FYS3155 Machine Learning - Project 1
Studying The Application of Linear Regression in Polynomial Parametrisation of Three-Dimensional Surfaces

Nils Johannes Mikkelsen
(Dated: October 8, 2018)

Three-dimensional surfaces are parametrised using polynomial models with coefficients determined by regressional analysis. Three regression methods are used, Ordinary Least Squares, Ridge and Lasso. No particular favourite is found, but the different methods have strengths and weaknesses depending on their application.

All material for project 1 may be found at:
https://github.com/njmikkelsen/machinelearning2018/tree/master/Project1

I. INTRODUCTION

To count the number of fields in which linear regression has played a pivotal role in the field's research methods is a fool's errand. In fact, the importance of linear regression cannot be understated. The aim of this project is to study one example in which linear regression is applied: namely polynomial parametrisations of three-dimensional surfaces. More concretely, this project aims to study surfaces $f = f(x, y)$ whose function values are only available as data sets $\{(x_i, y_i, f_i)\}$, $i = 1, \ldots, N$. The ideal is to be able to analyse the underlying map $f(x, y)$ by parametrising it with a polynomial map $z = z(x, y)$, whose coefficients are to be determined using linear regression.

In other words, the goal of this project is to study the behaviour of linear regression when it is used in conjuction with polynomial parametrisations of surfaces. In order to achieve this, the project looks at two different cases: the *well-studied* Franke's test function and a dataset from https://earthexplorer.usgs.gov/. The former case serves as an analysis of the applicability of the various linear regression methods, while the latter is included for the purpose of studying a real-world example surface.

II. LINEAR REGRESSION

A. The fundamentals

The basic idea behind linear regression is to fit a function value $y = y(\mathbf{x})$ to some model $f(\mathbf{x})$. Mathematically speaking, $f(\mathbf{x})$ maps the n-dimensional input value $\mathbf{x}$ to the single-valued estimate $f(\mathbf{x})$ of the function value y. The map $f(\mathbf{x})$ may itself be composed of a several submaps $f_j(\mathbf{x})$. The only requirement is that the main map should be a linear combination of the submaps with coefficients β_j, called the model parameters:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i = \beta_1 f_1(\mathbf{x}_i) + \cdots + \beta_M f_M(\mathbf{x}_i) + \varepsilon_i \quad (1)$$

The added error-term ε_i makes the equation an equality. The fundamental assumption of linear regression is that this error-term behaves according to a normal distribution $N(\mu, \sigma^2)$. Linear regression is concerned with the estimation of the model parameters such that the model $f(\mathbf{x})$ is "as close as possible" to y. This vague closeness-terminology is the actually the basis for the existence of different kinds of linear regression methods. This project employs three different kinds of linear regression methods: Ordinary Least Squares (OLS), Ridge and Lasso.

Mathematically speaking, "how close" $f(\mathbf{x})$ is to y is measured using a statistical measure. OLS defines closenes via the Mean Squared Error:

$$\text{MSE}(y, \mathbf{x}, f) = \sum_{i=1}^{N} \big(y_i - f(\mathbf{x}_i)\big)^2 \quad (2)$$

In addition to OLS's MSE measure, Ridge and Lasso includes an additional penalty term:

$$\text{penalty}(\lambda, p, \beta) = \lambda \sum_{j=0}^{M} |\beta_j|^p \quad (3)$$

The difference between Ridge and Lasso lies in the value of p: Ridge defines $p = 2$, while Lasso defines $p = 1$.

B. Matrix formulation

We now introduce the vector quantities

$$\mathbf{y} = \begin{pmatrix} y_1 & \cdots & y_N \end{pmatrix}^T \quad (4a)$$

$$\mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}^T \quad (4b)$$

$$\hat{\beta} = \begin{pmatrix} \beta_1 & \cdots & \beta_M \end{pmatrix}^T \quad (4c)$$

$$\varepsilon = \begin{pmatrix} \varepsilon_1 & \cdots & \varepsilon_N \end{pmatrix}^T \quad (4d)$$

Combining equation (1) for each $i \in \{1, \ldots, N\}$ into a single column vector we find

$$\begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} \beta_1 f_1(\mathbf{x}_1) + \cdots + \beta_M f_M(\mathbf{x}_1) \\ \vdots \\ \beta_1 f_1(\mathbf{x}_N) + \cdots + \beta_M f_M(\mathbf{x}_N) \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}$$

$$\begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{x}_1) & \cdots & f_M(\mathbf{x}_1) \\ \vdots & & \vdots \\ f_1(\mathbf{x}_N) & \cdots & f_M(\mathbf{x}_N) \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_M \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}$$

$$\mathbf{y} = \hat{X}\hat{\beta} + \varepsilon$$

Using this matrix formulation of linear regression, the matrix $\hat{X}$ is commonly refered to as the design matrix of a given model $f(\mathbf{x})$.

The next step is to rewrite the statistical measures using the same matrix formulation.

$$\text{MSE}(y, \mathbf{x}, f) = \left|\mathbf{y} - \hat{X}\hat{\beta}\right|^2 \tag{5}$$

$$\text{penalty}(\lambda, p, \hat{\beta}) = \lambda\left|\hat{\beta}\right|^p \tag{6}$$

C. Closed-form solutions

In the case of OLS and Ridge, there exists closed form solutions to the minimisation problem. As OLS is a special case of Ridge with $\lambda = 0$, we are going to tackle Ridge first. The function to minimise is the so-called loss function:

$$\text{Loss}(\mathbf{y}, \hat{X}, \hat{\beta}, \lambda) = \left|\mathbf{y} - \hat{X}\hat{\beta}\right|^2 + \lambda\left|\hat{\beta}\right|^p \tag{7}$$

First, the ℓ^2 norm of a vector $\mathbf{v}$ may be written as $\mathbf{v}^T\mathbf{v}$. Second, in order to minimise (7) we need to find the zero of the first derivative of the loss function:

$$\begin{aligned} \frac{\partial}{\partial\hat{\beta}}\text{Loss} &= \frac{\partial}{\partial\hat{\beta}}\Big[\big(\mathbf{y} - \hat{X}\hat{\beta}\big)^T\big(\mathbf{y} - \hat{X}\hat{\beta}\big) + \lambda\hat{\beta}^T\hat{\beta}\Big] \\ 0 &= -2\hat{X}^T\big(\mathbf{y} - \hat{X}\hat{\beta}\big) + 2\lambda\hat{\beta} \\ \hat{X}^T\mathbf{y} &= \big(\hat{X}^T\hat{X} + \lambda\hat{I}\big)\hat{\beta} \end{aligned}$$

which leads to

$$\hat{\beta} = \big(\hat{X}^T\hat{X} + \lambda\hat{I}\big)^{-1}\hat{X}^T\mathbf{y} \tag{8}$$

The OLS solution is now found simply by setting $\lambda = 0$. Equation (8) may be further simplified using a Singular Value Decomposition: $\hat{X} = \hat{U}\hat{D}\hat{V}^T$. Note that

$$\hat{X}^T\hat{X} = \hat{V}\hat{D}^T\hat{U}^T\hat{U}\hat{D}\hat{V}^T = \hat{V}\hat{D}^2\hat{V}^T$$

so that

$$\begin{aligned} \hat{\beta} &= \big(\hat{V}\hat{D}^2\hat{V}^T + \lambda\hat{I}\big)^{-1}\hat{V}\hat{D}^T\hat{U}^T\mathbf{y} \\ &= \big(\hat{V}(\hat{D}^2 + \lambda\hat{I})\hat{V}^T\big)^{-1}\hat{V}\hat{D}^T\hat{U}^T\mathbf{y} \\ &= \hat{V}(\hat{D}^2 + \lambda\hat{I})^{-1}\hat{D}^T\hat{U}^T\mathbf{y} \end{aligned} \tag{9}$$

In the case of OLS, (9) may be further simplified to

$$\hat{\beta}_{\text{OLS}} = \hat{V}\hat{D}^{-1}\hat{U}^T\mathbf{y} \tag{10}$$

D. Lasso solutions

As opposed to OLS and Ridge solutions, there are no closed form expressions for Lasso solutions. Instead, these solutions are calculated using the LARS algorithm via the `scikit learn` Python package. Specifically, the functionality `lars_path`, using the additional parameter `method="lasso"`.

III. METHOD

A. Franke's Function

As Franke's function has been studied numerous times, the focus of this report lies not in determining which polynomial model is most suited for parametrising the surface, but instead studying the behaviour of linear regression with respect to a pre-determined model. The model in question is a 5^{th} degree polynomial, consisting of 21 unique parameters. The entire polynomial (in all its extensive glory) has been included in appendix B for completeness.

Franke's function is essentially a weighted sum of four bivariate gaussian curves, located in the region about the origin. The full function is included in appendix B. The specific region of Franke's function studied in this project is $(x, y) \in [0, 0] \times [1, 1]$.

To simplify notation, $f(x, y)$ is used to denote Franke's function, while $z(x, y)$ is used to denote the polynomial parametrisation of $f(x, y)$.

1. *Without noise*

We begin by analysing a data set with no noise: $f(x, y)$ is evaluated at $N = 400$ coordinates (x_i, y_i) that are randomly drawn from $[0, 0] \times [1, 1]$ with equal probability. Labelling this data set D_0, we have

$$D_0 = \{(x_0, y_0, f_0), \ldots, (x_{N-1}, y_{N-1}, f_{N-1})\}. \tag{11}$$

The main objective is to fit $z(x, y)$ to D_0 using all three regression methods. In the case of Ridge and Lasso, a range of 100 regression penalties between 10^{-9} and $5 \cdot 10^{-2}$ are used. In addition, the Bootstrap algorithm with $K = 200$ iterations is used to estimate the models' biases and variances.

Having run all the curve fitting computations, the different methods are compared with respect to error scores such as MSE, R^2 and $\sum\sigma_\beta^2$, as well as comparison plots of the models' biases and variances. Seeing that this analysis is more focused on the behaviour of the regression methods as opposed to the accuracy of the specific model, all error scores and other comparisons will be viewed with respect to the regression penalties.

2. Introducing noise

As a final addition of complexity, normally distributed noise $N(0, \sigma_\varepsilon^2)$ is added to $f(x, y)$. This leads to the new data set:

$$D_1 = \{(x_0, y_0, f_0 + \varepsilon_0), \dots \\ \dots, (x_{N-1}, y_{N-1}, f_{N-1} + \varepsilon_{N-1})\}. \quad (12)$$

Incuding noise in this analysis is particularly important as all real data features noise. Without some kind of quantifiable information regarding how linear regression reacts to the addition of noise, drawing reasonable conclusions from the polynomial parametrisations would be inappropriate.

Similar analyses as those performed in the previous section is now re-run using three different noisy data sets: $\sigma_\varepsilon = \{10^{-2}, 10^{-1}, 10^{0}\}$.

B. Terrain data

The data set used is called "`SRTM1N59E009V3`" and was downloaded from https://earthexplorer.usgs.gov/ Wednesday, 3$^{\text{th}}$ of October. The file can be found the project's github page. The terrain data is centered about the Norwegian city Moss, just South of Oslo, the capital of Norway. The data features both sea level and some peaking mountains, much like Franke's Function. As the data set contains many data points, some calculations may become too expensive computationally speaking. In such cases the analysis will instead be performed on subsets with every n_x^{th} x coordinate and every n_y^{th} y coordinate.

As a first look at the data, a simple OLS fit is performed with different polynomial of degrees greater or equal to 5. The resulting models will be compared mostly on the basis of their MSE scores. This initial test serves as a quick survey of the the different polynomial models, in particular whether the model is applicable or not.

Having done a simple OLS fit, the next step is to apply the same machinery to the data as with Franke's function, this includes every regression method, using different penalties, and the Bootstrap resampling technique. Model selection will again be based on the error analysis of average MSE, model bias, etc. However, seeing that this data set represents a real surface, additional care will be taken with respect to possible real-world applications of the models. In particular, address whether the model is suitable for predicting the terrain of the region outside of the data.

IV. RESULTS

A. Franke's Function

A surface plot of Franke's function for $(x, y) \in [0, 0] \times [1, 1]$ is included in figure 4 in Appendix A. In addition, a surface plot of the simple OLS parametrisation, figure 5, and a contour plot of the difference between the OLS parametrisation and Franke's function, figure 6, is also included. From a visual standpoint, the parametrisation seems to mimick Franke's function quite well, despite some discrepancies. Most notably, the OLS parametrisation fails to model the flat surface for y values close to 1 and it begins a drastic descent for $x \approx y \approx 0$, while Franke's function actually smoothens out. Both of these inconsistencies are also visible in the contour plot. The MSE and R^2 scores for this parametrisation are $1.485395 \cdot 10^{-3}$ and $9.811110 \cdot 10^{-1}$ respectively. Furthermore, the sum of the model's parameter variances is $\sum \sigma_\beta^2 = 3.865426 \cdot 10^{1}$.

Figure 1 shows how the MSE and R^2 scores depend on the penalty, the OLS curve is added for comparison. While the Lasso increases rapidly with increasing penalty, the Ridge scores manage to remain fairly constant. It is evident that the sudden jumps in the Lasso scores is due to model parameters put to zero. Using the Bootstrap, figure 2 shows how the average MSE and R^2 scores depend on the penalty. The relationship is essentially the same, now however with a more smooth Lasso curve.

Moving on to the models' bias and variance, figure 3 displays estimated biases and variances for different penalties. While under a miniscule fluctuation (due to random sampling in the Bootstrap), the OLS estimates stay remain constant. The Ridge estimates are also well-behaved, meaning the Ridge penalty is not restricted in any particular way. The Lasso estimates on the other hand show strong dependence on the penalty. As the penalty increases, several model parameters are put to zero, which in turn leads to an overall decrease in the model variance. Meanwhile, the Bias is very stable for low penalties, however spikes for penalties above 0.04. It is apparent that the ideal penalty for the Lasso is somewhere between 0.01 and 0.02, where in which the variance is low and the bias is stable.

A second run using greater penalties was done to see whether the tendencies displayed in figures 1, 2 and 3 were inherently local in λ. The resulting figures are included in figures 7, 8 and 9.

B. Terrain Data

A contour plot of the terrain data is included in figure 11 in appendix A. In addition, OLS fits using polynomials of degrees 5, 6, 7, 8, 9 and 10 are incldued in figures 12 through 17. From the figures it is clear that the more

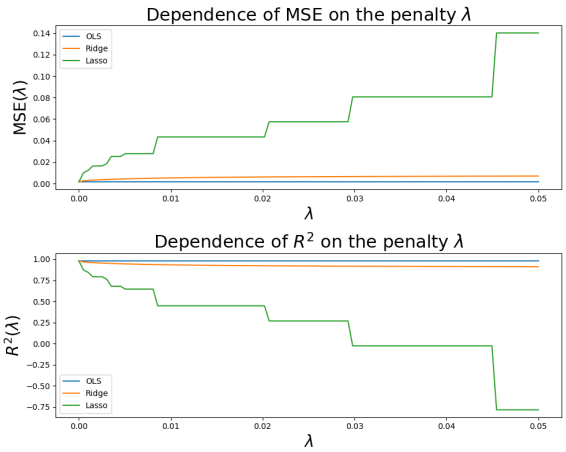

Figure 1. The dependence of MSE and R^2 on the penalty regression parameter λ.

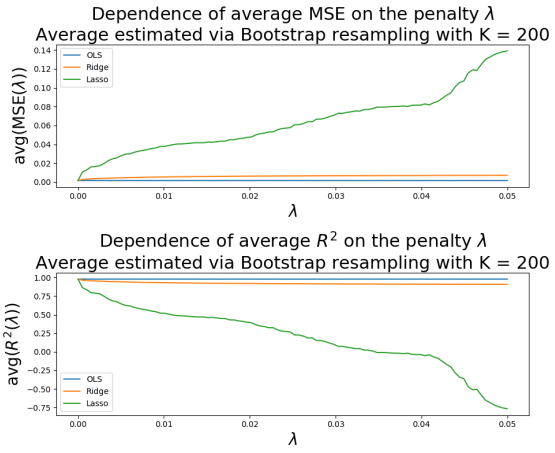

Figure 2. The dependence of average MSE and average R^2 on the penalty regression parameter λ. The averages are estimated using the Bootstrap algorithm with $K = 200$ iterations.

complex models, i.e., those of greater degree, are more realistic in terms of their curvature. The real data features a great deal of noise, which isn't captured by the parametrisations. Moreover, the sea level is heavily violated by most of the models. This is to be expected however as the models do not place any criteria on this.

In the case of the 10^{th} degree parametrisation, the MSE and R^2 scores were $4.47664167 \cdot 10^3$ and $7.39150511 \cdot 10^{-1}$ respectively. The size of the MSE score is most likely due to noise in the data set. As the R^2 score is normalised in comparison with the MSE, it does not show the same ridiculous numbers. Nonetheless, a R^2 of about 0.74 is not very impressive to say the least. The parametrisations are great for capturing the major structures in the data, but fail to reproduce the data.

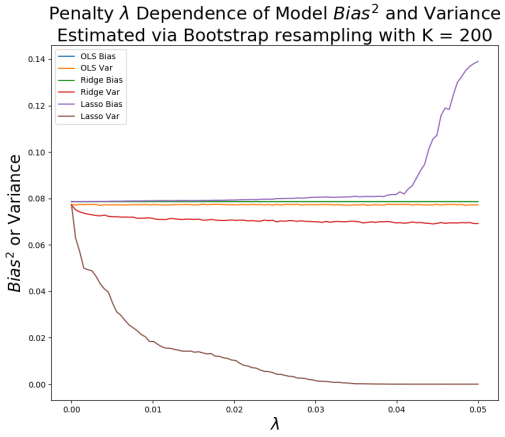

Figure 3. The dependence of estimated model bias^2 and variance on the penalty regression parameter λ. The estimations are made using the Bootstrap algorithm with $K = 200$ iterations.

V. DISCUSSION

The analysis of Franke's function illuminated how stable the penalty dependence of Ridge regression is compared to Lasso regression. This is likely due to Lasso's unique ability of completing freezing out some parameters, while Ridge on the other hand changes slowly with respect to λ.

The 10^{th} degree OLS parametrisation of the terrain data showed that these kinds of methods are very adaptible for describing the large structures and shapes of noisy data such as the terrain data. However, in order to reproduce the terrain, other methods should be used instead.

VI. CONCLUSION

Linear regression proved a fair condidate for parametrising surfaces using polynomial models. While the parametrisations are unable to fully repoduce the underlying surfaces, they do describe their overaching shape. This is useful in cases when detailed maps are uneccessary, and only a major description is needed.

[1] Trevor Hastie et. al. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2017. Second Edition, publisher: Springer.

[2] Morten Hjorth-Jensen. Fys-stk 3155: Machine learning project 1. url: https://compphysics.github.io/ MachineLearning/doc/Projects/2018/Project1/pdf/Project1.pdf.

Appendix A: Plots

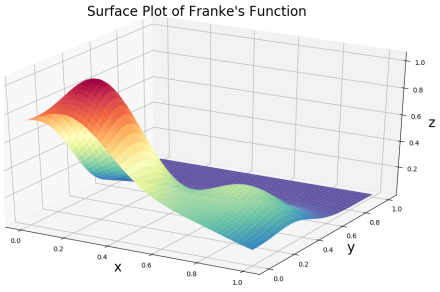

Figure 4. Surface plot of Franke's function for $(x, y) \in [0, 0] \times [1, 1]$.

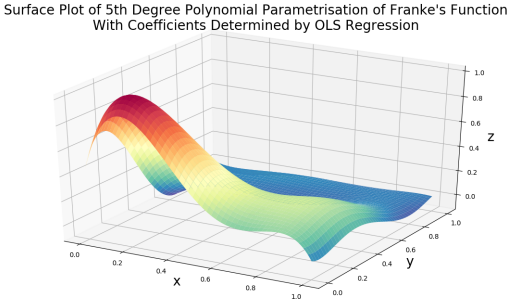

Figure 5. Surface plot of a 5^{th} degree polynomial parametrisation of Franke's function for $(x, y) \in [0, 0] \times [1, 1]$, with term coefficients determined via Ordinary Least Squares regression analysis.

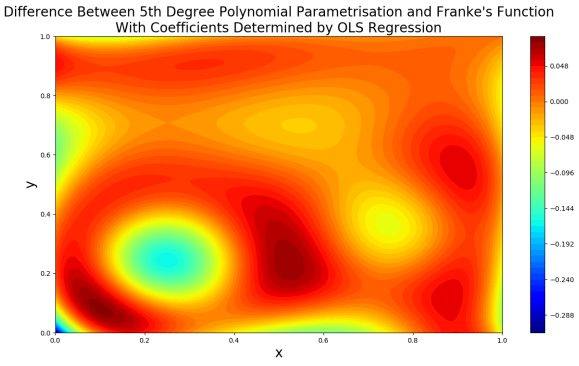

Figure 6. Contour plot of the difference between a 5^{th} degree polynomial parametrisation of Franke's function and Franke's function for $(x, y) \in [0, 0] \times [1, 1]$, with term coefficients determined via Ordinary Least Squares regression analysis.

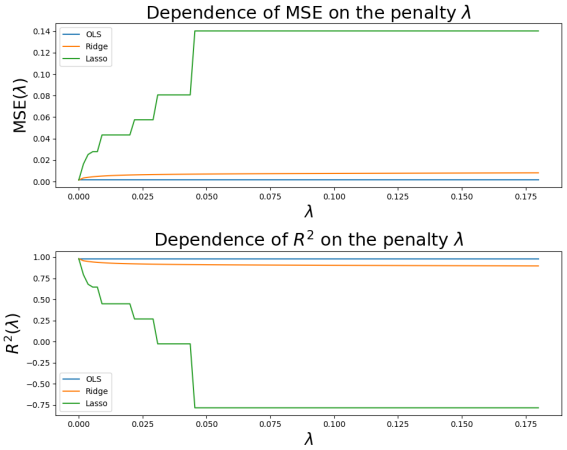

Figure 7. The dependence of MSE and R^2 on the penalty regression parameter λ.

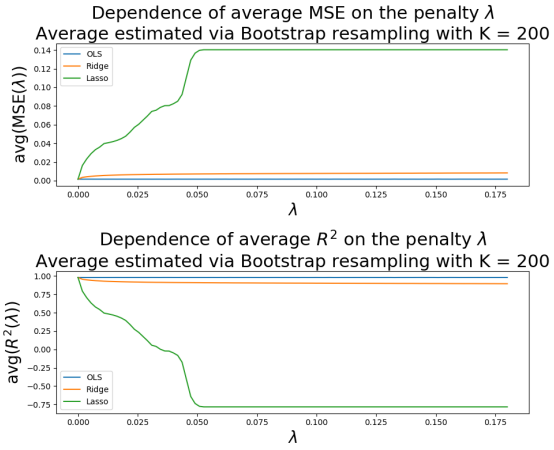

Figure 8. The dependence of average MSE and average R^2 on the penalty regression parameter λ. The averages are estimated using the Bootstrap algorithm with $K = 200$ iterations.

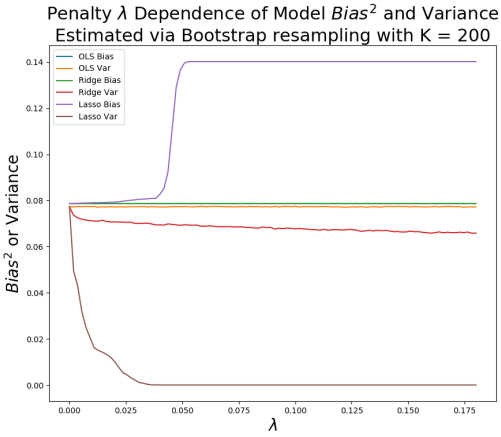

Figure 9. The dependence of estimated model bias^2 and variance on the penalty regression parameter λ. The estimations are made using the Bootstrap algorithm with $K = 200$ iterations.

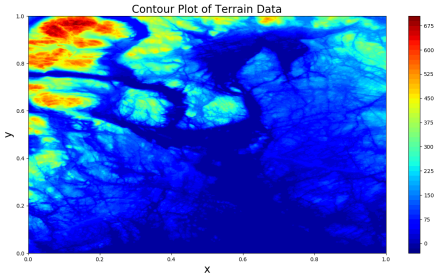

Figure 10. Contour plot of the terrrain data set.

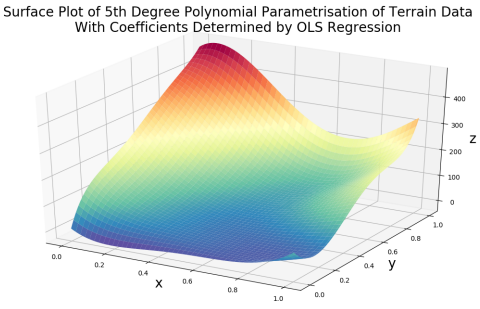

Figure 11. Surface plot of a polynomial parametrisation of the terrrain data set of degree 5.

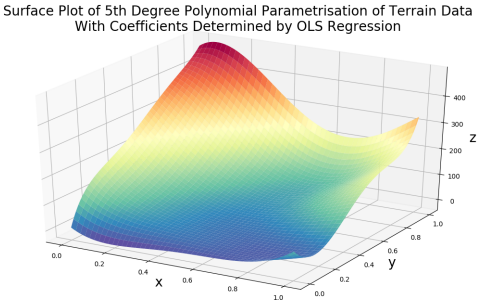

Figure 12. Surface plot of a polynomial parametrisation of the terrrain data set of degree 5.

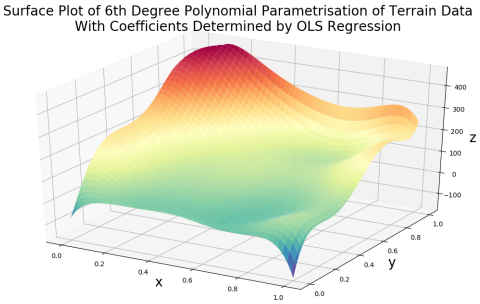

Figure 13. Surface plot of a polynomial parametrisation of the terrrain data set of degree 6.

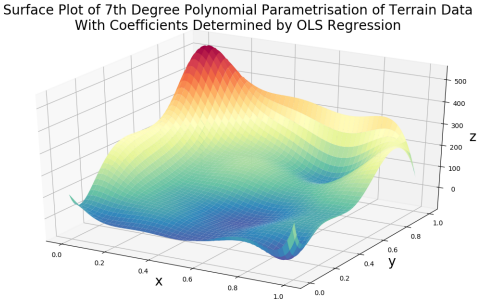

Figure 14. Surface plot of a polynomial parametrisation of the terrrain data set of degree 7.

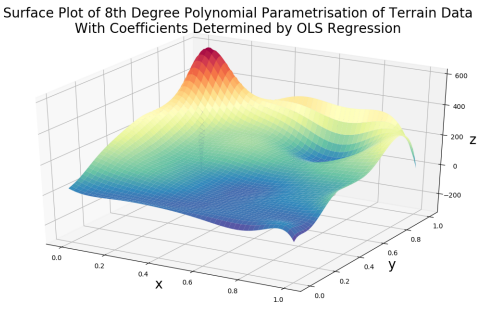

Figure 15. Surface plot of a polynomial parametrisation of the terrrain data set of degree 8.

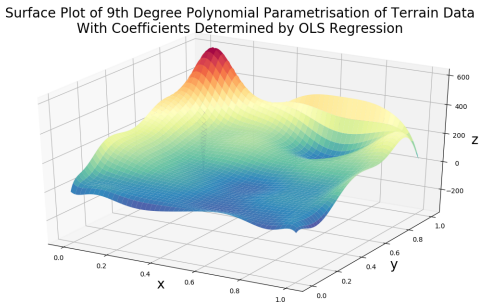

Figure 16. Surface plot of a polynomial parametrisation of the terrrain data set of degree 9.

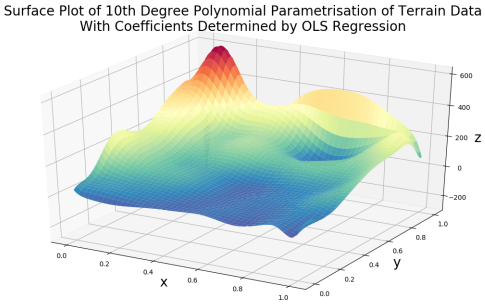

Figure 17. Surface plot of a polynomial parametrisation of the terrrain data set of degree 10.

Appendix B: Equations

Franke's test function is the following weighted sum of four bivariate gaussian curves:

$$f(x,y) = \frac{3}{4}e^{-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}} + \frac{3}{4}e^{-\frac{(9x+1)^2}{49}-\frac{(9y+1)^2}{10}} + \frac{1}{2}e^{-\frac{(9x-7)^2}{4}-\frac{(9y-3)^2}{4}} - \frac{1}{5}e^{-(9x-4)^2-(9y-7)^2} \tag{B1}$$

The explicit expression for the 5^{th} degree polynomial parametrisation used in the analysis of Franke's function is given below. Note that the ordering of the terms is important. To emphasise the difference between the terms, extra vertical spacing is added so that each line consists of same-degree terms (for the sake of clarity).

$$\begin{aligned} z(x,y) = {} & \beta_0 + \\ & \beta_1 x + \beta_2 y + \\ & \beta_3 x^2 + \beta_4 xy + \beta_5 y^2 + \\ & \beta_6 x^3 + \beta_7 x^2 y + \beta_8 x y^2 + \beta_9 y^3 + \\ & \beta_{10} x^4 + \beta_{11} x^3 y + \beta_{12} x^2 y^2 + \beta_{13} x y^3 + \beta_{14} y^4 + \\ & \beta_{15} x^5 + \beta_{16} x^4 y + \beta_{17} x^3 y^2 + \beta_{18} x^2 y^3 + \beta_{19} x y^4 + \beta_{20} y^5 \end{aligned} \tag{B2}$$

Hence, the entire model consists of 21 parameters: $\beta_0, \ldots, \beta_{20}$.