Comparing these equations together, it is clear that the most significant term is the second order term. Noticing that, the second order term contribute the most calculated power, which means the second order terms are of leading domain when fitting the noisy data. This phenomenon is extremely obvious when clean data is applied to these models, where the values of other terms can be approximated to zero.

On the other hand, with noise level increasing, the weight or coefficients of first order and constant number increases. This phenomenon indicates that, as the noise mixed in the starting points and end points are no not consistent as clean data, even the shape of curve is now less likely to be a squared wave but with more inflexions. The increasing weights of first term and constant coefficient tries to adjust the fitting curve to pass data points as more as possible, acting like a compensation measure to minimize the mixed errors.

Therefore, the square and second order model may be the suitable ways to fit the noisy data.

**-Polyfit() fitting**

The second figure set demonstrates how the polyfit() method fits the data under different noise levels. As mentioned before, the polyft() method can only return valid coefficients for second order, no purely square model can be obtained. Hence, the comparison will be carried without square model only includes second order and fourth order system.

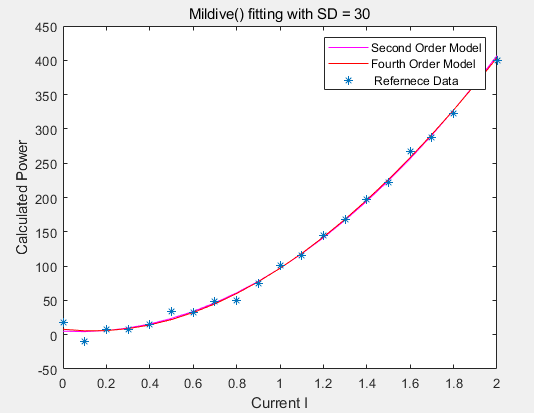
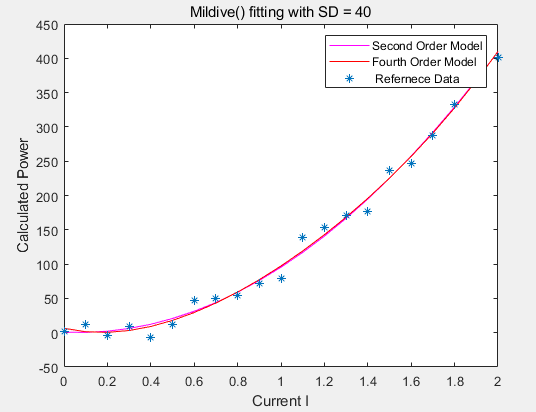
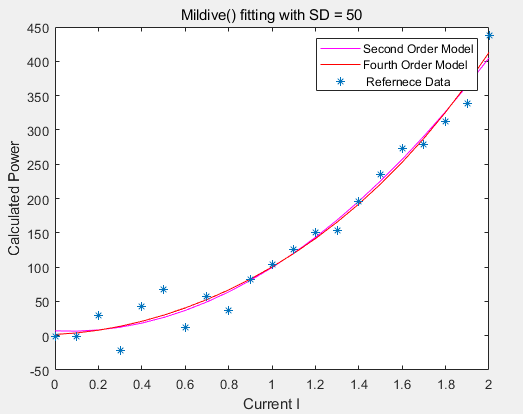
  

Figure set 2. polyfit() fitting plot

From the above plots, the second order curve and fourth order curve fit in with each other closely. With the noise level increasing, the crossing points of each curve drops. Differs from mldivide() method, the fourth and second order curve in polyfit() method do not have much divergences but they may be assumed as a one curve together. In other words, the plots of fitting cannot tell the differences between the second order and fourth order curve as there are almost considered as superposed. Same as mldivide() method, zooming into the curve equations of these two models.

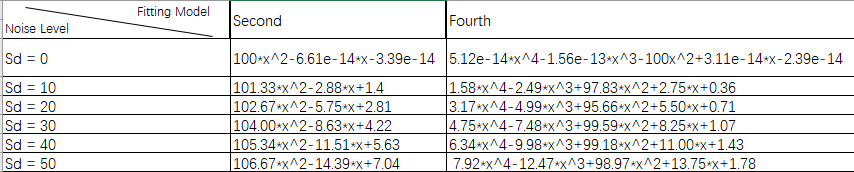


Table4. Curve Equations of polyfit() Model under different Noise Level

Same as mldivide() method, for the clean data, the second and fourth order are approximated as purely square formula. However, the coefficients of fourth order, thirds order and first order becomes larger, rather than 0.001 magnitude level in mldivide() method. Calculate the power contribution of each order, which may give more detailed understanding of the leading order in these fitting models.

The coefficients contribution is defined as follows,

Eqn.7

The n is the order term of , and N is the maximum order of . For example, the fourth order of is , where a is the coefficients of corresponding order term. As the magnitudes of calculated power in each noise level are within the same range, typically up to , choosing SD=20 and SD=50 two sets to calculate contribution factor is representative enough.

**Discussion**