Final Exam: Interpolation and Model Fitting
Math 6310

Kennedy Dorsey

05/06/2020

Introduction

Using interpolation and other model fitting techniques, models were trained on the A-Data set, and final models were tested using the B-Data set. The best model chosen had a reasonably low training error and test error in relation to the other models, and the reasoning behind each model choice is explained. The data sets given were

3.1 $y = f(x)+$ noise

A - Data										
x	0.4313	7.223	1.088	9.032	3.848	6.835	2.588	2.13	6.601	8.981
y	0.7732	92.35	3.455	136.5	31.48	88.1	15.93	11.77	78.3	130.5

B - Data					
x	5.023	1.684	8.65	3.278	5.077
y	49.67	8.059	124.8	23.33	53.54

3.2 $z = f(x, y)+$ noise

A - Data									
x	1.159	1.601	1.693	0.4127	0.9846	0.5608	2.929	0.5614	2.296
y	0.8187	4.969	4.437	3.948	1.244	4.968	1.018	2.667	3.645
z	6.486	63.05	51.88	34.21	7.592	54.03	13.29	17.03	40.84

B - Data						
x	4.063	0.8043	2.339	4.976	1.926	3.602
y	3.14	4.268	4.27	2.809	2.946	4.621
z	45.57	43.21	53.93	44.34	29.72	70.19

3.3 $(x, y) = \vec{f}(t) + \text{noise}$, or $\begin{cases} x = f_1(t) + \text{ noise} \\ y = f_2(t) + \text{ noise} \end{cases}$ which is a parametric curve.

A - Data									
t	1.0	3.0	5.0	7.0	9.0	11.0	13.0	15.0	17.0
x	0.1168	0.3085	0.4041	0.792	0.9742	0.88	1.409	1.695	1.373
y	0.01081	-0.0594	0.02837	0.1055	-0.164	0.0009737	0.2359	-0.2279	-0.09356

B - Data								
t	2.0	4.0	6.0	8.0	10.0	12.0	14.0	16.0
x	0.2364	0.3395	0.5665	0.9583	0.8912	1.071	1.677	1.508
y	-0.01665	-0.05229	0.1152	-0.02328	-0.1678	0.2025	0.03829	-0.3065

3.4 $w = f(x, y, z)+$ noise

A - Data										
x	1.86	1.165	1.483	1.921	1.429	1.057	1.925	1.658	1.133	1.533
y	1.558	1.916	1.633	1.283	1.365	1.092	1.373	1.347	1.705	1.649
z	1.026	1.446	1.239	1.954	1.908	1.862	1.092	1.977	1.412	1.458
w	2.475	4.045	3.324	1.745	2.779	2.654	2.709	1.369	3.878	2.865

B - Data					
x	1.899	1.248	1.03	1.072	1.874
y	1.041	1.212	1.001	1.139	1.212
z	1.526	1.294	1.972	1.181	1.303
w	2.154	4.014	2.244	3.474	2.298

Analysis

3.1 Since the sample of data points given contains noise, the actually data points of $y = f(x)$ is not known, and therefore, it is better to create a model that does not fit the data points exactly.

Starting with the first data set, since we are only given 10 data points from the equation $y = f(x)$ + noise and have no knowledge of the underlying distribution of $f(x)$, it helps to plot the points on a graph. Below, you can see the figure.

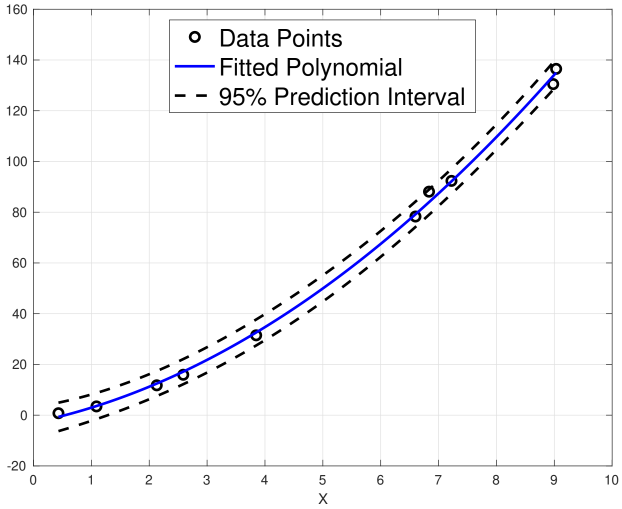

Figure 1: A - Data Set for $y = f(x) + noise$ Plotted with 95% Confidence Interval

From just looking at the points plotted, the data does not seem to be linear. The points seem to curve upwards, which leads to the idea of fitting the data with a polynomial of degree greater than 1. Since there are 10 data points, a polynomial of degree 9 would go through each data point exactly, however, this would produce a model that overfits the training data, which means it would lead to a higher error on the test data since the training data contains noise. That is why the polynomial of degree 2 is fitted to the data points using MATLAB function *polyfit*. It fits the data well, and all of the points are within the 95% confidence interval. Going with a low degree polynomial removes the negatives that come with fitting higher degree polynomials and it fits the data given the noise.

Polynomials of degree 1 to 9 were fitted to the data and the mean squared error was computed for each polynomial. The errors computed are given below.

Degree	1	2	3	4	5	6	7	8	9
MSE	61.3420	3.5755	2.5588	2.5560	2.5349	2.2995	1.1625	1.0069	0.0000

3.2 The second A - Data Set with underlying function $z = f(x, y) + noise$ is fit with a model. The plotted points are shown below. From the figure and the table, as x increases, y increases, then decreases, then increases and decreases again. The output, z, seems to have a linear relationship with y, as it increases when y increases and decreases when y decreases.

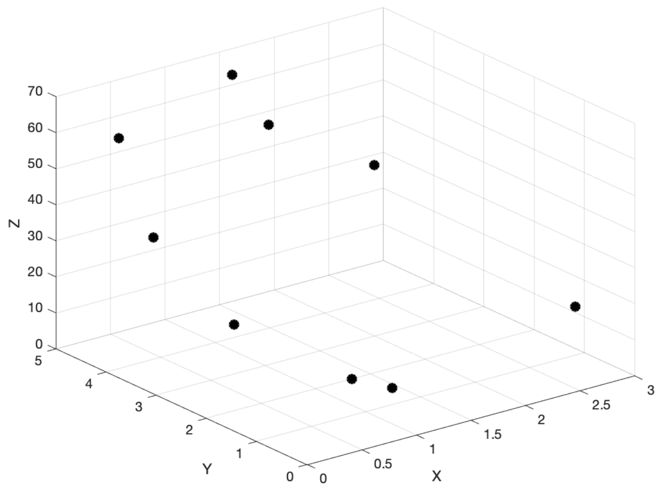

Figure 2: A - Data Set for $z = f(x, y) + noise$ Plotted

The next figure shows the result of piecewise cubic interpolation on the data points using the MATLAB *meshgrid* and *griddata* functions. As with the previous data set, the points are not actual points from the function $f(x, y)$, so interpolating each point is not the best approach to approximate f as it fits the data points too closely.

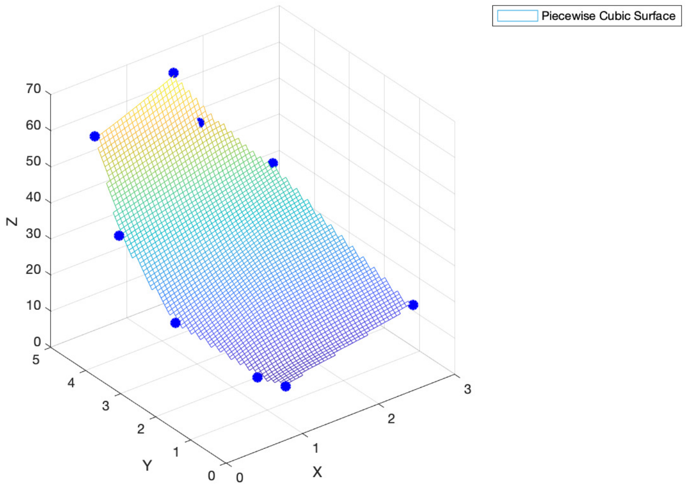

Figure 3: Piecewise Cubic Interpolation for $z = f(x, y) + noise$

The next figure shows a quadratic polynomial fit for the data points that yielded the equation $z = f(x, y) + noise$ as $2.903 + 2.201 * x - 1.924 * y + 1.153 * xy + 2.295 * y^2$ after using the MATLAB Curve Fitting ToolBox. This is a smoother approximation and is more flexible with how closely the data points are approximated.

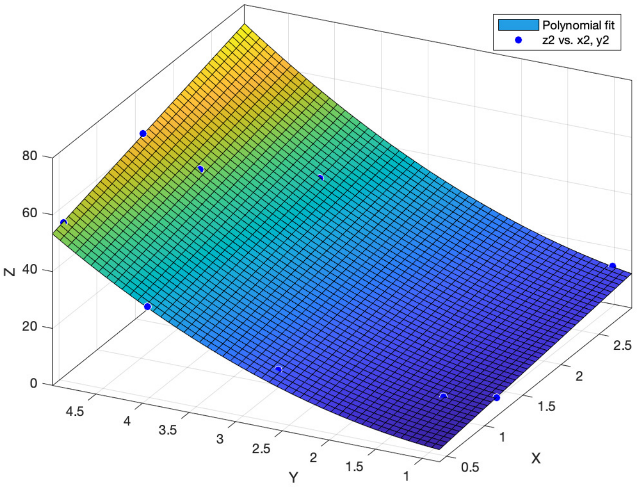

Figure 4: Polynomial Fit for $z = f(x, y) + noise$ as $2.903 + 2.201 * x - 1.924 * y + 1.153 * xy + 2.295 * y^2$

3.3 To create a model fit for the function $(x, y) = \vec{f}(t)$ + noise, or $\begin{cases} x = f_1(t) + \text{ noise} \\ y = f_2(t) + \text{ noise} \end{cases}$, the x and y data points were plotted as a function of t.

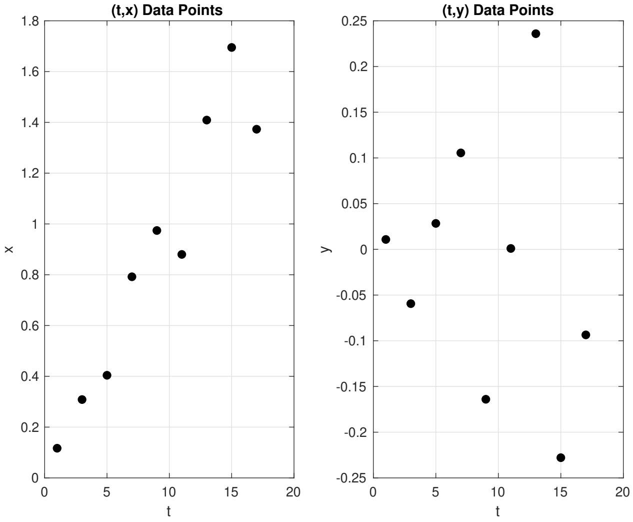

Figure 5: Data Points for $(x, y) = \vec{f}(t) + noise$

When accounting for noise, x as a function of t appears to be linear and there is no noticeable bend in the data points. When fitting a model to x, a polynomial of degree one, two, and three was fit . As expected, fitting a higher degree polynomial did not make the fit significantly better, so it is best to stick with the lowest degree polynomial to prevent overfitting. The degree one polynomial was chosen and the function is $0.09402 * t + 0.03746$.

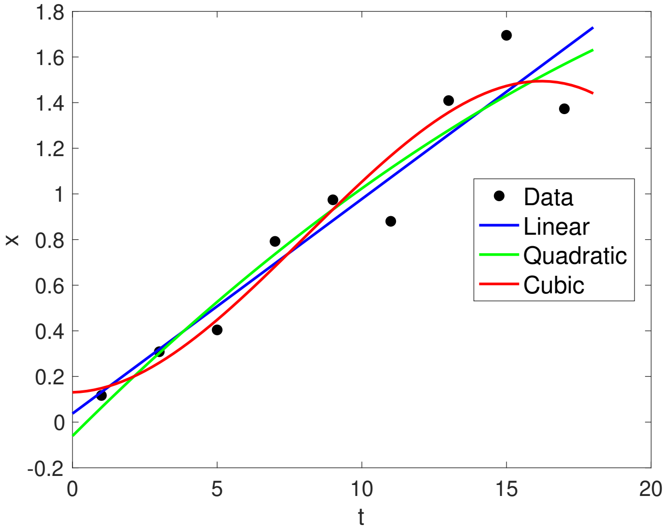

Figure 6: Polynomial Fit for Degree 1,2, and 3 for x as a Function of t

On the other hand, y as a function of t appears to be nonlinear. When fitting the data points, the equations $y = 0.02 * t \cos t$ and $y = 0.175 * \sin(0.9693 * t + 1.966)$ (sum of sine model with one term) were chosen after fitting the data with the right coefficients using $y = 0.02 * t \cos t$ and $y = a0 * \sin(a1 * t + b1) + a2$ as the models. The equation $y = 0.02 * t \cos t$ fits the data extremely well and is simple, which may be help since the data points are noisy and the line chosen for $x = f_1(t)$ has a bit of error between the data points. While the sum of sine model does follow the data points, it seems to stray away too much at points. A Fourier model was also fit to the data points with 3 terms to produce an equation $y = a0 + a1 * cos(t * w) + b1 * sin(t * w) + a2 * cos(2 * t * w) + b2 * sin(2 * t * w) + a3 * cos(3 * t * w) + b3 * sin(3 * t * w)$. Since it fit very closely and the model is very complicated compared to the other two, the simpler model was chosen because the data points include noise. In the end, $y = 0.02 * t \cos t$ was chosen because it follows the overall trend of the data points and is not very complicated to account for overfitting. The sum of sine model with two terms fit almost identically to the $y = 0.02 * t \cos t$ model, so it was not chosen for this reason.

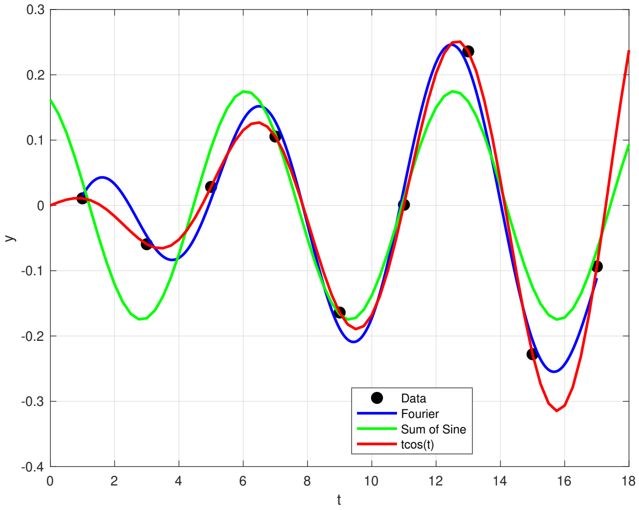

Figure 7: Model Fits for y as a Function of t

3.4 The fourth A - Data Set with underlying function $w = f(x, y, z) + noise$ is fit with a model. The plotted points are shown below.

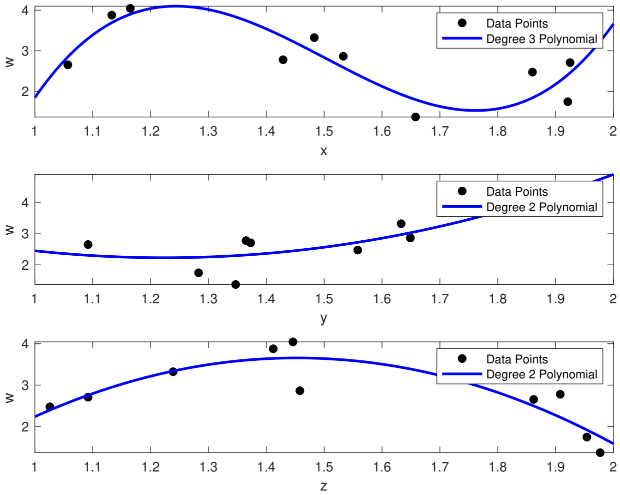

Figure 8: Model Fits for x, y, and z as a function of w for $w = f(x, y, z) + noise$

When plotting each independent variable against the dependent variable, it seems that they all were able to be approximated nicely by polynomial functions. A polynomial of degree 3 fit x as a function of w, and polynomials of degree 2 fit y and z as a function w using the same ideas that were discussed before. Because of this, a linear model seems like a good choice. Using the MATLAB function *fitlm*, a linear regression model was fit to the data. Four models were tested, $w = x + y + z$, $w = x + x^2 + x^3 + y + z$, $w = x + y + y^2 + z$, and $w = x + y + z + z^2$ to if certain variables contributed more significantly to the model. If so, then the model could be simplified. If not, then more terms would be included, which would add more complexity to the model and decrease its generalizability.

To test this, leave one out cross validation was performed on thew A - Data Set to see which model had the lowest mean squared error (MSE) when tested on the part of the training data that was reserved for testing. The model with the lowest MSE was the 2 model, $w\ x + x^2 + x^3 + y + z$. The results are shown in the table below.

Model	1	2	3	4
MSE	0.2728	0.0193	0.8484	0.3394

It is evident that the MSE is improved when using the second model over the others, and is very acceptable. The coefficient of determination and the adjusted coefficient of determination was 0.997 and 0.994. Since the model does not take into account any other terms such as y^2 or z^2, the model is simplified yet still fits the data well. However, the original model is very simple (and therefore, the most generalizable), and also has a low MSE value of 0.363, and 0.874 and 0.811 as the coefficient of determination and the adjusted coefficient of determination, respectively. Since the data contains noise, going with a generalizable, yet significant model is the best choice.

3.5 COVID-19 case numbers in Louisiana from 3/09/2020 to 3/30/2020 are used as the training data set, and the number of cases from 3/31/2020 to 4/09/2020 will be used as the test set. First the number of cases are plotted as a function of the date. The data source can be found in the references.

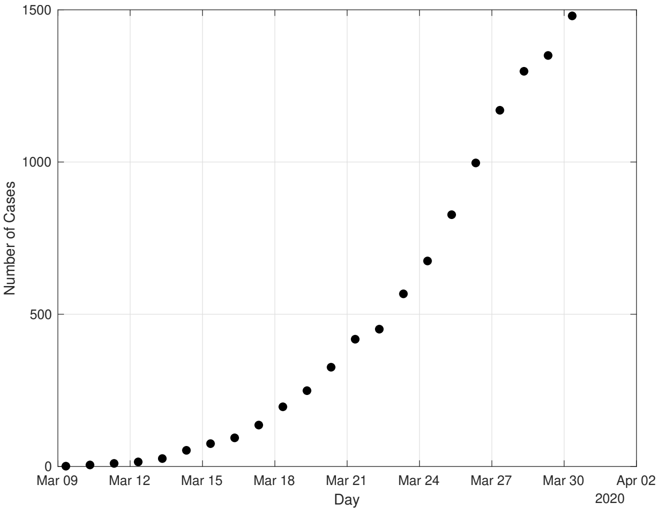

Figure 9: COVID-19 Data Plotted from 3/09/20 to 3/30/20

This data differs from the others as there is knowledge about the underlying function that produced these data points. It is well known now that infection rates tend to increase exponentially when people come into contact with one another, and once a large proportion of the population develops immunity or gets infected, the infection rate begins to plateau. The next figure shows the best fit lines for polynomial of degrees two, three, and four, as well as Gaussian and exponential best fit lines computed in MATLAB.

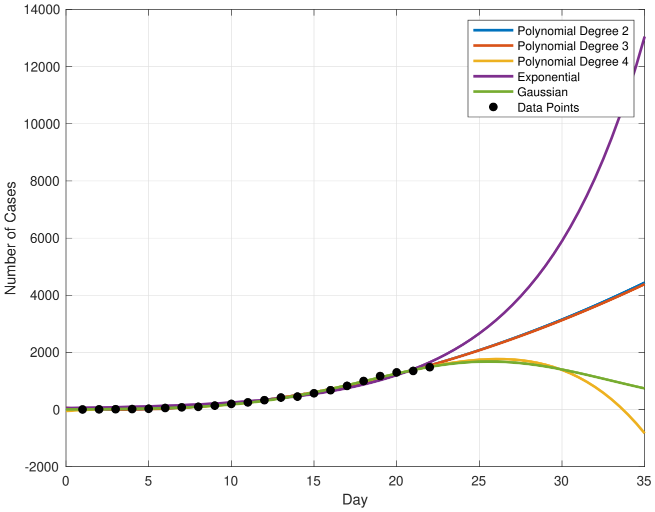

Figure 10: Model Fits for COVID-19 Data

3.5 The function extends a little past the data points to get an idea of what trend the data is expected to follow. The exponential best fit line extends upwards more quickly than any other best fit line. The best fit using these models would be a fit that is somewhat in the middle, since the number of cases are increasing, but maybe not as fast as the exponential best fit line would suggest. However, the exponential best fit line seems to be the most accurate since the others seem to taper off too quickly. Using knowledge of the COVID-19 situation in Louisiana, since March was only the beginning of the pandemic and many were not in self quarantine, it can be expected that the number of cases would not drop off so suddenly, and that is why the exponential line is chosen as the best fit for this data and for the test data.

Results

3.1 For the first function, plotting the B - Data Set is shown in the figure below. The test points line up very closely and the mean squared error between the test points and the best fit line was 1.9051, lower than the training error.

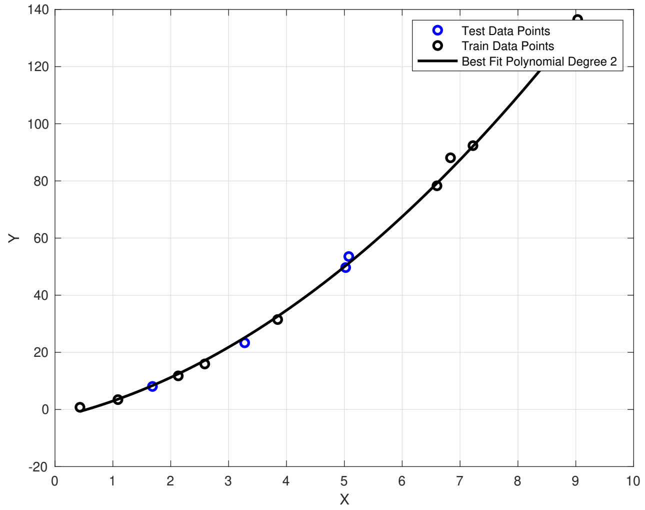

Figure 11: Test Results for 3.1

3.2 For the second function, the B - Data Set was plotted, and is shown in the figure below. The test points follow closely and the MSE computed was 2.2279.

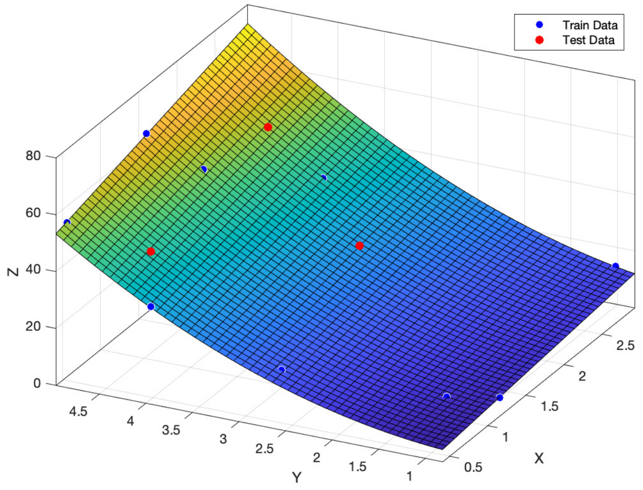

Figure 12: Test Results for 3.2

3.3 For the third function, the B - Data Set was plotted. For this function, the test points do not lie too closely to the graph, but neither do the training data.

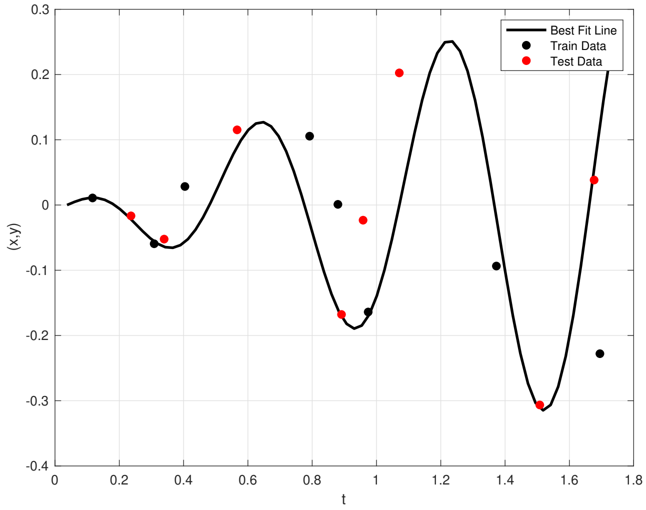

Figure 13: Test Results for 3.3

3.4 For the fourth function, the B - Data was tested using the linear model chosen in the analysis.The MSE, R^2, and adjusted R^2 computed using the test data with the same linear model was 4.2147e-08, 1, and 1, respectively. The linear model was a great fit for the test data.

3.5 The fifth function test points are plotted below. As predicted, the test data increases more rapidly than the Gaussian and polynomial fits, but less rapidly than the exponential after a few days. A better fit would be a model that starts to decline at around day 28.

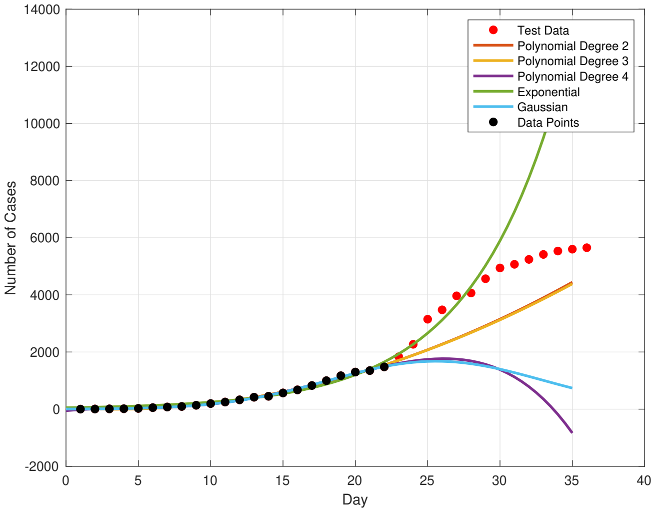

Figure 14: Test Results for 3.5

Conclusions

In conclusion, when fitting noisy data it is better to choose a simple, more general model to avoid overfitting. However, if the amount of noise is unknown, fitting can be tricky, and it is better to stay closer to the data points rather than picking a model that is farther away from the training points. A model was constructed for these five different functions using training data, and then tested using test data. Tools used to fit the models ranged from curve fitting functions, to linear regression fitting functions, to polynomial interpolation functions in MATLAB.

References

1. "COVID-19 Pandemic in Louisiana". WikiPedia. *https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Louisiana.*

2. Uri M. Ascher and Chen Grief. *A First Course in Numerical Methods.*