Analysis of topographic data using linear regression methods

Frieda Wik

September 2023

1 Abstract

Topographic data is widely used for scientific and technical studies, including hydrology, agriculture and forestry. Such data must be prepared prior to usage and a well fitting parameterisation is crucial to any further analysis [1]. Here we retrieve and pre-process terrain data from the U.S. Geological Survey [4] and parameterise it using linear regression models such as Ordinary Least Squares, Ridge and Lasso regression. We study polynomials of increasing complexity and the impact of the regularisation parameters λ. We also study the behaviour of the polynomial coefficients and the bias-variance trade-off of the error between our predicted values and the corresponding true values from our test data set. We find that the Ordinary Least Squares regression is the most appropriate model to approximate the terrain surface. However, the model complexity must be limited to avoid instability and excessive computing time.

2 Introduction

In this project 3D digital terrain data from Western Norway was parameterised using three different linear regression models: Ordinary Least Squares (OLS), Ridge and Lasso regression. The goal was to create a surface model accurate enough to depict fjords and mountains, producing a numerical map suitable for subsequent analysis. Polynomial functions up to 20th degree were tested with various penalty parameters. The adequateness of the models were evaluated by looking at the Mean Squared Error (MSE) and the R^2 score function. The data was divided into a train and a test dataset and the Bootstrap and the Kfold cross- validation resampling methods were used when evaluating the performance of the models. For development purpose, a data set with the two-dimensional Franke's function was created. This function is popular when testing algorithms [3] and was used to assure the correctness of the models prior to using them for the analysis of topographic data. Polynomials up to fifth-degree were tested on the Franke's function. The theoretical background and the model set-up are described in the method section, while the results from the Franke's function and the terrain data are presented in the result section. Additional material can be found at the GitHub repository of the project at https://github.com/friedawik/FYS-STK4155-.

3 Methods

3.1 Linear Regression

In a linear regression model the relationship between the input vector $X^T = (X_1, X_2, X_3...X_p)$ and the output Y is assumed to be linear [2]. The linear relationship is modelled via the approximation $\hat{Y}$ as described by equation 1, where the hat denotes the solution when using the optimal coefficients.

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \tag{1}$$

If we include $\hat{\beta}_0$ in the vector of coefficients $\hat{\beta}$ and add the constant variable 1 in X, then we can write the linear model as the inner product $\hat{Y} = X^T \hat{\beta}$ and further on in matrix notation as $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$. In this report, the matrix $\boldsymbol{X}$ will be referred to as the design matrix. Thus, $\hat{\boldsymbol{y}}$ approximates the true $\boldsymbol{y}$ which is given by $\boldsymbol{y} = \boldsymbol{f(x)} + \boldsymbol{\epsilon}$, where $\boldsymbol{f(x)}$ is the continuous true function and $\boldsymbol{\epsilon}$ is the normally distributed error $\boldsymbol{\epsilon} \sim N(0, \sigma^2)$. The expectation value of $\boldsymbol{y}$ for a given element i is given by equation 2.

$$\begin{aligned}
\mathbb{E}[y_i] &= \mathbb{E}[f(x_i) + \epsilon_i] \\
&= \mathbb{E}[f(x_i)] + \mathbb{E}[\epsilon_i] \\
&= \mathbb{E}[f(x_i)] + 0 \\
&= \mathbb{E}[\hat{y}_i] \\
&= \sum_j x_{ij}\beta_j \\
&= \boldsymbol{X}_{i,*}\boldsymbol{\beta}
\end{aligned} \tag{2}$$

Where the expectation of the error is $\mathbb{E}[\epsilon_i] = 0$, as defined above. The variance of $\boldsymbol{y}$ for a given element i is given by equation 3. In this project, the values of the design matrix $\boldsymbol{X}$ are fixed, meaning they have no inherent stochastisity and do not contribute to the error. Then, the variance for a given element i of $\boldsymbol{y}$ is equal to the variance of the error ϵ [5]:

$$\begin{aligned}
Var(y_i) &= \mathbb{E}[[y_i - \mathbb{E}[y_i]]^2] \\
&= \mathbb{E}[y_i^2 - 2y_i\mathbb{E}[y_i] + \mathbb{E}[y_i]^2] \\
&= \mathbb{E}[y_i^2] - 2\mathbb{E}[y_i]^2 + \mathbb{E}[y_i]^2 \\
&= \mathbb{E}[y_i^2] - \mathbb{E}[y_i]^2 \\
&= \mathbb{E}[(X_i\beta + \epsilon_i)^2] - (X_i\beta)^2 \\
&= \mathbb{E}[(X_i\beta^2 + 2X_i\beta\epsilon_i + \epsilon_i^2] - X_i\beta^2 \\
&= (X_i\beta)^2 + 2X_i\beta\mathbb{E}[\epsilon] + \mathbb{E}[\epsilon^2] - (X_i\beta)^2 \\
&= 2X_i\beta \times 0 + \mathbb{E}[\epsilon^2] \\
&= \sigma^2
\end{aligned} \tag{3}$$

The two equations show that the data $\boldsymbol{y}$ is assumed to have a mean value of $\mathbf{X}_{i,*}\boldsymbol{\beta}$ with variance σ^2. The best fit of the linear model is found by calculating the unknown coefficients $\boldsymbol{\beta}$ that minimize a so-called cost function. The three linear regression methods used in this project are described below, as well as their corresponding cost functions.

The OLS regression
In the OLS model, the estimated coefficients $\beta = (\beta_0, \beta_1, ...\beta_p)^T$ of a model fitting a data set $(x_1, y_1)..(x_N, y_N)$ are found by minimizing the cost-function that corresponds to the residual sum

of squares as described in equation 4.

$$\begin{aligned} C(\boldsymbol{X}, \boldsymbol{\beta^{OLS}}) &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \\ &= \frac{1}{N}\sum_{i=0}^{N-1}(y_i - \tilde{y}_i)^2 \\ &= \mathbb{E}[(\boldsymbol{y} - \tilde{\boldsymbol{y}})^2] \end{aligned} \tag{4}$$

The minimum of equation 4 can be found by differentiating with respect to $\boldsymbol{\beta}$. In matrix-vector notation the optimal $\hat{\boldsymbol{\beta}}$ giving rise to the minimum can be expressed as

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y} \tag{5}$$

This holds as long as the matrix $\mathbf{X}^T\mathbf{X}$ is non-singular such that the least squares coefficients $\hat{\boldsymbol{\beta}}$ are uniquely defined. The expectation values of the optimal coefficients $\hat{\boldsymbol{\beta}}$ are given by equation 6.

$$\begin{aligned} \mathbb{E}[\hat{\boldsymbol{\beta}}] &= \mathbb{E}[(\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\boldsymbol{y}] \\ &= (\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\mathbb{E}[\boldsymbol{y}] \\ &= (\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\ &= \boldsymbol{\beta} \end{aligned} \tag{6}$$

This result show that the optimal $\hat{\boldsymbol{\beta}}$ coefficients give a model that is unbiased [3]. The Gauss-Markov theorem states that the OLS is the best estimator of all linear and unbiased estimators, meaning it gives the lowest MSE. However, biased methods such as the Ridge regression and Lasso regression may result in lower prediction error by reducing the variance while allowing for some bias [2]. The variance of the optimal $\hat{\boldsymbol{\beta}}$ is given by equation 7. The variance measures the spread in the optimal coefficients and can be used to calculate the confidence interval of our solution [3].

$$\begin{aligned} Var(\hat{\boldsymbol{\beta}}) &= \mathbb{E}[[\hat{\boldsymbol{\beta}} - \mathbb{E}(\hat{\boldsymbol{\beta}})][\hat{\boldsymbol{\beta}} - \mathbb{E}(\hat{\boldsymbol{\beta}})]^T] \\ &= \mathbb{E}[[(\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{\beta}][(\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{\beta}]^T] \\ &= (\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T)\boldsymbol{X}(\boldsymbol{X^T X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= (\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2)\boldsymbol{X}(\boldsymbol{X^T X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= ((\boldsymbol{X^T X})^{-1}(\boldsymbol{X^T X})\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2(\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T)\boldsymbol{X}(\boldsymbol{X^T X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= (\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{X}^T + \sigma^2(\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T)\boldsymbol{X}(\boldsymbol{X^T X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \boldsymbol{\beta}\boldsymbol{\beta}^T(\boldsymbol{X^T X})(\boldsymbol{X^T X})^{-1} + \sigma^2(\boldsymbol{X^T X})^{-1}(\boldsymbol{X^T X})(\boldsymbol{X^T X})^{-1} - \boldsymbol{\beta}\boldsymbol{\beta}^T \\ &= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} \end{aligned} \tag{7}$$

Where $\mathbb{E}(\boldsymbol{y}\boldsymbol{y^T}) = \boldsymbol{X}\boldsymbol{\beta}\boldsymbol{\beta}^T\boldsymbol{XT} + \sigma^2\boldsymbol{I}_{nn}$

The Ridge regression
The Ridge regression imposes a penalty on the regression coefficients by shrinking them by a parameter λ and thus introducing a bias to the model. This regularisation is based on the L2 norm and the optimal $\hat{\boldsymbol{\beta}}$ can be found by minimizing the sum of squares with the penalty parameter λ added as described by equation 8. The data needs to be standardised before applying Ridge regression to avoid that the intercept is penalised [2].

$$C(\beta^{Ridge}) = \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{8}$$

In vector-matrix notation the optimal regression coefficients can be expressed as shown in equation 9.

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

where $\mathbf{I}$ is the identity matrix. When $\lambda \to \infty$ the coefficients $\beta \to 0$ and the MSE stabilises. When $\lambda \to 0$ the solution approaches the OLS solution.

The Lasso regression
The Lasso regression also shrinks the regression coefficients, but does so based on the L1 norm. In Lasso regression the cost-function to be minimized is on the form given in equation 10. Again, the Lasso output are scaling-dependent and the data needs to be standardised before training the model.

$$C(\boldsymbol{\beta}^{Lasso}) = \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{10}$$

The Lasso penalty makes the solution non-linear, such that $\hat{\boldsymbol{\beta}}$ cannot be found by matrix inversion. In this project the Lasso regression has been performed using the Scikit-learn toolkit.

3.2 Resampling techniques

The best approach for model selection and assessment is to set aside a validation set for parameter tuning so that the test set can be used for evaluation of the final model only. Evaluating the model on the test data will lead to underestimation of the prediction error on new data [2]. In this project, the data has been split in train and test data only and there was therefore no validation data set available. However, resampling techniques can be used for accuracy assessment when data is too scarse to set aside a validation set [5]. In this project, the Bootstrap and the Kfold cross-validation resampling techniques were employed to find the MSE of the training and test data. The Python time module was used to benchmark the time consumption of these methods.

In the *Bootstrap* resampling technique, samples are drawn randomly with replacement from the training data. When this is done B times, there will be B bootstrap samples created, each sample with the same size as the original data. This mimics drawing new samples from the population, and allows for assessment of the models [2]. Some of the data will be drawn multiple times while others will never be used. The Bootstrap method is computing expensive, and the number of bootstrap samples were limited to 30 in this project, to avoid the analyses to be overly time consuming. The data was split in train and test set before bootstrapping so that the model evaluation was done on independent data. The Bootstrap implementation to calculate MSE is shown in algorithm 1.

Algorithm 1 Bootstrap resampling technique

```
 1: procedure BOOTSTRAP($X, z$)                                   ▷ The full data set
 2:     Divide in train and test sets
 3:     Choose number n of bootstraps
 4:     for i in $range(n)$ do
 5:         Draw bootrap samples with replacement from train data
 6:         Define the vector x* that contains the bootstrap samples
 7:         Compute $\beta$* with values from x*
 8:         Compute predicted z
 9:         Compute MSE
10:     Compute average MSE                                        ▷ From all n models
11:     return Average MSE
```

In the *Kfold cross-validation* method, the data is split i k parts and every part is used once as test data and k-1 times as training data. This ensures that the model is tested on independent data, since the remaining fold is always kept aside for model evaluation. The error is averaged from the individual MSEs calculated from each fold. Typically, when increasing the number of folds, the bias decreases but the variance increases [2]. In this project five to ten folds were tested. The Kfold cross-validation implementation to calculate MSE is shown in algorithm 2.

Algorithm 2 Kfold cross-validation

```
1: procedure KFOLD CROSS-VALIDATION($X, z$)                        ▷ The full data set
2:     split in k+1 data sets                                      ▷ $k$ folds + test set
3:     for $i$ folds up to $k$ do
4:         Compute $\beta$                                         ▷ Train model on k folds
5:         Compute predicted z
6:         Compute MSE                                             ▷ Evaluate model on last fold
7:     Compute average MSE                                         ▷ From all k models
8:     return Average MSE
```

3.3 The bias-variance trade-off

Given that the real data $\boldsymbol{y}$ represents the true values $\boldsymbol{f}$ with an approximation error ϵ assumed to have expectation value $E(\epsilon) = 0$ and variance $Var(\epsilon) = \sigma_\epsilon^2$, then the prediction error can be decomposed into the squared bias, variance and an irreducible error. To show this, we define an approximation of $\boldsymbol{f}$ in terms of the coefficients $\boldsymbol{\beta}$ and the design matrix $\boldsymbol{X}$, such that $\boldsymbol{f} \approx \tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$. The decomposition of the error between the real data $\boldsymbol{y}$ and the approximation $\tilde{\boldsymbol{y}}$ is given in equation 11.

$$\begin{aligned}\mathbb{E}[(\boldsymbol{y}-\tilde{\boldsymbol{y}})^2] &= \mathbb{E}[\boldsymbol{y^2 - 2y\tilde{y}} + \tilde{y}^2] \\ &= \mathbb{E}[\boldsymbol{y^2}] - 2\boldsymbol{y}\mathbb{E}[\tilde{\boldsymbol{y}}] + \mathbb{E}[\boldsymbol{\tilde{y}^2}] \\ &= \boldsymbol{f^2} + \boldsymbol{\sigma^2} - 2f\mathbb{E}[\tilde{\boldsymbol{y}}] + var(\tilde{\boldsymbol{y}}) + \mathbb{E}[\tilde{\boldsymbol{y}}]^2 \\ &= \boldsymbol{(f - \mathbb{E}[\tilde{y}])^2} + var(\tilde{\boldsymbol{y}}) + \boldsymbol{\sigma^2} \\ &= (bias[\tilde{\boldsymbol{y}}])^2 + var(\tilde{\boldsymbol{y}}) + \boldsymbol{\sigma^2}\end{aligned} \tag{11}$$

Where we have used the definition of variance given in equation 12.

$$\mathbb{E}[\boldsymbol{\tilde{y}^2}] = var(\tilde{\boldsymbol{y}}) + \mathbb{E}[\tilde{\boldsymbol{y}}]^2 \tag{12}$$

And rewritten $\mathbb{E}[\boldsymbol{y^2}]$ as shown in equation 13, using that $\boldsymbol{f}$ is deterministic and thus $\mathbb{E}[\boldsymbol{f}] = \boldsymbol{f}$.

$$\begin{aligned}\mathbb{E}[\boldsymbol{y^2}] &= \mathbb{E}[\boldsymbol{(f+\epsilon)^2}] \\ &= \mathbb{E}[\boldsymbol{(f^2 + 2f\epsilon + \epsilon^2)}] \\ &= \mathbb{E}[\boldsymbol{f^2}] + 2\mathbb{E}[\boldsymbol{f\epsilon}] + \mathbb{E}[\boldsymbol{\epsilon}^2] \\ &= \boldsymbol{f^2} + \boldsymbol{\sigma^2}\end{aligned} \tag{13}$$

The bias term cannot be calculated unless we know the true value $\boldsymbol{f}$. Since the true value is generally unknown, we approximate $\boldsymbol{f}$ with $\boldsymbol{y}$, as given in equation 14 [3].

$$Bias[\tilde{\boldsymbol{y}}] = \boldsymbol{(f - \mathbb{E}[\tilde{y}])} \approx \boldsymbol{(y - \mathbb{E}[\tilde{y}])} \tag{14}$$

The variance is given by equation 15.

$$var[\tilde{\boldsymbol{y}}] = \mathbb{E}[(\boldsymbol{\tilde{y} - \mathbb{E}[\tilde{y}]})^2] = \frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - \mathbb{E}[\tilde{\boldsymbol{y}})^2 \tag{15}$$

The variance of the model is the deviation of the model $\tilde{\boldsymbol{y}}$ around its own mean, while the bias of the model is the difference of the average of the model against the true mean $\boldsymbol{f}$. The irreducible error is the error in the data around its true mean, and can therefore not be reduced [2].

3.4 Pre-processing of the data

All models were first tested on the two-dimensional Franke's function, given by equation 16. This function consists of four Gaussian bump terms, each containing both exponential and quadratic terms. A dataset was generated with $x, y \in [0,1]$ and stochastic noise with normal distribution N(0,1). The magnitude of the noise was varied, as can be seen in figure 1. The experiments in this project were conducted with an added noise of $\sigma^2 = 0.05$.

$$\begin{aligned}f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+\frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right)\end{aligned} \tag{16}$$

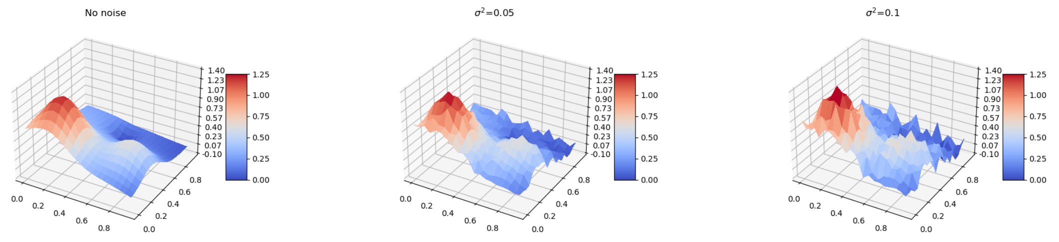


Figure 1: Franke's function without noise and with added stochastic noise of $\sigma^2 = 0.05$ and $\sigma^2 = 0.1$.

The data was scaled to avoid penalising the intercept when performing the Ridge and Lasso regression, since "the ridge solutions are not equivariant under scaling of the inputs, and so one normally standardizes the inputs before solving" [2]. The intercept is the target value z when x=0 and y=0. The scaling was performed by removing the intercept from the design matrix and then centering the data around zero by subtracting the mean. The mean was later added back to the predicted values.

The terrain data was downloaded as a .tif file from the course github page and converted to a numpy array using the imageio.imread() function. A smaller array with every 10th element was created, to allow for faster analysis. This also had a smoothening effect on the surface as some of the details of the topography were excluded. The terrain data covered an area of one degree longitude and latitude, while the z value was given in meter. Western Norway lays within 60-61° N and 8-9° W, where the span of one degree in longitude and latitude covers approximately 55x110 km. For the regression analysis, a square corresponding to approximately 55x55 km was created and the target values z were scaled such that the x and y had the same unit as the elevation. This was done by defining x and y between 0 and 1, thus producing a surface with dimension of 1x1, and then scaling the z values by $z_{scaled} = z \times 1/55e3$.

The data was split into a train and a test set using the Scikit-learn function train_test_split() where 4/5 of the data was used for training and the remaining part was used to test the models. The training data itself can not be used to find the suitable parameters, as increasing the complexity of the model would eventually lead to an interpolating fit and zero residual. Such a model would be overfitting and not predict future data [2].

The Scikit-learn library was used to test the self-written codes prior to applying them to the terrain data. For the self-written codes the pseudo invers function of Python was used.

4 Results

In this project digital terrain data from Western Norway was parametrised using OLS, Ridge and Lasso regression. Polynomial functions of the form $[x, y, xy, x^2, y^2 ...]$ up to 20th-degree were fitted to the data and the fit was evaluated in terms of MSE and r^2 score. The optimal complexity and regularisation parameter λ were found by screening over a range of different values. Prior to modelling the terrain data, the models were tested on the two- dimensional Franke's function, but only with polynomials up to 5th-degree. The Bootstrap and the Kfold cross-validation resampling techniques were used to assess the error as a function of polynomial degree and magnitude of penalization.

4.1 Test on Franke's function

The codes were developed while testing on data generated from the Franke's function. To ensure the correctness of the models, the codes were compared to equivalent functions available from the Scikit-learn library. These comparisons can be found in the GitHub repository of the project. All models and their parameters were assessed by evaluating the MSE and r^2 score performance metrics on a dataset set aside for testing. The relationship between the MSE of the test and train data of the OLS model can be seen in figure 2. As expected, the MSE of the training data decreased with increasing complexity, while the MSE of the test data started to increase again at higher-degree polynomials due to the model failing to generalise on new data, indicating overfitting. To make this plot, models up to 25th-degree polynomials were tested, since no overfitting was occuring at the fifth-degree polynomial. The training data typically yields a smaller error than the true error since the same data is being used to fit the model and to measure its error. This results in a too optimistic estimate of the generalization error. The optimism will decrease as the training sample size increases due to the improvement of the model and it will increase with the number of inputs we use [2].

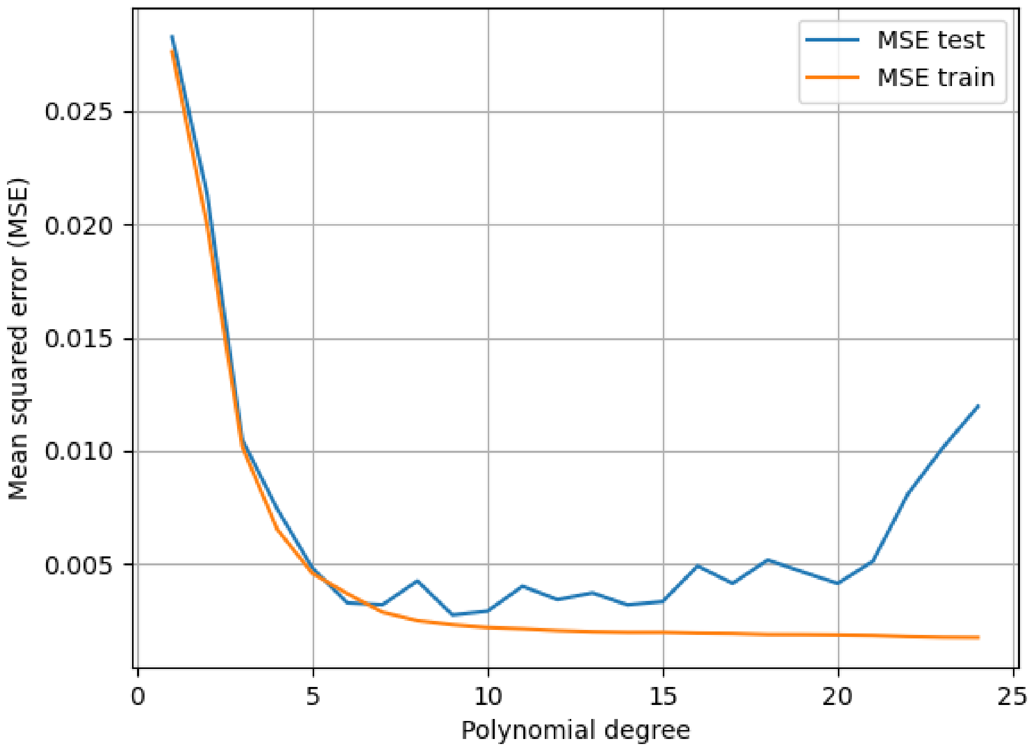
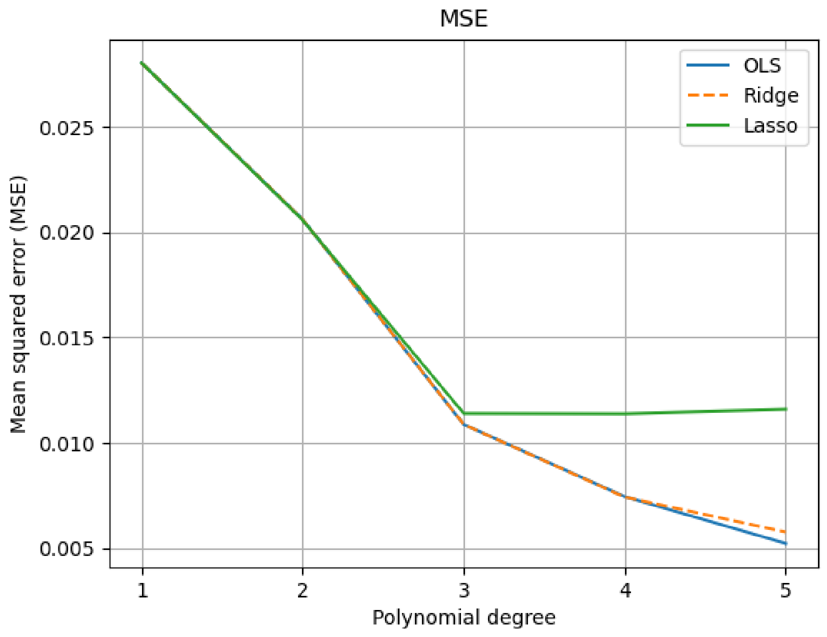


Figure 2: The MSE of the train and test datasets. The error of the training data declines with increasing complexity, while the error of the test data increases at higher-degree polynomials as overfitting starts to occur.

To determine the best degree of polynomial fit, both Bootstrap and Kfold cross-validation resampling techniques were used. Figure 3 shows the OLS, Ridge and Lasso methods performed with Kfold cross-validation, where the two latter methods were run with regularisation parameter $\lambda = 10^{-4}$. In this experiment, the Kfold cross-validation was performed by dividing the data into 5 folds. All regression methods showed a sharp decline in MSE from first to third-degree polynomials. After this, the OLS and Ridge MSEs continued to decrease, while the Lasso regression flattened. Based on the findings of figure 3, a polynomial of fifth-degree was chosen for the subsequent analysis, as the OLS and Ridge hit their lowest MSE at this degree.



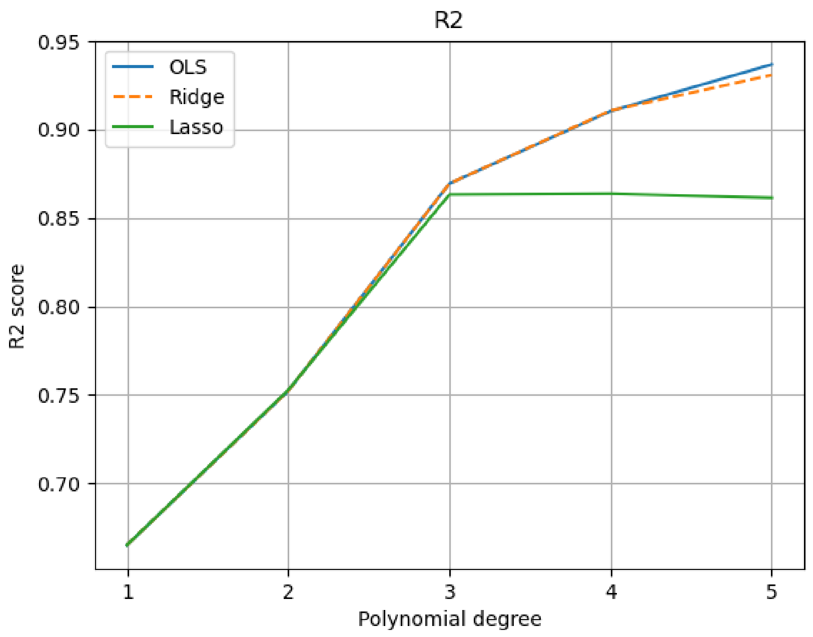


Figure 3: Mean squared error and R2 score for polynomials up to fifth-degree.

The Kfold cross-validation method was tested with five to ten folds on all three regression models. As can be seen in figure 4, no clear difference could be detected when increasing the number of folds on a fourth-degree polynomial. Typically, the bias is prone to increase at lower number of folds if data is too scarce. On the other side of the scale, the leave-one-out cross-validation, where K equals the data size, is approximately unbiased but may have high variance [2]. Since no change in MSE was detected between five and ten folds, a split of five folds was chosen for the rest of the analysis.

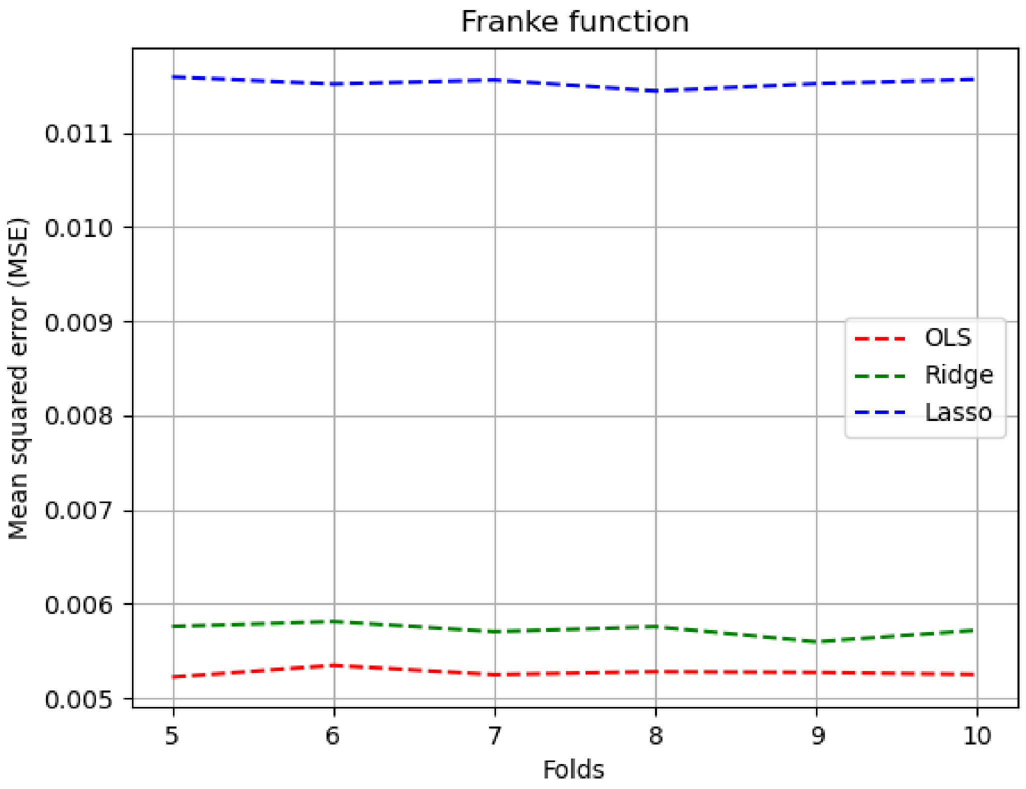


Figure 4: Kfold cross-validation with five to ten folds.

The Ridge and Lasso regression models were tested with regularisation parameters ranging from $10^{-8} < \lambda < 10^{8}$ as shown in figure 5. The MSE of both models stabilised at the same values, as their $\boldsymbol{\beta}$ coefficients shrank to 0 with increasing regularisation, resulting in predicted values that approach zero. The Lasso regularisation shrank the coefficients faster than the Ridge regularisation, such that the plateau was reached at a lower λ. For this implementation of the Franke's function, which had an added normally distributed stochastic noise of $\epsilon \sim N(0, 0.05)$, a regularisation parameter of approximately $\lambda < 10^{-5}$ and $\lambda < 10^{-4}$ resulted in the lowest MSE for Lasso and Ridge regression, respectively. For the rest of the analysis, $\lambda = 10^{-5}$ was chosen as it gave low and stable results for both Ridge and Lasso regression. With a λ this low, the penalised regression models approach the non-penalised OLS model, which indicates that the OLS is the best model for fitting the Franke's function.

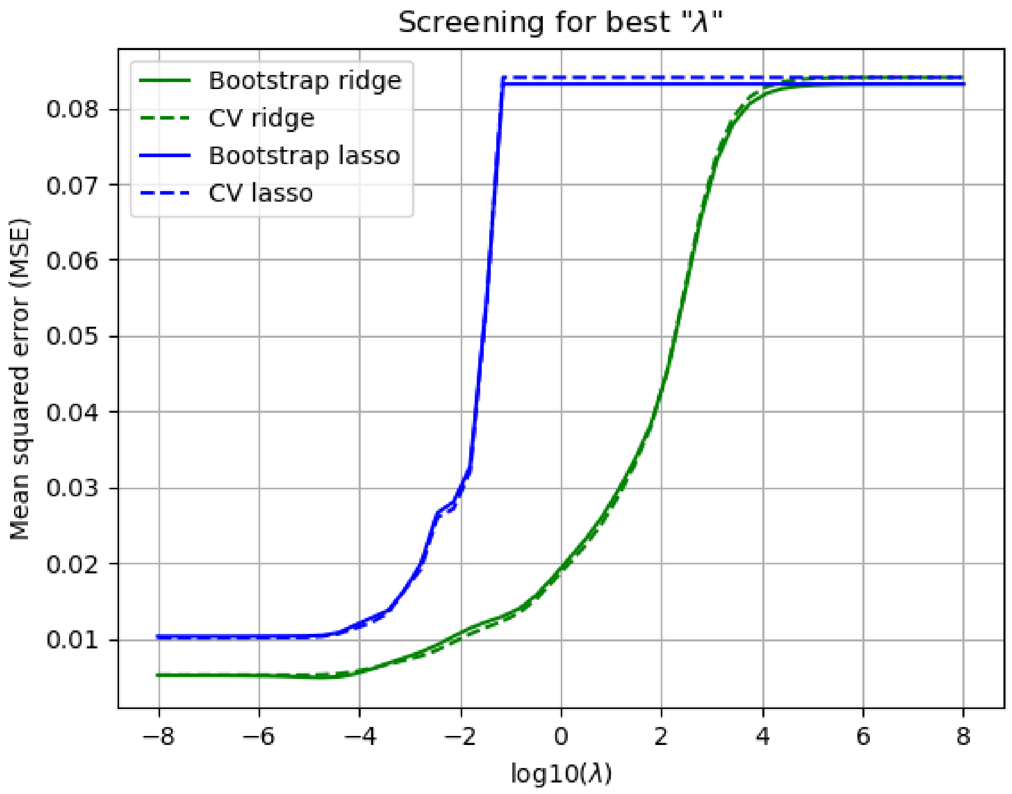


Figure 5: Cross-Validation and Bootstrap with different λ.

The bias-variance trade-off was evaluated for all three regression models. Because the threshold for the variance increase was not reached at lower-degree polynomials, the complexity was increased until the variance started to increase around eighth-degree polynomials. Figures 6 to 8 show the mean squared error decomposed in squared bias and variance for polynomials up to 10th degree.

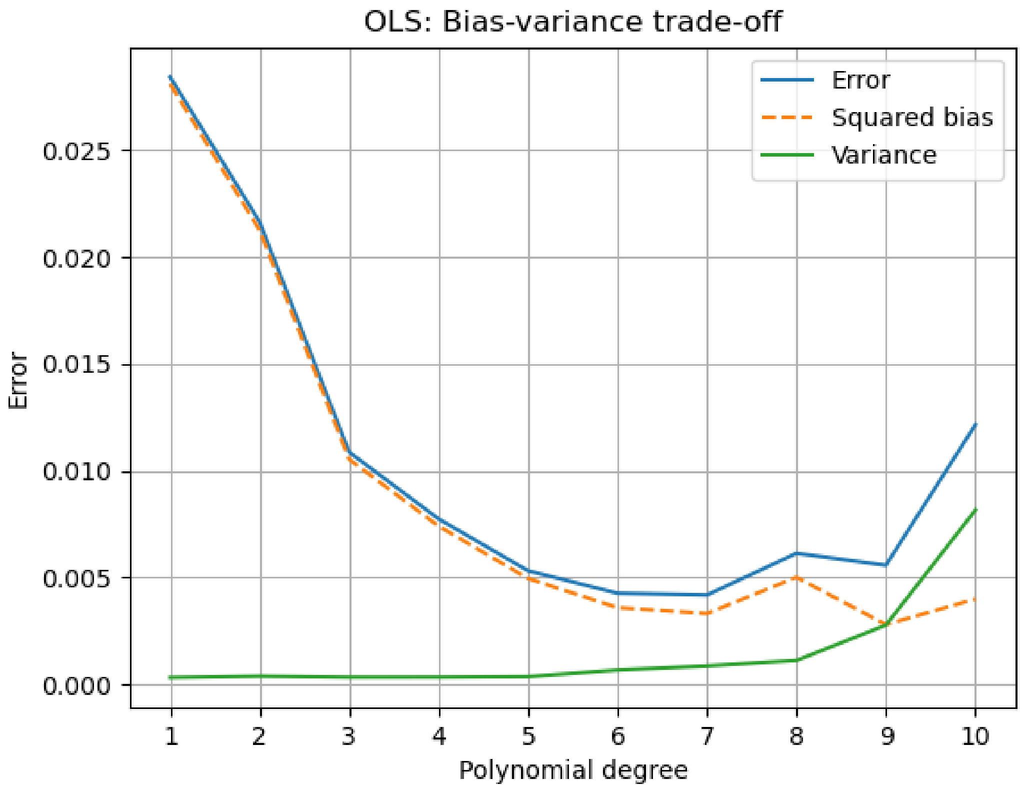


Figure 6: The bias-variance trade-off for OLS regression.

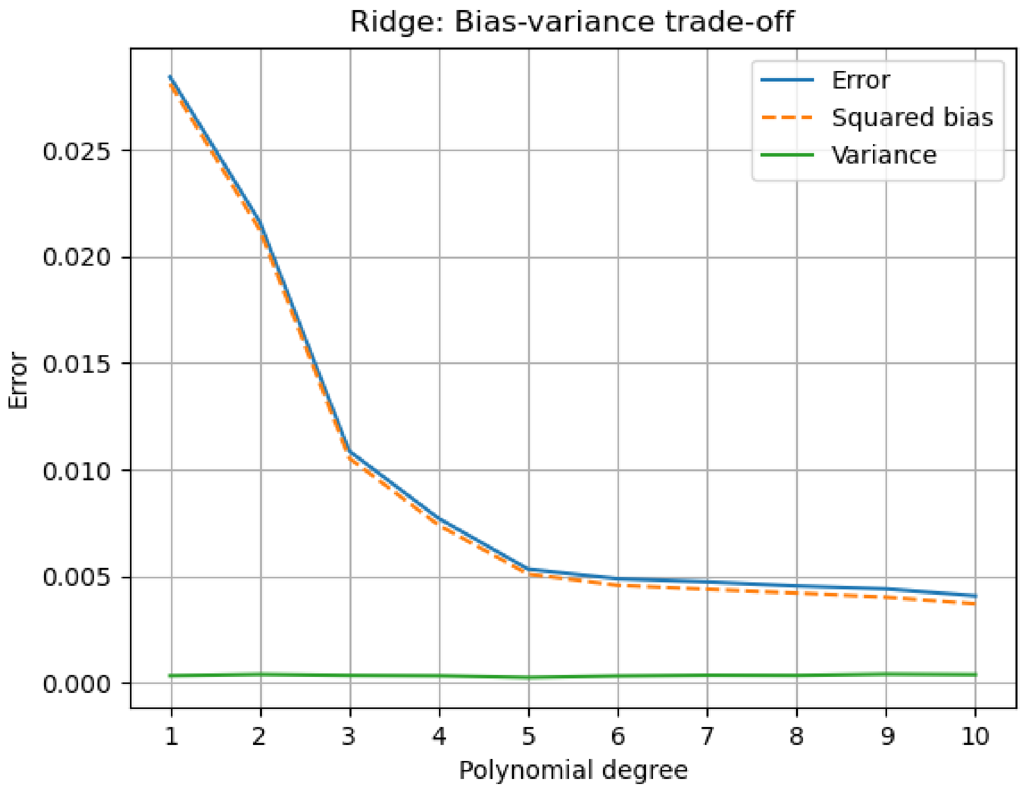


Figure 7: The bias-variance trade-off for Ridge regression with $\lambda = 1e - 4$.

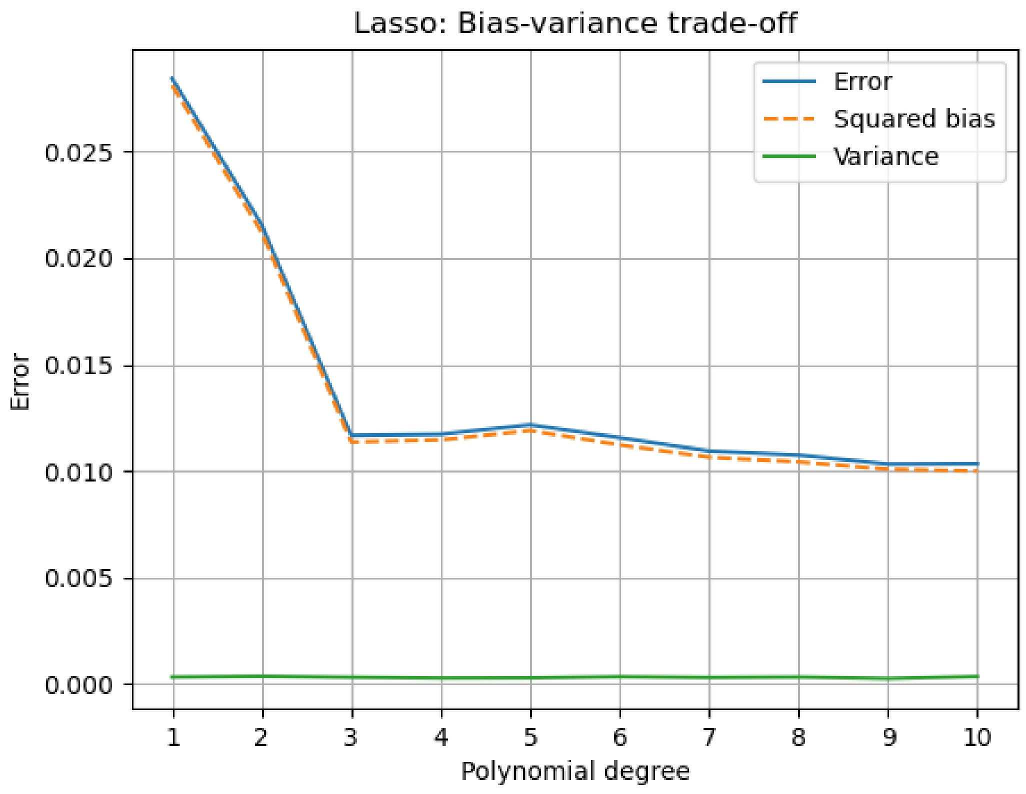


Figure 8: The bias-variance trade-off for Lasso regression with $\lambda = 1e - 4$.

For all models, the full error was composed of the squared bias and no variance at start. For the OLS model, the variance started to increase after eight-degree polynomials, leading to an increase of the total error. The variance of the Ridge and Lasso models showed no sign of increasing due to the penalty imposed on the coefficients. The regularisation term λ is proportional to the square of the magnitudes of the coefficients in the regression equation and forces the coefficients to be smaller. This shrinkage reduces the model's sensitivity to individual data points and makes it less prone to overfitting. As the penalty term decreases, the model's flexibility decreases, leading to higher bias but lower variance [2]. This can be directly observed in figures 9 to 11, where the OLS and Ridge coefficients range from approximately $-9 < \beta < 9$ while the Lasso coefficients range from less than $-2 < \beta < 2$. Whereas the coefficients for OLS and Ridge follow each other tightly, the Lasso model returns strongly dampened coefficients. This corresponds to a stronger penalisation by the penalty factor λ. For this experiment, $\lambda = 10^-5$ was chosen.

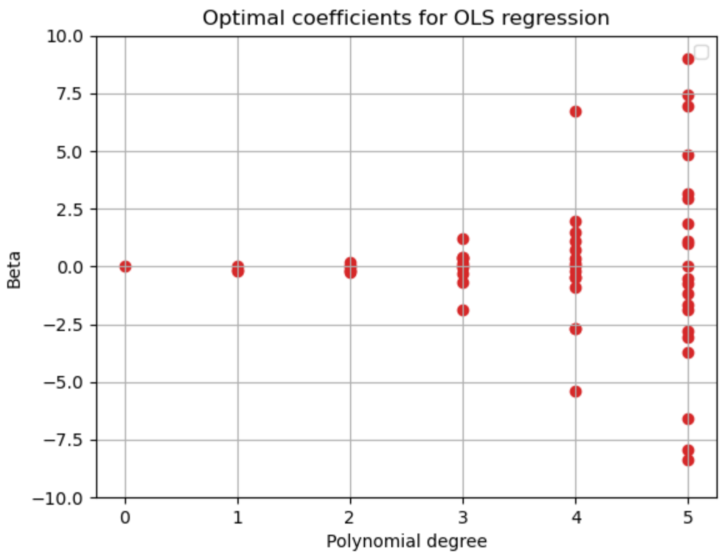


Figure 9: The optimal coefficients for the OLS regression.

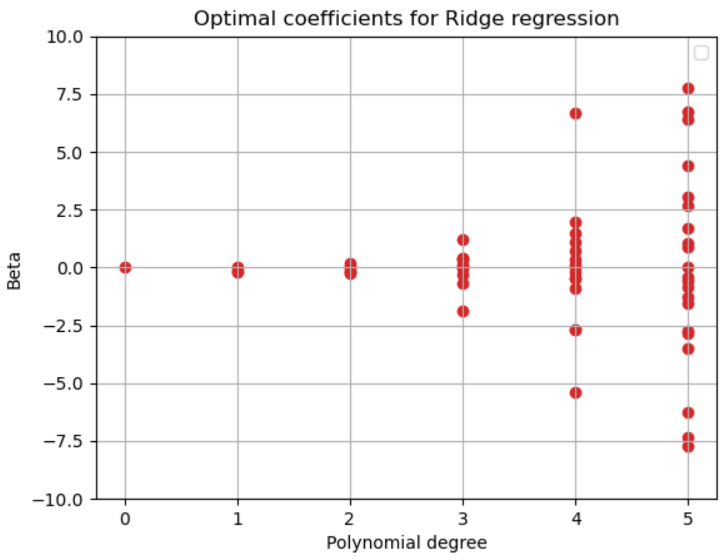


Figure 10: The optimal coefficients for the Ridge regression with $\lambda = 1e - 4$.

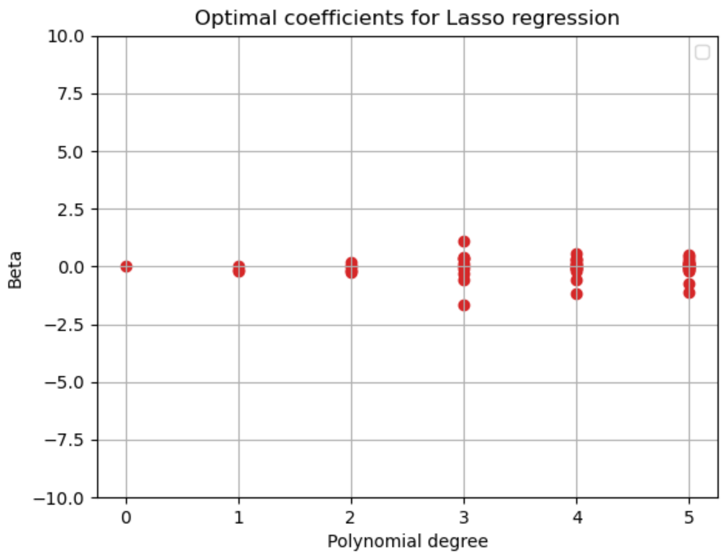


Figure 11: The bias-variance trade-off for Lasso regression with $\lambda = 1e - 4$.

Using the optimal coefficients from figures 9 to 11 and a polynomial of fifth-degree, the solutions resulting from all regression models were plotted in a 2D plot in figure 12 and as a 3D surface in figure 13. New data was created for the plots, to avoid that potential overfitting would be camouflaged by the fact that the model was trained on the identical data points. The upper left panel is the Franke's function with added noise of $\sigma^2 = 0.05$ and the other three plots are the surfaces resulting from the OLS, Ridge and Lasso regression models, with $\lambda = 1e - 4$ used for the two latter. All models seem to catch the large features of the data but Ridge and Lasso have a somewhat flatter appearance than the OLS.

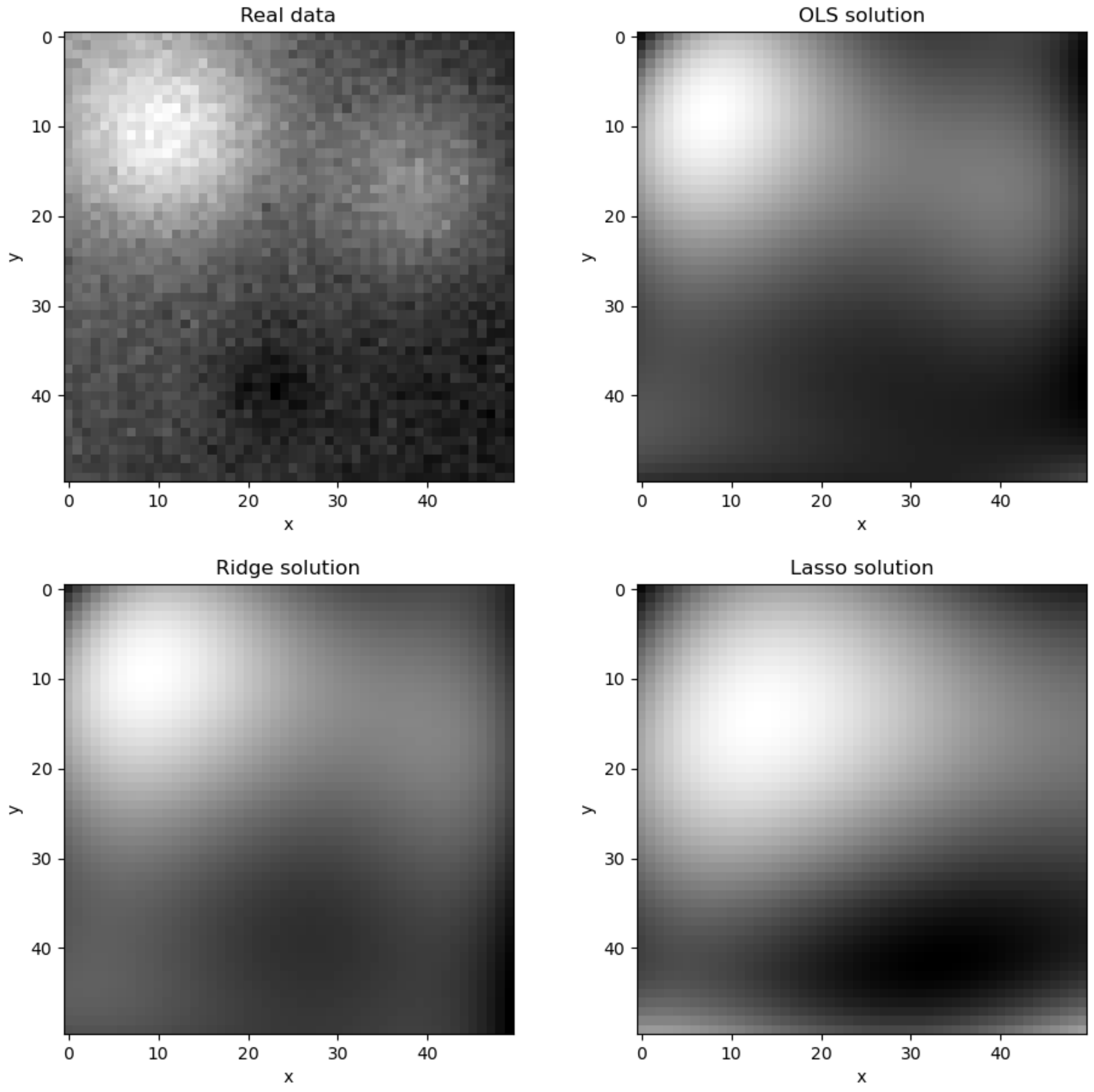


Figure 12: Franke's function parameterised using Ordinary Least Squares, Ridge and Lasso regression. The two later were fit with $\lambda = 10^{-4}$.

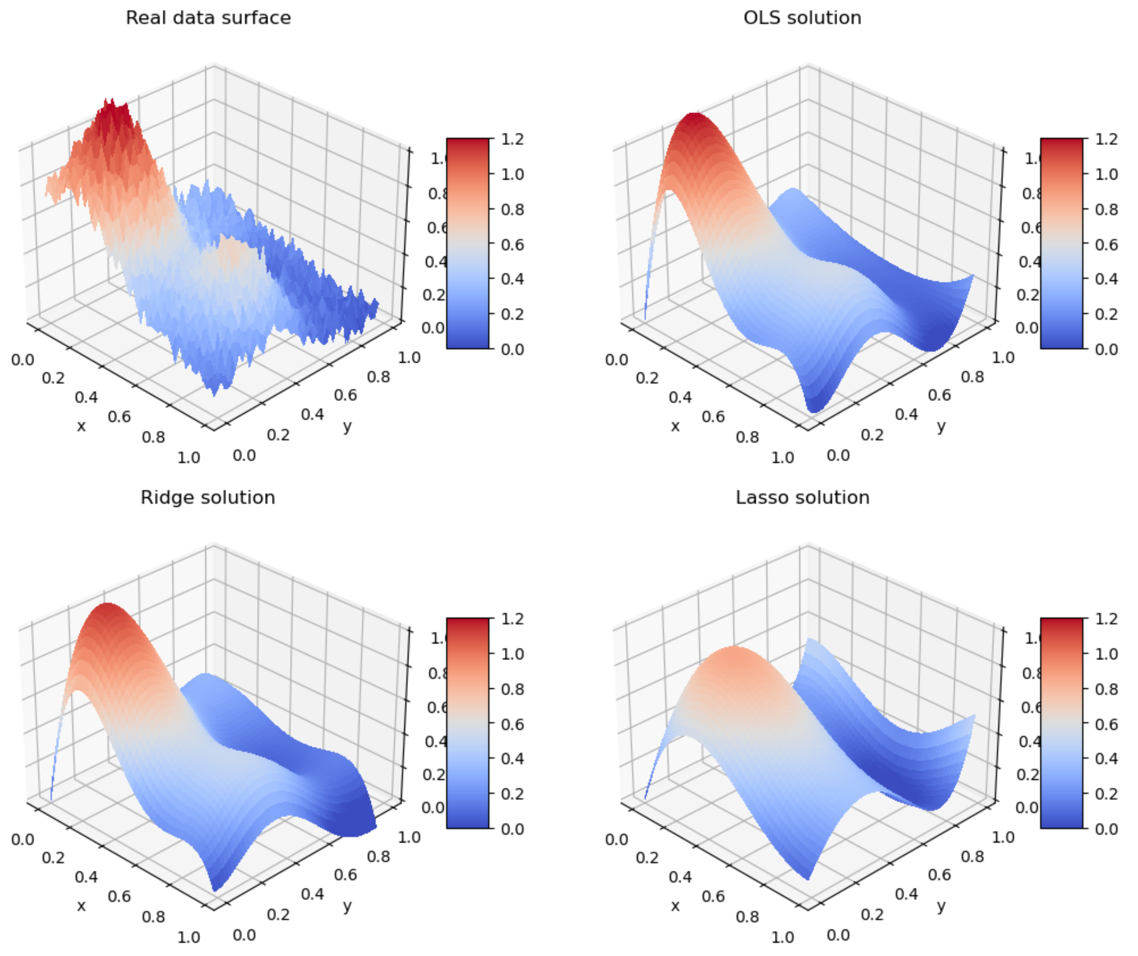


Figure 13: The original data with added noise of $\sigma^2 = 0.05$ and the corresponding OLS, Ridge and Lasso regression fits.

4.2 Modelling of topographic data

The topographic data was parameterised using the same codes as the ones developed and tested on the Franke's function. The optimal polynomial degree and hyperparameter λ were found by performing a search over a range of degrees and λ values. The models were evaluated on a separate test data set.

The terrain data was fitted with polynomials up to 20th degree. Figure 14 shows the MSE and the r^2 score of the OLS, Ridge and Lasso regression models. All three models performed similarly until third-degree polynomials, whereupon the Lasso error started to flatten while the OLS and Ridge error continued to decrease. At the seventh-degree polynomial, the OLS started to outperform the Ridge regression as well, rendering it the optimal regression method of the three. From these plots, it can be seen that the OLS at the 20th-degree polynomial performed the best.

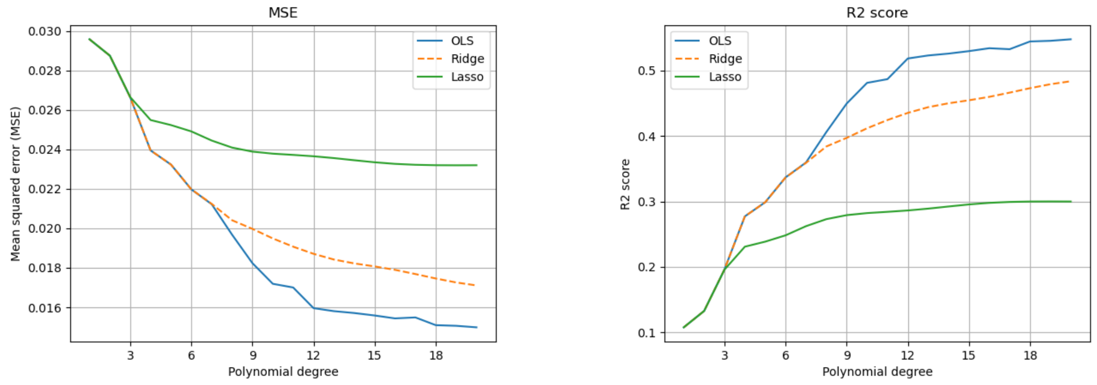


Figure 14: MSE and r^2 Score for all regression models.

However, as the complexity increased, the β coefficients of the OLS regression continued to grow considerably, reaching levels of $\beta = \pm 10^4$ at the 20th-degree polynomial. Such high coefficients could challenge the stability of the model, as small changes in the input data can generate large change in the predicted values. Increasing the complexity should also eventually lead to overfitting, as the model would start to capture noise rather than the underlying pattern. However, for this data set, the bias-variance trade-off show a negligible contribution of the variance to the total error, as can be seen in figures 15 to 17. The bias-variance trade-off was studied for all three regression methods and the squared bias completely dominated the error contribution, indicating that the models were far from overfitting. A similar obeservation was made when plotting the MSE of the test and train data up to 40th-degree polynomials, as can be seen in figure 18. Both errors were on a declining slope, even at high complexities. For the error and the bias-variance trade-off analysis, the regularisation hyperparameter was set to $\lambda = 10^{-6}$ since this obtained the lowest MSE (see figure 22).

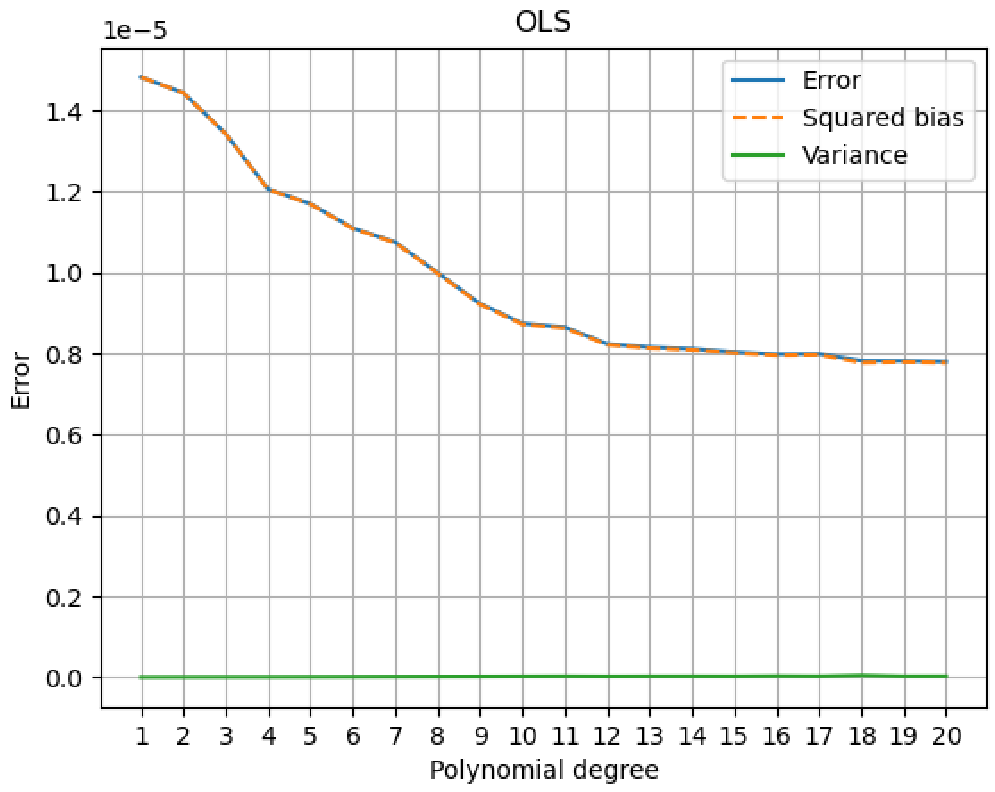


Figure 15: The bias-variance trade-off for OLS regression.

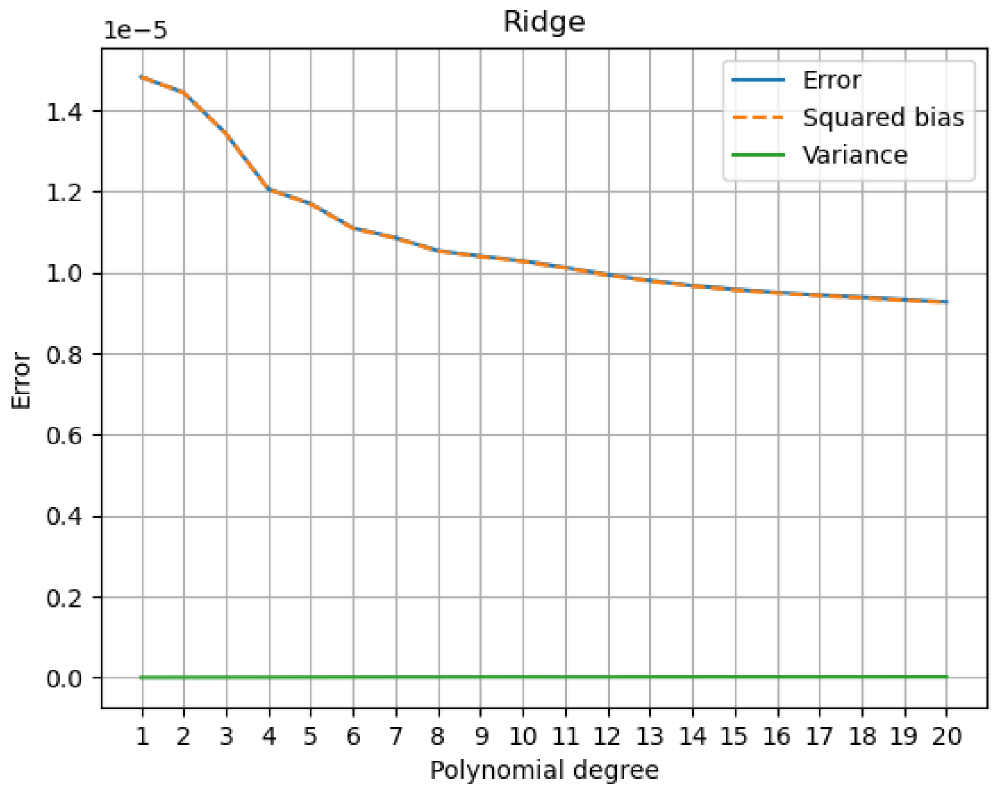


Figure 16: The bias-variance trade-off for Ridge regression with $\lambda = 10^{-6}$.

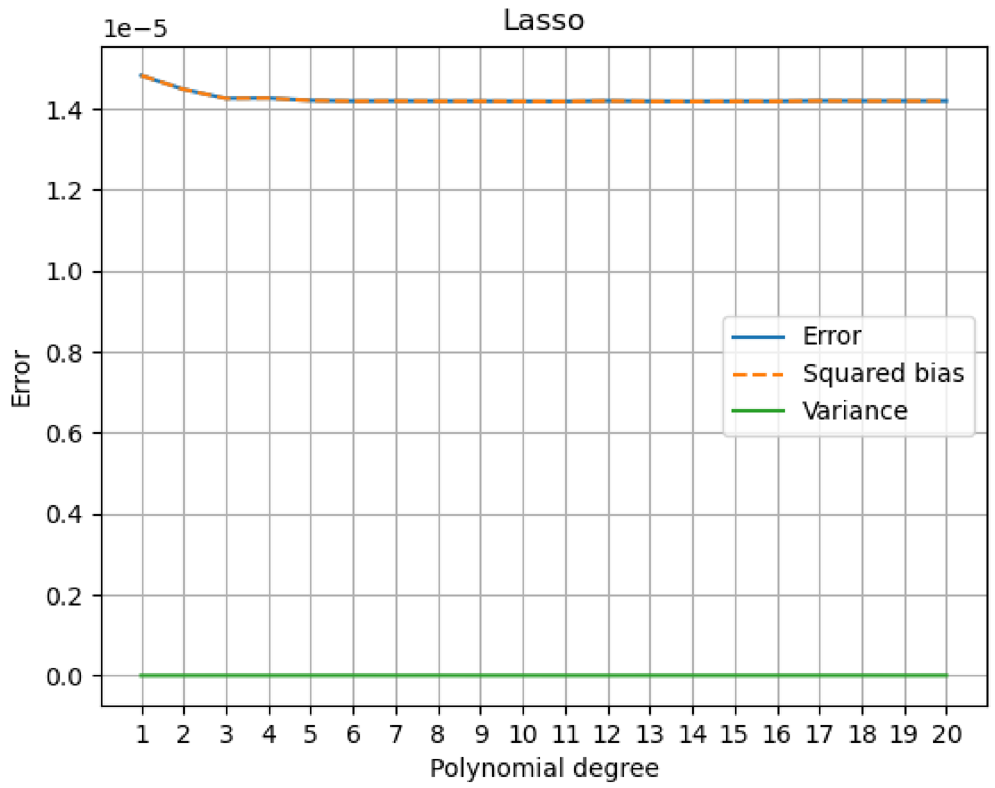


Figure 17: The bias-variance trade-off for Lasso regression with $\lambda = 10^{-6}$.

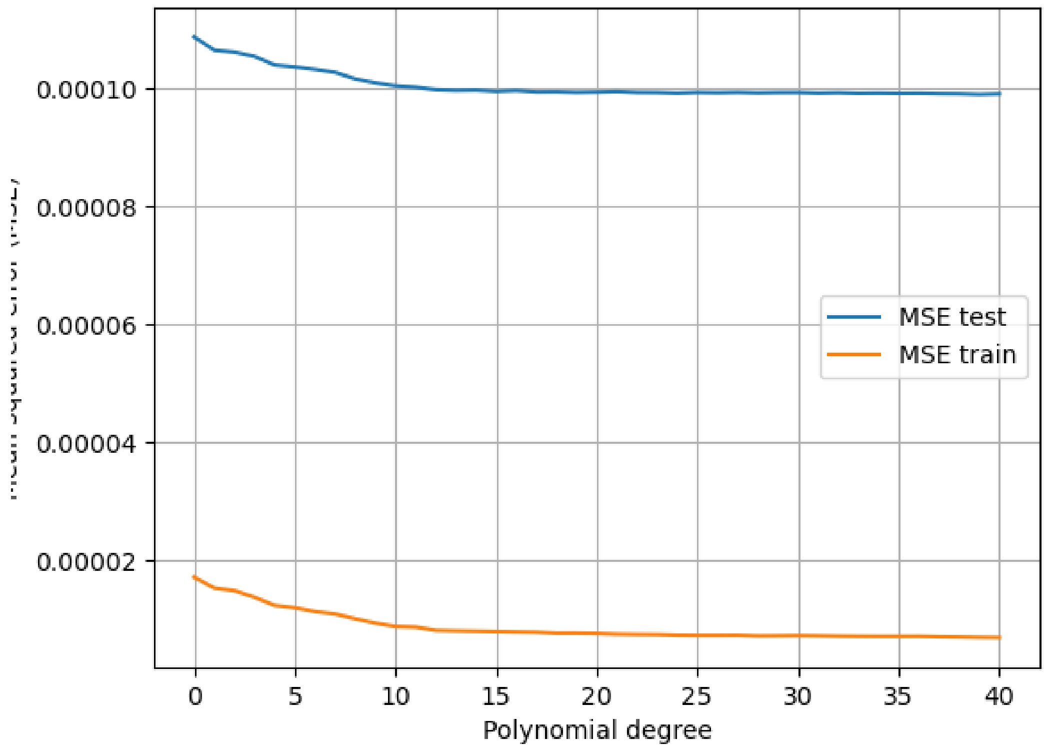


Figure 18: The MSE of the test and train datasets of the terrain data. The test MSE shows no sign of increasing, even at 40th-degree polynomials.

The figures 19 to 21 show the optimal coefficients for the three regression models up to polynomials of 20th-degree. While the Ridge and Lasso coefficients are controlled by λ, the OLS coefficients are growing exponentially and are out of any reasonable range at higher-degree polynomials.

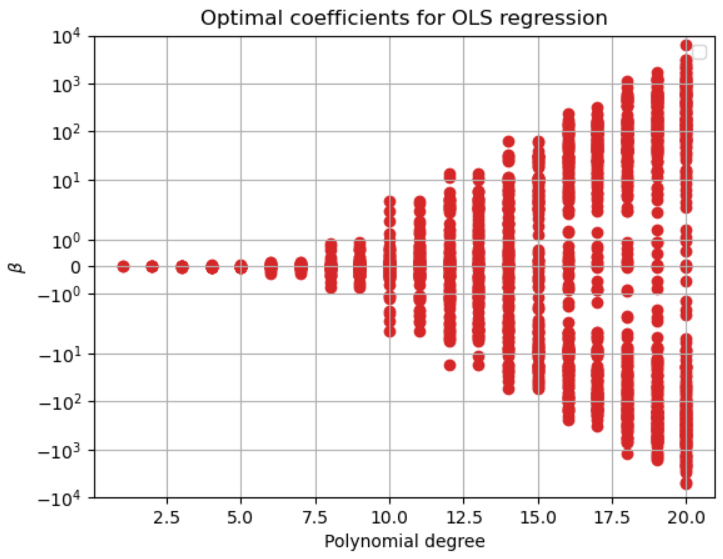


Figure 19: The optimal coefficients for the OLS regression.

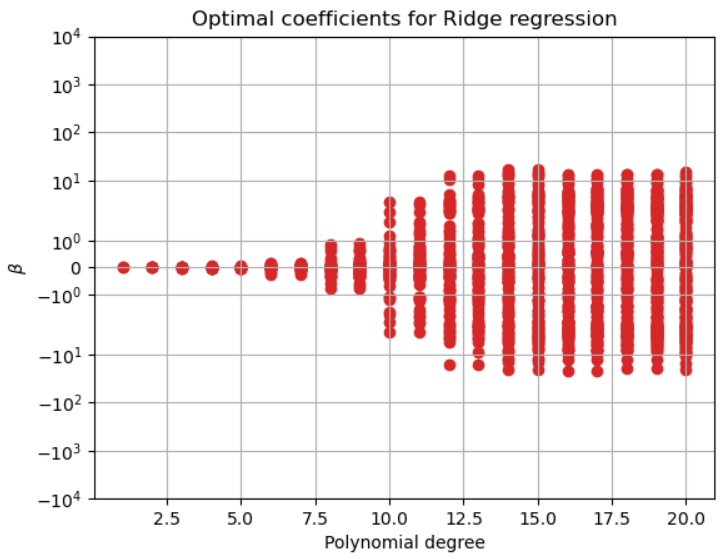


Figure 20: The optimal coefficients for the Ridge regression with $\lambda = 1e - 6$.

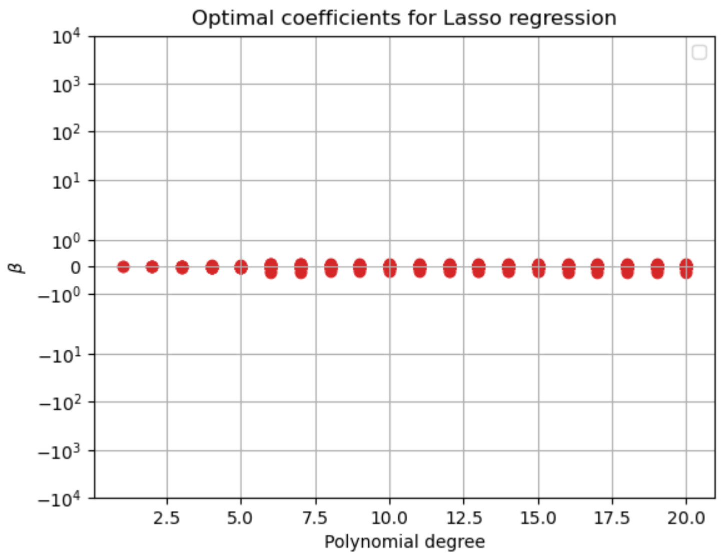


Figure 21: The optimal coefficients for the Lasso regression with $\lambda = 1e - 6$.

The Bootstrap and the Kfold cross-validation techniques were used as resampling techniques when screening for the optimal Ridge and Lasso regularisation parameters. For the Bootstrap analysis, 30 bootstrap datasets were created. As can be seen from figure 22, the error was more or less constant up to approximately $\lambda < 10^{-6}$ for Lasso regression and $\lambda < 10^{-4}$ for Ridge regression. As a comparison, the mean squared error for OLS with the same set-up was 0.7e-5, meaning that again the OLS regression outperformed both Ridge and Lasso regression. In this plot, a tenth-degree polynomial was used. The Kfold cross-validation returned a slightly higher error than the Bootstrap method, which could be due to the upward bias that can occur when the data is divided in too few folds [2].

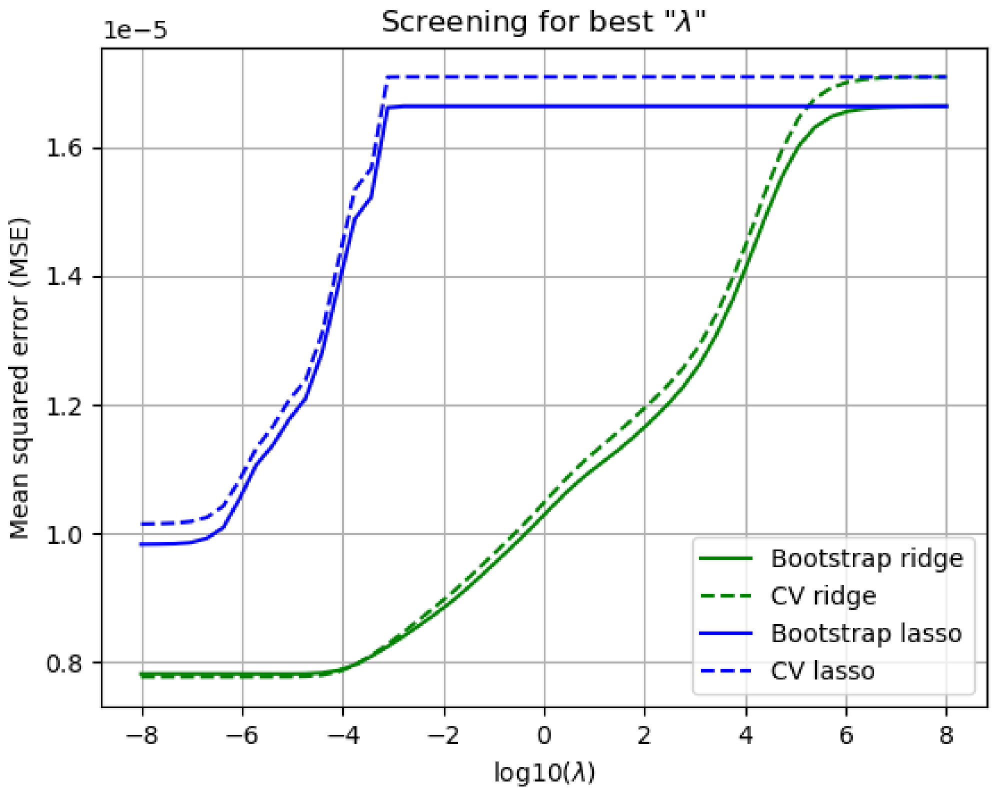


Figure 22: Cross-validation and bootstrap with different λ.

Finally, all three regression models were used to model the terrain data. Figure 23 shows the 2D plots all models using a polynomial of 20th-degree and figure 24 shows the corresponding 3D surfaces. As a reference, a plot of the original data is included in the upper left panel in both figures. The OLS model is clearly mapping the terrain with more detail than the two other methods.

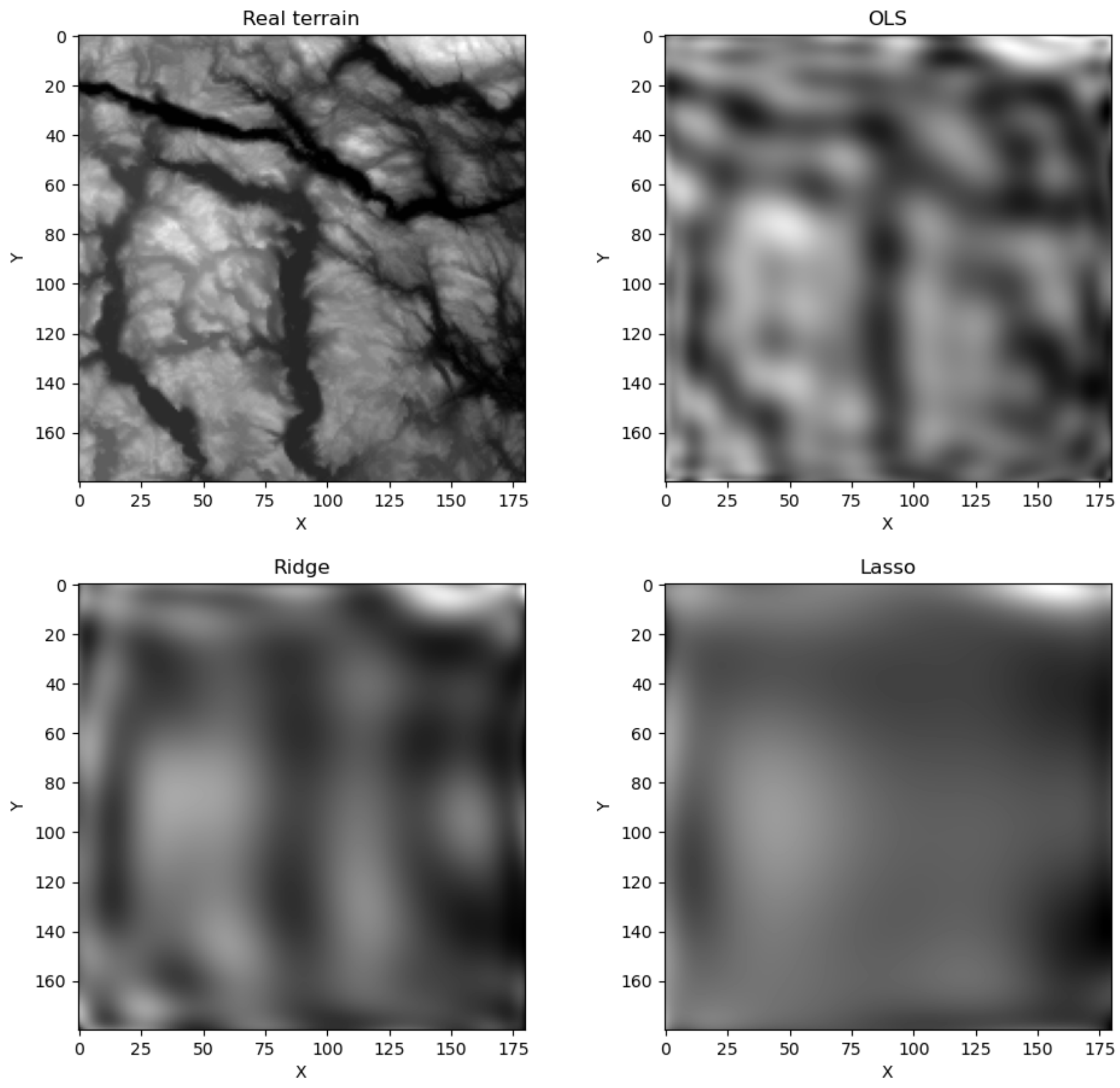


Figure 23: The OLS, Ridge and Lasso solutions when modelling the terrain data.

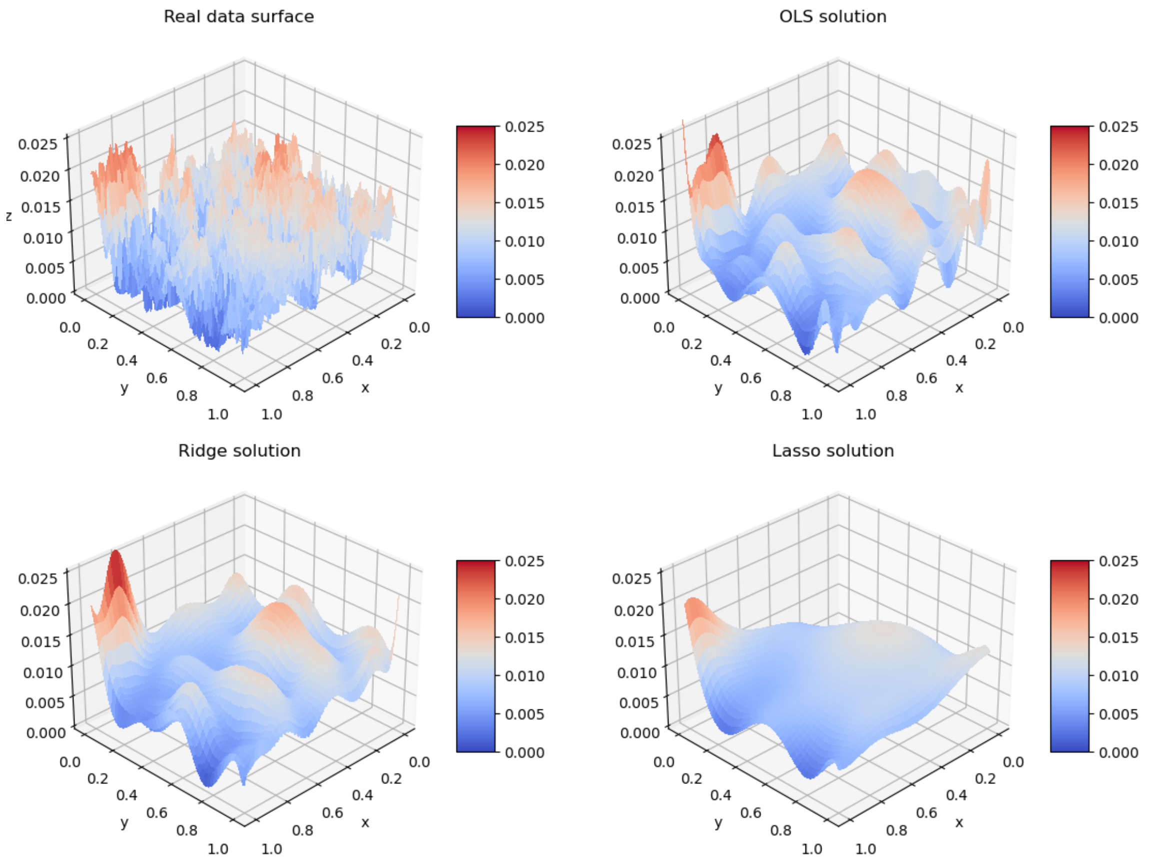


Figure 24: The real terrain data plotted as a 3D surface along with the surfaces produced with OLS, Ridge and Lasso regression. A 20th-degree polynomial was used with $\lambda = 10^{-6}$ for both Ridge and Lasso regression models.

5 Discussion

There was a stark contrast between the exploding coefficients of the OLS model and the bias-variance trade-off plots showing negligible variance. Even at high complexity, the variance was found to be almost non-existing for all three regression methods, while we would expect the variance to increase with complexity, and certainly when the coefficients are allowed to grow without restrictions as in the case of the OLS solution [2]. However, considering the roughness of the topography, it is possible that the small variance is truly a feature of the data, meaning that the squared bias would completely rule out the variance of the model due to the high complexity needed to approach the model $\tilde{\boldsymbol{y}}$ to the true data $\boldsymbol{y}$. Since the terrain data has strong variation in its elevation, it is reasonable to assume that a high complexity model is needed to decrease the bias.

When evaluating the MSE, the Kfold cross- validation resampling method was tested with five to ten folds. At such a low number of folds, this estimation method could have substantial bias, depending on the size of the training set [2]. To limit the computing time, the Bootstrap analysis was performed with only 30 Bootstrap samples, which could also lead to inaccurate estimates.

The Lasso regression was implemented using the Scikit-learn functionality. However, the Lasso function would often return the following warning: *"ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation."* Several adjustments were done to avoid the warning, such as increasing the tolerance and and the maximum number of iteration but the error message was still returned. Since the model was used to screen over all parameters, including some that were not optimal, it may be expected that the Lasso function fails to converge while keeping to a reasonable number of iterations. We did not investigate to what extend this could have underestimated the performance of the Lasso regression.

6 Conclusion

In this project, codes for OLS, Ridge and Lasso regression were developed to map terrain data using polynomials of different degrees. A GeoTIF file covering a surface of one degree latitude and longitude from Western Norway was used as input data. The developed codes were also tested on the Franke's function prior to the terrain data analysis.

For both the terrain data and the Franke's function, the OLS outperformed the Lasso and Ridge regression at higher polynomial degrees. The 2D and 3D plots revealed that the OLS and Ridge models can capture some of the intermediate sized features in the topographic data, such as fjord arms and mountain peeks, whereas the Lasso regression smoothened the surface such that only large-scale features were recognisable. However, when increasing the complexity of the model, the β coefficients of the OLS model increased to levels that can lead to instability. If the goal is to capture fine details of the topography, the regression models used in this project meet their limitation as their coefficients either grow out of reasonable range or become too penalised to follow the data in detail. However, if the interest is to capture large-scale features, the OLS regression model at lower complexity or the Ridge and Lasso at higher complexity but with an appropriate penalisation could be good choices.

References

[1] Giuseppe Amatulli et al. “A suite of global, cross-scale topographic variables for environmental and biodiversity modeling”. In: *Scientific Data* 5.1 (Mar. 2018), p. 180040. ISSN: 2052-4463. DOI: `10.1038/sdata.2018.40`. URL: `https://doi.org/10.1038/sdata.2018.40`.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction.* 2nd ed. Springer, 2009. URL: `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.

[3] Morten Hjorth-Jensen. *Lecture notes in Applied Data Analysis and Machine Learning.* Sept. 2023.

[4] U.S. Geological Survey. *EarthExplorer.* 2023. URL: `https://earthexplorer.usgs.gov/` (visited on 09/25/2023).

[5] Wessel N. van Wieringen. *Lecture notes on ridge regression.* 2023. arXiv: `1509.09169 [stat.ME]`.

List of developed codes:

- utils.py
- franke_plots.py
- terrain_plots.py
- solution_plots.py
- scikit_comparison.py

All codes can be found at https://github.com/friedawik/FYS-STK4155-