Project 1. FYS-STK4155 Data Analysis and Machine Learning

Kristian Joten Andersen

October 7, 2019

1 Introduction

Linear regression is a mathematical tool widely used in data analysis. It gives us the possibility of fitting models to data and test their predictive value.

In this report we will take a look at three common regression methods, namely ordinary least squares, Ridge and Lasso regressiong. We will use these methods in order to fit 2-dimensional polynomials of n'th order to a toy model terrain data set with artificially added Gaussian noise and compare their predictive powers. We will then move on to real terrain data and attempt to model this data using the regression methods.

First we will present the theory and algorithms needed in order to perform the linear regression, as well as the toy model terrain data we use. Then we will perform rigorous testing of the regression methods on the toy model in order to encapture all the aspects of the regression methods and how they depend on certain model and method parameters. Finally we will present true terrain data and perform linear regression fit on that data in an attempt to model it. Due to the extrem number of figures compared to text, all figures have been added after the appendix for increased readability

2 Theory and algorithms

In general when one takes measurements with an instrument there is always some error in the resulting measured values compared to the true values. If we define the measurement value as y, the error as $\varepsilon$and the true value f, then for sample i one has

$$y_i = f(x_i) + \varepsilon_i, \tag{1}$$

where x_i denotes the position where the measurement was made. $f(x)$ is a graph if x is 1-dimensional, or a surface if x is 2-dimensional, i.e. $\mathbf{x} = (x_1, x_2)$. Let us now define a model function $g(x)$. If the noise has zero mean, i.e. $< \varepsilon >= 0$, then

$$\tilde{y}_j = g(x_j) \tag{2}$$

is an estimate of a measurenet done at position x_j.

Let us assume $g(x)$ is a polynomial of order p, then

$$g(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p, \tag{3}$$

where β_i are coefficients than need to be fitted. If the position x is 2-dimensional, then

$$\begin{aligned} g(\mathbf{x}) &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_1 x_2 + \beta_5 x_2^2 + \cdots + \beta_m x_2^p \\ &= \sum_{i=0}^{n} g_i(\mathbf{x}) \end{aligned} \tag{4}$$

where

$$g_i(x, y) = \beta_m x^i + \beta_{m+1} x^{i-1} y + \beta_{m+2} x^{i-2} y^2 + \cdots + \beta_{m+i-1} x y^{i-1} + \beta_{m+i} y^i. \tag{5}$$

Here, m is an index which starts at the value given by the number of β-parameters in a polinomial of one order less than i. For a 1-dimensional polynom the number of parameters are $n_\beta = p + 1$ for a polynomial of order p. For a 2-dimensional polynomial the number of parameters is given by

$$n_\beta(p) = \frac{(p+1)(p+2)}{2}. \tag{6}$$

If we now set $\mathbf{x}$to be all points at which we sample a surface $f(x_1, x_2)$, where $\mathbf{x}_i = (x_{1,i}, x_{2,i})$, then we may write

$$\tilde{\mathbf{y}} = g(\mathbf{x}). \tag{7}$$

We further define $\boldsymbol{\beta} = [\beta_0, b_1, \ldots, \beta_n]$ and $\mathbf{X}$ as a $N \times n$ matrix, where N is the number of measurements, and row i of $\mathbf{X}$ is

$$\mathbf{X}_i = \begin{bmatrix} 1 & x_{1,i} & x_{2,i} & x_{1,i}^2 & x_{1,i}x_{2,i} & x_{2,i}^2 & \ldots & x_{2,i}^p \end{bmatrix}. \tag{8}$$

This way we can rewrite $\tilde{\mathbf{y}}$ as

$$\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} \tag{9}$$

2.1 Regression methods

Every regression method that aims to fit a function to a data set aims to minimize a cost function C. If the measured data set is expected to follow eq. (1) and the error ε_i is assumed to follow a normal distributoion with zero mean and variance σ^2, i.e. ε_i $\mathcal{N}(0, \sigma^2)$, then ordinary least squares (OLS) regression gives the highest likelyhood for the parameters in the fitting function. The cost function of OLS is given by

$$C(\mathbf{X}, \boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2 = \mathbb{E}\left[(\mathbf{y} - \tilde{\mathbf{y}})^2\right], \tag{10}$$

which is known as the mean square error (MSE). The best-fit-parameters are defined as the parameters which minimizes the cost function. In Hjorth-Jensen (2019b) it is shown that

$$\frac{\partial C(\mathbf{X}, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0 = \mathbf{X}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}), \tag{11}$$

which we rewrite to

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\boldsymbol{\beta}. \tag{12}$$

This can further be rewritten to

$$\boldsymbol{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{13}$$

which will have give us the OLS solution to $\boldsymbol{\beta}$ as long as $\mathbf{X}^T\mathbf{X}$ is not singular, i.e. it is invertable.

Tanking a second look at the cost function we see that

$$\mathbb{E}\left[(\mathbf{y}-\tilde{\mathbf{y}})^2\right] = \mathbb{E}\left[(\mathbf{f}+\boldsymbol{\varepsilon}-\tilde{\mathbf{y}})^2\right] \tag{14}$$

$$= \frac{1}{n}\sum_i \mathbb{E}\left[(f_i+\varepsilon_i-\tilde{y}_i)^2\right] \tag{15}$$

and adding and subtracting $\mathbb{E}[\tilde{\mathbf{y}}]$ gives

$$\mathbb{E}\left[(\mathbf{y}-\tilde{\mathbf{y}})^2\right] = \frac{1}{n}\sum_i \mathbb{E}\left[\{(f_i-\mathbb{E}[\tilde{\mathbf{y}}])-(\tilde{y}_i-\mathbb{E}[\tilde{\mathbf{y}}])+\varepsilon_i\}^2\right] \tag{16}$$

$$\mathbb{E}\left[(\mathbf{y}-\tilde{\mathbf{y}})^2\right] = \frac{1}{n}\sum_i \mathbb{E}\left[(f_i-\mathbb{E}[\tilde{\mathbf{y}}])^2\right] + \frac{1}{n}\sum_i \mathbb{E}\left[(\tilde{y}_i-\mathbb{E}[\tilde{\mathbf{y}}])^2\right] + \sigma^2 + h(\mathbf{y},\tilde{\mathbf{y}}), \tag{17}$$

where

$$h(\mathbf{y},\tilde{\mathbf{y}}) = \frac{1}{n}\sum_i \mathbb{E}\left[2\varepsilon_i\{(f_i-\mathbb{E}[\tilde{\mathbf{y}}])-(\tilde{y}_i-\mathbb{E}[\tilde{\mathbf{y}}])\}-2(f_i-\mathbb{E}[\tilde{\mathbf{y}}])(\mathbf{y}_i-\mathbb{E}[\tilde{\mathbf{y}}])\right]. \tag{18}$$

f is not a stochastics variable, idem for $\tilde{\mathbf{y}}$. If we have $\mathbb{E}[\mathbf{f}] = \mathbb{E}[\tilde{\mathbf{y}}]$ then eq. (18) is reduced to zero and the last term in eq. (17) vanishes. We then recognize the first term in eq. (17) as the bias, the second trem as the model variance and the third term as the variance of the noise. The bias can be viewed as the error one does when the model is to simple to predict the data, while the model variance is the result of fitting noise. both these effects will be shown in sec. 3

In some cases, if the number of free parameters if large compared to the number of data points, it is possible for OLS regression to fit the noise. This makes it so that the models predictive value gets worse, as will be discussed later. A tool to reduce the likelyhood of sampling noise is to limit the values one allows the fitting parameters to take. Ridge regression has a slightly different cost function than OLS and is given by (Hjorth-Jensen 2019b)

$$C^{\text{Ridge}}(\mathbf{X},\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i-\tilde{y}_i)^2 + \sum_{i=0}^{n_\beta-1}\lambda\beta_i^2, \tag{19}$$

where λ is a parameter > 0. On vector formwe get

$$C^{\text{Ridge}}(\mathbf{X},\boldsymbol{\beta}) = (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}, \tag{20}$$

to which the solution that minimizes C^{Ridge} is

$$\boldsymbol{\beta} = \left(\mathbf{X}^T\mathbf{X} - \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{21}$$

Another regression method is known as Lasso regression where the cost function, like Ridge regression, is slightly different from OLS and given by (Hjorth-Jensen 2019b)

$$C^{\text{Lasso}}(\mathbf{X},\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}}. \tag{22}$$

There is no straight forward solution to the best fit parameter values of Lasso regression and so a iterative solution method is required. We won't introduce any such method here but rather use the algorithm of the *scikit-learn* library in Python.

We can show that the variance of the fitted parameters for OLS and ridge regression are given by the following equations (see Hjorth-Jensen 2019b)

$$\text{Var}\left[\boldsymbol{\beta}^{\mathbf{OLS}}\right] = \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1} \tag{23}$$

$$\text{Var}\left[\boldsymbol{\beta}^{\text{Ridge}}\right] = \sigma^2\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{X}\left\{\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\right\}^T, \tag{24}$$

where the variance of a parameter is given by the diagonal element, i.e.
$\beta_i = \text{Var}[\boldsymbol{\beta}]_{ii}$.

2.2 Cross validation

In order to se how well a regression method and model fit the given data set it is important to perform cross validation (CV) tests. This means that one need to split the data into training data and test data, and then fit the model to the trainging data. After the fit we then perform a goodness-of-fit test to both the training data and the test data, and the goal is to see which model and method that gives the best fit for both training and test data.

There are many ways of performing the CV test. Three common ways are the following: First and most simple is to split data into training and test data with 1/3 to 1/5 of the data as test data, then fitting and performing the goodnes-of-fit. The second test is known as a k-fold test, where the data is divided into k equal sized groups and then each group takes turn akting as the test data while the rest is treated as the training data. The last test is to do a k-fold CV where $k = N$, i.e. the number of data points. This is known as the leave-one-out cross validation (LOOCV), though it might take a long time for large data sets.

2.3 Problem at hand

The problem at hand is to see how the 3 regression methods described in sec. 2.1 perform on a predefined function, given a model that does not fit the true data. The function from which we draw our data is the Franke function defined as

$$\begin{aligned} f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) \\ &+ \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ &+ \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) \\ &- \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned} \tag{25}$$

which is defined on the region $x, y \in [0, 1]$. Our fitting model will be a 2-dimensional polynomial as defined in eq. (4), where $x_1 = x$ and $x_2 = y$. We will primarily perform k-fold CV in order to see how the different regression methods and models perform on data drawn from eq. (25) where we will add randomly generated Gaussian noise (white noise) with zero mean and variance of $\sigma^2 = 0.01$ to make the data more realistic. We will see how our choice of λ, k, p, and the number of data points contribute to the various fits.

Finally we will take a terrain image of Norway and try to implement our regression methods in order to best fit a model to the given terrain data.

3 Results and discussion

As our measure of the goodness-of-fit we choose to use the mean square error (MSE) as defined by the cost function of OLS regression, eq. (10). Furthermore, as argued in sec. 2.2, we choose to use k-fold CV in order to have a more robust estimate, since one split might coincide with getting large outliers in either the test or training data, and LOOCV will take too long to compute. Therefore the first question arise as to what value of k to use? literature says that a value of 3 to 5 is Ideal, but we chose to check this. Therefore, we sampled the Franke function in eq. (25) with a 21×21 equidistant grid forom 0.0 to 1.0 in each direction. Then we added Gaussian noise with zero mean and variance of 0.01, as described in sec. 2.3. The sampled Franke function, both with and without noise, can be seen in fig. 1.

The model chosen was a 5th order polynomial and the method chosen to test the k-value was OLS and Ridge regression, the last one with a λ-value of $1.0 \cdot 10^{-3}$. Then, for each k-value between 2 and 9 we performed 10 random splits of the noisy data set and for each split we fitted the model to the training data and calculated the MSE value of both the test and training data. The results are shown in fig. 2. We see from the figure that the MSE value of the test data for both methods are falling between $k = 2$ to $k = 5$. additionally, the MSE of the training data is slightly increasing in the same domain. This is likely due to the fact that there are fewer data points to fit and therefor

one is more likely to fit some of the noise. Additionally, if the goups are large (e.g. $k = 2$), there is a chance that some structure might be under sampled and therefore gets less weighted in the fit. Another feature that is visible is that the standard deviation of the test data is increasing from $k = 4$, likely due to less data points in the test group, meaning that if there are many large noise values in some groups and fewer in other groups, this might lead to a larger spread in the MSE values. From the data we choose to use $k = 4$ moving forward as it seems to reduce both the total MSE and its overall variance.

An important indicator of what method and model to use in order to fit a given data set is the complexity tradeoff. To check this we performed a 4-fold split with on our sampled data and performed a 4-fold CV test for all regression methods (with different values of λ for Ridge and Lasso) while varying the complexity, i.e. polynomial order, of our model. The results can be seen in figs. 3–6. It is evident from the figures that the model performs porely for all regression methods when complexity is small. The reason for this is that the data is underfitted, i.e. one doesn't capture all the structure of the true data, in other words we have a large model bias. On the contrary, as is evident for the OLS regression, to high complexity results in overfitting the data, i.e. fitting the noise, resulting in a rise in the MSE value of the test data. The size of the error bars in figs. 3 and 4 is as expected if compared to fig. 2, as the test data experience a larger variance in MSE value.

In general it might look like the higher the complexity the better the fit for Ridge and Lasso regression, as the λ-parameter supresses the large β-values commonly associated with fitting noise. This is not the case in fig 6 where we only fit on a 10×10 data grid, thus having less data points in the training data. We se that for the smallest values of λ the MSE of the test data is rising at higher complexity, thus being a sign that the methods are also fitting noise and we have high model variance. Another feature from all the complexity figures is that a large λ-value supresses the β-values too much, and thus one is underfitting the data all together. This is a proof that for any model and dateset, if one chooses to use Ridge or Lasso regression, one has to tune the λ-value in order to optimize the fit. The last thing that is evident from all the figures is that the test data MSE-value, i.e. the variance of the test data, is always larger than the noise variance, which is what we would expect.

Coninuing with our analysis, considering the trend as seen in fig. 6, it is of interest to see how the MSE change with the number of data points. Using a 4-fold CV on a 5th order polynomial we checked this for all regression methods, including different values of λ for Ridge and Lasso. The results are shown in fig. 7 where we have omitted the standard deviation for readability. We see that for most of the (smaller) λ-values the smallest MSE values for the test function, and the smallest difference between training and test data happens at larger number of grid points. This is as expected as one fits less noise when one has more data because the distance between grid points reduces and higher complexity (i.e. larger gradients) is needed to sample the noise.

Lastly in our analysis of the regression methods on the Franke function we take a look at the dependency of λ. At this point it is quite clear that so far the Lasso regression needs a smaller λ-value. Fitting 1st to 7th order polynomials with a 4-fold CV with varying λ-values we mapped the Ridge and Lasso regression methods' dependency on λ. The results are shown in figs. 8 and 9. In fig. 8 we see a turning point for the test

data MSE for the smaller λ-values in the case of a 10×10 data grid. This is not seen in the 21×21 data grid case nor for Lasso regression at all. It is also clear that for large λ-values, Lasso regression reduces to what looks like a 0th order polymnomial fit. Furthermore, we see that for a given complexity the fit stops improving under a certain λ-value. This is because the β-values are no longer being supressed by the λ.

To get a more visual feeling of how the lower order polynomial fits look like, the OLS regression fits to the 21×21 grid of the Franke function in fig. 1 is plotted in figs. 10−12 for polynomial model functions of 0th to 5th order. It is easy to see that the fit goes from a mean to a lane to a parabola and then into more complex shapes. After that, in fig. 13−15 both Ridge and Lasso rgression have been used modelling a 5th order polynomial to the Franke function, wtih 3 different values of λ. We see from fig. 13 what we expected to be the case for Lasso regression from fig. 9, that large λ-values is fitting a constant value to the data set. Furthermore, we see that with decreasing λ the fitted functions gaing more features and more accurately represents the Franke function.

In figs. 16−18 we have plotted the best fit β-values for a 5th order polynomial for all three regression methods, Ridge and Lasso with $\lambda = 1.0 \cdot 10^{-4}$ and $\lambda = 1.0 \cdot 10^{-6}$ respectively. For OLS and Ridge the 95% confidence interval is also plotted, calculated from the equations for the variance of $\boldsymbol{\beta}$, see eqs. (23) and (24). We see from the figures that the shape of the plots are quite similar, but the overall β-values are smaller for Ridge and (especially) Lasso regression. This is more clearly seen in fig. 19 where the β-values are overplotted for OLS and Ridge/Lasso for several λ-values. We see that for small λ-values, Ridge regression is close to identical with OLS, while Lasso regression never really reaches the same values as OLS. This last case might be due to the fact that Lasso regression has an iterative method, and that the values have not completely converged. A soludion to this would be to increase the number of iterations for the Lasso regression, but this would on lead to much longer runtime for the fit and is only recomended if absolutely necessary.

We also tested the equations for the variance of $\boldsymbol{\beta}$, eqs. (23) and (24). This we did by a 4-fold CV on 100 different noise realisations, with different splits for different noise realizations. The results for β_7 for OLS and Ridge ($\lambda = 1.0 \cdot 10^{-4}$) regression for all splits, together with their mean values, and calculated and estimated standard deviations are shown in fig. 20. As we see from the figure, the equation for calculating the variance of $\boldsymbol{\beta}$ are performing quite well.

Now we are ready to look at some real terrain data. WE experienced some problems downloading the terrain data from https://earthexplorer.usgs.gov/ so we used the terrain file provided in Hjorth-Jensen (2019a). The terrain file is an image of a part of Norway. In order to make it possible to process with our regression code we first rescaled it. First we rescaled the range of the image so thet it took values between 0 and 1, then we reduced the number of pixels in each axis direction to 1/50 of the original, i.e. each pixel in the reduced image is the average of 50×50 pixels in the original image. Both the original and reduced image can be seen in fig. 21.

In order to fit the rescaled image with polynomials of a given order we further defined the pixels' x and y values to have a maximum range of 1, i.e. since y direction was the largest, pixel 0 was set to 0.0 and the largest pixel number was set to 1.0 with all pixels in between scaled by the inverse of the largest pixel number. The pixels in

x-direction was scaled with the same factor to keep the relative shape of the terrain. After that, by choise only, the terrain was centered on (0,0) before we started fitting the terrain with 4-fold CV for initially OLS and Ridge regression. In figure 22 we see the average MSE values for both training and test data for the terrain fits of different polynolial orders. We see that OLS performs the best of these, but stops imporving after polynomials of order 25. At this point one has 351 parameters to fit, while for the 45th order polynomials one has 1081. Due to the choise of x and y values, these high order betas will have very little say except when both (x, y) is close to $(1, 1)$ which in the case of our chosen coordinates doesn't happen. The best fit OLS of a polynomial of 30th order is plotted together with the reduced image in fig. 23. It is clear that the regression method has underfitted the image as it is much smoother than the reduced image.

There is the possibility that our choise of coordinates for the pixels in the image are causing the smoothing. To check this we use a new scaling of the x- and y-values of the pixels so that maximum extension is 2.0 in y-direction and we no longer center on (0,0). The following MSE results are shown in fig. 24. From the figure it looks like the best fit is for $p = 15$ and $\lambda = 1.0 \cdot 10^{-8}$. In figure 25 this fit is plotted against the rescaled image, and we see quite clearly that we do not do much better with the new coordinate system. The only positive result was a reduction in the time used by a factor of 10 due to not fitting very high order polynomials.

Another possibility that has to be considered is that our model of using polynomials of p'th order might not be the optimal model for fitting the images. Other possible models could be sinusoidal functions of x and y with different periods and β-values as amplitudes. This would though take some time to implement in the code, and at the time this is not anything we have time to perform.

4 Conclusions

From our results we see that regression fitting of models to data requires several considerations. In order to maximize the precission of the fit one should prferably have many more data points than free parameters to fit in order to not fit noise. One also need to have a model that is complex enough so that the model data is not undersampled. We have seen that a 4 or 5 splits are optimal for a k-fold cross validation. Further, we see that the choice of coordinates might affect the precission and time of a fit if the coordinates may be chosen independent of the data one wants to fit.

References

Hjorth-Jensen, M. (2019a). Data analysis and machine learning: git repository. https://github.com/CompPhysics/MachineLearning.git.

Hjorth-Jensen, M. (2019b). Data analysis and machine learning: Linear regression and more advanced regression analysis.

A Code

All code and the most important figures produced can be found in the following git repository: https://github.com/krisjand/FYS-STK4155.git The necessary code is in the path "src/python/", while there exists code for some fortran code for linear matrix operations in "src/fortran", though this proved obsolete. Warning: The code outputs many plots and images in the current status, the main plots as shown in this report is in the folder "src/python/main_figs"

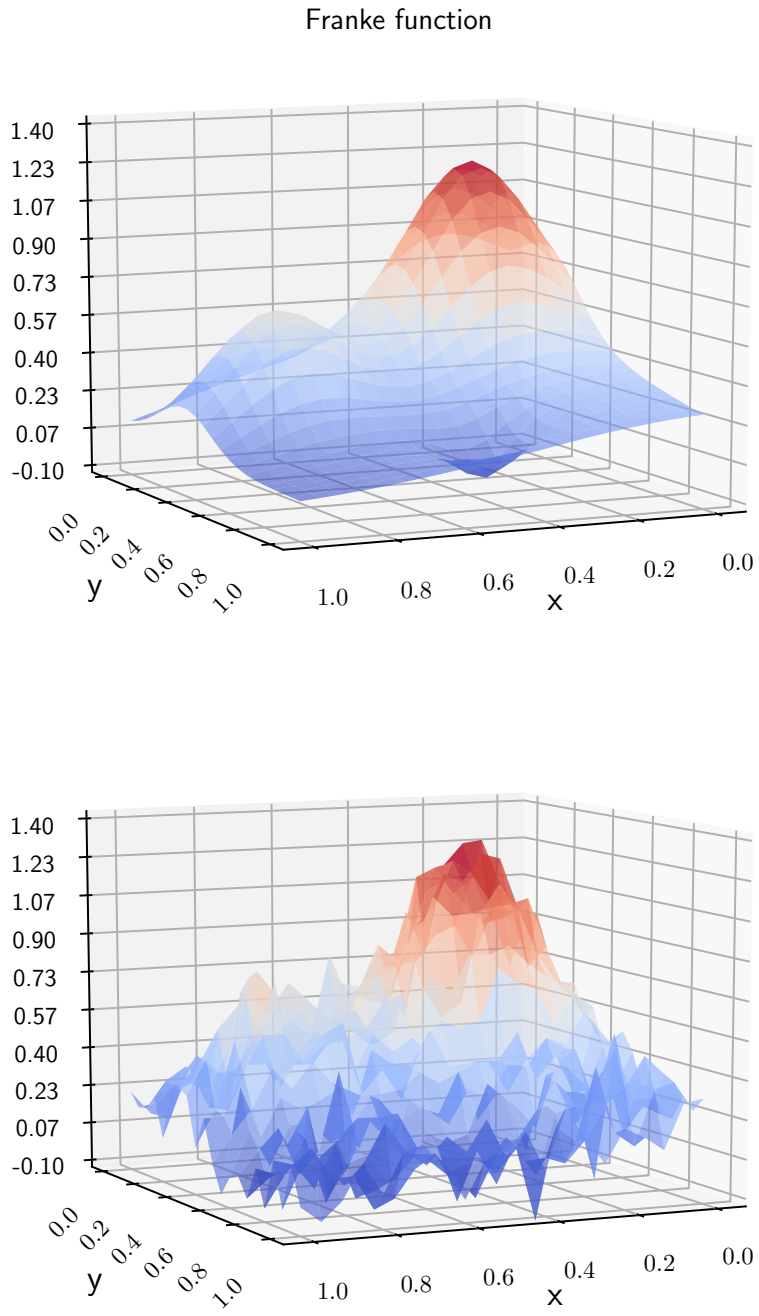

Figure 1: Franke function,using a 21×21 equidistant grid of (x, y) coordinates between (0,0) and (1,1). The bottom plot shows the Franke function with gaussian random noise with zero mean and 0.01 variance.

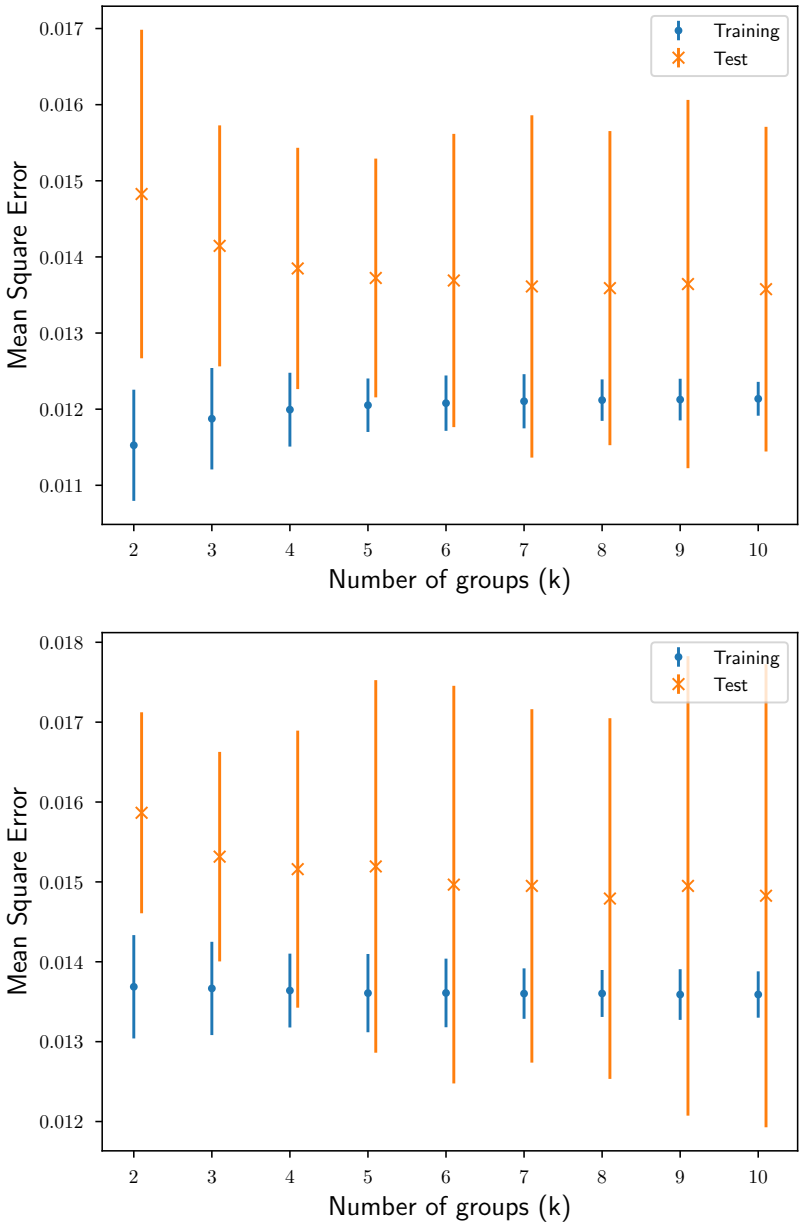

Figure 2: Number of cross validation groups/splits tradeoff plot for training and test data for OLS (*top*) fit and Ridge regression (*bottom*) fit with $\lambda = 1.0 \cdot 10^{-3}$. The plots show the average mean square error with calculated standard deviations of a 5th order polynomial as a function of number of groups/splits.

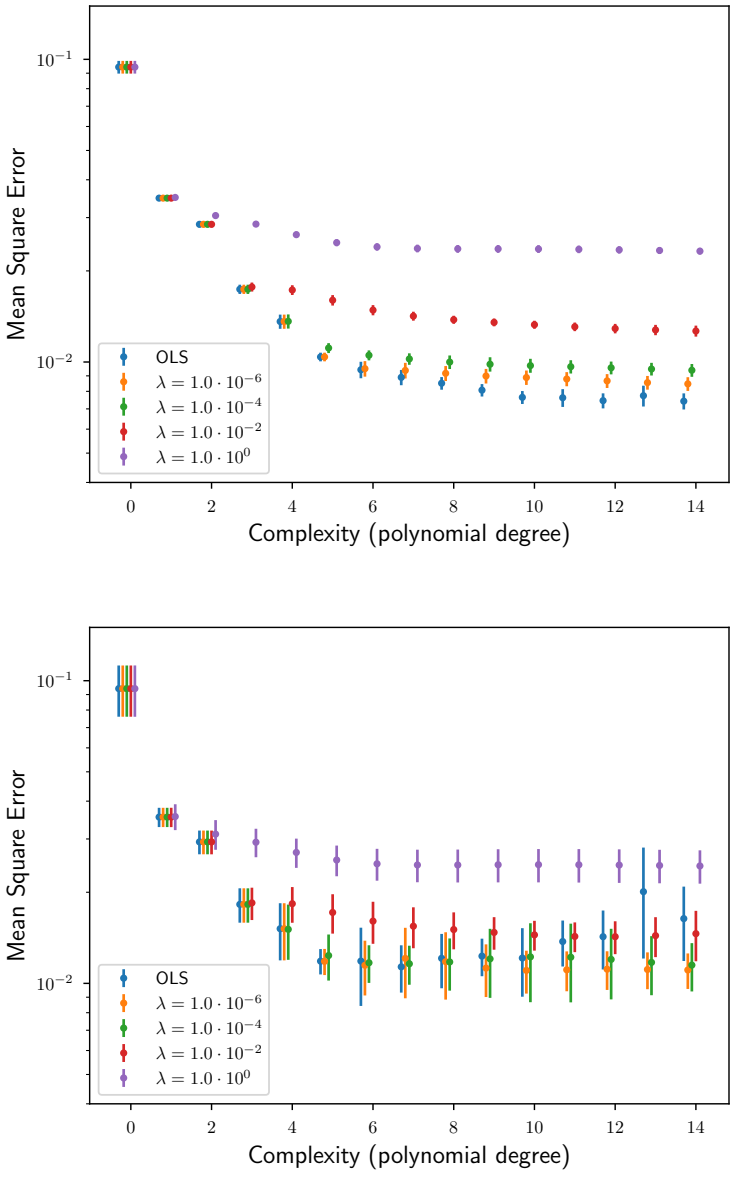

Figure 3: Complexity tradeoff plot of fitting the Franke function using Ridge regression on a 21×21 equidistant grid with different values of λ. The top plot shows the average mean square error of the training data as a function of complexity for a 4-fold cross validation test, while the bottom plot shows a similar plot for the test data. The error bars are calculated standard deviations. The blue points are the values of the ordinary least squares fit, equivalent to $\lambda = 0$.

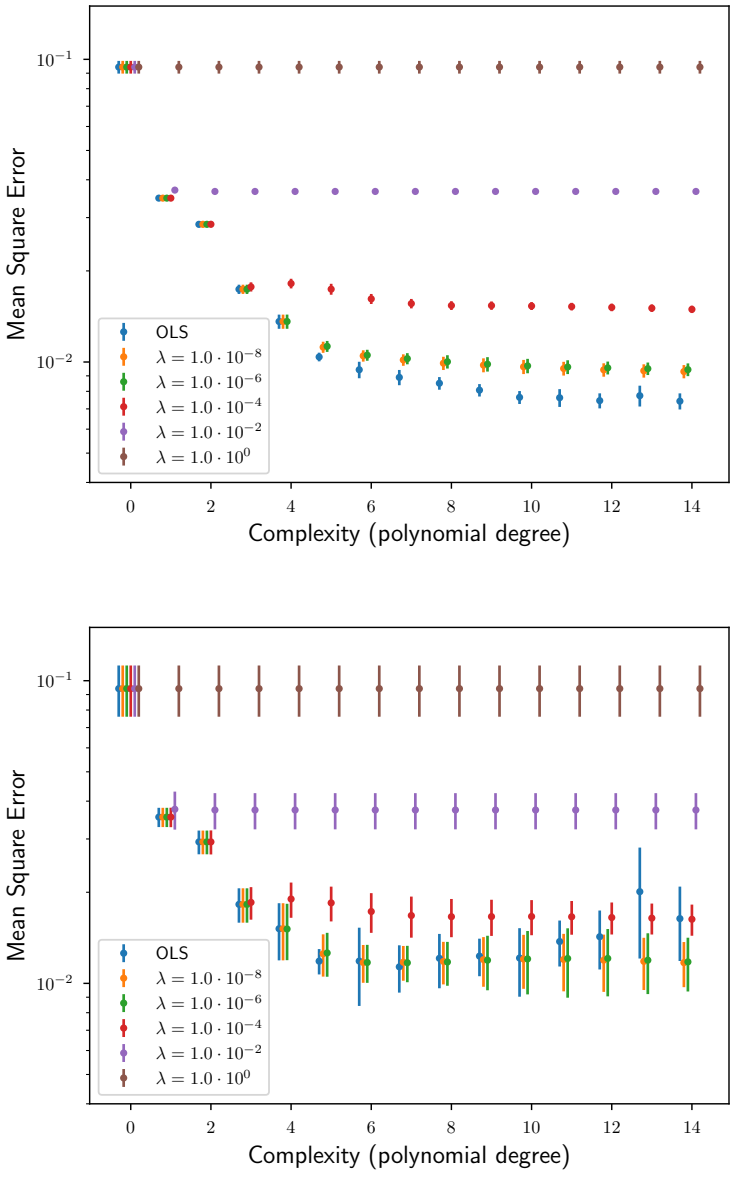

Figure 4: Complexity tradeoff plot of fitting the Franke function using Lasso regression on a 21×21 equidistant grid with different values of λ. The top plot shows the average mean square error of the training data as a function of complexity for a 4-fold cross validation test, while the bottom plot shows a similar plot for the test data. The error bars are calculated standard deviations. The blue points are the values of the ordinary least squares fit, equivalent to $\lambda = 0$.

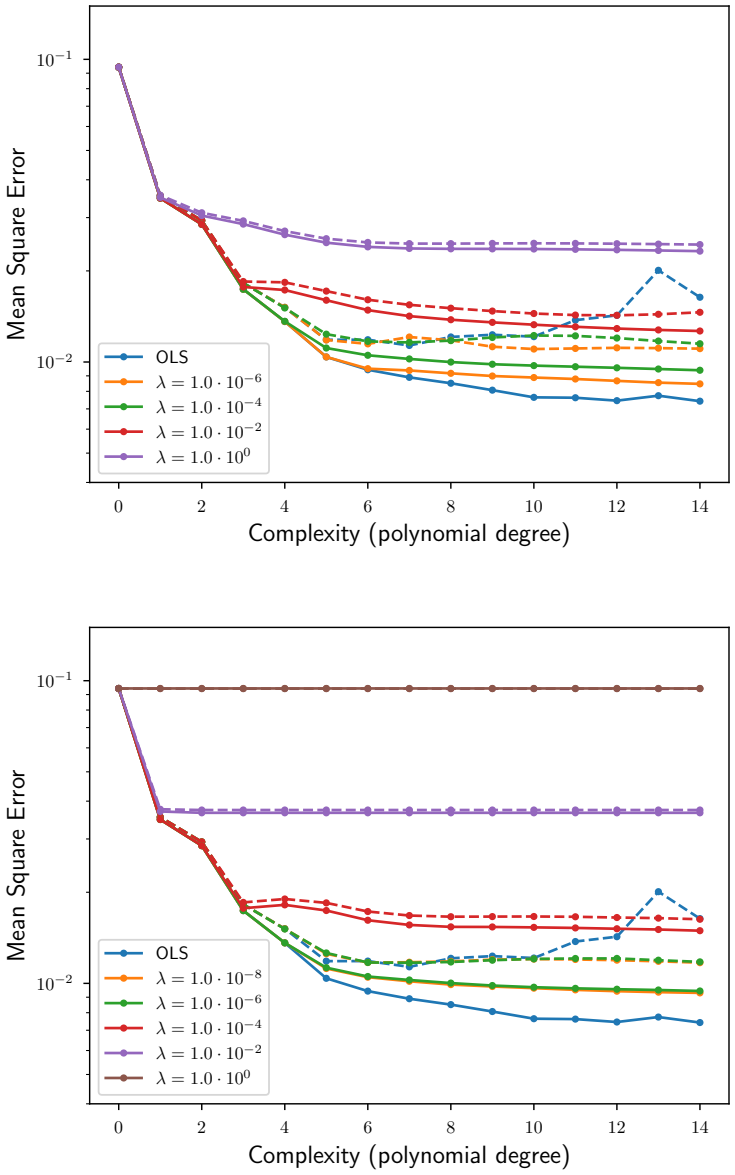

Figure 5: Complexity tradeoff plot for training and test data with different values of λ, fitted on a 21×21 equidistant grid. The top plot shows the average mean square error as a function of complexity for the Ridge regression as seen in fig. 3 with broken lines for test data and full lines for training data. The bottom plot shows a similar plot for the Lasso regression fit. The blue points are the values of the ordinary least squares fit, equivalent to $\lambda = 0$.

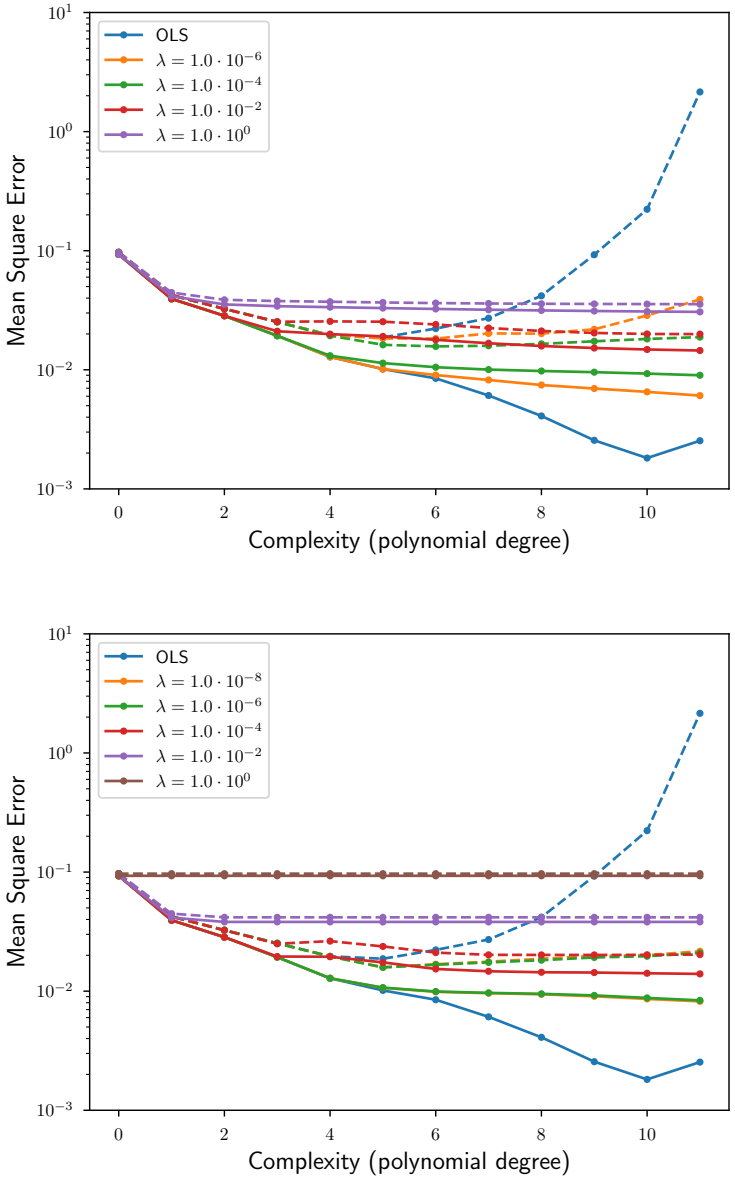

Figure 6: Complexity tradeoff plot for training and test data with different values of λ, now fitting on a 10×10 equidistant grid. The top plot shows the average mean square error as a function of complexity for the Ridge regression with broken lines for test data and full lines for training data. The bottom plot shows a similar plot for the Lasso regression fit. The blue points are the values of the ordinary least squares fit, equivalent to $\lambda = 0$.

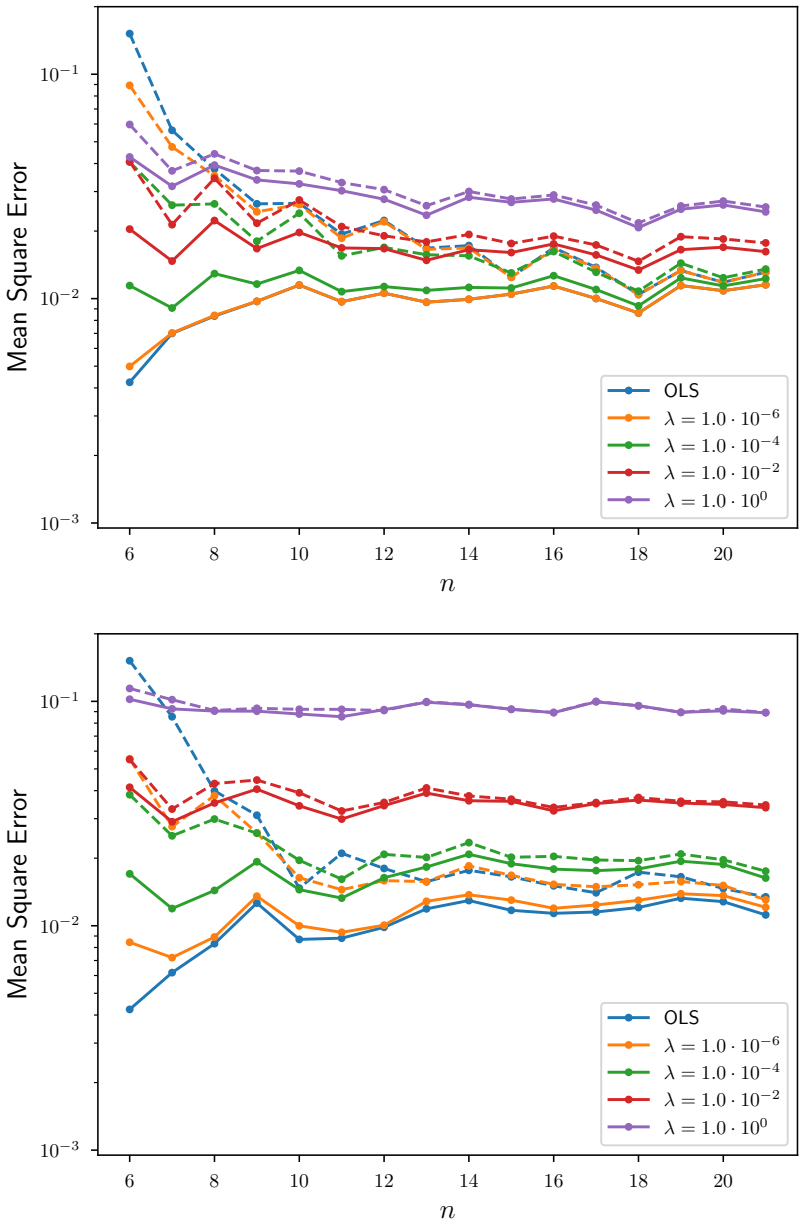

Figure 7: Number of grid/data points tradeoff plot for training and test data for Ridge (*top*) and Lasso (*bottom*) regression fit, showing the average mean square error of a 5th order polynomial as a function of number of number of data points along both the x- and y-axis. In all cases a 4-fold cross validation fit was used. The broken lines for represent the test data and full lines the training data.

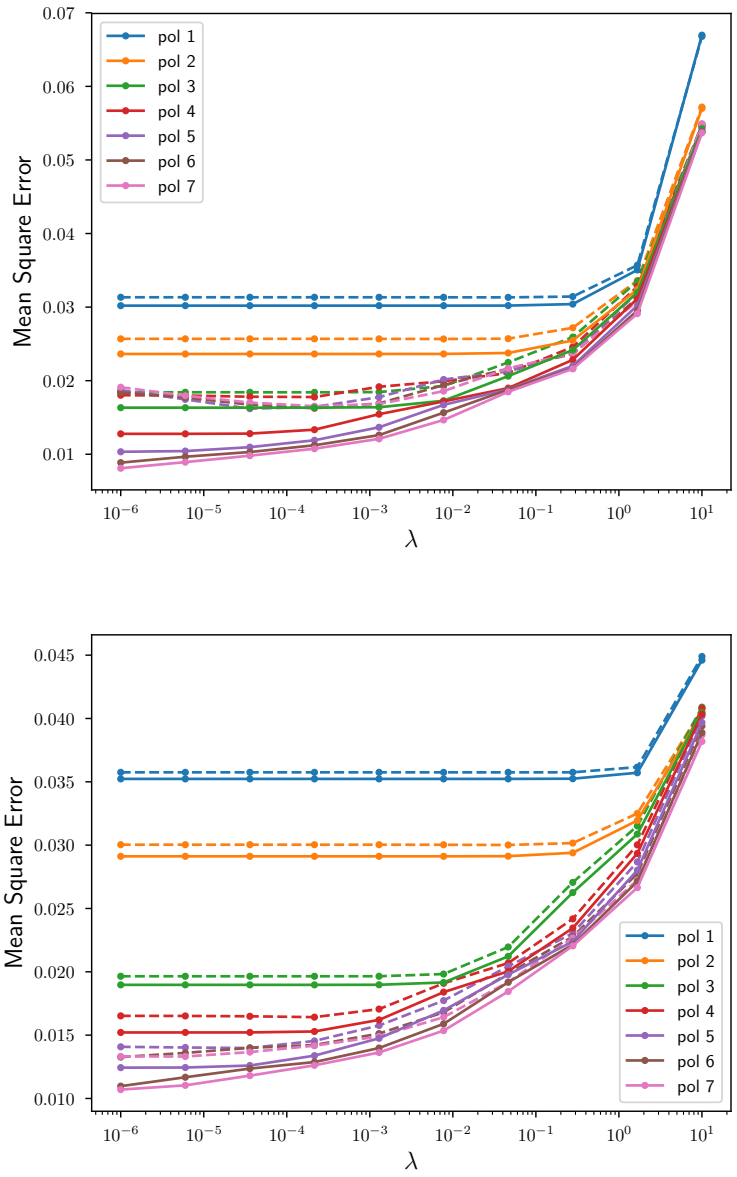

Figure 8: λ tradeoff plot for training and test data for Ridge regression fit, showing the average mean square error as a function of λ for different polynomial orders. The top plot shows the fit on a 10×10 equidistant grid, while the bottom plot shows the fit on a 21×21 grid. The broken lines for represent the test data and full lines the training data.

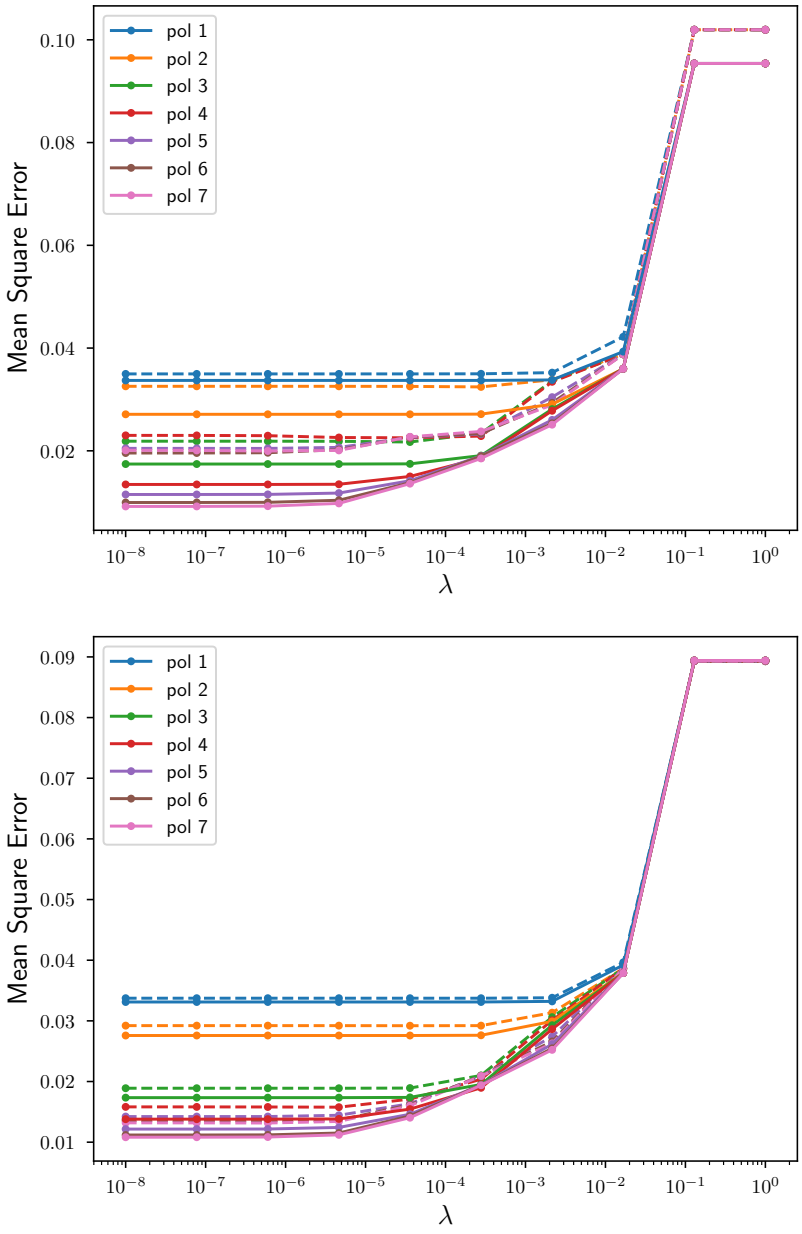

Figure 9: λ tradeoff plot for training and test data for Lasso regression fit, showing the average mean square error as a function of λ for different polynomial orders. The top plot shows the fit on a 10×10 equidistant grid, while the bottom plot shows the fit on a 21×21 grid. The broken lines for represent the test data and full lines the training data.

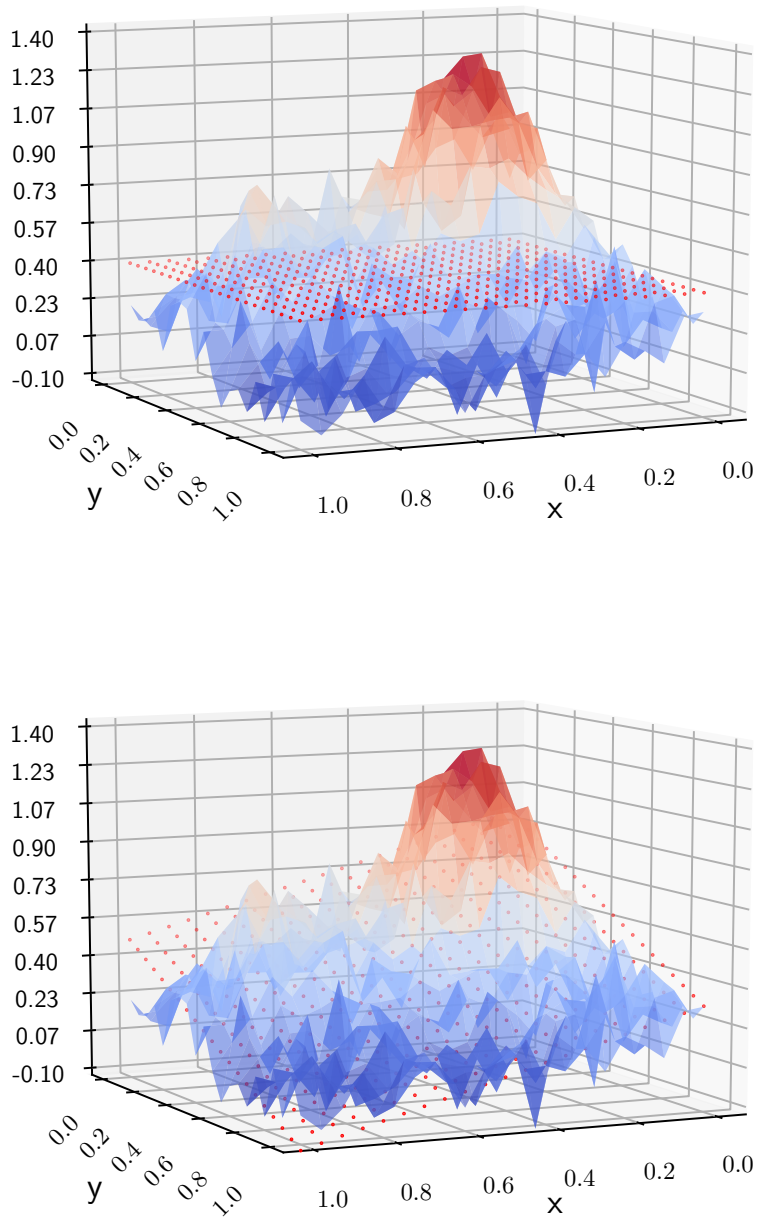

Figure 10: Ordinary least squares fit of the noisy Franke function, fitting a 0th (*top*) and 1st (*bottom*) order polynomial. The red points are the estimated values from the best fit OLS regression.

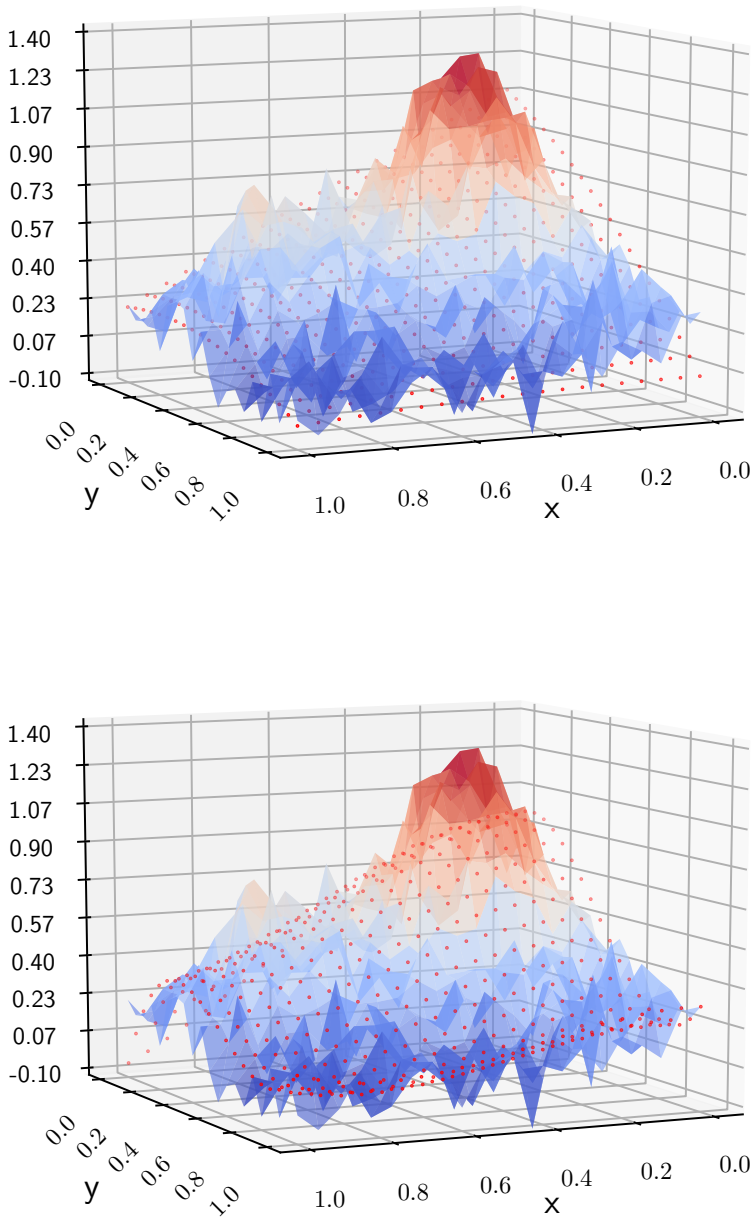

Figure 11: Ordinary least squares fit of the noisy Franke function, fitting a 2nd (*top*) and 3rd (*bottom*) order polynomial. The red points are the estimated values from the best fit OLS regression.

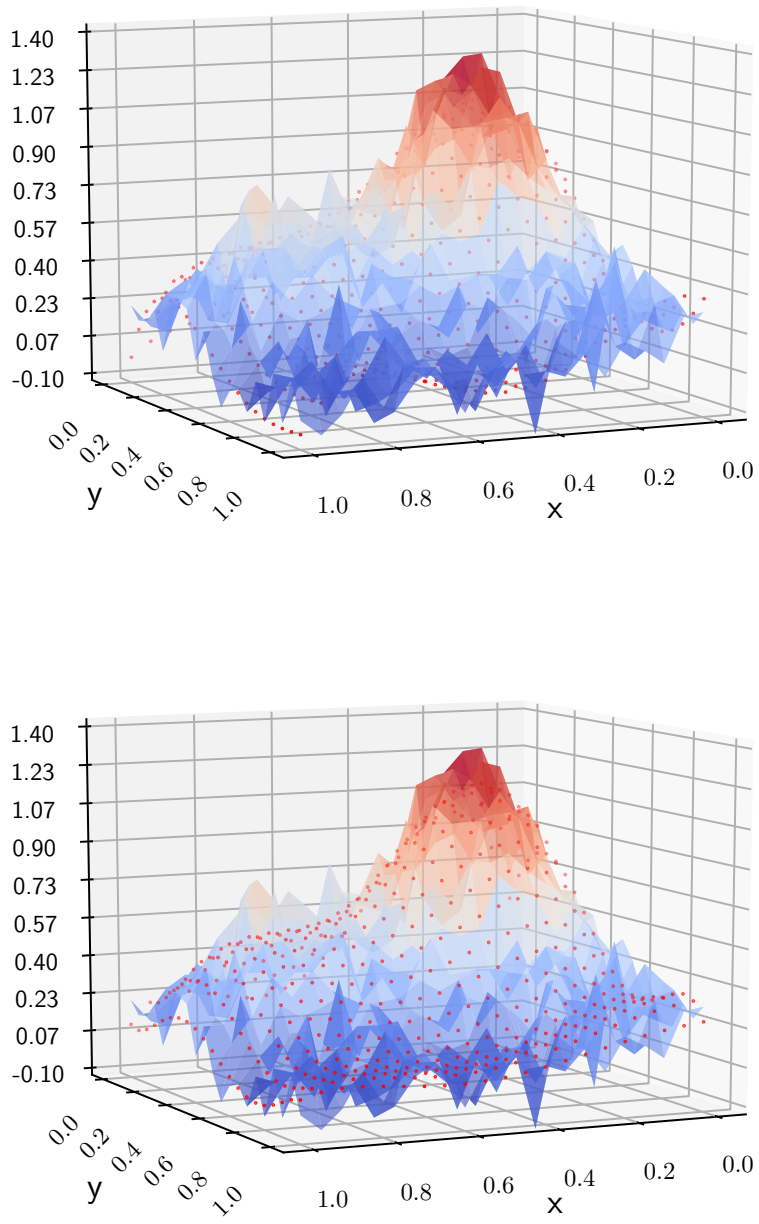

Figure 12: Ordinary least squares fit of the noisy Franke function, fitting a 4th (*top*) and 5th (*bottom*) order polynomial. The red points are the estimated values from the best fit OLS regression.

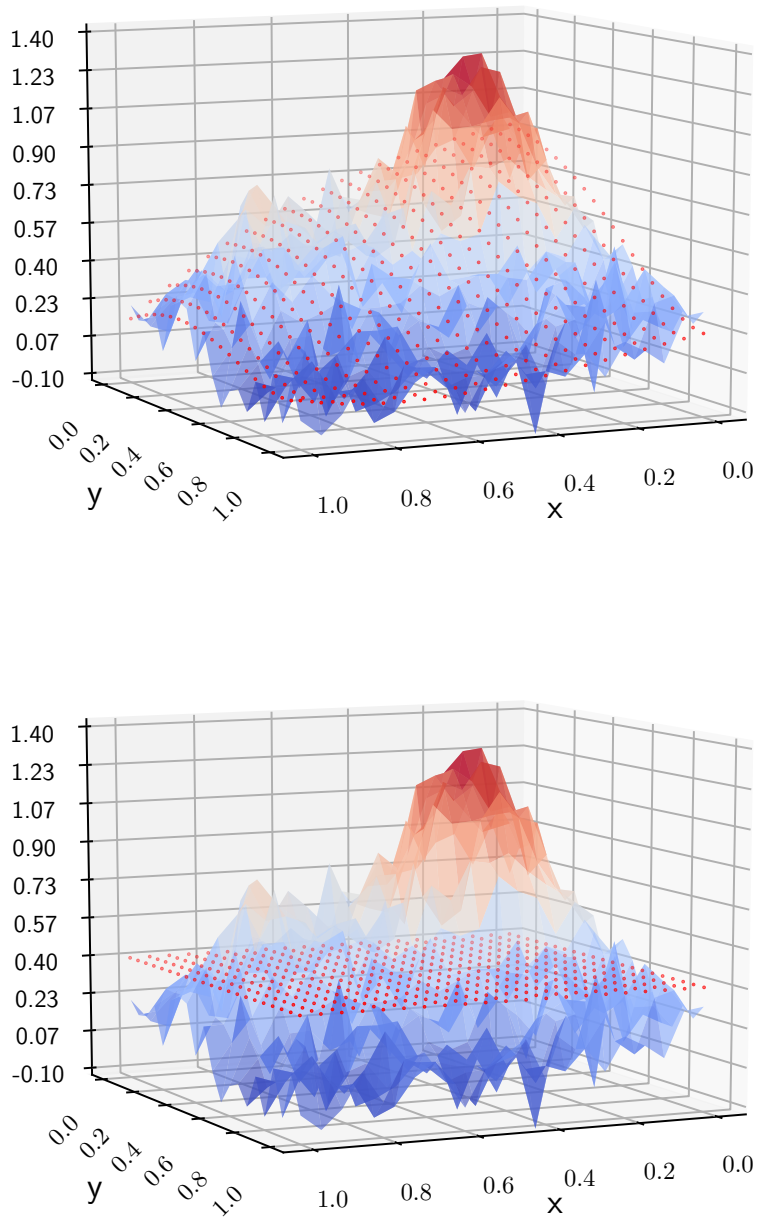

Figure 13: Ridge and Lasso regression fit of the noisy Franke function, fitting a 5th order polynomial with $\lambda = 1.0$. Top plot shows the Ridge fit and the bottom plot shows the Lasso fit. The red points are the estimated values from each regression fit.

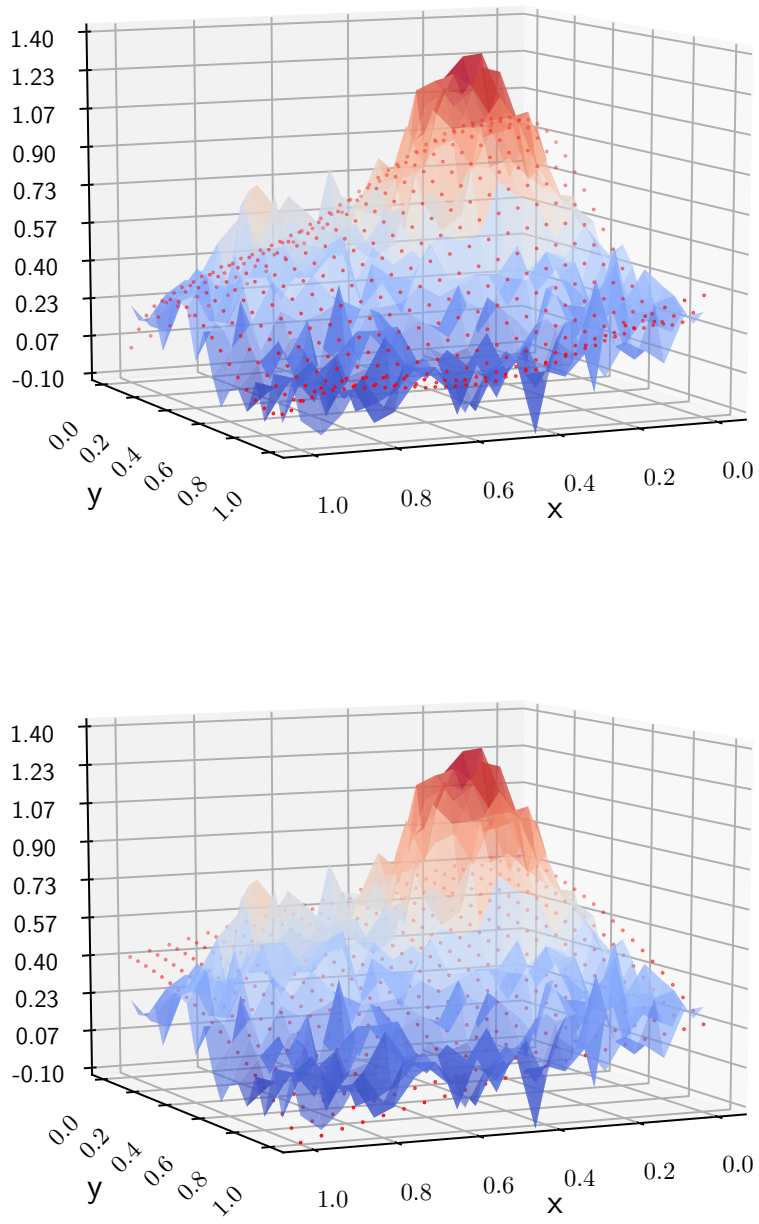

Figure 14: Ridge and Lasso regression fit of the noisy Franke function, fitting a 5th order polynomial with $\lambda = 0.01$. Top plot shows the Ridge fit and the bottom plot shows the Lasso fit. The red points are the estimated values from each regression fit.

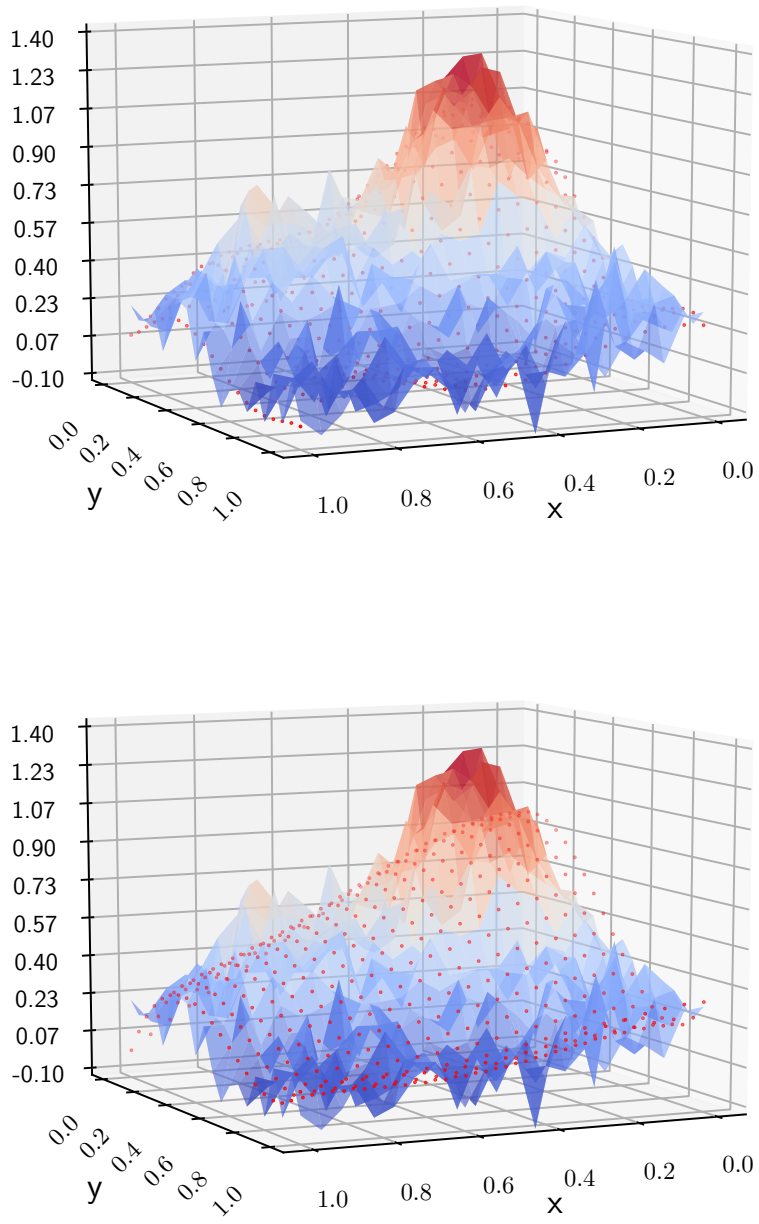

Figure 15: Ridge and Lasso regression fit of the noisy Franke function, fitting a 5th order polynomial with $\lambda = 1.0 \cdot 10^{-4}$. Top plot shows the Ridge fit and the bottom plot shows the Lasso fit. The red points are the estimated values from each regression fit.

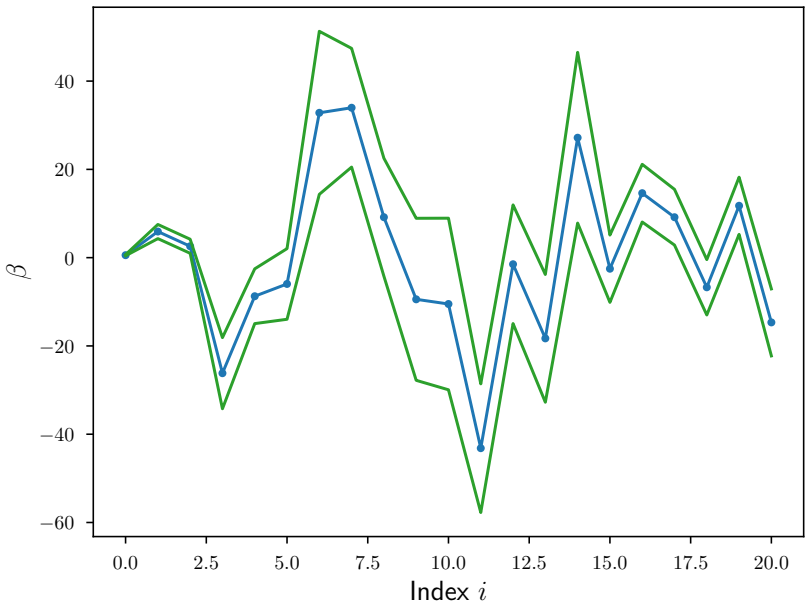

Figure 16: Best fit OLS β-values for a 5th order polynomial on the noisy Franke function, using a 21 $\times$ 21 equidistant grid of (x, y) coordinates between (0,0) and (1,1). The blue line is the estimated β-values while the green lines show the estimated 95% confidence interval using eq. (23)

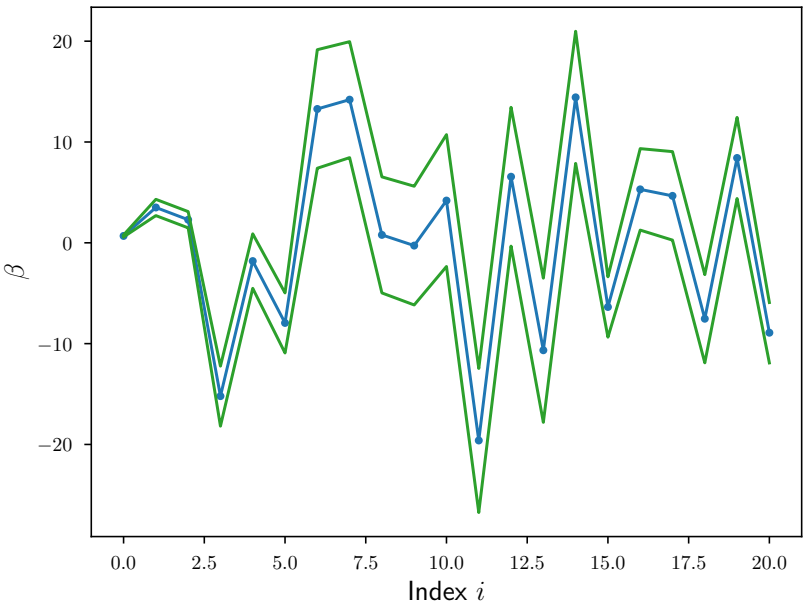

Figure 17: Best fit Ridge regression β-values for a 5th order polynomial and $\lambda = 1.0 \cdot 10^{-4}$, using a 21×21 equidistant grid of (x, y) coordinates between (0,0) and (1,1). The blue line is the estimated β-values while the green lines show the estimated 95% confidence interval using eq. (24)

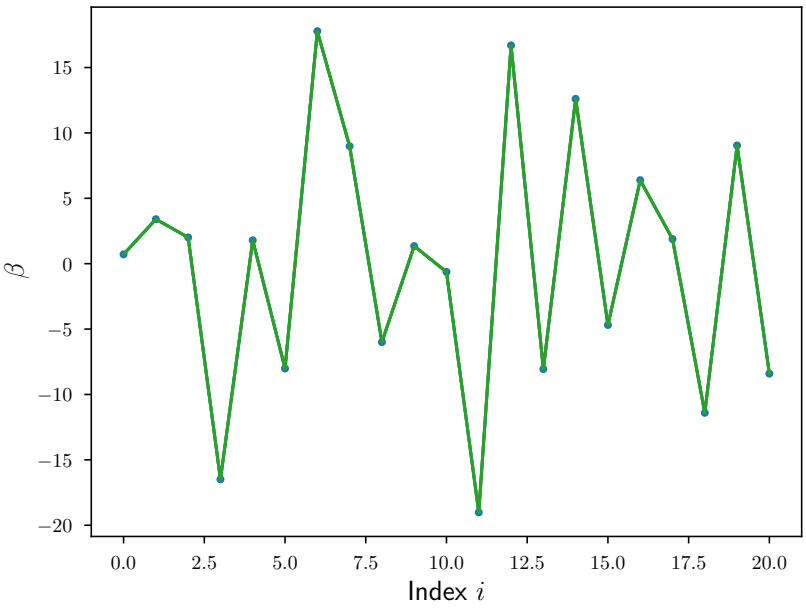

Figure 18: Best fit Lasso regression β-values for a 5th order polynomial and $\lambda = 1.0 \cdot 10^{-6}$, using a 21×21 equidistant grid of (x, y) coordinates between (0,0) and (1,1).

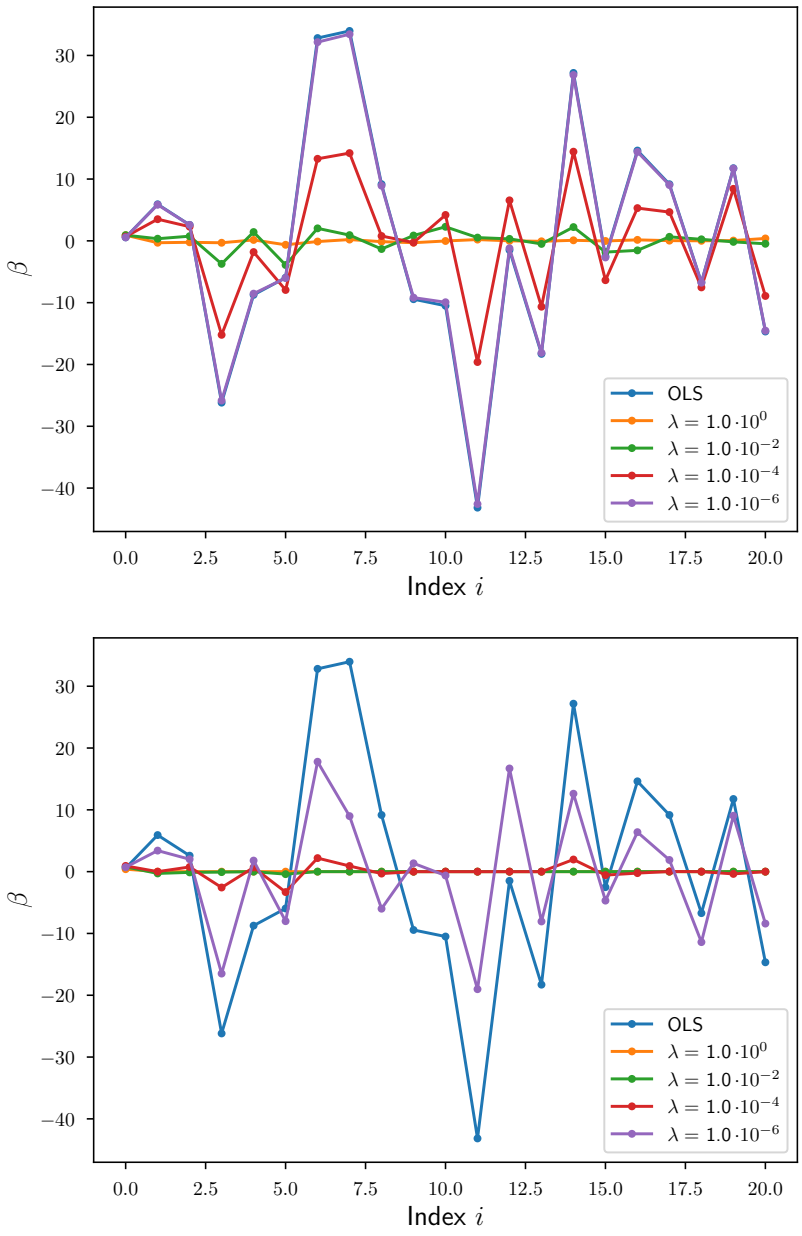

Figure 19: Best fit Ridge (*top*) and Lasso (*bottom*) regression β-values for a 5th order polynomial for different values of λ. The fit was performed on a 21×21 equidistant grid and a 4-fold cross valdation. The blue line represents the ordinary least squares solution, i.e. $\lambda = 0.0$.

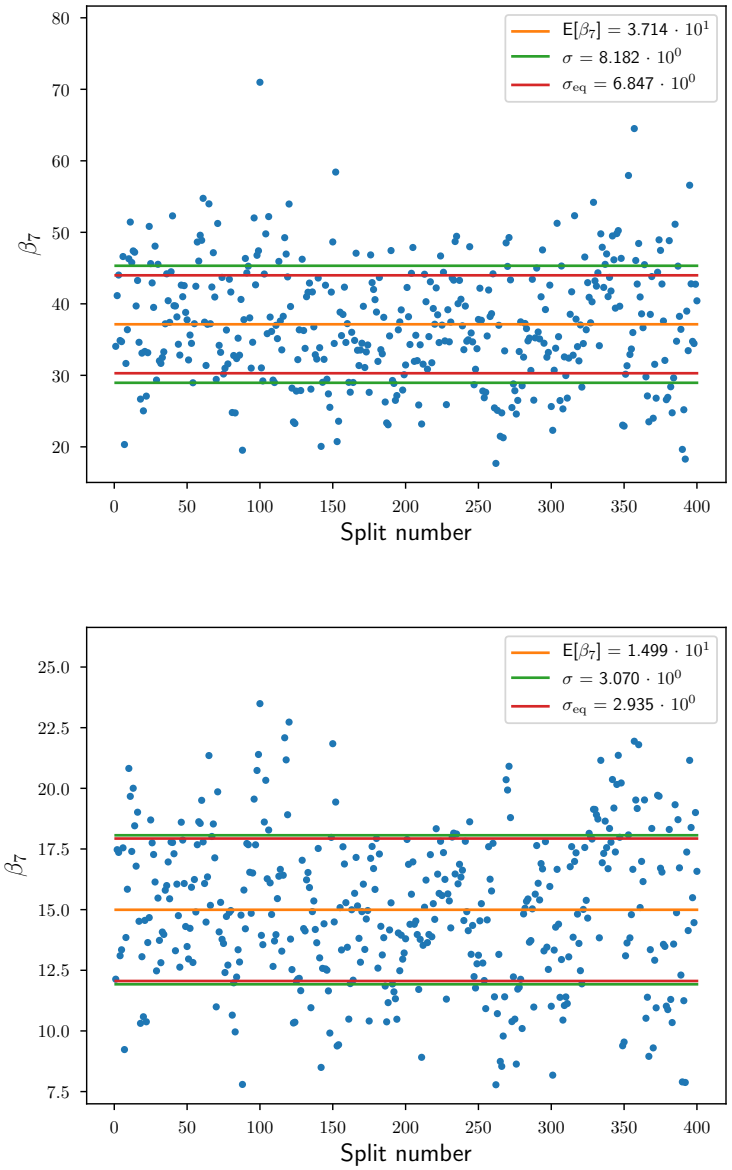

Figure 20: Scatter plot of all fitted β_7-values (i.e. the x^2y term) from a 4-fold cross validation of 100 different realizations of noise on the Franke function. The fitting function was a 5th order polynomial to a 21×21 equidistant grid. The top plot is from an ordinary least squares fit while the bottom plot is from a Ridge regression fit with $\lambda = 1.0 \cdot 10^{-4}$. The yellow line is the average value of all fitted β_7-values, the green lines the calculated standard deviation from the resulting β_7-values, and the red line is the average standard deviation as calculated for each split through eqs. (23) and (24) for each regression method respectively.

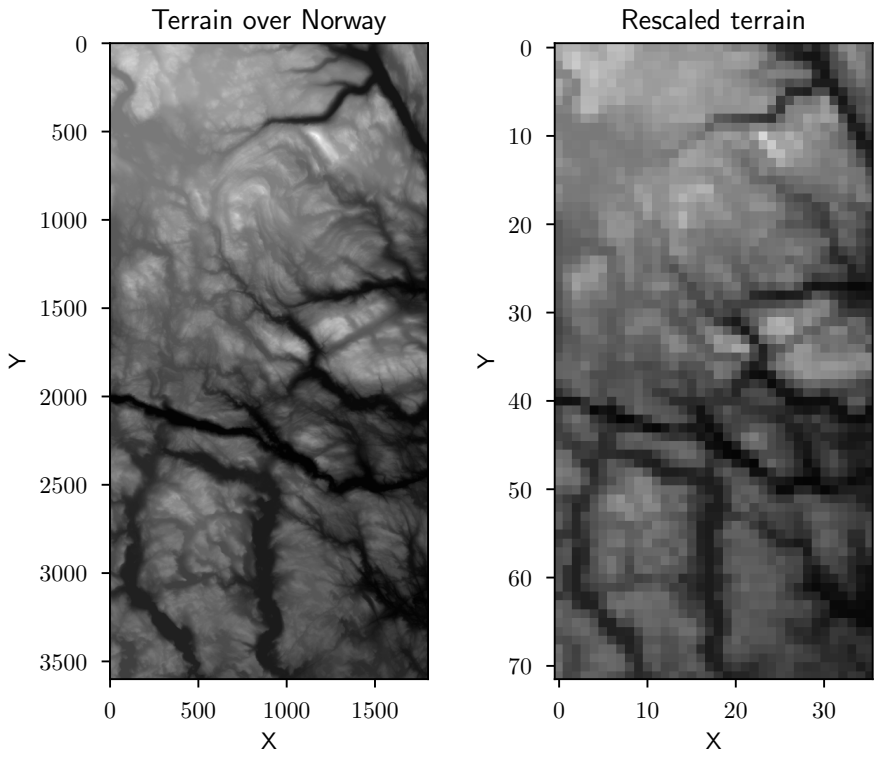

Figure 21: An image showing terrain of a part of Norway. The image to the right is a 1/50 rescaling of the left image. The axis values are pixel numbers in x- and y-direction respectively.

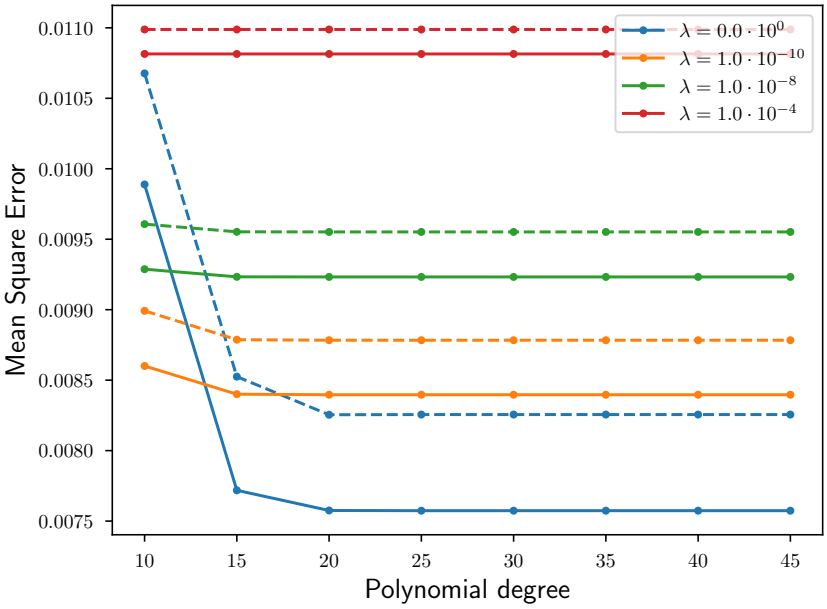

Figure 22: A plot of the mean square error as a function of the model complexity (polynomial order) for the different Ridge regression fits of the rescaled terrain seen in fig. 21. The λ-values of each fit is given, where $\lambda = 0.0$ is equivalent to ordinary least squares. The values are the average of a 4-fold cross validation test, where the test data and training data values are the broken and solid lines respectively. The image pixels have taken values between -0.5 and 0.5 in y-direction with an equal scaling and relative shift in x-direction so that the image in centered on (0,0).

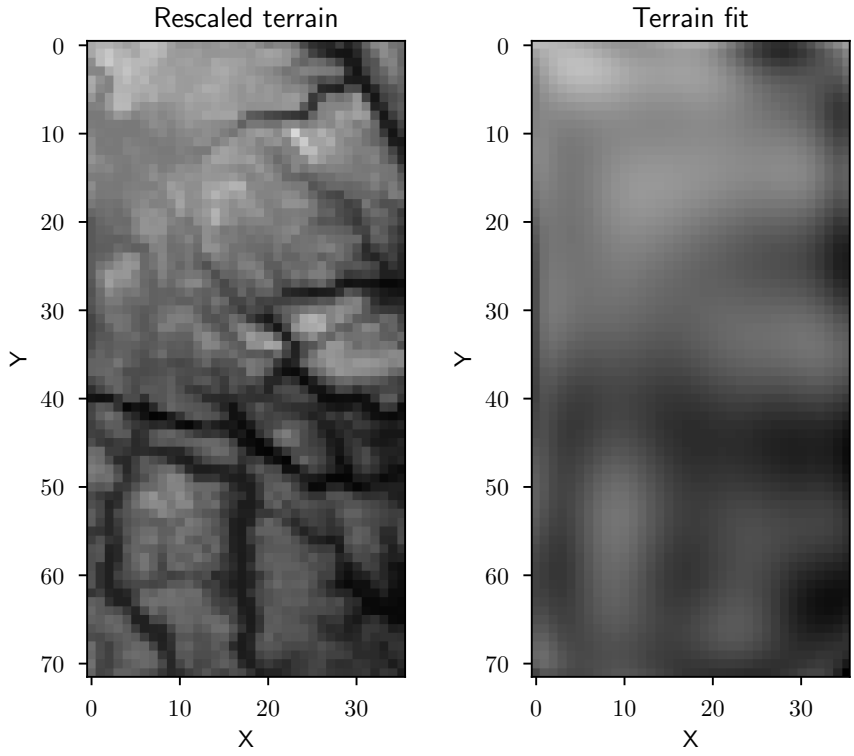

Figure 23: An image showing the rescaled terrain and the fit with the lowest mean square error shown in fig. 22. The fit was an ordinary least squares fit with a ploynomial of 30th order.

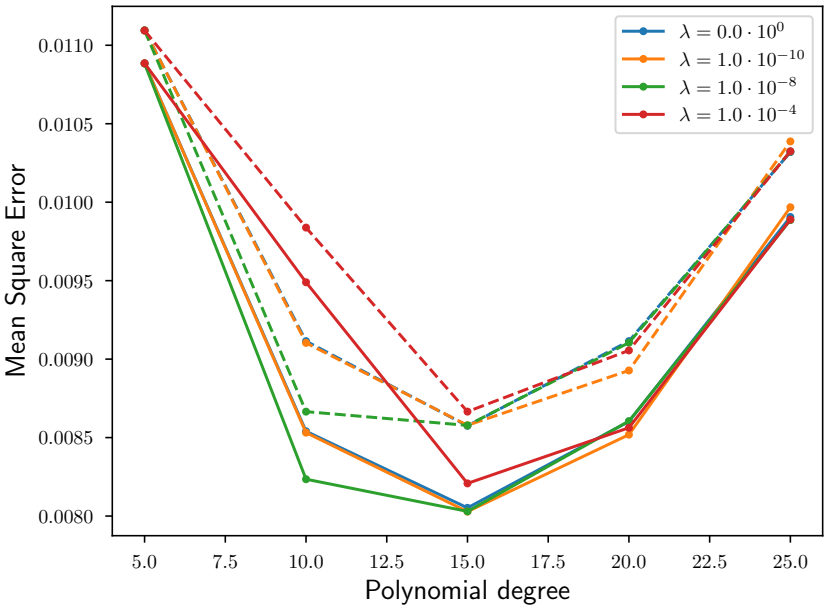

Figure 24: A plot of the mean square error as a function of the model complexity (polynomial order) for the different Ridge regression fits of the rescaled terrain seen in fig. 21. The λ-values of each fit is given, where $\lambda = 0.0$ is equivalent to ordinary least squares. The values are the average of a 4-fold cross validation test, where the test data and training data values are the broken and solid lines respectively. The image pixels have taken values between 0 and 2 in y-direction with an equal scaling in x-direction.

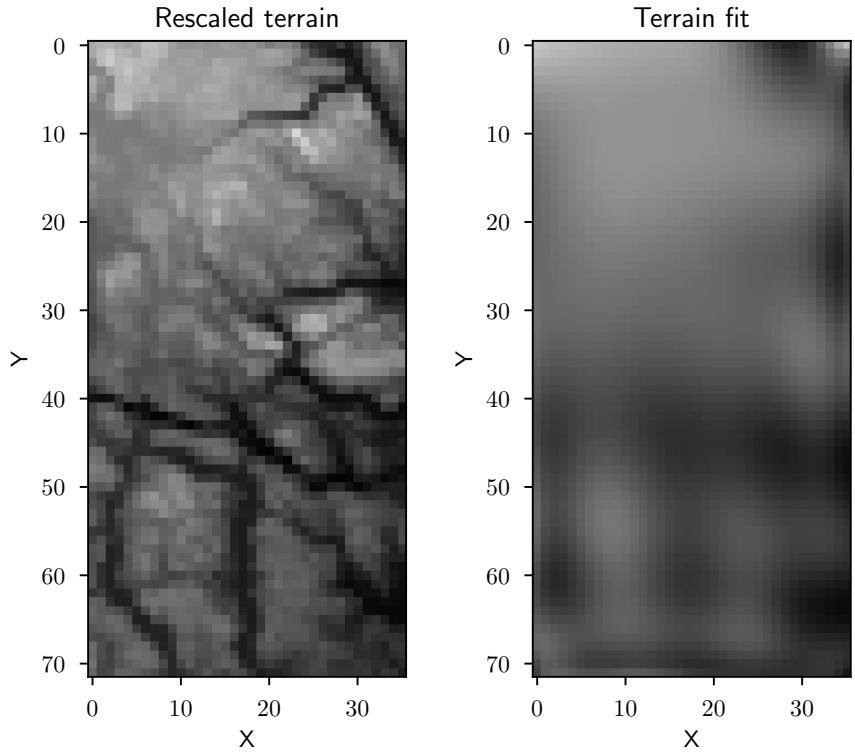

Figure 25: An image showing the rescaled terrain and the fit with the lowest mean square error shown in fig. 22. The fit was an ordinary least squares fit with a ploynomial of 30th order.