FYS-STK4155 - Project 1

Frida Larsen

This report investigates the parametrisation of surfaces using two dimensional polynomials by implementing three linear regression methods. Resampling methods are used for model analysis. The polynomial approach proved able to reproduce general shapes, but failed to capture irregular details typically found in real terrain data.

I. INTRODUCTION

The aim of this project is to study how well surface data can be parametrised using two dimensional polynomials. The polynomials are determined using the linear regression methods ordinary least squares (OLS), ridge and lasso. We will also explore the use of resampling methods for improved model analysis.

The project consists of two main parts. First, we will attempt to parametrise a known function called the Franke function. Then we will use what we have learned and employ the same tools for predicting terrain data.

All relevant code may be found in the GitHub repository 'FYS-STK4155'[1] under the Project1 folder. This folder also includes a Figures folder, which holds all the figures presented in this text and produced during the project.

II. METHODS

The methods and theory presented here are based on the lecture notes from Morten Hjorth-Jensen [1][2], Hastie et al's book *The Elements of Statistical Learning* [3] and James et al's book *An Introduction to Statistical Learning* [4].

A. Polynomial regression

We begin by assuming that the height of the surface z is a function of the plane coordinates x and y: $z = f(x, y)$. In addition there will be an unavoidable measuring error, ϵ, which satisfies $E[\epsilon] = 0$ and $Var[\epsilon] = \sigma^2$. That is

$$z = f(x, y) + \epsilon. \tag{1}$$

We want to find an nth degree polynomial approximation of $f(x, y)$. For example, the second order approximation would be

$$f(x, y) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 xy + \beta_4 x^2 + \beta_5 y^2, \tag{2}$$

where the βs are regression coefficients to be determined.

In the 0th order approximation, β_0 would simply be the average of z. This is also the case for higher order polynomials. We therefore choose to center z (remove mean value), in which case $\beta_0 = 0$ always. This will simplify our calculations. Furthermore, we will also scale z by its standard deviation in order to avoid numerical issues caused by e.g. outliers. Scaling z will not affect x and y, since we can choose to include the scaling factor in the coefficients. In summary,

$$z \to z = \frac{z - E[z]}{\sqrt{Var[z]}}. \tag{3}$$

We want to determine the βs using a data set of n points (x_i, y_i, z_i). Each z-coordinate can be expressed as

$$\begin{aligned} z_1 &= f(x_1, y_1) + \epsilon \\ z_2 &= f(x_2, y_2) + \epsilon \\ &\vdots \\ z_n &= f(x_n, y_n) + \epsilon. \end{aligned}$$

By expanding the polynomials we may formulate our problem as a matrix equation:

$$\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & \cdots \\ x_2 & y_2 & \cdots \\ \vdots & \vdots & \ddots \\ x_n & y_n & \cdots \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{pmatrix} + \begin{pmatrix} \epsilon \\ \epsilon \\ \vdots \\ \epsilon \end{pmatrix} \tag{4}$$

Or, in a more compact form:

$$\mathbf{z} = X\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{5}$$

In order to avoid numerical errors, we choose to center and scale each column in X. As for the scaling of z, this will not affect the problem since we can absorb any additional factors into the coefficients.

The general approach to determining $\boldsymbol{\beta}$ is to minimise a cost function.

[1] https://github.com/fridalarsen/FYS-STK4155

B. Regression methods

1. Ordinary Least Squares (OLS)

The approach of the OLS method for finding the regression coefficients is to find the βs that minimise the mean squared error (MSE) of the approximation. The MSE is given by

$$\mathrm{MSE}(\mathbf{z}, \tilde{\mathbf{z}}) = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2, \tag{6}$$

where z_i is the true observation and $\tilde{z}_i$ is the approximated value.

The OLS coefficients are given by

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{z}, \tag{7}$$

where X is the design matrix and z contains the data points. Since OLS is a special case of the ridge regression method with penalty $\lambda = 0$, the derivation of the expression for β is included in the ridge regression section that follows.

2. Ridge regression

The approach of the ridge method for finding the regression coefficients is to find the βs that minimise the mean squared error (MSE) of the approximation plus a penalty which is proportional to the square of the coefficient estimate. In other words, the ridge method aims to minimise

$$MSE + L^2\text{-penalty} = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 + \lambda\sum_{j=1}^{p}\beta_j^2. \tag{8}$$

$\lambda \geq 0$ is known as the penalty parameter and must be predetermined.

We want to minimise the above function. We begin by defining

$$C \equiv MSE + L^2\text{-penalty}$$

for simplicity of notation. We then write C as a vector function:

$$C = \frac{1}{n}(\mathbf{z} - X\boldsymbol{\beta})^T(\mathbf{z} - X\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

The function C will reach its minimum when its derivative with respect to β is 0. We have

$$\frac{\partial C}{\partial \boldsymbol{\beta}} = -\frac{1}{n}2X^T(\mathbf{z} - X\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta}$$

Setting this equal to 0 yields

$$\begin{aligned} 0 &= -\frac{1}{n}2X^T(\mathbf{z} - X\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta} \\ \Rightarrow \frac{1}{n}X^T\mathbf{z} &= (\frac{1}{n}X^TX + \lambda I)\boldsymbol{\beta} \\ \Rightarrow \boldsymbol{\beta} &= \frac{1}{n}(\frac{1}{n}X^TX + \lambda I)^{-1}X^T\mathbf{z} \\ &= (X^TX + n\lambda I)^{-1}X^T\mathbf{z} \end{aligned}$$

We rename $n\lambda$ as simply λ and end up with

$$\boldsymbol{\beta} = (X^TX + \lambda I)^{-1}X^T\mathbf{y} \tag{9}$$

We note again that $\lambda = 0$ gives the OLS coefficients.

3. Lasso regression

The approach of the lasso method for finding the regression coefficients is to find the βs that minimise the mean squared error (MSE) of the approximation plus a penalty which is proportional to the coefficient estimate. In other words, the lasso method aims to minimise

$$MSE + L^1\text{-penalty} = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 + \lambda\sum_{j=1}^{p}\beta_j. \tag{10}$$

Unlike the OLS and ridge regression methods, there is no analytical function for the regression coefficients which minimise the lasso cost function. For this project we have chosen to use the sklearn function `LassoLars`[2] [5] to calculate the coefficients.

C. Model analysis

1. Model evaluation

In order to determine the quality of our model, we cannot test it on the same data we used to create the model. It goes without saying that the model will fit the data it was trained on better than unknown data. In order to evaluate our model then, we will need to split our original set of observations into a training and a test set. Throughout this project we have split the data randomly into a training and a test set where approximately 35% of the original data is used as the test set.

In order to quantify the accuracy of the model predictions on the test set we use the MSE and R^2-scores. The MSE is defined in equation 6. The R^2-score is given by

$$\mathrm{R}^2(\mathbf{z}, \tilde{\mathbf{z}}) = 1 - \frac{\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n-1}(z_i - \bar{z})^2}, \tag{11}$$

[2] Scikit-learn LassoLars

where $\bar{z}$ is the average of $\mathbf{z}$ given by

$$\bar{z} = \frac{1}{n}\sum_{i=0}^{n-1} z_i. \tag{12}$$

2. Kfold cross validation

The basic method of splitting the observational data into a training and a testing set described in the previous section is not perfect. The MSE and R^2 calculated do not provide the full picture, since they are computed only for a specific grouping of the data. The main idea of the Kfold cross validation method is to evaluate the model on a variety of training and testing split configurations by dividing the data into K groups. The algorithm is as follows:

Algorithm 1 Kfold cross validation

1: Shuffle the data set and split it into K numbered groups.
2: **for** $k = 1, 2, \ldots, K$ **do**
3: Use group k as the testing set and combine the remaining groups into a training set.
4: Fit a model on the training set, test on the testing set and evaluate MSE and R^2.
5: **end for**
6: Compute the mean and standard deviation of the K MSEs and R^2-scores.

We use the standard deviations as an estimate of the error of the Kfold estimates of MSE and R^2. For this project we have chosen to perform 5fold cross validation.

3. Bias-variance tradeoff

The bias of a model is the systematic difference between the average model prediction and the true value:

$$\text{Bias}(\tilde{z}) = f(x,y) - \mathbb{E}[\tilde{z}]. \tag{13}$$

The variance of a model is the systematic difference between each model prediction and the average of the model predictions:

$$\text{Var}(\tilde{z}) = \mathbb{E}\big[(\tilde{z} - \mathbb{E}[\tilde{z}])^2\big] \tag{14}$$

The variance provides a measure of how the predicted values of the model are spread. A high variance indicates an overfitted model, since it then follows the training data too closely.

Both the bias and the variance will contribute to the overall prediction error (MSE) of the model. We wish to quantify just how much they each contribute.

Before we consider the full set of points, we simply look at one data point. The MSE at this point is given by

$$\mathbb{E}[(z-\tilde{z})^2] = \mathbb{E}[(f(x,y)+\epsilon-\tilde{z})^2],$$

where $\tilde{z}$ is the model prediction of $z = f(x,y)+\epsilon$. Adding and subtracting $\mathbb{E}(\tilde{z})$ yields

$$\begin{aligned}\mathbb{E}[(z-\tilde{z})^2] &= \mathbb{E}[f(x,y)+\epsilon-\tilde{z}+\mathbb{E}(\tilde{z})-\mathbb{E}(\tilde{z}))^2] \\ &= \mathbb{E}\Big[\big((f(x,y)-\mathbb{E}(\tilde{z}))+\epsilon+(\mathbb{E}(\tilde{z})-\tilde{z})\big)^2\Big] \\ &= \mathbb{E}\Big[\big(f(x,y)-\mathbb{E}(\tilde{z})\big)^2\Big] + \mathbb{E}\Big[2\big(f(x,y)-\mathbb{E}(\tilde{z})\big)\epsilon\Big] \\ &\quad + \mathbb{E}\Big[\epsilon^2\Big] + \mathbb{E}\Big[2\big(f(x,y)-\mathbb{E}(\tilde{z})\big)\big(\mathbb{E}(\tilde{z})-\tilde{z}\big)\Big] \\ &\quad + \mathbb{E}\Big[2\epsilon\big(\mathbb{E}(\tilde{z})-\tilde{z}\big)\Big] + \mathbb{E}\Big[\big(\mathbb{E}(\tilde{z})-\tilde{z}\big)^2\Big],\end{aligned}$$

where we have used that the mean $\mathbb{E}$ is linear. Since $(f(x,y)-\mathbb{E}(\tilde{z}))$ is simply a number, we have

$$\mathbb{E}\Big[\big(f(x,y)-\mathbb{E}(\tilde{z})\big)^2\Big] = \big(f(x,y)-\mathbb{E}(\tilde{z})\big)^2$$

and similarly

$$\mathbb{E}\Big[2\big(f(x,y)-\mathbb{E}(\tilde{z})\big)\epsilon\Big] = 2\big(f(x,y)-\mathbb{E}(\tilde{z})\big)\mathbb{E}(\epsilon) = 0,$$

since $\mathbb{E}(\epsilon) = 0$. We also see that

$$\sigma^2 = Var(\epsilon) = \mathbb{E}(\epsilon^2) - \mathbb{E}(\epsilon)^2 = \mathbb{E}(\epsilon^2)$$

For the fourth term, $2(f(x,y)-\mathbb{E}(\tilde{z}))$ factors out, which leaves

$$\mathbb{E}\big(\mathbb{E}(\tilde{z})-\tilde{z}\big) = \mathbb{E}(\tilde{z}) - \mathbb{E}(\tilde{z}) = 0.$$

The fifth term is 0 for the same reason. Thus we are left with

$$\begin{aligned}\mathbb{E}[(z-\tilde{z})^2] &= \big(f(x,y)-\mathbb{E}(\tilde{z})\big)^2 + \sigma^2 + \mathbb{E}\big[\mathbb{E}((\tilde{z})-\tilde{z})^2\big] \\ &= \text{Bias}(\tilde{z})^2 + \sigma^2 + \text{Var}(\tilde{z})\end{aligned} \tag{15}$$

Averaging over all the points in the data set yields

$$\text{MSE} = \frac{1}{n}\sum_i \text{Bias}(\tilde{z}_i)^2 + \sigma^2 + \frac{1}{n}\sum_i \text{Var}(\tilde{z}_i). \tag{16}$$

In summary, we see that the bias and variance both contribute to the MSE. However, they are not the only constituents. There will always be an irreducible error contribution, σ^2.

The above expressions for bias and variance assume that we know the true function $f(x,y)$. However, this is not the case in real-life applications.

4. Bootstrap

In the previous section we found the expression for the MSE in terms of the model bias and variance. The expressions we used for bias and variance assume that we know the true function $f(x, y)$ and $\mathbb{E}(\tilde{z})$. However, this is not the case in real-life applications. We must therefore use the data points available to us to make an estimate of these statistics. The function $f(x, y)$ can simply be estimated by the observations z, but estimating $\mathbb{E}(\tilde{z})$ proves more challenging.

We want to use the bootstrap resampling method for determining $\mathbb{E}(\tilde{z})$. The general idea is to average over the predicted $\tilde{z}$-values resulting from models trained on training sets sampled from the same original training data. The algorithm is as follows:

Algorithm 2 The bootstrap method for estimating bias and variance

1: Train main model on training set A which has N points
2: Compute MSE of main model on testing set B
3: **for** $b = 1, 2, \ldots, B$ **do**
4: Draw N points from the training set with replacement and train a model on this set
5: Compute predictions on testing set B
6: **end for**
7: Calculate the mean of all the predictions made and use this as $\mathbb{E}(\tilde{z})$
8: Estimate bias and variance for each coordinate (x_i, y_i) using equations 13 and 14 by replacing $f(x, y)$ with z and averaging over the bootstrap samples.
9: Compute the average (over all points) and standard deviation of the pointwise biases and variances from the previous step.

We use the standard deviation as an estimate of the error of the average biases and variances across all points.

D. Procedures

1. The Franke function

In order to get a better feel for how our methods work and the quality of our models, we will begin by studying a known function with added noise. The function we will be using is known as the Franke function:

$$\begin{aligned} f(x,y) =& \frac{3}{4}\exp\Big(-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}\Big) \\ &+\frac{3}{4}\exp\Big(-\frac{(9x+1)^2}{49}-\frac{(9y+1}{10}\Big) \\ &+\frac{1}{2}\exp\Big(-\frac{(9x-7)^2}{4}-\frac{(9y-3)^2}{4}\Big) \\ &-\frac{1}{5}\exp\Big(-(9x-4)^2-(9y-7)^2\Big) \end{aligned} \tag{17}$$

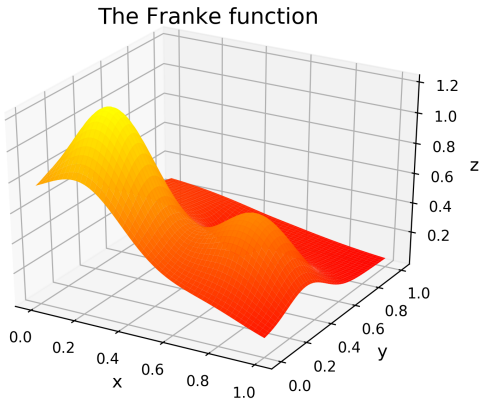


FIG. 1. The Franke function

The function is defined for $x, y \in [0, 1]$. Figure 1 shows a 3D plot of the Franke function.

For our purposes we need to add some noise to the Franke function. We add a noise which is normally distributed with mean 0 and standard deviation 1.

We will first study the OLS model. We begin by investigating the naive approach by finding the MSE as a function of model complexity. Then we move on to a more in-depth analysis with the bias-variance tradeoff and 5fold cross validation. We then use the results from these processes to select and optimal degree for approximation. Finally, we compute the squared difference between this prediction and the true Franke function.

As previously mentioned, the plain training/testing MSE does not give an accurate estimate of the actual error. We will therefore skip this step for the other methods.

For the ridge regression method we begin by studying the bias variance tradeoff and the 5fold cross validation in order to determine the best value of the penalty λ. We then make an approximation using this value. The errors are studied in the same way as for OLS.

When it comes to lasso regression, the procedure will be the same as for ridge regression.

2. *Predicting terrain data*

The next step is to use our models to predict terrain data. The original terrain data[3] is plotted in figure 2.

For this part of the project we will repeat the analysis done for the Franke function as described in the previous section. The only difference is that for ridge and lasso we will perform a grid search in order to obtain a first rough estimate of the best polynomial degree and λ-combination. This will be done by computing (naively) the MSE for a set of combinations. This process will be used to determine a polynomial degree for which we will then use 5fold cross validation and the bias variance tradeoff for determining the best penalty λ.

III. RESULTS

Figure 3 shows how the MSE varies as a function of the model complexity (polynomial degree) for the OLS approximation of the Franke function. Figure 4 shows the bias variance tradeoff for the OLS approximation of the Franke function using bootstrap resampling. Figure 5 shows the corresponding 5fold cross validation results. Based on the latter plot, an OLS approximation using a 4th order polynomial was made. This is presented in figure 6. The squared difference between the approximation and the true Franke function is shown in figure 7.

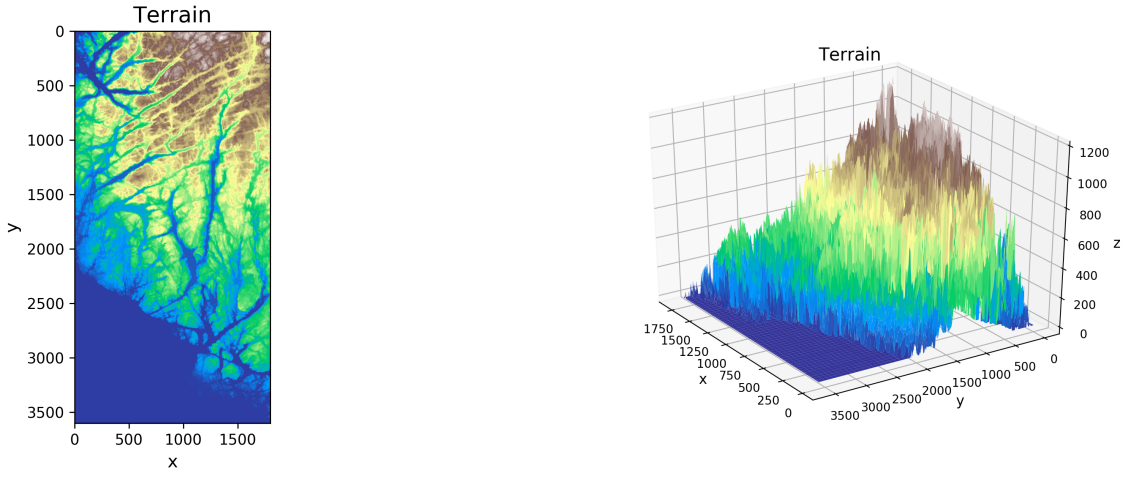
Figure 8 shows the bias variance tradeoff of the 5th degree ridge approximation of the Franke function for a variety of penalties λ. Figure 9 shows the corresponding 5fold cross validation. Based on the latter, $\lambda = 10^{-2}$ was chosen. The resulting approximation and error is shown in figures 10 and 11 respectively.

Figure 12 shows the bias variance tradeoff of the 5th degree lasso approximation for a variety of penalties λ. Figure 13 shows the corresponding 5fold cross validation. Based on the latter, $\lambda = 10^{-4}$ was chosen. The resulting approximation and error is shown in figures 14 and 15 respectively.

Figure 16 shows the bias variance tradeoff of the terrain data OLS approximation.The corresponding 5fold cross validation is shown in figure 17. Figure 18 shows the OLS approximation using a 5th order polynomial. The squared difference between the prediction and the original terrain data is shown in figure 19.

Figure 20 shows the test MSE for each combination of penalties λ and polynomial degrees for ridge regression of the terrain data. Figure 21 shows the bias variance tradeoff for the 4th degree polynomial approximation. Figure 22 shows the corresponding 5fold cross validation. Figure 23 shows the ridge approximation of the terrain data with a 4th degree polynomial and penalty $\lambda = 10$. The corresponding errors are shown in figure 24.

Figure 25 shows the test MSE for each combination of penalties λ and polynomial degrees for lasso regression of the terrain data. Figure 26 shows the bias variance tradeoff for the 3rd degree polynomial approximation. Figure 27 shows the corresponding 5fold cross validation. Figure 28 shows the lasso approximation of the terrain data with a 3rd degree polynomial and penalty $\lambda = 10^{-4}$. The corresponding errors are shown in figure 29.



(a) 2D plot of terrain data.

(b) 3D plot of terrain data.

FIG. 2. The terrain of Møsvatn Austfjell in Norway.

[3] The data were downloaded from the MachineLearning GitHub.

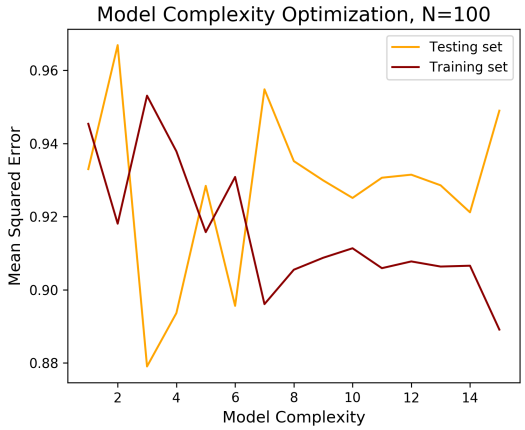


FIG. 3. MSE as a function of model complexity for the Franke function OLS approximation.

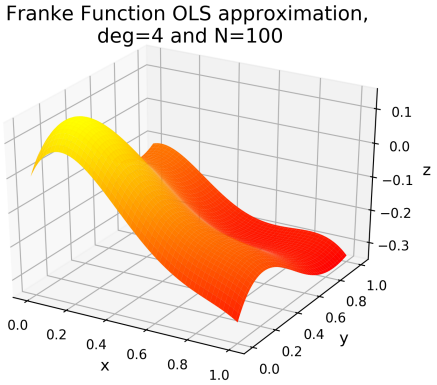


FIG. 6. OLS approximation of the Franke function by a 5th degree polynomial.

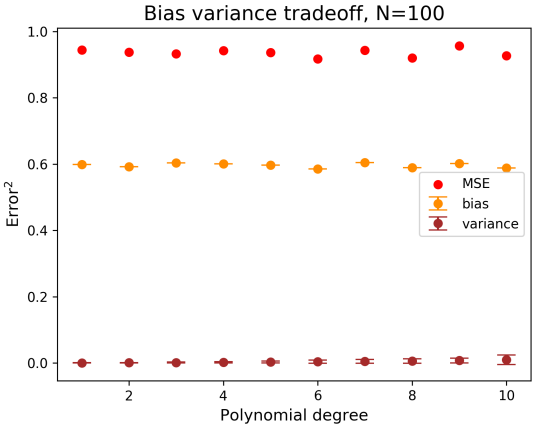


FIG. 4. Bias variance tradeoff of the Franke function OLS approximation.

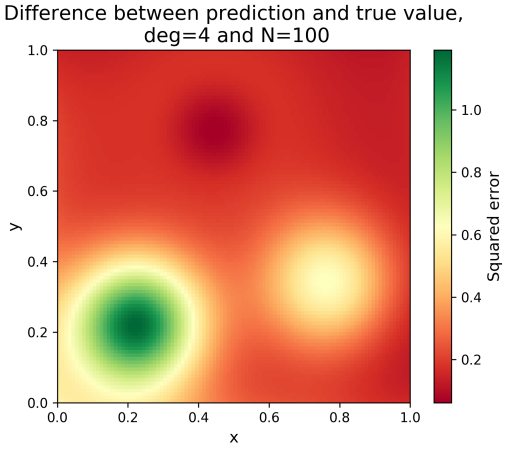


FIG. 7. The errors of the 5th degree OLS approximation of the Franke function.

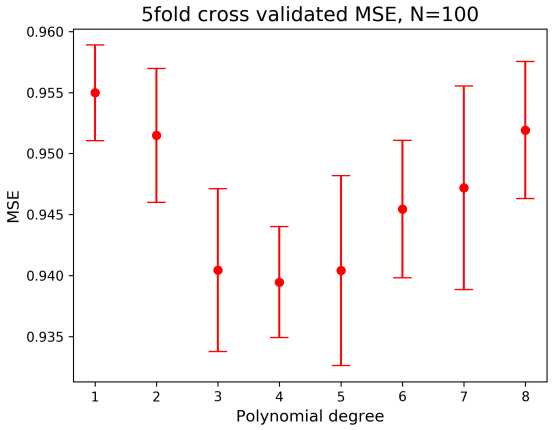


FIG. 5. Cross validated MSE as a function of model complexity for the Franke function OLS approximation.

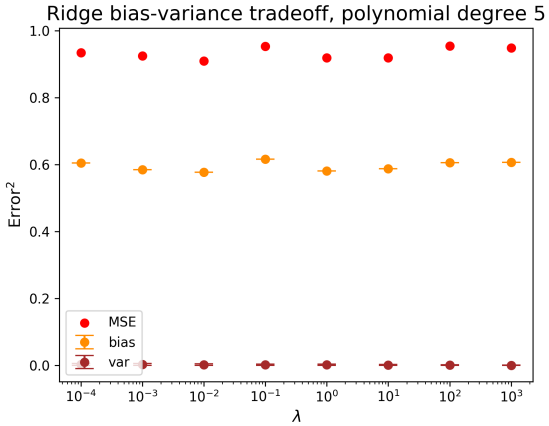


FIG. 8. Bias variance tradeoff of the Franke function ridge approximation.

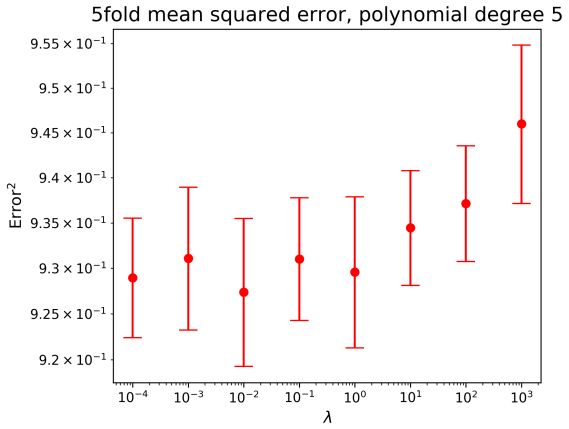


FIG. 9. Cross validated MSE as a function of penalties λ for the Franke function ridge approximation.

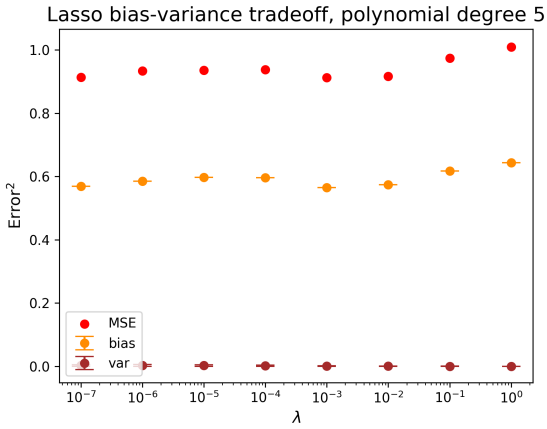


FIG. 12. Bias variance tradeoff of the Franke function lasso approximation.

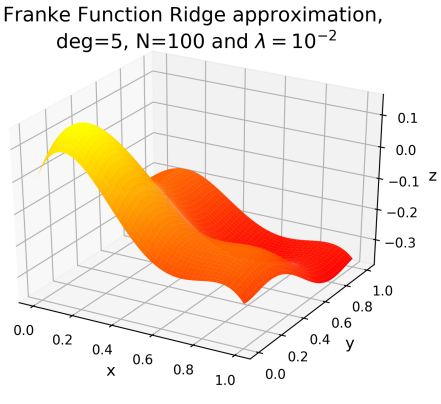


FIG. 10. Ridge approximation of the Franke function by a 5th degree polynomial using a penalty $\lambda = 10^{-2}$.

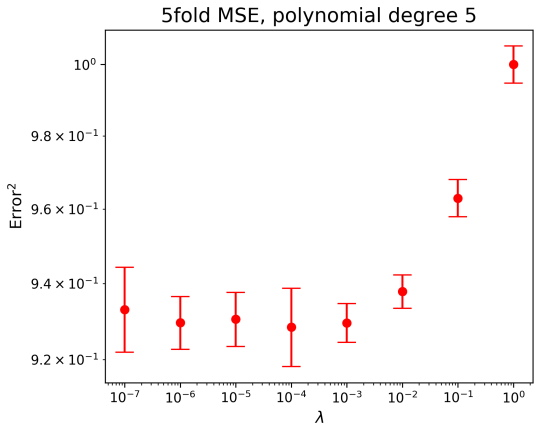


FIG. 13. Cross validated MSE as a function of penalties λ for the Franke function lasso approximation.

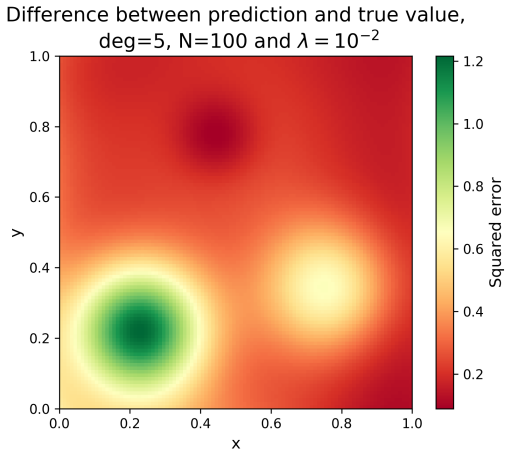


FIG. 11. The errors of the 5th degree ridge approximation of the Franke function with penalty $\lambda = 10^{-2}$.

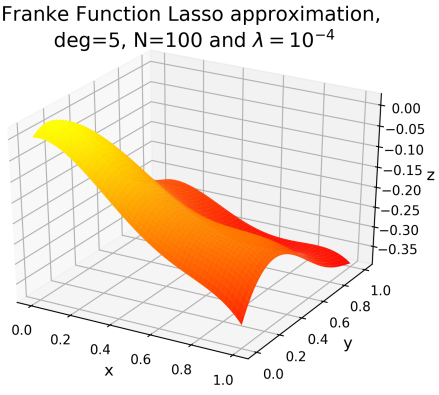


FIG. 14. Lasso approximation of the Franke function by a 5th degree polynomial using a penalty $\lambda = 10^{-4}$.

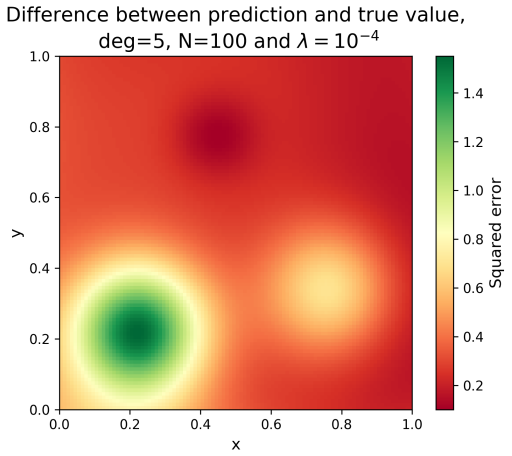


FIG. 15. The errors of the 5th degree lasso approximation of the Franke function with penalty $\lambda = 10^{-4}$.

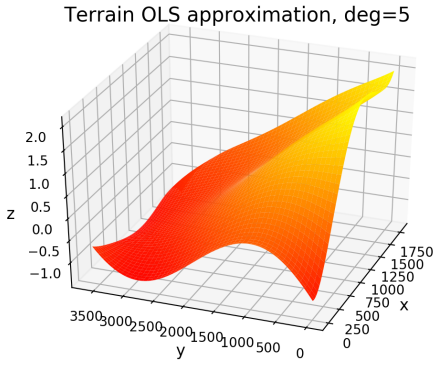


FIG. 18. OLS approximation of the terrain data by a 5th degree polynomial.

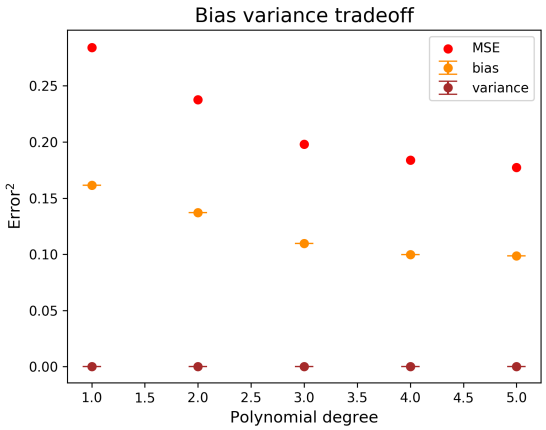


FIG. 16. Bias variance tradeoff of the terrain data OLS approximation.

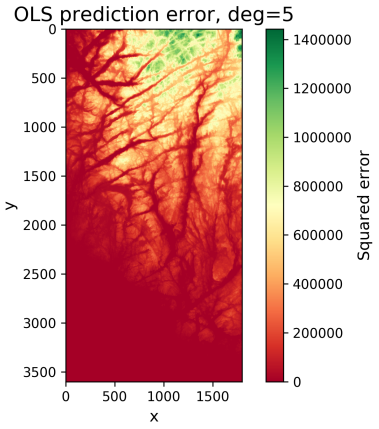


FIG. 19. The errors of the 5th degree OLS approximation of the terrain data.

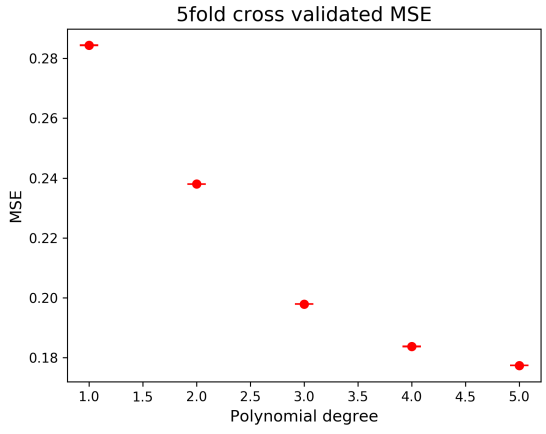


FIG. 17. Cross validated MSE as a function of model complexity for the terrain data OLS approximation.

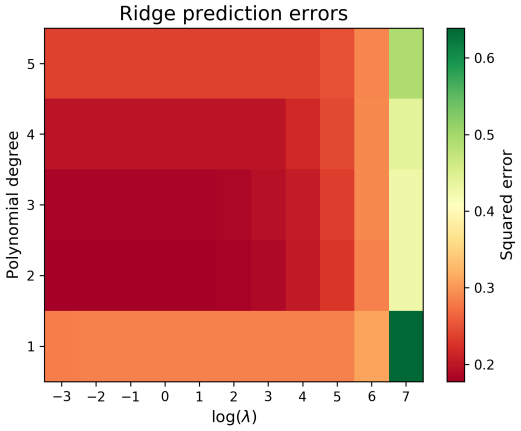


FIG. 20. The test MSE for each combination of penalties λ and polynomial degrees for ridge regression of terrain data.

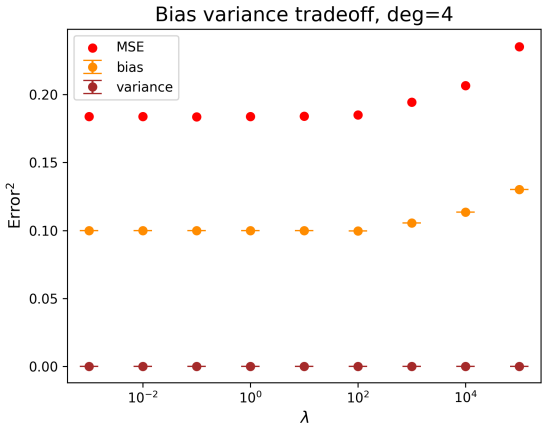


FIG. 21. Bias variance tradeoff for a 4th degree ridge approximation on terrain data.

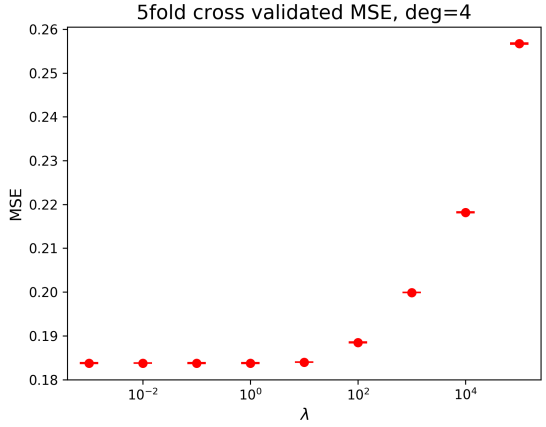


FIG. 22. 5fold cross validation of penalties λ for a 4th degree ridge approximation on terrain data.

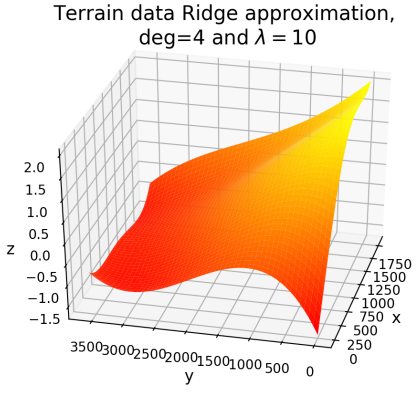


FIG. 23. 4th degree ridge approximation of terrain data with penalty $\lambda = 10$.

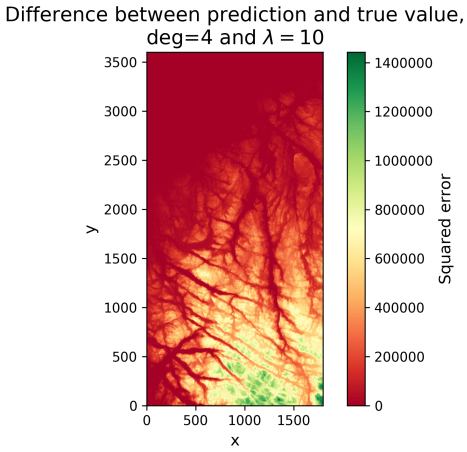


FIG. 24. Errors of the 4th degree ridge approximation of terrain data with penalty $\lambda = 10$.

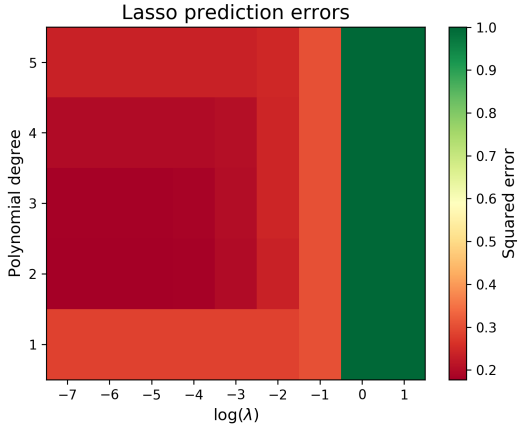


FIG. 25. The test MSE for each combination of penalties λ and polynomial degrees for lasso regression of terrain data.

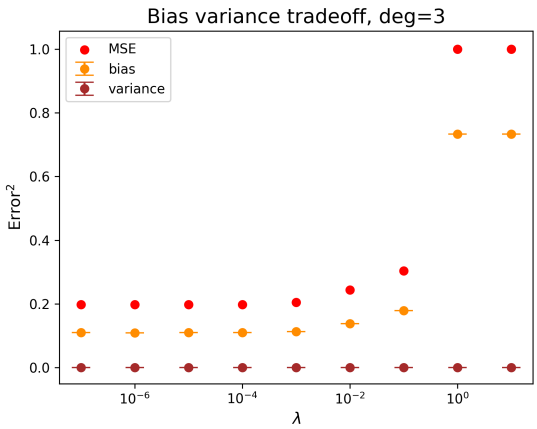


FIG. 26. Bias variance tradeoff for a 3rd degree lasso approximation on terrain data.

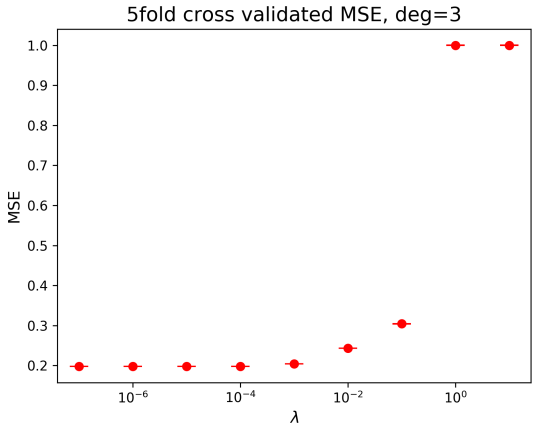


FIG. 27. 5fold cross validation of penalties λ for a 3rd degree lasso approximation on terrain data.

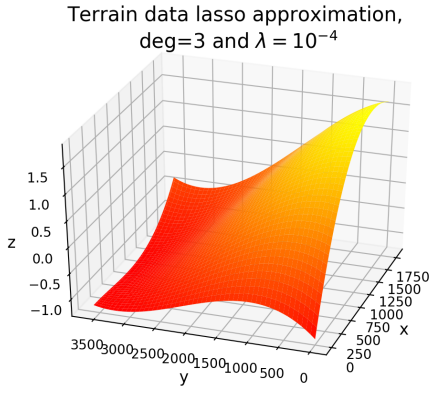


FIG. 28. 3rd degree lasso approximation of terrain data with penalty $\lambda = 10^{-4}$.

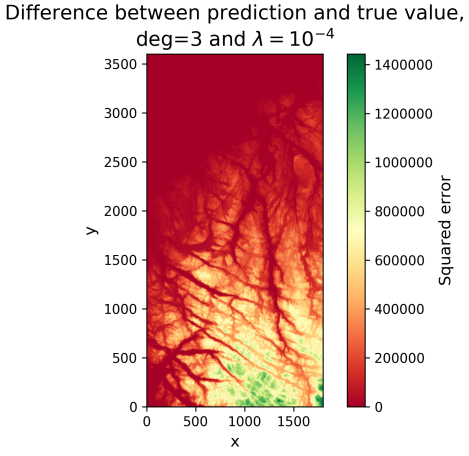


FIG. 29. Errors of the 3rd degree lasso approximation of terrain data with penalty $\lambda = 10^{-4}$.

IV. DISCUSSION

For the Franke function, we noticed that a larger generated data set gave a better approximation. This makes sense, since the training set size increases when the total number of points increases. In addition, the higher polynomial degrees require more points for a fit to be possible, due to the high number of coefficients necessary.

For the Franke function we got a bias variance tradeoff that showed no clear trends. This was the case for all three methods used. We would expect the variance to increase with the increase of polynomial degree (overfitting), but no such trends are visible. In addition, we would expect the bias to increase with the decrease of polynomial degree (underfitting). This trend is also absent. However, the 5fold cross validation gave results which were easy to interpret: a clear minimum error could be found. The optimal values for polynomial degree and penalty were therefore chosen based on the cross validation and not the bias variance plots.

We found that the ridge and lasso methods were not significantly better than the OLS method for predicting the Franke data. It is possible that a better combination of penalty and polynomial degree could provide a better result. However, we found the computational cost of finding a better λ to be very large compared to the observed benefits. A possible solution could be to investigate whether a different family of functions can provide better results than the polynomial family.

The bias variance tradeoff was easier to interpret for the terrain data. Here we saw a clear variation. The 5fold cross validation results agreed with the bias variance tradeoff results. This is most likely due to the large size of the data set.

The approximation error plots of the terrain data looked very different from the corresponding Franke plots. The vein-like patterns observed in the terrain error is a result of the highly irregular terrain surface. The Franke function is smooth, so the corresponding Franke error does not display this feature. All in all, the Franke function provided a good substitute for a terrain data set.

V. CONCLUSION

The aim of this project was to study parametrisation of surface data using linear regression methods with two dimensional polynomials. We used ordinary least squares, ridge and lasso regression and investigated polynomial degrees up to maximum 15. We found that while ridge and lasso provide an additional penalty parameter for adjusting the regression coefficients, the additional benefits were too small to make up for the added computational cost.

Bootstrap was used to estimate the bias variance tradeoff in our models, but these results proved challenging to interpret. 5fold cross validation was used to estimate the true MSE. These results were much more interpretable and provided a clear picture of what parameters gave the best approximation.

In general, we found that to parametrise surfaces using polynomials provided reasonable results. However, the polynomial approach was not able to capture irregular details typically found in real terrain data.

[1] Morten Hjorth-Jensen. Resampling techniques and ordinary least square. Lecture slides, Sep 2020.

[2] Morten Hjorth-Jensen. Ridge and lasso regression. Lecture slides, Sep 2020.

[3] Hastie et al. *The Elements of Statistical Learning*. Springer, 2 edition, 2009.

[4] James et al. *An Introduction to Statistical Learning*. Springer, 2013.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.