Final version October 9, 2020

Preprint typeset using LaTeX style emulateapj v. 12/16/11

ANALYSIS OF THE DATA WITH THE LINEAR REGRESSION METHODS WITH APPLICATION OF RESAMPLING TECHNIQUES APPLIED DATA ANALYSIS AND MACHINE LEARNING, FYS-STK4155 PROJECT 1

Line G. Pedersen[1] and Maria L. Markova[1]

Final version October 9, 2020

ABSTRACT

The following project is focused on the data analysis by means of the linear methods for regression, namely, the Ordinary Least Squares, ridge and LASSO methods. This analysis is combined with application of two resampling techniques, the bootstrap and k-fold cross-validation method. The first part of the project aims at testing the proposed methods on the simplest Franke's function. The predicting power of the models was studied in terms of the mean squared errors (MSE) for the test and train data subsets as well as the bias-variance tradeoff for a variable model complexity. Additionally, models were studied as functions of the hyperparameter λ, added noise amplitude, number of data points and relative size of the test and train data subsets. The ridge regression in the combination with cross-validation and LASSO regression with bootstrap were found to be the most effective for minimizing the test MSE for a model with the polynomial degree 5 for Franke's data. In the second part of the project, developed programs were applied to the real terrain data, and the behaviour of fit scores was studied for the combination of all regression methods and with cross-validation resampling. The ridge regression method with 8-fold cross-validation was found to be the most efficient as applied to minimizing the test MSE, as compared to OLS, and significantly less time consuming as compared to the LASSO regression.

Subject headings: Linear Methods for Regression, Ordinary least squares, Shrinkage Methods, ridge regression, LASSO regression, bias-variance tradeoff, cost function, Bootstrap resampling technique, k-fold Cross-Validation

1. INTRODUCTION

A common problem in all fields of science is finding a good model that fits experimentally obtained data. In principle, a model should be built on a balance between being able to reproduce all significant features of the data without falling into the problem of poor predictability due to overfitting. Here, selecting an effective fit method together with optimal fit parameters is a main key to success.

Machine learning is one of these fields, where a correct and reliable reproduction of data with a certain model is one of the central problems. This field of artificial intelligence has gained a lot of traction in the recent years with the development of continuously stronger computers. Today it is used in a variety of fields, as most of modern science is based on big data sets which can be difficult or even impossible to analyze manually. A common introduction to machine learning is different regression methods, which aim is to let the machine learn from a big set of training data, and then let it predict a fit for a given dataset. However, a good scientific model needs an equally good and reliable error estimation. The fits are thus usually tested with different methods on a separate set of data not used as a part of the training data.

In this report, three central regression methods to fit a three-dimensional set of data will be explored. These are the Ordinary Least Squares (OLS) method, the Ridge regression, and Least Absolute Shrinkage and Selection Operator (LASSO), outlined in the theoretical background sections 2.1.1, 2.1.2, and 2.1.3. In addition, the error estimates with these methods will be improved by combining them with the resampling techniques, such as bootstrap and cross-validation, discussed in sec.2.2.1 and sec.2.2.2. The code developed to realize all these methods is described in detail in sec.4.

The testing of the regression methods is two-fold. The first part consists of fitting the data generated with the analytical Franke's function with added random noise. The effectiveness of the three methods will be tested by calculating mean squared errors (MSE), R^2 scores, as well as by performing bias-variance tradeoff analyses, which are described in details in sec.5 and sec.7.1. The second part of the project consists of applying the regression methods to the more complex dataset, based on the real terrain data. In contrast to the Franke's data,

l.g.pedersen@fys.uio.no
maria.markova@fys.uio.no
[1] Department of Physics, University of Oslo, P.O. Box 1048 Blindern, N-0316 Oslo, Norway

the data are characterized by numerous features of the real landscape, and the aim here is to find a function of polynomial degrees reproducing the general terrain details in an optimal way. Again, the success of the various methods will be determined from MSE and R^2, as well as the bias-variance trade off. The main results on this dataset can be found in sec.6. The detailed discussion of obtained scores is given in sec.7.2. The summary of all results and conclusion can be found in sec.9.

2. THEORETICAL BACKGROUND

2.1. *Regression analysis in general and linear regression*

Each data analysis problem in machine learning application, as well as in the generic statistical analysis, starts with the input, or predictor, random variables $\mathbf{X} = \{x_0, x_1, ..., x_{n-1}\} \in \mathbb{R}^n$ resulting in the measurement of the response, or observed random variables $\mathbf{Y} = \{y_0, y_1, ..., y_{n-1}\} \in \mathbb{R}^n$. Naturally, one should pose the following question: what is the functional dependence between both sets of variables? The whole analysis will then be focused on finding a certain likelihood (regression) function $\mathrm{P}(\mathbf{Y}|\mathbf{X})$ expressing the distribution of variables in $\mathbf{Y}$ given variables from $\mathbf{X}$ as a condition. Setting the relationship between both datasets as $\mathbf{Y}{=}f(\mathbf{X})$ will allow us to go further in the sense of making predictions of an outcome for new input data and inferring casual relationships between both data sets. In a vast majority of cases, no prior knowledge on the functional form $\mathbf{Y}{=}f(\mathbf{X}){=}(\mathbf{Y}|\mathbf{X})$ is available, and it might be reasonable to assume the linearity of the likelihood function with respect to the input parameters $\mathbf{X}$. This provides us with a significant simplification, since a linear regression model can be simply built as (1):

$$\tilde{\mathbf{Y}} = f(\mathbf{X}) = \beta_0 x_0 + \sum_{i=1}^{n-1} \beta_i x_i, \tag{1}$$

where $\tilde{\mathbf{Y}}$ is a vector of predicted values, $\beta_0, \beta_1, ..., \beta_{n-1}$ are the regression parameters to be found. In a general case, each element of the $\mathbf{X}$ set can be rewritten as $\mathbf{X} = \mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_{n-1}$, where each vector $\mathbf{X}_i = \{x_i^0, x_i^1, ..., x_i^{p-1}\}$ contains additional distribution of the explanatory variables over p different features. In this case the vector $\mathbf{X}{\in}\ \mathbb{R}^n$ transforms into a so-called design, or feature, matrix $\mathbf{X}{\in}\ \mathbb{R}^{n\times p}$. The response variables can then be rewritten in the vector form as:

$$\mathbf{Y} = \tilde{\mathbf{Y}} + \boldsymbol{\epsilon} = f(\mathbf{X}) + \boldsymbol{\epsilon} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{2}$$

where $\mathbf{Y}{=}\{y_0, y_1, ..., y_{n-1}\}^T$, $\boldsymbol{\beta}{=}[\beta_0, \beta_1, ..., \beta_{p-1}]^T \in \mathbb{R}^p$, and $\boldsymbol{\epsilon}{=}[\epsilon_0, \epsilon_1, ..., \epsilon_{n-1}]^T$ is the vector of approximation errors.

Since both vectors $\boldsymbol{\beta}$ and $\boldsymbol{\epsilon}$ are unknown, the goal now is to estimate parameters $\boldsymbol{\beta}$ minimizing the difference between the observed and predicted values, or so-called residuals $\mathbf{Y} - f(\mathbf{X})$. It is natural to assume, that the sum of residuals for each sample y_i might serve as an estimator of how well the model fits the observed data. However, in this case, one might run into the problem of mutual cancellation of some terms with the opposite sign, which might mask an issue of poor fitting. It might be improved to a certain extent by taking the sum of absolute values of residuals or the relative errors (RE) (2):

$$\mathrm{RE}_i = \left| \frac{y_i - \tilde{y_i}}{y} \right|. \tag{3}$$

The new issue arising here is related to the existence of the minimum for such sum, since it may not be present at all. In order to ensure the existence of parameters $\boldsymbol{\beta}$ minimizing the deviation of an observation from a prediction, it is convenient to introduce the residual sum of squares (1):

$$\mathrm{RSS} = \sum_{i=0}^{n-1} (y_i - f(x_i))^2 = (\mathbf{Y} - \tilde{\mathbf{Y}})^T (\mathbf{Y} - \tilde{\mathbf{Y}}). \tag{4}$$

The minimized function is often called the cost function $C(\mathbf{Y}, f(\mathbf{X}))$. The primary goal is now to find the set of parameters $\boldsymbol{\beta}$ (model) minimizing the cost function which can be accomplished with the following procedures.

2.1.1. *Ordinary Least Squares*

The cost-function specified in the previous section serves as the base for the simplest, and widely used regression method, the ordinary least squares method (OLS) (1). The cost function is defined by the mean squared error for a given set of observable variables $\mathbf{Y}$ and the corresponding predictions (in the vector form):

$$\begin{aligned} C(\mathbf{Y}, f(\mathbf{X})) =& \frac{1}{n} \left(\mathbf{Y} - f(\mathbf{X})\right)^T (\mathbf{Y} - f(\mathbf{X})) \\ =& \frac{1}{n} \left(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\right)^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \,. \end{aligned} \tag{5}$$

Therefore, the search of the regression parameters $\boldsymbol{\beta}$ is defined by the following minimization problem to be solved:

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{n} \left(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\right)^T \left(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\right) \right\}. \tag{6}$$

Taking the derivatives with respect to sequential parameters β_i provides us with:

$$\frac{\partial C(\mathbf{Y}, f(\mathbf{X}))}{\partial \beta_i} = \frac{\partial}{\partial \beta_i} \left\{ \frac{1}{n} \sum_{j=0}^{n-1} (y_i - \beta_0 x_j^0 - \sum_{k=1}^{p-1} \beta_k x_j^k)^2 \right\} = 0, \tag{7}$$

or in the compact vector form:

$$\frac{\partial C(\mathbf{Y}, f(\mathbf{X}))}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) = 0. \tag{8}$$

In the case of linear independence of the elements (columns) in $\mathbf{X}$ (*i.e.* the design matrix $\mathbf{X}$ has the full column rank), one can rewrite the previous expression as:

$$-2\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) = 0 \Rightarrow \boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}, \tag{9}$$

thus setting the unique relation for the regression coefficients β_i. Linear dependence of the sample columns in the design matrix results in a singular square matrix $\mathbf{X}^T\mathbf{X}$, for which the determinant $det(\mathbf{X}^T\mathbf{X}) = 0$ and the inverse matrix can no longer be determined. This phenomenon is called perfect multicollinearity and can be limited by imposing an additional assumption of the linearly independent columns. This eliminates the problem of singularity for $\mathbf{X}^T\mathbf{X}$ and ensures an existence of the inverse matrix. If not applied, it is still possible to perform prediction of $\mathbf{Y}$ values. However, the regression coefficients β_i will then not be uniquely defined. In a vast majority of real cases, generated data might contain a considerable fraction of correlated variables. This leads to a particularly poor performance of the described linear regression method: the regression coefficients can be poorly defined and, if applied for the fit, result in a large variance for the predicted data. A particularly large positive coefficient on one variable might be cancelled out by a particularly large negative coefficient of the correlated variable. This problem might be solved by introducing the so-called shrinkage methods, described in the next section.

The one of underlying assumptions in the ordinary least squares method is the independent and identically distributed sample pairs (x_i, y_i) or, at least, the presence of a conditional independence of y_i given the input value x_i if the x_i values are not drawn randomly. Another assumption to be made is the form of the distribution for the approximation error $\boldsymbol{\epsilon}$. Considering normally distributed error $\epsilon_i \sim \mathbb{N}(0, \sigma^2)$, one could estimate the variance of the parameters $\boldsymbol{\beta}$ with the following steps:

$$\begin{aligned}\mathbb{E}(\mathbf{Y}) =&\mathbb{E}(\mathbf{X}\boldsymbol{\beta}) + \mathbb{E}(\boldsymbol{\epsilon}) = \mathbf{X}\boldsymbol{\beta} \Rightarrow \\ \Rightarrow&\mathbb{E}(\boldsymbol{\beta}) = \mathbb{E}((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}) = \\ =&(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbb{E}(\mathbf{Y}) = \boldsymbol{\beta},\end{aligned} \tag{10}$$

and, finally:

$$\begin{aligned}Var(\boldsymbol{\beta}) =&\mathbb{E}\left\{(\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta}))^T(\boldsymbol{\beta} - \mathbb{E}(\boldsymbol{\beta}))\right\} = \\ =&\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}.\end{aligned} \tag{11}$$

In the case of this project, the first part implies introduction of an additional noise to the analytical Franke's function, *i.e.* the error is user-defined and sampled from the normal distribution. The noise is multiplied by the amplitude α, $\epsilon \to \alpha\epsilon$, meaning that the variance of the parameters β becomes a quadratic function of α: $Var(\beta) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} \to \alpha^2\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$, which will be used in this project. Otherwise, the variance σ^2 could be estimated as (1):

$$\sigma^2 = \frac{1}{n-p-1}(\mathbf{Y} - \tilde{\mathbf{Y}})^T(\mathbf{Y} - \tilde{\mathbf{Y}}). \tag{12}$$

This expression will be exploited in the work with the real terrain data.

2.1.2. *Shrinkage methods: Ridge regression*

As the amount of data points and parameters p increases, the dimensions of the design matrix increases together with the probability to find correlating components. As it was discussed above, it leads to a poor predicting power of the model as the regression coefficients can not be uniquely estimated. Overall, an absence of an inverse matrix $(\mathbf{X}^T\mathbf{X})^{-1}$ might be corrected by introducing an additional parameter $\lambda \geq 0$ so that:

$$\mathbf{X}^T\mathbf{X} \to \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}, \tag{13}$$

where $\mathbf{I}$ is the $p \times p$ identity matrix. This forms the basic idea of the ridge regression method. In this case, the regression coefficients are being shrunk by imposing an additional penalty on their size via putting a constraint on the diagonal of $(\mathbf{X}^T\mathbf{X})$ before an inversion. The parameter λ is often called the regularization parameter (or hyperparameter) and introduces an additional effective degree of freedom into the regression procedure. The cost function can the be redefined as (2):

$$C(\mathbf{Y}, f(\mathbf{X})) = \frac{1}{n}||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2, \tag{14}$$

where norm-2 vectors are defined as $||x||_2 = \sqrt{\sum_i x_i^2}$. Alternatively:

$$C(\mathbf{Y}, f(\mathbf{X})) = \left\{(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})\right\} + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}. \tag{15}$$

In this case, the regression parameters $\boldsymbol{\beta}^{Ridge}$ are minimizing the penalized residual sum of squares. Differentiating the new cost function with respect to parameters beta results in an explicit form of ridge parameters:

$$\min_{\beta\in\mathbb{R}^p} C(\mathbf{Y}, f(\mathbf{X})) \Rightarrow \boldsymbol{\beta}^{Ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{16}$$

This is equivalent to putting an additional constraint on the OLS cost function minimization problem, namely $\sum_{j=1}^{p-1}\left(\beta_j^{Ridge}\right)^2 \leq t$, where t is a finite positive number. It is important to notice some difference in notations used in the literature. In some sources the the 0th term in β, or intercept, is added (instead of $\beta_0 x_i^0$ into the fit function and removed from the penalty term to avoid dependence on the origin of the chosen vector $\mathbf{Y}$ (see eq.14). The intercept could be easily found by the value of $\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$. In the case of our project both notations are equivalent since x_i^0 corresponds to 1 (zeroth degree of x_i). Another important feature of the ridge regression parameters is its relation to the corresponding OLS $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}^{Ridge} = \frac{\boldsymbol{\beta}^{OLS}}{1+\lambda}, \tag{17}$$

meaning that the regularization parameter, more precisely $1/(1+\lambda)$ scales the OLS $\boldsymbol{\beta}$. It is expected that all ridge estimators converge to 0 (and each other) with the increasing regularization parameter. This fact will be further discussed in the application to the Franke data.

2.1.3. *Shrinkage methods: LASSO regression*

Another method putting an additional constraint on the regression estimators is the Least Absolute Shrinkage and Selection Operator (LASSO). The basic idea of the

method is quite similar to the shrinkage performed with the ridge regression with one main exception: the 1-norm is used instead of 2-norm for the penalty term of the cost function (2):

$$C(\mathbf{Y}, f(\mathbf{X})) = \frac{1}{n}||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1, \qquad (18)$$

where 1-norm is defined as $||x||_1 = \sum_i |x_i|$. This form is equivalent to limiting the minimization problem for the cost function of OLS with an additional condition of $\sum_{j=1}^{p-1} \left|\boldsymbol{\beta}_j^{LASSO}\right| \leq t$. This important difference changes the nature of shrinkage. LASSO regression translates each coefficient β_i by λ and truncates at 0, meaning that the coefficient β_i can be set exactly to zero depending on the hyperparameter λ, in contrast to the ridge regression, asymptotically shrinking β_i to 0 with the increasing λ. In addition, the 1-norm penalty makes the solution of the minimization problem nonlinear with respect to $\mathbf{Y}$, and no closed form of the regression parameter $\boldsymbol{\beta}^{LASSO}$ exists.

2.2. *Resampling techniques*

As one has succeeded in building up a model, it should be further assessed how well this model fits a given dataset, *i.e.* to asses the predicting power of the model. It is the usual case in machine learning to divide a given dataset into the test and training subsets (3). The training data are used for performing a fit of the model parameters, or, practically, for building up the model. The fit parameters are kept and subsequently implemented for fitting the data in the test set. In the neural network application, it is also common to distinguish the validation set used for the tuning of the model parameters used to asses the performance of a model on the test data. However, in the present project the test dataset performs the same functions as the validation set, therefore, the latter was not included in the study. Practically, this division allows to both train the model on the data from a given set (majority of the data) by using *e.g.* OLS, ridge or LASSO regression method and estimate how well the model performs on the rest of the data. This might imply estimation of the mean squared error, bias, variance or the R^2 score for the test and train sets and their comparison. For the increasing complexity of the model and study of its accuracy it might be tempting to collect additional data, gaining higher statistics. However, real large-scale experiments and Monte Carlo simulations might be time consuming, as the time of an experiment as well as the data gained are usually limited. Since the new data are not often accessible, the resampling techniques are used to derive an additional information of the performed fit. The main idea implies repetitive selection of a data sample from the training set, refitting the model on each sample, assessing the performance of this fit on the rest of the (test) data and collecting its quantitative characteristics (mean squared error, R^2 score, *etc.*). This allows us to obtain several assessments of how well the model performs on the new test datasets (in each iteration of resampling) without introducing the new data and infer more information on the fitting model than can be inferred by performing the fitting once. Two widely used resampling techniques will be described and used in this project - the bootstrap and k-fold cross-validation.

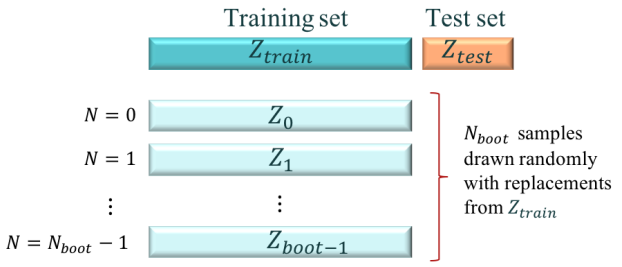

FIG. 1.— Principal scheme of the bootstrap resampling technique.

2.2.1. *Bootstrap*

The principal idea behind the bootstrap resampling technique is based on the repetitive random drawing of the data with replacement from the original training set (1). Let us assume having a set of observables combined in the train set $\mathbf{X}_{train} = \{x_1, x_2, ..., x_n\}$. In each bootstrap iteration i, a data set of the same size as the original training set is drawn with the replacement from $\mathbf{X}_{train}$. For each bootstrap sample X_i the model $\theta_i = \theta(X_i)$ is computed by evaluating the function of interest $\theta = \theta(X)$ under the observations from X_i. The procedure is performed N_{boot} times (*e.g.* 100) and provides us with the set $\theta_1, \theta_2, ..., \theta_N$ which can be subsequently used to estimate the mean value:

$$\bar{\theta} = \frac{\sum_{i=1}^{N_{boot}} \theta_i}{N_{boot}}, \qquad (19)$$

and the standard deviation of the sample:

$$STD(\theta) = \sqrt{\frac{\sum_{i=1}^{N_{boot}} (\theta_i - \bar{\theta})^2}{N_{boot}}}. \qquad (20)$$

The principal advantage of the bootstrap is its relative simplicity. It can be easily applied for the estimation of the mean squared error, bias and variance for both a training set (in this case each bootstrap sample plays a role of the training set, whilst the original training set becomes the test set) and the original test set. This becomes especially convenient in the perspective of the bias-variance tradeoff analysis for the different model complexity. However, this advantage is accompanied by a drawback: the model fitted as well as the prediction made by this model might be dependent on a given representative sample. This is primarily due to the random nature of performed sampling from $\mathbf{X}_{train}$: the bootstrap train sample and the initial training set, playing role of a test sample, have observations in common and the bootstrap set might have the same element x_i drawn from $\mathbf{X}_{train}$ more than once. The overlap of the data in the test and train datasets might lead to masking of the overfitting problem. This drawback might be eliminated with another widely applied resampling technique described below.

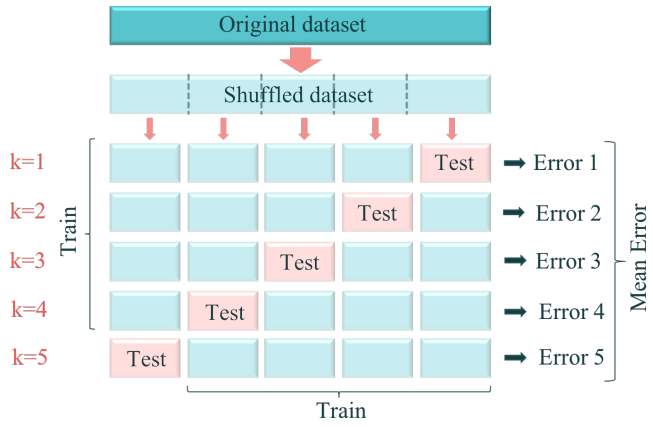

FIG. 2.— Principal scheme of the k-fold cross-validation resampling technique.

2.2.2. *k-fold Cross-Validation*

The unbalanced influence from particular samples drawn from the $\mathbf{X}_{train}$ happening due to the random drawing of the samples with replacements might be eliminated by splitting the initial dataset into k equally sized exhaustive and mutually exclusive subsets, thus preventing the same elements from being found simultaneously in the train and the test sets (2). One of k batches is used as the test set while the remaining $k-1$ are used for the model training. In general, the procedure involves the following steps:

1. The dataset $\mathbf{X}$ is shuffled randomly.
2. The dataset is split into k equally sized groups.
3. Keep one of k batches as the new train set and use the remaining $k-1$ for fitting the model.
4. Evaluate and keep the estimate of the prediction error for a current iteration.
5. Perform steps 3-4 (refitting of the model) $k-1$ times.
6. Estimate an average prediction error by summing up k collected estimates and dividing by the total number of iterations, k.

In a particular case of $k = n$, so-called leave-one-out cross-validation, the procedure is performed n times and all training sets have an especially high degree of resemblance (difference is only in one swapped element). It might lead to an increased variance and almost unbiased error estimate. In the present work 5-fold cross-validation will be the main tool for the study of the mean squared error for the train and the test sets. The study of the bias-variance tradeoff in this case is complicated, as compared to the bootstrap resampling analysis. Calculating the bias and variance for the test dataset at each iteration step and simple averaging over k iterations results in erratic decomposition of the mean squared error into the bias and variance components. Therefore, the bootstrap was used in each case of the bias-variance tradeoff study.

2.3. *Overfitting, underfitting and bias-variance tradeoff*

The final goal of the regression analysis is to find the 'best' model, *i.e.* the model catching all particularities of the data it is trained on and possessing good predicting power of reproducing the details of the data outside of the training set. It might be tempting to use the mean squared error, or the residual sum-of-squares, for the training set to asses the predicting power of the model on the test data. However, the idea of each regression method implies finding the regression parameters minimizing the sum for the training set, ensuring the minimum test MSE for a given model complexity. Increasing complexity of a model for a given set of training data results in a model that adapts better to the observed test data and becomes more rigidly fixed to the current training dataset. As a general rule, such rigidity complicates further generalization of this model if applied to other data. This problem is called overfitting and was briefly mentioned in the previous sections. It might appear in the form of additional noise, or the spikes in the fit function, as in case of this project. In order to understand the nature of this phenomenon, it is common to consider the decomposition of the MSE into the variance, bias and irreducible error terms and to study how all three terms change for the test and train datasets with the increasing model complexity.

Let us consider the case studied in this project. The space of predictor variables is given by $x \in \mathbf{X} \in \mathbb{R}^n$ and $y \in \mathbf{Y} \in \mathbb{R}^n$, as the response variables are $z_i \in \mathbf{Z} \in \mathbb{R}^n$. Further, let us assume that the observed data can be written as:

$$z_i = f(x_i, y_i) + \epsilon_i, \tag{21}$$

with a normally distributed error (noise) $\epsilon_i \in \epsilon \sim \mathbb{N}(0, \sigma^2)$. The aim of our analysis is to find an appropriate approximation of the function $f(\cdot)$, $\tilde{z}$. The mean squared error could then be written as:

$$\mathrm{MSE}(z, \tilde{z}) = \frac{1}{n}\sum_i (z_i - \tilde{z}_i)^2 = \mathbb{E}\left(f(\mathbf{X}, \mathbf{Y}) + \epsilon - \tilde{z}\right)^2. \tag{22}$$

To simplify the decomposition, it is convenient to add and extract $\mathbb{E}(\tilde{z})$:

$$\begin{aligned}\mathrm{MSE}(z, \tilde{z}) &= \mathbb{E}\left(f + \mathbb{E}(\tilde{z}) - \mathbb{E}(\tilde{z}) + \epsilon - \tilde{z}\right)^2 = \\ &= \mathbb{E}\left((f - \mathbb{E}(\tilde{z})) + \epsilon - (\tilde{z} - \mathbb{E}(\tilde{z}))\right)^2\end{aligned} \tag{23}$$

Writing the term under the $\mathbb{E}$ sign explicitly one gets:

$$\begin{aligned}\mathrm{MSE}(z, \tilde{z}) &= \mathbb{E}\left(f - \mathbb{E}(\tilde{z})\right)^2 + \mathbb{E}\left(\tilde{z} - \mathbb{E}(\tilde{z})\right)^2 + \mathbb{E}(\epsilon)^2 - \\ &- 2\mathbb{E}\left((f - \mathbb{E}(\tilde{z}))(\tilde{z} - \mathbb{E}(\tilde{z}))\right) + 2\mathbb{E}\left(\epsilon(f - \mathbb{E}(\tilde{z}))\right) - \\ &- 2\mathbb{E}\left(\epsilon(\tilde{z} - \mathbb{E}(\tilde{z}))\right).\end{aligned} \tag{24}$$

According to the definition, function f does not have a stochastic nature and gives deterministic correspondence between its value and a chosen pair (x_i, y_i). Here it is assumed for simplicity that the pairs (x_i, y_i) are predetermined non-randomly (*e.g* by introducing a grid). Similar is applicable to the deterministic value $\mathbb{E}(\tilde{z})$. For the error ϵ $\mathbb{E}(\epsilon_i) = 0$ and $\mathrm{Variance}(\epsilon_i) = \mathbb{E}(\epsilon_i^2) = \sigma^2$. The

fourth term can be then simplified as:

$$2\mathbb{E}\left((f-\mathbb{E}(\tilde{z}))(\tilde{z}-\mathbb{E}(\tilde{z}))\right) = 2(f-\mathbb{E}(\tilde{z}))\mathbb{E}(\tilde{z}-\mathbb{E}(\tilde{z})) = \\ = 2(f-\mathbb{E}(\tilde{z}))(\mathbb{E}(\tilde{z})-\mathbb{E}(\tilde{z})) = 0. \tag{25}$$

By analogy:

$$2\mathbb{E}\left(\epsilon(f-\mathbb{E}(\tilde{z}))\right) = 2(f-\mathbb{E}(\tilde{z}))\mathbb{E}(\epsilon) = 0. \tag{26}$$

Finally, the last term can be transformed as:

$$2\mathbb{E}\left(\epsilon(\tilde{z}-\mathbb{E}(\tilde{z}))\right) = 2\mathbb{E}(\epsilon)\mathbb{E}(\tilde{z}-\mathbb{E}(\tilde{z})) - \mathrm{cov}(,\tilde{z}-\mathbb{E}(\tilde{z})) = 0, \tag{27}$$

where covariance term equals to zero due to independence of the error ϵ and the predicted value $\tilde{z}$. The remaining terms provide us with the following decomposition of the MSE:

$$\mathrm{MSE}(z,\tilde{z}) = \mathbb{E}(f-\mathbb{E}(\tilde{z}))^2 + \mathbb{E}(\tilde{z}-\mathbb{E}(\tilde{z}))^2 + \sigma^2 = \\ = \mathrm{Bias}^2(z,\tilde{z}) + \mathrm{Variance}(\tilde{z}) + \sigma^2. \tag{28}$$

Here the unknown function f was substituted by the actual values z. The first term of the decomposition is the squared bias, determined as a mean difference between the observed value z and the average of a model prediction $\tilde{z}$:

$$\mathrm{Bias}(z,\tilde{z}) = \mathbb{E}(z-\mathbb{E}(\tilde{z})). \tag{29}$$

This term provides an additional error due to the simplifications of the model $\tilde{z}$. The second term of the MSE decomposition reflects the mean squared deviation of the predicted value from its mean value. Both terms are basically kept under control by setting a certain complexity of the model. In contrast, the last term is called an irreducible error and can not be altered or reduced while creating the model.

As it was mentioned, the MSE of the training set tends to decrease with the model complexity. However, the minimization of the test MSE is defined by the tradeoff of the corresponding bias and variance terms. For higher complexity the average difference between the observed values and their predictions for the test set tends to decrease, resulting in the decreasing squared bias. On the other hand, higher complexity results in a larger variability for the predicted values, meaning that the variance tends to increase. Practically, the overfitting is observed for the models providing high variance and low bias for the predicted value. High bias and low variance models, on the contrary, correspond to the simpler models with low complexity which might demonstrate underfitting. Such models are unable to describe all features of the fitted data and result in a relatively high MSE for both the test and training data. In the perspective of abovementioned MSE decomposition, its minimization for the test data is always related to the bias-variance tradeoff: it is impossible to minimize both values simultaneously. For the present project the MSE will be considered a function of both the polynomial degree (complexity) and the hyperparameter λ for the ridge and LASSO regression methods. In the most general case, search of the minimal test MSE should be performed with the grid search in a two dimensional space of λ and polynomial degree. Such search will be presented for the case of the terrain data in sec.6.

3. STUDIED MODELS

This project aims at establishing the model, fitting the proposed data by using three methods of linear regression, the OLS, Ridge and LASSO. All models set by these methods will be subsequently assessed by combining the analysis with two resampling techniques, the bootstrap and k-fold cross-validation, as described in the previous section. The first part of the project deals with the simple Franke's function, widely applied to test various fitting and interpolation algorithms. The function is denoted by $z = f(x,y)$ and could be written as a weighted sum of 4 exponential functions of variables $x \in [0,1]$ and $y \in [0,1]$:

$$\begin{aligned} z(x,y) =& \frac{3}{4}exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \\ &+\frac{3}{4}exp\left(-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}\right) + \\ &+\frac{1}{2}exp\left(-\frac{9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \\ &-\frac{1}{5}exp\left(-(9x-4)^2 - (9y-7)^2\right). \end{aligned} \tag{30}$$

The fit will be performed in terms of different polynomial degrees of variables x and y in form of $1, x, y, xy, x^2, y^2, x^2y, xy^2, x^3, y3, ...$ Practically, this is performed by creating a grid of equally spaced pairs (x_i, y_i) with $x \in [0,1]$ and $y \in [0,1]$, $i = 0, ..., N-1$. The number of data points $N \times N$ is kept fixed with $N = 20$ throughout the comparison of different methods and set to $N = 10, 20, 30$ for each resampling technique separately in order to study how the increasing number of input parameters (increasing dimension of the design matrix) affects the mean squared errors for the predictions on both test and train datasets. The first step of the analysis implies building up the design matrix $\mathbf{X}$, which, in a case of polynomial degree p, will have a dimension of $(N \times N) \times m$, with $m = \frac{(p+1)(p+2)}{2}$. The explicit form of the design matrix will be the following:

$$\mathbf{X} = \begin{bmatrix} 1 & x_0 & y_0 & x_0^2 & y_0^2 & x_0y_0 & \dots \\ 1 & x_0 & y_1 & x_0^2 & y_1^2 & x_0y_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N-1} & y_{N-1} & x_{N-1}^2 & y_{N-1}^2 & x_{N-1}y_{N-1} & \dots \end{bmatrix}. \tag{31}$$

Before creating the design matrix, the generated dataset was scaled in order to improve stability of a future fit. The design matrix created with the scaled x and y variables as well as the vector of the corresponding z values are split into the test and training subsets. Throughout the whole work 20% of data were used to asses the performance of the fit created on the remaining 80% of the training data. All three regression methods were applied in order to fit a given data subset (see sec.2.1.1,2.1.2,2.1.3) with in-built resampling techniques.

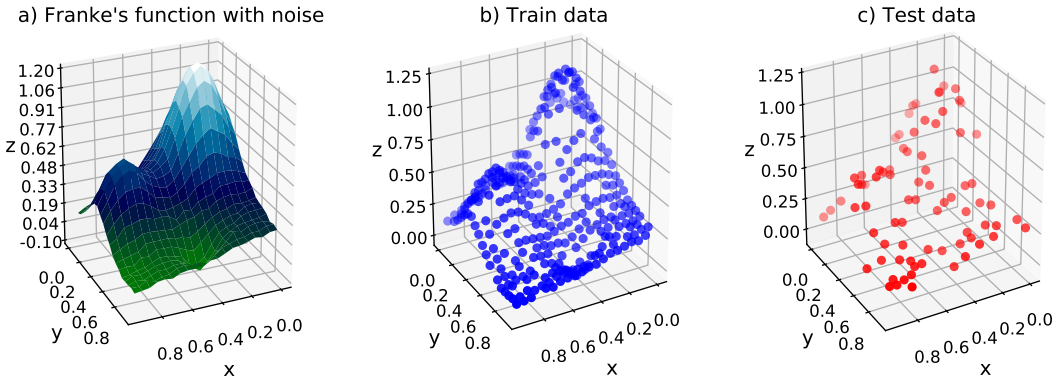

FIG. 3.— Franke's function, test and training daatasets in case of $N = 20$, and total number of (x_i, y_i) data pairs $N \times N = 400$.

All above-mentioned procedures were initially tested for the Franke's function and applied to the real, more complex, terrain data. This dataset presents a landscape over a region of Møsvatn Austfjell and was taken from (4).

The performance of the model (performed fit) in both cases was assessed by introducing and analysing two statistical functions, the mean squared error (MSE) and the coefficient of determination, denoted by R^2 score. Here we present the recap of the definitions for these functions:

$$\mathrm{MSE}(z, \tilde{z}) = \frac{\sum_{i=0}^{N\times N-1}(z_i - \tilde{z}_i)^2}{N \times N}, \tag{32}$$

which presents the sum of squared residuals and, if taken for the training data, coincides with the cost function for the OLS or the ridge and LASSO regression methods (with the corresponding conditions limiting the sum of squared β (ridge) or their absolute values (LASSO)). MSE is a non-negative value or, in presented cases, function of the model complexity or λ. Smaller values of the MSE correlate with the better performance of the model, or the fit. In the present work the MSE for both the test and train data sets are compared.

The coefficient of determination can be written as:

$$R^2(z, \tilde{z}) = 1 - \frac{\sum_{i=0}^{N\times N}(z - \tilde{z})^2}{\sum_{i=0}^{N\times N}(z - \bar{z})^2}, \tag{33}$$

where $\bar{z}$ is the mean value of z. This score reflects how well the built model $\tilde{z}$ fits the data as compared to the baseline model, predicting the mean value $\bar{z}$ for each z_i. In case of $R^2 = 0$, the model gives exactly the same prediction as the baseline model. In case of the best fit, the predicted values are close to the observed values and $R^2 \approx 1$. Finally, the predictions worse than the average prediction result in the negative R^2 score. By the value of R^2 approaching 1 one can judge on the improvement of the fit while tuning the parameters of the model.

4. CODE

The study of all three regression methods combined with the bootstrap and cross-validation resampling techniques was performed with the set of Python3 programs which can be accessed from Appendix.A (10). All steps of calculations for the first part of the project are collected in the *main.py* file which is based on a fork-like structure. A user sets the options for the different routines to be initialized (resampling technique, regression method, study of complexity-dependence or λ-dependence) and the program is run with the current settings. The standard python packages are imported together with the functions from the side files *regression_methods.py*, *resampling_methods.py*, *statistical_functions.py* and *data_processing.py*. Firstly, the data are generated by setting variables $\mathbf{X}$ and $\mathbf{Y}$ and setting the Franke's function using functionalities from *data_processing.py*. The latter file contains setting of the function to be fit, the corresponding design matrix for a given polynomial degree and adding a user-defined noise. Before performing the statistical analysis, the data are scaled by means of *DataScaling(...)* function included into the data processing file in order to increase a stability of the performed fit. This function exploits the in-built *scikit−learn* functionality *preprocessing.StandardScaler()* which sets a scaler used further for rescaling of the data after all calculations are performed. The latter procedure is required to perform plotting of the correctly scaled fit alongside the original Franke's function.

Since one of the main tasks of the project is to com-

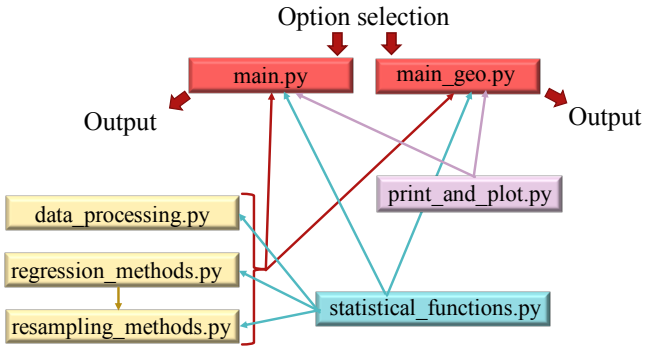

FIG. 4.— Principal scheme of the code used for the project.

pare the bootstrap and k-fold cross-validation technique, set of calculations are either performed for the first or the second resampling option, depending on the users's choice. For each choice the user defines the parameter with respect to which the test and train MSE, test variance test bias and R^2 scores will be studied (λ or polynomial degree dependence). The calculations are combined in three functions - *Bootstrap(...)*,*Cross_Validation(...)* and *NoResampling(...)*. Splitting of the input data into the training and test datasets is included in the *main.py* file and performed before running the bootstrap function instead of including it into the bootstrap function. The splitting is done with the *scikit-learn* functionality *train_test_split()* which performs random splitting into the test and train subset according to the defined fraction of the data to be distributed into the test subset. When the MSE dependence on λ is studied, we wanted to make sure that the only variable parameter is λ and the only randomness is present while selecting the resampled training data on each bootstrap iteration with no redistributing into the test and train subset for each new value of λ. The example code for the cross-validation method is presented below:

```
def Cross_Validation(nrk, x_scaled, y_scaled,
    z_scaled , lamda, pol_deg, regression_method
    ):
# Setting a current design matrix
  X = DesignMatrix(x_scaled,y_scaled,pol_deg)

# Data shuffling
  shuffled_indices = np.arange(len(x_scaled))
  np.random.shuffle(shuffled_indices)
  shuffled_matrix = np.zeros(X.shape)
  shuffled_z = np.zeros(len(x_scaled))
  for i in range(len(x_scaled)):
    shuffled_matrix[i] = X[shuffled_indices[i]]
    shuffled_z[i] = z_scaled[shuffled_indices[i
    ]]

# Splitting into k batches
  split_matrix = np.split(shuffled_matrix,nrk)
  split_z = np.split(shuffled_z,nrk)

# Setting empty arrays
  MSError_test_CV= np.zeros(nrk)
  MSError_train_CV= np.zeros(nrk)
  R2_curr = np.zeros(nrk)
  z_plot_stored = np.zeros(len(z_scaled))

  for k in range(nrk):

 # Setting the test data
    X_test = split_matrix[k]
    z_test = split_z[k]

        # Setting the training data
    X_train = split_matrix
    z_train = split_z
    X_train = np.delete(X_train,k,0)
    z_train = np.delete(z_train,k,0)
    X_train = np.concatenate(X_train)
    z_train = np.ravel(z_train)

    # Performing regression
    if(regression_method == "Ridge"):
      beta = Ridge_beta(X_train, z_train, lamda,
     pol_deg)
      z_tilde = Ridge(X_train, z_train, X_train,
     lamda, pol_deg)
      z_predict = Ridge(X_train, z_train, X_test
    , lamda, pol_deg)
      z_plot = Ridge(X_train, z_train, X, lamda,
     pol_deg)
    elif(regression_method == "LASSO"):
      beta = LASSO_SKL_beta(X_train, z_train,
    lamda)
      z_tilde = LASSO_SKL(X_train, z_train,
    X_train, lamda)
      z_predict = LASSO_SKL(X_train, z_train,
    X_test, lamda)
      z_plot = LASSO_SKL(X_train, z_train, X,
    lamda)
    elif(regression_method == "OLS"):
      beta = OLS_beta(X_train, z_train)
      z_tilde = OLS(X_train, z_train, X_train)
      z_predict = OLS(X_train, z_train, X_test)
      z_plot = OLS(X_train, z_train, X)
    else:
      print("Don't forget to pick regression
    method!")

    # Collecting the scores
    z_plot_stored = z_plot_stored + z_plot
    R2_curr[k] = R2(z_test,z_predict)
    MSError_test_CV[k] = MSE(z_predict,z_test)
    MSError_train_CV[k] = MSE(z_tilde,z_train)

  return np.mean(MSError_test_CV), np.mean(
    MSError_train_CV), np.mean(R2_curr),
    z_plot_stored/nrk
```

Inside each resampling function, one of three regression methods is used to make a prediction of a trained model on the test data. This is done by choosing one of the options *OLS(...)* (*OLS_beta(...)*), *Ridge(...)* (*Ridge_beta(...)*) or *LASSO_SKL(...)*(*LASSO_SKL_beta(...)*). For the OLS and ridge, the regression coefficients and an estimate of the predicted value are written explicitly as in sec.2.1.1 and sec.2.1.2, and for the LASSO regression the *scikit-learn* functionality *linear_model.Lasso()* with 10^6 iterations was used. Similar *scikit-learn* functionality for ridge and OLS were also included in the program and tested against the self-made ridge and OLS functions. All statistical functions studied in this project, namely the MSE, R^2, variance and bias are written explicitly and collected in *statistical_functions.py*. In addition, the variance of the regression parameters can be calulated with *BetaVar(...)*. Addressing the issue of a matrix inversion, which becomes significant for the increasing polynomial number and number of input data points, the *np.linalg.pinv()* was used to avoid this issue. This function returns the generalized inverse of a matrix produced with its singular-value decomposition (). This was also performed by using *scipy.linalg.svd()*, but all calculations were performed with the former functionality. All results are written to files and plotted by using the functions collected in the *print_and_plot.py* file.

The work on the second part of the project including the terrain data was performed in the separated *main_geo.py* file and can be characterized by the linear order: the required parts of the analysis are performed in a row and end up with plotting of the collected results. The terrain data are read out from the *SRTM_data_Norway_2.tif* file with the *SetTerrainCompression(...)* function from the data analysis set of functions. This function takes the compression factor as an argument, and both dimensions of the terrain matrix are reduced by the chosen factor. Instead of performing rebinning which will distort the original heights of

the landscape, every n-th element is kept. It was tested that the compression factor $n = 50$ (reduces dimensions from thousands to hundreds) applied preserves the general shape and the features of the landscape and decreases the amount of input data points so that all regression methods can be compared on the same footing. This is especially relevant for the search of the optimal parameter λ with the LASSO regression, as particularly small parameters might require increasing of number of iterations and considerable CPU time. After reading, the terrain data are scaled and fitted with OLS, ridge or LASSO with either 8-fold cross-validation or no resampling included. Number of fold was selected to ensure division of the data into equally-sized batches. If the *search* option is on, the search for an optimal hyperparameter will be performed for a chosen polynomial degree. User might also switch on the calculation of test and train MSE as a 2 dimensional function of the polynomial degree and λ with plotting of the surface fit corresponding to the optimal pair of parameters.

5. RESULTS

5.1. *Ordinary Least Squares on Franke's function*

As a first step, the Ordinary Least Squares (OLS) method was used on the Franke's function with added noise defined by the amplitude parameter α. Both the MSE and R^2 were calculated for polynomial degrees up to $p = 5$ and some selected values of α. The results are shown in Fig.5 (MSE) and Fig.6 (R^2). From these figures it is clear that the MSE approaches zero and R^2 approaches one as polynomial degree is increased, signifying a better fit for the higher-order polynomials. This is the general case for all noise amplitudes, $\alpha = 0.1, 0.05, 0.01$.

In Fig.7 the values of 21 regression parameters β, corresponding to a polynomial degree $p = 5$ are plotted against a number of a current parameter β. The confidence interval here is set by one standard deviation. The relative errors vary for each β_i significantly (from $\approx 1\%$ for $\beta \approx -1.536$ to $\approx 200\%$ for the smallest $\beta \approx 0.003$), but are still small in absolute values. Here we present the case for $\alpha = 0.1$ for which the errors could be seen in Fig.7.

5.2. *Bias-variance tradeoff for OLS with the bootstrap resampling technique*

In order to study the bias-variance tradeoff, the bootstrap resampling technique was implemented. First, the MSE for the train and test datasets were calculated as functions of complexity in order to estimate which polynomial degrees yielded the best fit results for the test data and presented in Fig.8. Here we see that the MSE for both test and train data approach zero as the complexity of the model is increased up to polynomial degree $p \approx 10$. For higher polynomials, the MSE for the test data increases, meaning that the fit gradually degrades

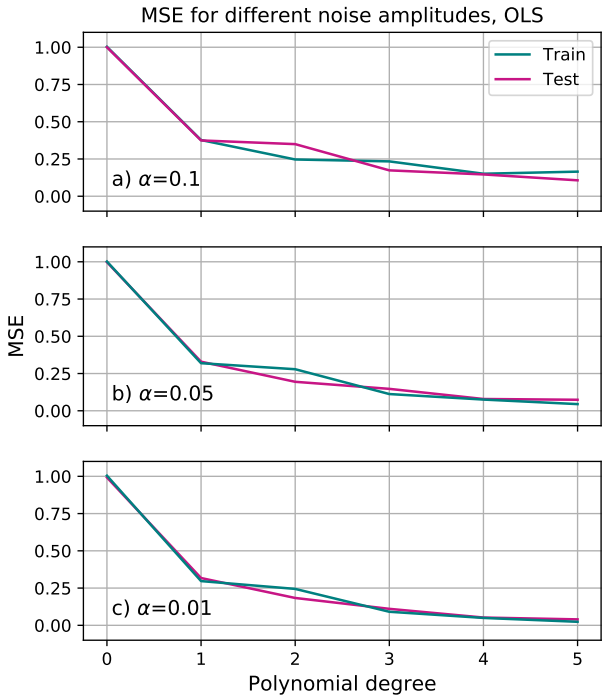

FIG. 5.— Mean squared error for the test and train datasets as a function of polynomial degree for three selected values of the noise amplitude $\alpha = 0.1, 0.05, 0.01$.

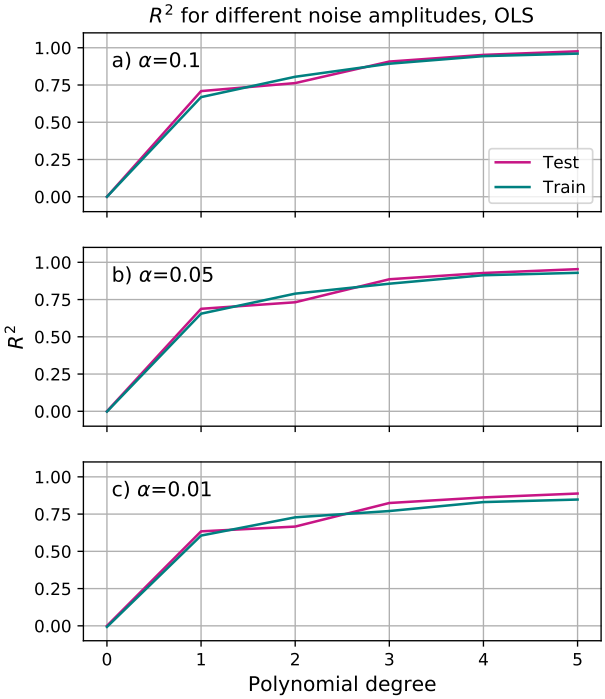

FIG. 6.— R^2 score for the test and train datasets as a function of polynomial degree for three selected values of the noise amplitude $\alpha = 0.1, 0.05, 0.01$.

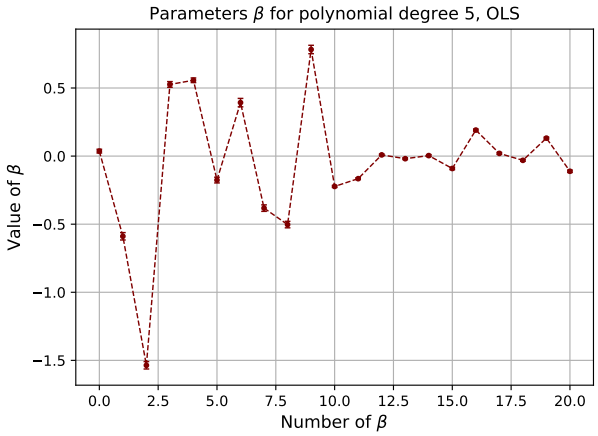

FIG. 7.— Regression coefficients β with the confidence intervals for the polynomial degree $p = 5$ and $\alpha = 0.1$. The dashed line is added for better visualization.

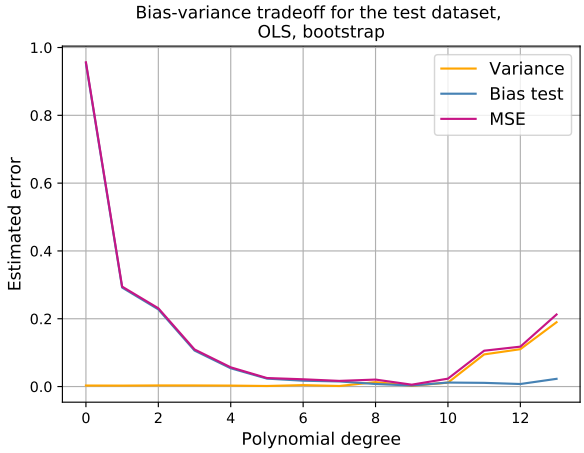

FIG. 9.— Mean squared error, bias and variance for the test dataset produced with 100 bootstraps and $\alpha = 0.01$ with the OLS regression method.

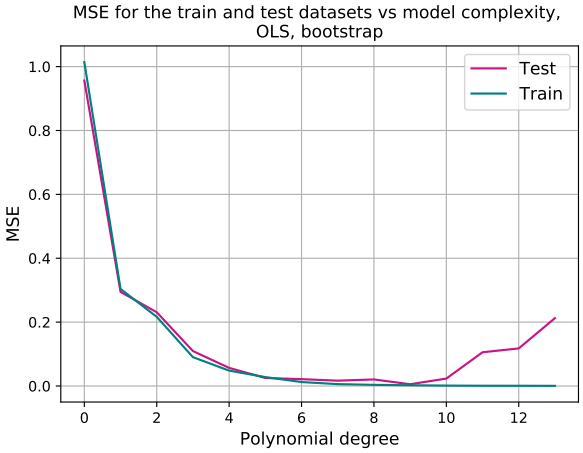

FIG. 8.— Mean squared error for the test and train datasets produced with 100 bootstraps and $\alpha = 0.01$ with the OLS regression method.

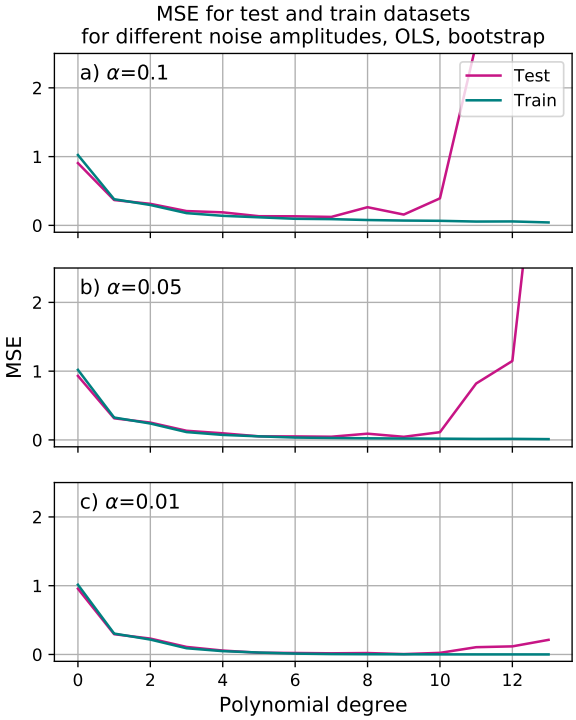

FIG. 10.— Mean squared error for the test and train datasets produced with 100 bootstraps for three selected values of $\alpha = 0.1, 0.05, 0.01$ with the OLS regression method.

for high complexities, and the test data become overfitted. This result is expected, and matches well with the prediction given in Fig.2.11 in (1).

Moving on to the bias-variance trade-off, Fig.9 shows computed variance, bias and MSE for the test data, created with 100 bootstraps and $\alpha = 0.01$. The figure illustrates the decomposition of the MSE into the sum of the variance and bias, and how the two components evolve as a function of complexity. From the plot it is evident that for low complexities the MSE is dominated by the high bias, while for higher complexities it is the variance that becomes a dominating component, and the bias approaches zero value.

One factor that changes the MSE calculations is the noise added to the data. Fig.10 shows MSE for test and train data for three different values of α. For the smallest test-value, $\alpha = 0.01$ the discrepancy between MSE for test and train data is the smallest up to relatively high polynomial degrees. For the higher α-values we see that the test data quickly get overfitted and are accompanied by relatively large MSE values.

Another factor that might affect the MSE is the number of data points in the total data set. The effect of increasing number of data points is shown in Fig.11. For a small number of data points, it is not possible to produce a fit with low MSE for both test and train data. With the increasing number of data points, the MSE for both test and train goes to zero, and will stay close to zero value up to higher polynomial degrees.

The last aspect studied in regards to the test and train MSE was the division between test and train data. In other words, we studied the MSE as the ratio between

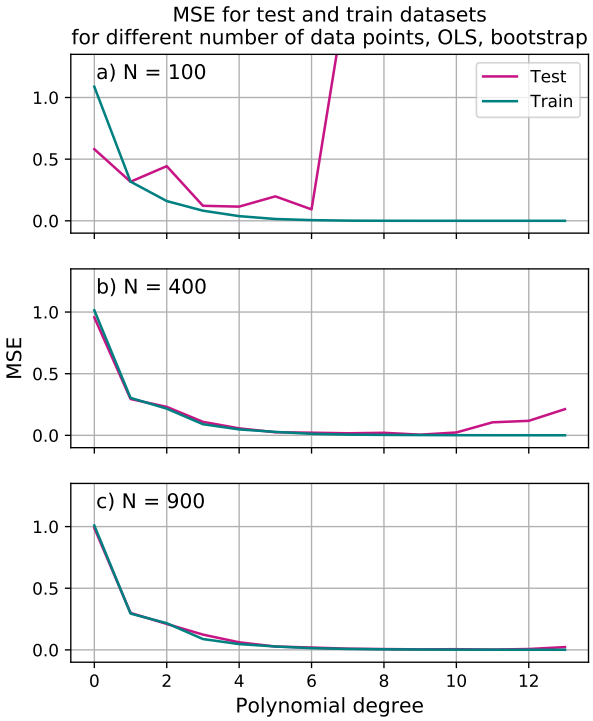

FIG. 11.— Mean squared error for the test and train datasets produced with 100 bootstraps for $N = 100, 400, 900$ with the OLS regression method. Here, the number of data points N is what was previously denoted by $N \times N$.

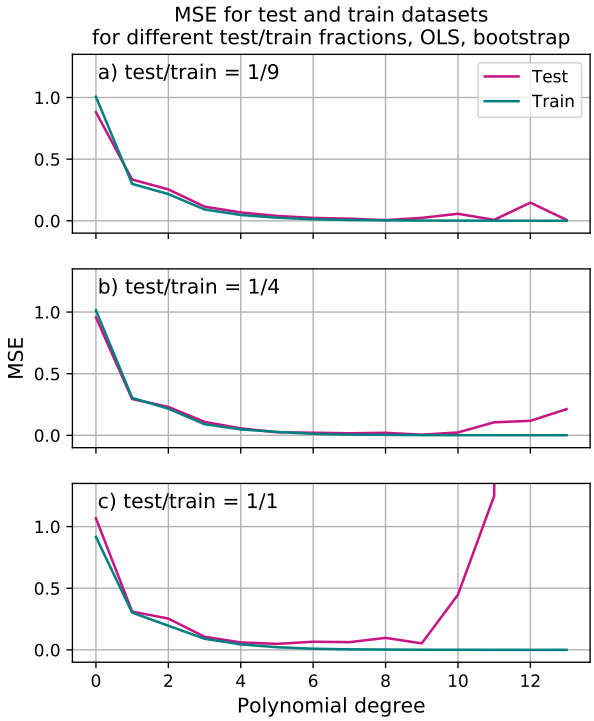

FIG. 12.— Mean squared error for the test and train datasets produced with 100 bootstraps for three different fractions of the test/-train data - 10%,20% and 50% with the OLS regression method.

test and train data was changed. The results are presented in Fig.12. The differences between test/train data of 1/9 and 1/4 are minor compared to the differences between test and train MSE for the 1/1 ratio. In the latter case, the amount of test and train data are the same, leading to a big MSE in the test data.

5.3. *The cross-validation method*

Fig.13 shows the MSE for test and train data when using the implemented cross-validation. Using different number of k-folds, the figure shows how the fit gets better when using a higher number of folds. Compared with the MSE results for the bootstrap method, we see that the cross-validation method can be used with higher degree polynomials before the test MSE begins to grow noticeably.

Table 1 summarizes the MSE and R^2 for the OLS method in three different cases: without any resampling, with the bootstrap method, and with 5-fold cross-validation. In all three cases, MSE is close to zero and R^2 is close to one, which indicates that the fit worked well for all combinations of methods.

5.4. *Ridge regression on the Franke function*

Implementing the rigde regression method and testing it on the Franke's function, we first present the test and train MSE as a function of polynomial degree, as shown

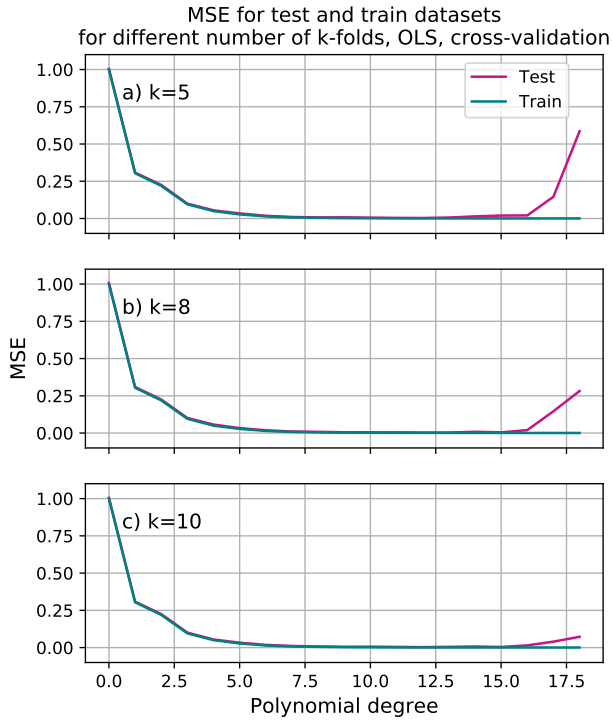

FIG. 13.— Mean squared error for the test and train datasets for $k = 5, 8, 10$ in k-fold cross-validation, OLS regression method, $\alpha = 0.01$.

TABLE 1
THE MEAN SQUARED ERROR AND R^2 SCORE FOR THE REFERENCE CASE WITH POLYNOMIAL DEGREE $p = 5$ PRODUCED WITH OLS WITH BOOTSTRAP ($N_{boot} = 100$), WITH 5-FOLD CROSS-VALIDATION AND NO RESAMPLING INCLUDED.

MSE	R^2
Without resampling	
0.033	0.961
Bootstrap with $N_{boot} = 100$	
0.039	0.960
5-fold cross-validation	
0.034	0.965

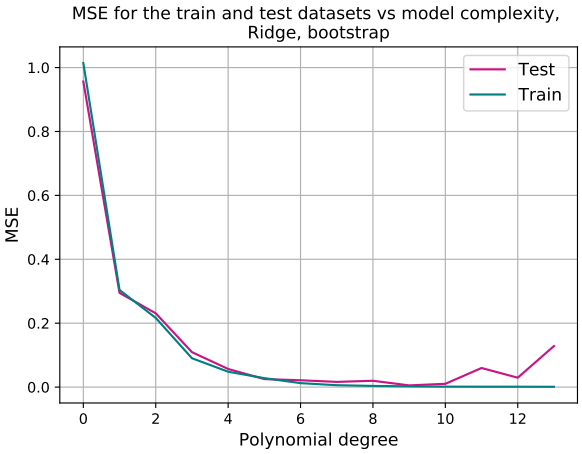

FIG. 14.— Mean squared error for the test and train datasets produced with 100 bootstraps and $\alpha = 0.01$ with the ridge regression method, $\lambda = 0.01$.

in Fig.14. Here, with $\lambda = 0.01$, it seems to be as good as the the OLS method for 100 bootstraps as discussed earlier (see Fig.8), but with slightly lower MSE-values after surpassing polynomial degree $p \approx 10$.

Similarly, Fig.15 shows variance, bias and MSE for the test data set with $\lambda = 0.01$ using the ridge regression method. Again, for these values, it is quite similar to the OLS bias-variance tradeoff as presented in Fig.9, except for slightly lower values for all types of error estimations after polynomial degree 10.

The real difference in MSE for Ridge regression becomes apparent when we study it as a function of λ, as is done in Fig.16. Here it is seen that the MSE for both test and train data remains close to zero for all λ up to 10, after which the MSE increases indicating a degrading fit.

Performing the bias-variance decomposition again, but now with varying λ-values, we obtain results presented in Fig.17. This figure shows that the variance always is close to zero using the ridge regression. The MSE is instead mainly dominated by the bias, which starts to increase for $\lambda > 10$.

In order to find out how high values of λ affect the fit, it

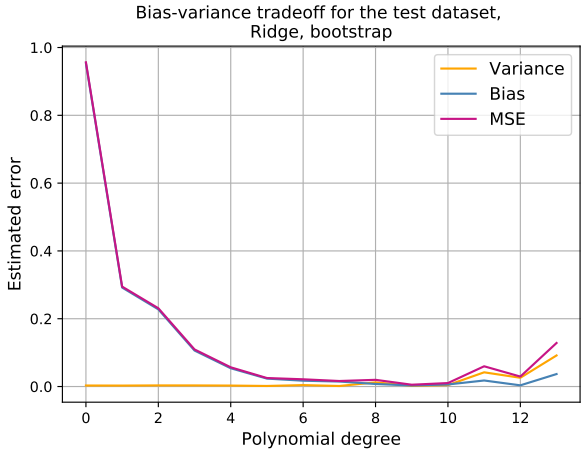

FIG. 15.— Mean squared error, bias and variance for the test dataset produced with 100 bootstraps and $\alpha = 0.01$ with the ridge regression method, $\lambda = 0.01$.

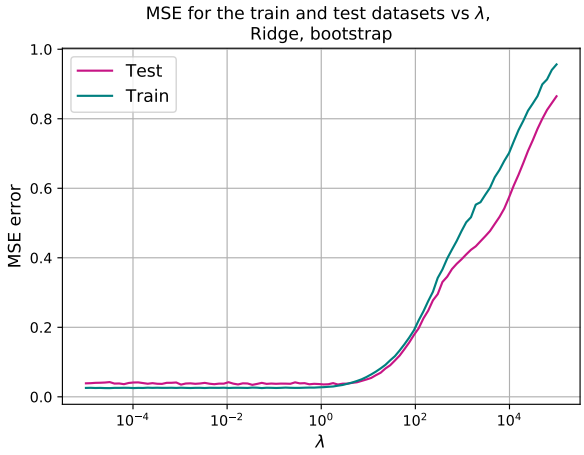

FIG. 16.— Mean squared error for the test and train datasets as functions of λ, produced with 100 bootstraps and $\alpha = 0.01$ with the ridge regression method, polynomial degree $p = 5$.

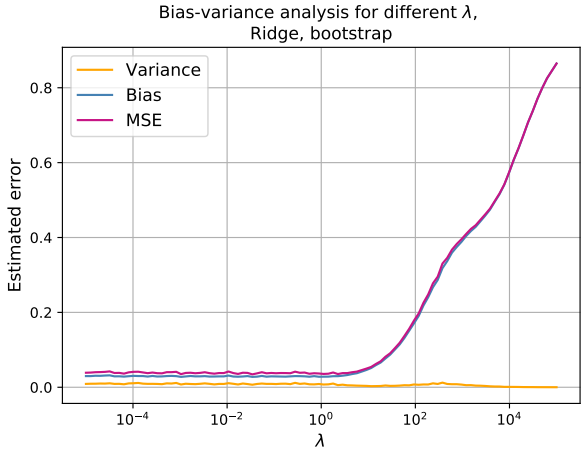

FIG. 17.— Mean squared error, bias and variance for the test dataset as a function of λ, produced with 100 bootstraps and $\alpha = 0.01$ with the ridge regression method, polynomial degree $p = 5$.

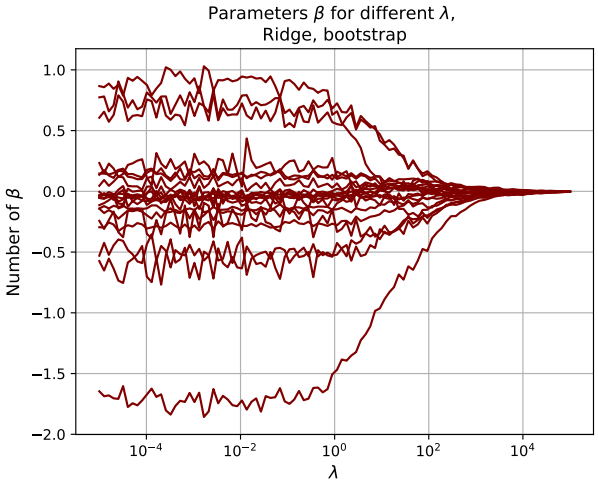

FIG. 18.— Evolution of the regression coefficients β with λ for ridge method, produced with 100 bootstraps and $\alpha = 0.01$.

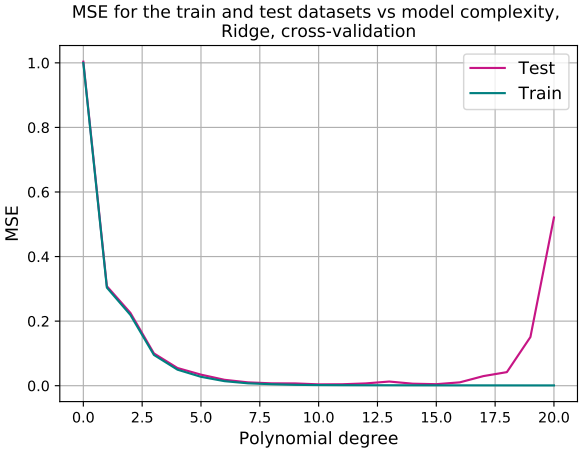

FIG. 19.— Mean squared error for the test and train datasets for 5-fold cross-validation, ridge regression method, $\alpha = 0.01$, $\lambda = 0.01$.

is possible to study the evolution of regression parameters while varying λ. This is demonstrated in Fig.18, which shows that for $\lambda > 1$, the values of β start converging to zero (getting shrunk), leading to a worse fit.

5.5. *Combining cross-validation and ridge regression*

In this section, we present the further study of ridge regression now used with cross-validation instead of the bootstrap method. To start with, the MSE for test and train data are plotted in Fig.18. The MSE stays at zero for higher degree polynomials as compared to the case of bootstrap. In order to overfit the data, complexity higher that 16th degree polynomials are needed with the cross-validation. This is significantly higher than for the bootstrap (Fig.14), where the fit could only go up to tenth order before the test MSE starts increasing.

Next is the ridge regression with cross-validation study of MSE as a function of λ, which is plotted in Fig.20. For the test set it follows the same shape as the one for ridge with bootstrap (Fig.16), but constantly stays at a slightly

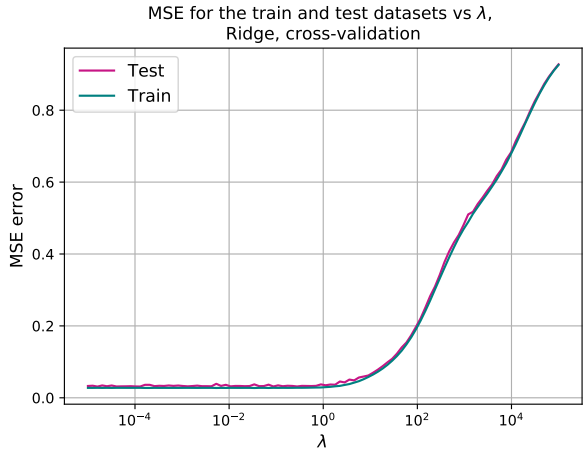

FIG. 20.— Mean squared error for the test and train datasets for 5-fold cross-validation as functions of λ, ridge regression method, $\alpha = 0.01$, polynomial degree $p = 5$.

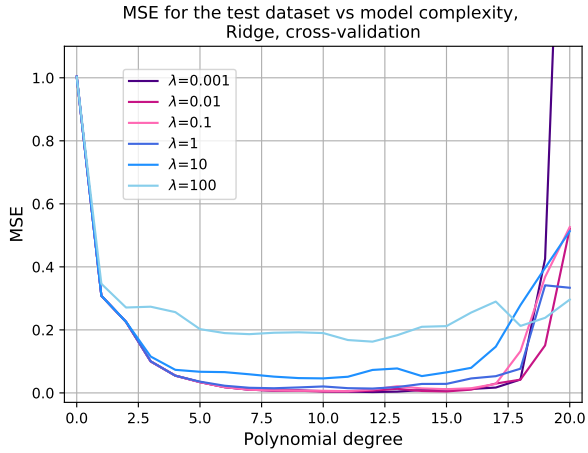

FIG. 21.— Mean squared error for the test set with the ridge regression method and 5-fold cross-validation for $\lambda = 0.001, 0.01, 0.1, 1, 10, 100$, $\alpha = 0.01$.

lower value. The cross-validation also gives MSEs which are very close for both test and train for all λ-values tested.

In order to further study how test MSE varies with both the value of the hyperparameter and complexity, we perform the study of the test MSE as a function of polynomial degree for six selected λ=0.001, 0.01, 0.1, 1, 10, 100, as shown in Fig.7. As expected from the previous studies, the lower λ values give lower MSEs. However, it is here interesting to note that for very high polynomial degrees, the bigger λ values seem to result in a more mild overfitting as compared to lower λ values.

MSE and R^2 for optimal λ are shown in Table 2 for ridge combined with the bootstrap, cross-validation, and no resampling. This also shows good results for all methods, but a λ on the order of 10^{-5} is needed for the cross-validation and no resampling case to give reliable fits.

5.6. *LASSO regression on the Franke function*

TABLE 2
THE MINIMAL MEAN SQUARED ERROR AND R^2 SCORE FOR THE REFERENCE CASE WITH POLYNOMIAL DEGREE $p = 5$ PRODUCED WITH RIDGE AND BOOTSTRAP ($N_{boot} = 100$), WITH 5-FOLD CROSS-VALIDATION AND NO RESAMPLING INCLUDED.

$\lambda_{min} \times 10^{-5}$	MSE	R^2
Without resampling		
8.111	0.023	0.974
Bootstrap with $N_{boot} = 100$		
343.047	0.034	0.964
5-fold cross-validation		
1.592	0.031	0.967

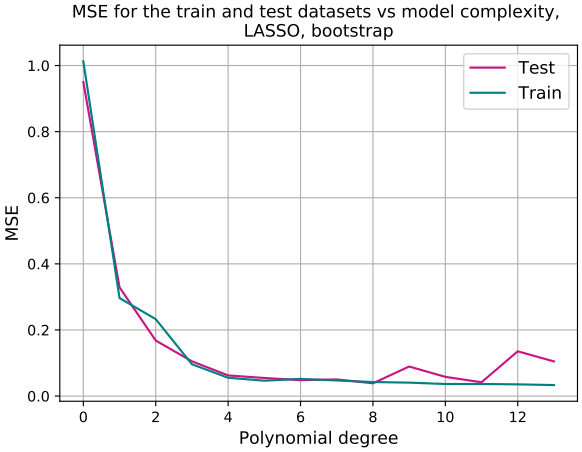

FIG. 22.— Mean squared error for the test and train datasets produced with 100 bootstraps and $\alpha = 0.01$ with the LASSO regression method, $\lambda = 0.01$.

The last regression method implemented and tested is the LASSO regression, for which the *scikit-learn* built-in LASSO regression functionality was used.

As usual, the first result presented in Fig.22 shows the MSE for the test and train data, now produced with LASSO regression and the bootstrap method. Here we keep the same value of the hyperparameter $\lambda = 0.01$ as was used in the same study with ridge. Compared to the previous methods, the current results are significantly worse for the test data, with test and train diverging already at polynomial degree 8. Similarly, the bias-variance tradeoff for LASSO as presented in Fig.23 shows that the bias decreases slower to zero for shown polynomial degrees as compared to the previous methods.

Studying the MSE as a function of λ, it becomes evident why the previously calculated MSEs for LASSO are particularly far from approaching zero. Fig.24 demonstrates MSE for the test and train data plotted for varying λ. This shows that λ must be at least smaller than 10^{-3} in order to get the MSE down to zero. The calculated MSE in Fig.22 and 23 were found with $\lambda = 0.01$, which explains bad performance of the fit.

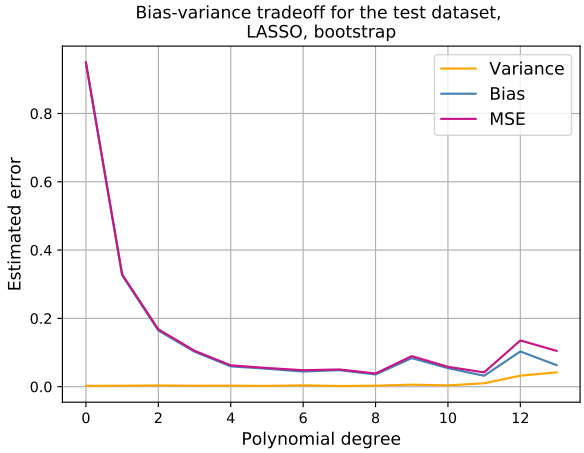

FIG. 23.— Mean squared error, bias and variance for the test dataset produced with 100 bootstraps and $\alpha = 0.01$ with the LASSO regression method, $\lambda = 0.01$.

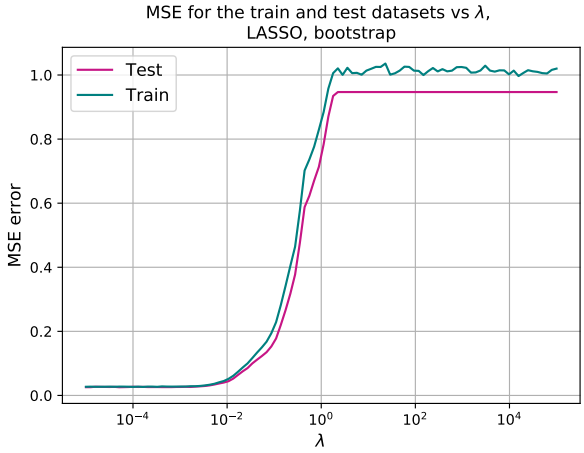

FIG. 24.— Mean squared error for the test and train datasets as functions of λ, produced with 100 bootstraps and $\alpha = 0.01$ with the LASSO regression method, polynomial degree $p = 5$.

Similarly, the bias-variance study as a function of λ in Fig.25 shows that we must use a λ smaller than at least 10^{-3} in order to achieve low MSE and bias results with LASSO regression.

This effect could be seen explicitly for $p = 5$ if regression parameters are studied as a function of λ. In Fig.26 it might be seen that β converges to 0 when $\lambda \geq 10^{-3}$. Overall, the values of β are getting shrunk faster as one moves from the lower to higher values of λ as compared to ridge.

5.7. *Combining cross-validation and LASSO regression*

Performing the LASSO regression with cross-validation, we get the MSE for test and train data as displayed in Fig.27. This shows that with cross-validation, the LASSO regression is improved as the test MSE stays close to the train MSE for higher polynomials degrees, but it stays at values relatively far from zero for $\lambda = 0.01$ as compared to the corresponding ridge case.

Studying the MSE as a function of λ again, this time

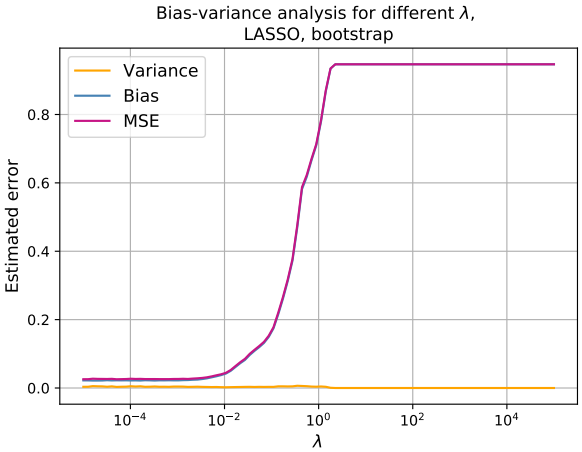

FIG. 25.— Mean squared error, bias and variance for the test dataset as a function of λ, produced with 100 bootstraps and $\alpha = 0.01$ with the LASSO regression method, polynomial degree $p = 5$.

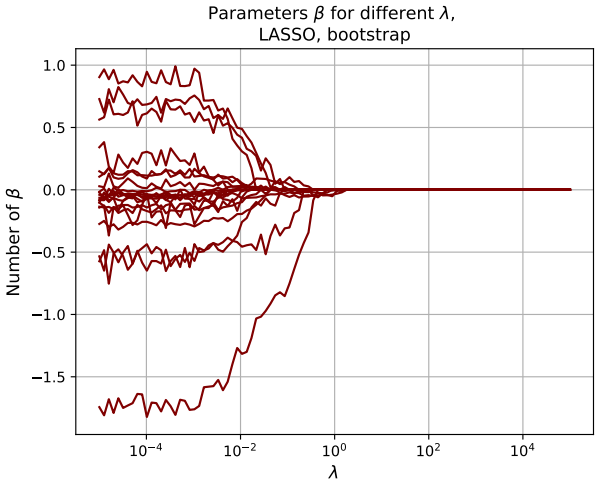

FIG. 26.— Evolution of the regression coefficients *beta* with λ for the LASSO regression method, produced with 100 bootstraps and $\alpha = 0.01$.

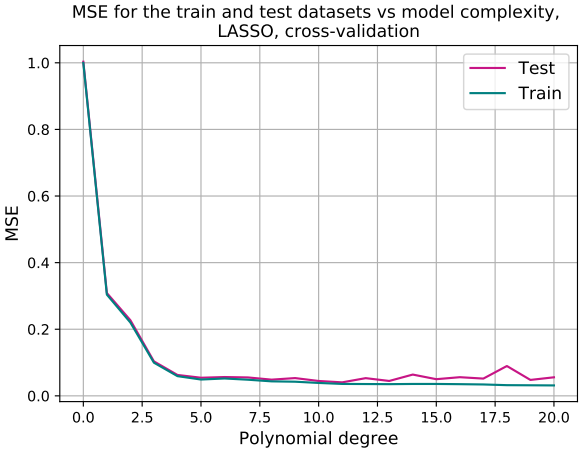

FIG. 27.— Mean squared error for the test and train datasets for 5-fold cross-validation, LASSO regression method, $\alpha = 0.01$, $\lambda = 0.01$.

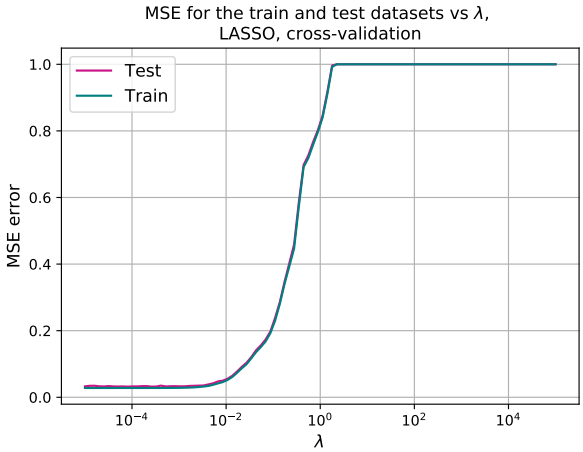

FIG. 28.— Mean squared error for the test and train datasets for 5-fold cross-validation as functions of λ, LASSO regression method, $\alpha = 0.01$, polynomial degree $p = 5$.

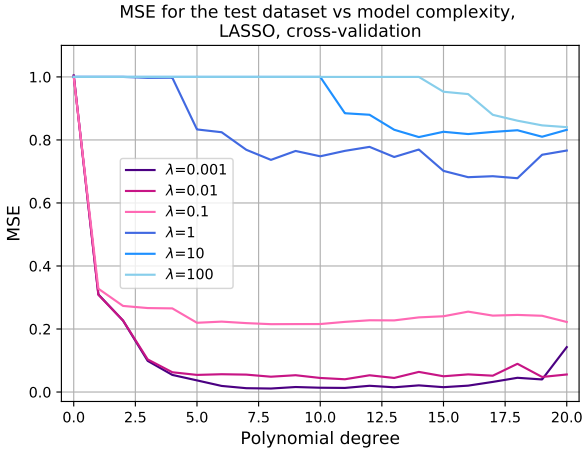

FIG. 29.— Mean squared error for the test set with the LASSO regression method and 5-fold cross-validation for $\lambda = 0.001, 0.01, 0.1, 1, 10, 100$, $\alpha = 0.01$.

with cross-validation, we get the plot shown in Fig.28. This demonstrates more or less the same as the previous study of MSE vs. λ without cross-validation, except now the train and test are particularly close to each other.

Finally, Fig.29 shows MSE as a function of polynomial degree for a number of selected λ values. Out of the chosen values, only $\lambda = 0.001$ comes relatively close to zero, while $\lambda > 0.1$ gives MSE estimates of 1 for lower polynomial degrees. In overall, $\lambda > 0.1$ should not be considered while selecting the best fitting model.

To summarize, Table 3 shows optimal λ, MSE, and R^2 for the LASSO regression for the three cases bootstrap, cross-validation, and without resampling. Based on the collected values, one can conclude that with low λ-values, LASSO regression also gives reasonable results, close to those obtained with ridge and OLS.

6. OLS, RIDGE, AND LASSO REGRESSION WITH RESAMPLING ON TERRAIN DATA

6.1. *The OLS method on terrain data*

TABLE 3
THE MINIMAL MEAN SQUARED ERROR AND R^2 SCORE FOR THE REFERENCE CASE WITH POLYNOMIAL DEGREE $p = 5$ PRODUCED WITH LASSO WITH BOOTSTRAP ($N_{boot} = 100$), WITH 5-FOLD CROSS-VALIDATION AND NO RESAMPLING INCLUDED.

$\lambda_{min} \times 10^{-5}$	MSE	R^2
Without resampling		
< 1.000	0.023	0.976
Bootstrap with $N_{boot} = 100$		
5.094	0.026	0.973
5-fold cross-validation		
1.592	0.032	0.968

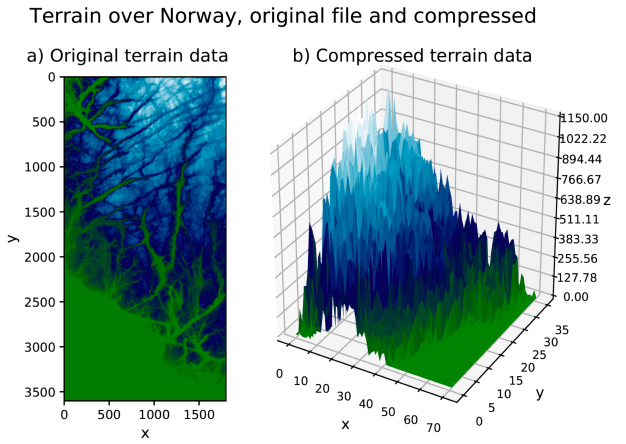

FIG. 30.— 3D-plot of the terrain data studied (a), and the compressed version of the data (b).

After having tested the code thoroughly on the Franke's function, it was applied to the real terrain data. The dataset used in this section is terrain data over a region of Møsvatn Austfjell. A graphical viewing of the original file can be seen in Fig.30 a. Since we wanted to study all three methods on the same footing, the dimensions of the terrain matrix were decreased by the factor of 50. This factor does not distort the general shape of the landscape (see Fig.30) and allows to run the LASSO regression for high polynomial degrees (at least $p > 10$) and low values of λ (at least $\lambda < 0.01$) with reasonable CPU run times.The compression is graphically shown in Fig.30 b.

Fits of the terrain data were first produced with the OLS method and with an 8-fold cross-validation, and are shown in Fig.31. The number of folds, $k = 8$, was chosen to ensure division of the data into the equally big batches. From the surface plots, it might be seen that increased polynomial degrees reproduce more details in the topography. While $p = 6$ gives the general shape of the mountains, higher polynomial degrees also show more detailed shapes.

The MSE and R^2 score for the terrain data fitted with OLS are shown in Table 4 for polynomial degree p = 10, calculated first with no resampling and later with 8-fold

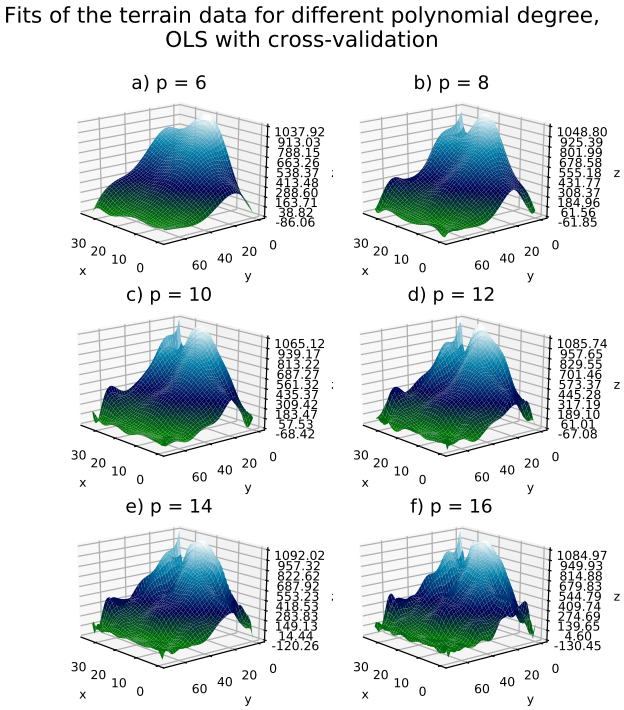

FIG. 31.— Fits performed by the OLS method with 8-fold cross-validation for different polynomial degrees.

cross-validation. For both MSE and R^2 there is a small improvement when introducing the cross-validation, but the difference is first apparent in the third decimal.

TABLE 4
THE MEAN SQUARED ERROR AND R^2 SCORE FOR TERRAIN DATA FOR THE REFERENCE CASE WITH POLYNOMIAL DEGREE $p = 10$ PRODUCED WITH OLS, 8-FOLD CROSS-VALIDATION AND NO RESAMPLING INCLUDED.

MSE	R^2
Without resampling	
0.162	0.837
8-fold cross-validation	
0.161	0.838

In Fig.32 the MSE for test and train data selected from the compressed terrain data is plotted as a function of polynomial degree, calculated with the OLS method and 8-fold cross-validation. From the figure, it might be seen that the test and train are approximately the same up to polynomial degree 10, after which they start to split, indicating that the test data have been overfitted by the algorithm.

The R^2 score for the terrain test data is plotted as a function of polynomial degree in Fig.33, both with and without cross-validation. Here it might be seen that both follow the same trend, but the R^2 calculated without resampling fluctuates more with polynomial degree than the one for cross-validation.

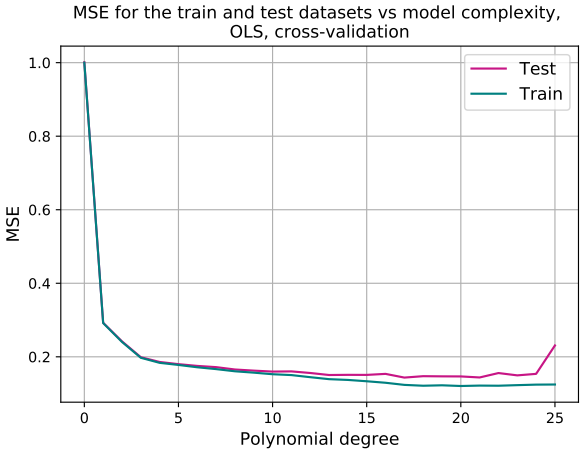

FIG. 32.— MSE for train and test datasets of the terrain data, calculated with the OLS method and 8-fold cross-validation.

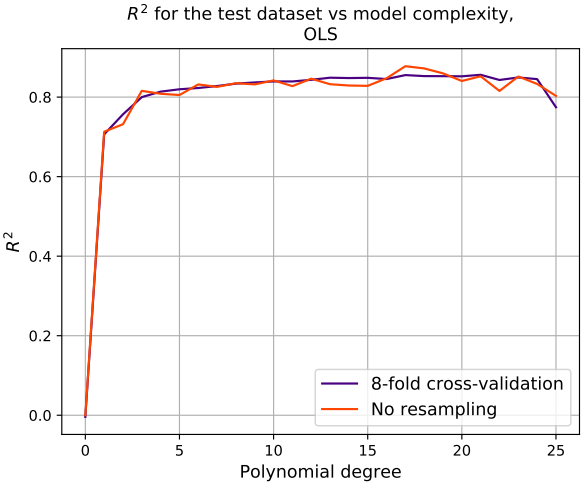

FIG. 33.— R^2 for test data of the terrain data calculated with the OLS method, both with cross-validation and without resampling.

6.2. *Ridge regression on the terrain data*

The terrain data was also studied with the ridge regression method. Fits produced for different polynomial degrees with ridge and 8-fold cross-validation and fixed value of $\lambda = 0.01$ is shown in Fig.34. Here we see the same trend as for OLS, where $p = 6$ gives the general outline of the mountains, while increasing polynomial degrees reveal more details. There are however small differences in which details are being displayed.

Table 5 shows MSE and R^2 for terrain data calculated with ridge regression and polynomial degree 10, both with and without cross-validation. Compared to the similar table for OLS, the difference between cases with and without cross-validation here is being in the second decimal.

MSE for the test and train datasets obtained with ridge regression for the terrain data as a function of polynomial degree is shown in Fig.35. Similar to OLS, the test and train MSE seem to split at polynomial degree $p \approx 10$, but with ridge the split seems to be slightly smaller meaning

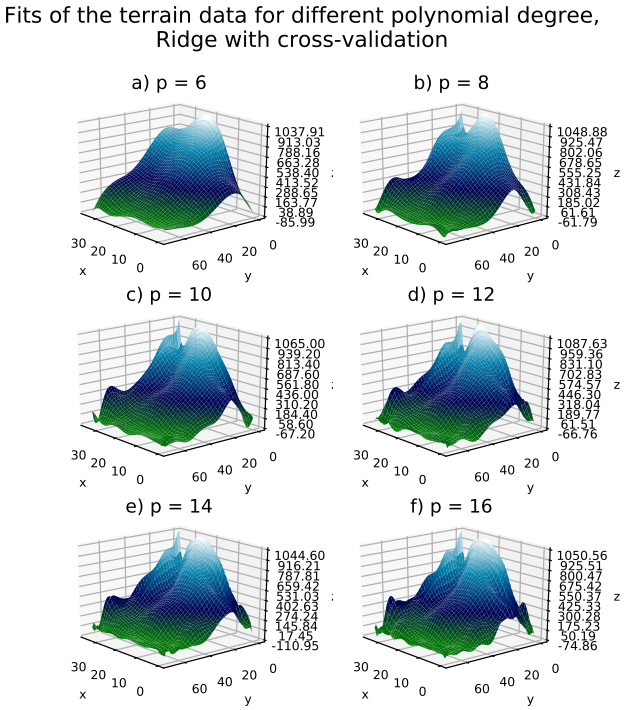

FIG. 34.— Terrain data fits performed with the ridge regression method with 8-fold cross-validation for different polynomial degrees.

TABLE 5
THE MINIMAL MEAN SQUARED ERROR AND R^2 SCORE FOR TERRAIN DATA FOR THE REFERENCE CASE WITH POLYNOMIAL DEGREE $p = 10$ PRODUCED WITH RIDGE REGRESSION, 8-FOLD CROSS-VALIDATION AND NO RESAMPLING INCLUDED.

λ_{min}	MSE	R^2
Without resampling		
0.351	0.136	0.872
8-fold cross-validation		
0.004	0.159	0.839

the data are slightly less overfitted for higher polynomial degrees.

The R^2 for the test data using ridge is shown in Fig.36. Again, the two graphs with and without cross-validation show the same trend, but more variations for the one without resampling.

6.3. *LASSO regression on the terrain data*

Lastly, the LASSO regression was applied to study the terrain data. A fit of the terrain for different polynomial degrees with $\lambda = 0.01$ using LASSO is presented in Fig.37. Here it is clear that with LASSO, significantly less details are shown in the fit, and it requires higher polynomial degrees to reproduce the main features of the terrain correctly for a given λ.

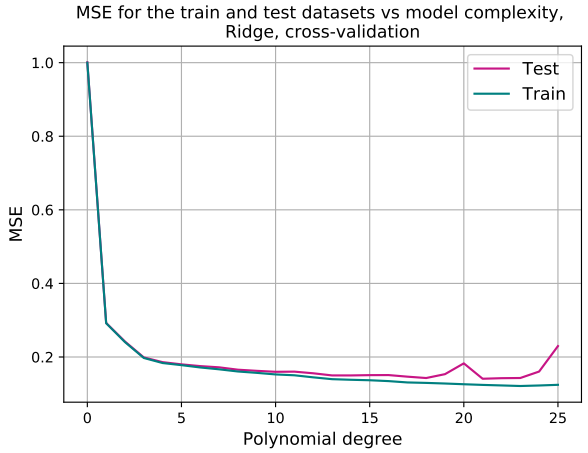

FIG. 35.— MSE for train and test datasets of the terrain data, calculated with the ridge method and 8-fold cross-validation.

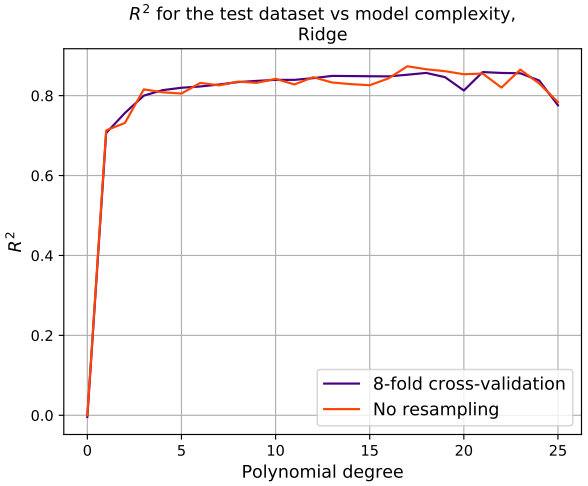

FIG. 36.— R^2 for test data of the terrain data calculated with the ridge method, both with cross-validation and without resampling.

The MSE and R^2 scores for LASSO on the terrain data are presented in Table 6. In order to receive optimal results with LASSO, a small $\lambda \approx 10^{-5}$ is required, making the simulation time significantly bigger than for OLS and ridge. However, with this small λ the MSE and R^2 get quite good, slightly worse than for ridge.

TABLE 6
THE MINIMAL MEAN SQUARED ERROR AND R^2 SCORE FOR TERRAIN DATA FOR THE REFERENCE CASE WITH POLYNOMIAL DEGREE $p = 10$ PRODUCED WITH LASSO, 8-FOLD CROSS-VALIDATION AND NO RESAMPLING INCLUDED.

$\lambda_{min} \times 10^{-5}$	MSE	R^2
Without resampling		
5.674	0.157	0.818
8-fold cross-validation		
17.586	0.160	0.838

In Fig.38, the test and train MSE for LASSO with cross-validation are presented as functions of polynomial

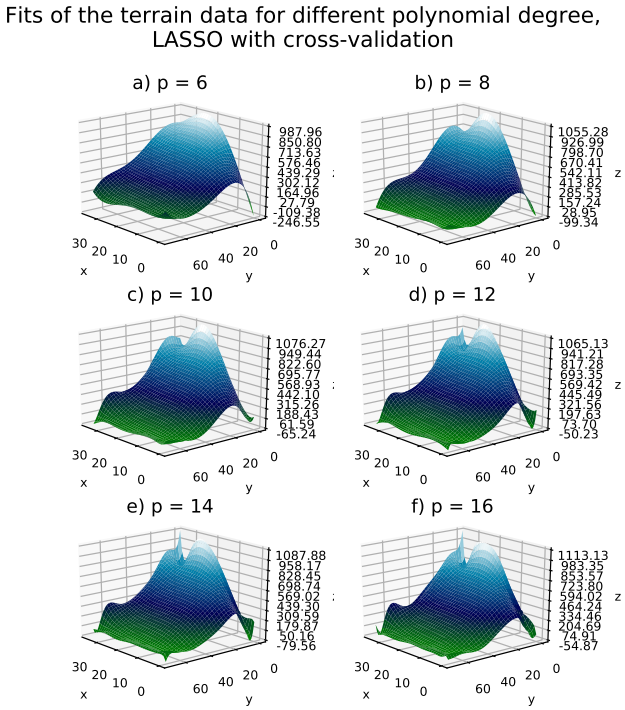

FIG. 37.— Terrain data fits performed with the LASSO regression method with 8-fold cross-validation for different polynomial degrees.

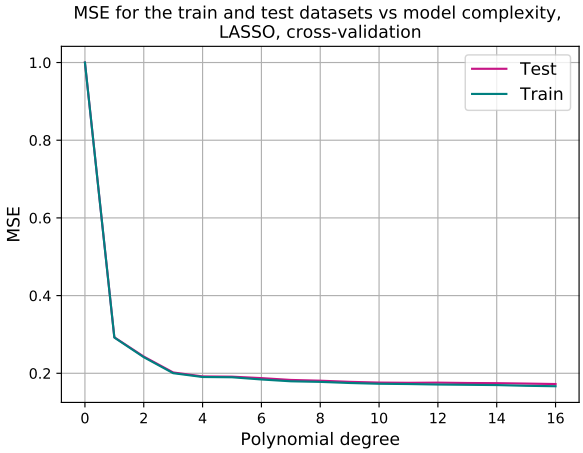

FIG. 38.— MSE for train and test datasets of the terrain data, calculated with the LASSO method and 8-fold cross-validation.

degree. The MSEs are only plotted for up to p = 16 due to the dramatically increasing CPU time it takes to calculate LASSO regression for high polynomial orders. However, it can be still seen that the test and train data slowly start to split after $p = 12$ and the overfitting occurs.

The R^2 score for LASSO with cross-validation is presented in Fig.39. Again, as in the previous cases, this plot shows how the R^2 varies from point to point if no resampling was included.

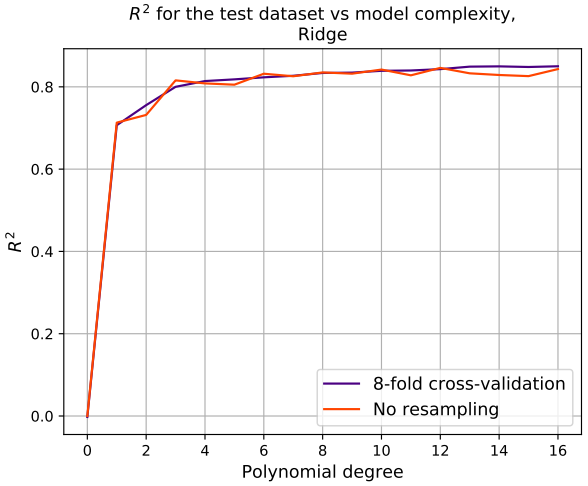

FIG. 39.— R^2 for test data of the terrain data calculated with the LASSO method, both with cross-validation and without resampling.

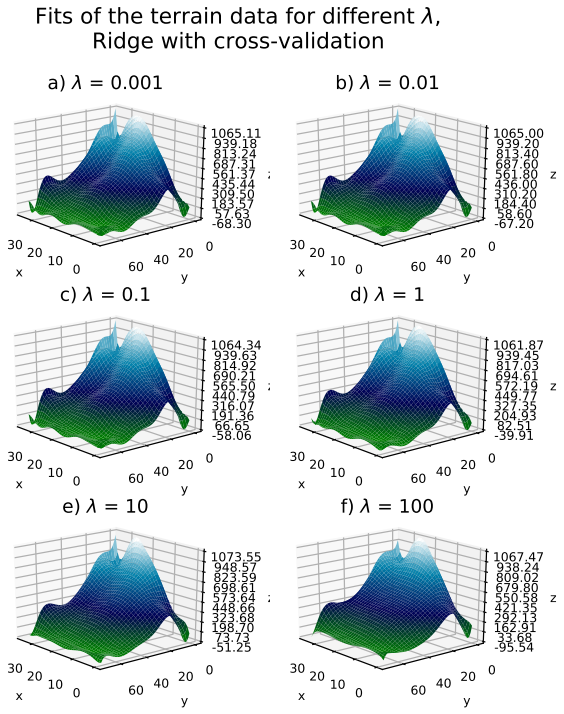

FIG. 40.— Terrain data fits performed with the ridge regression method with 8-fold cross-validation for different values of λ for polynomial degree $p = 10$.

6.4. *Further studies of the terrain data*

In this section, we explore what happens to the terrain data fits when changing λ for ridge and LASSO regression. The first case is ridge regression with cross-validation, which is plotted for different λ in Fig.40. This shows that the main features are clearly replicated for all λ-values, although more details can be included with the smaller λ-values.

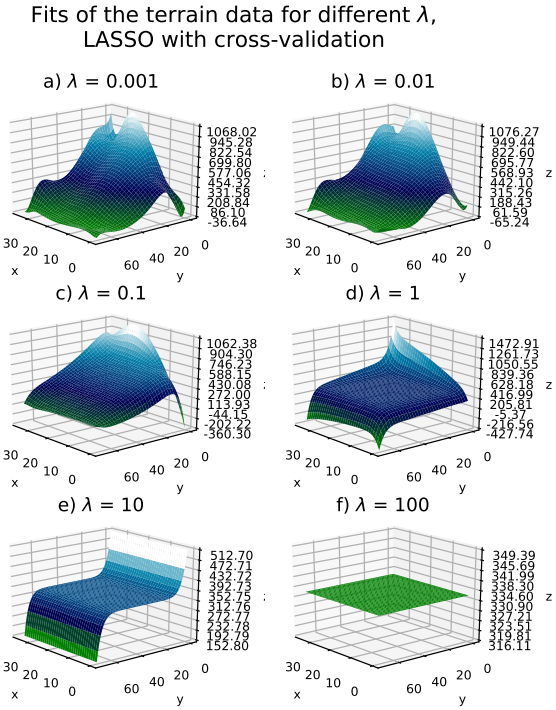

FIG. 41.— Terrain data fits performed with the LASSO regression method with 8-fold cross-validation for different values of λ for polynomial degree $p = 10$.

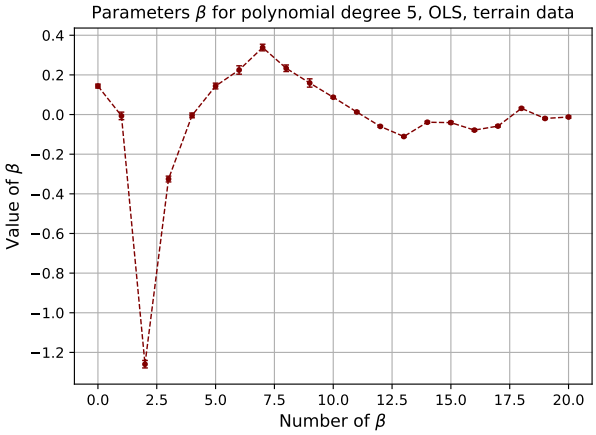

FIG. 42.— Regression coefficients β with the confidence intervals for the polynomial degree $p = 5$ produced for OLS with no cross-validation. The dashed line is added for better visualization.

This is on the other hand not the case for the LASSO regression (see Fig.41), where only $\lambda = 0.001$ and $\lambda = 0.01$ are able to clearly show the two mountain peaks. $\lambda = 0.1$ shows a vague mountain-shape, but the bigger λ-values fail at fitting the terrain in a meaningful way.

By analogy with the case of the Franke's data, the regression parameters for the $p = 5$ and OLS with no resampling applied are shown together with the corresponding error bars in Fig.42. In contrast to analytical Franke's function with a normally distributed noise, the value of σ is no longer defined by the user. It could be es-

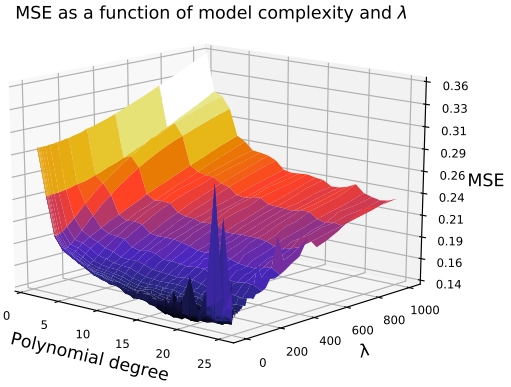

FIG. 43.— The mean squared error produced for ridge regression and 8-fold cross-validation as a function of polynomial degree and λ.

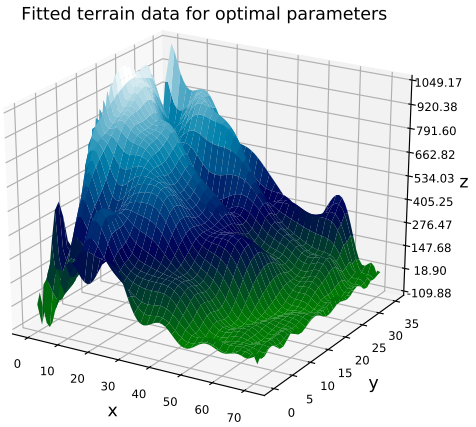

FIG. 44.— The fit surface obtained with the ridge regression method and 8-fold cross-validation for $p = 17$ and $\lambda = 9.475\times10^{-5}$.

timated according to equation 12 and implemented into equation 11 for the simplest OLS case (1).

Throughout this paper, the MSE has been studied as a function of both a polynomial degree and λ. Here we choose the ridge regression, as the method providing us with the best scores and timing performance, to study the test MSE in a two dimensional space of p and λ, shown in Fig.43. This figure clearly shows that the general shape suggests a lower λ-value and a higher polynomial degree, but only up to a certain degree. For very low λ and high polynomial degrees, the data will be overfitted and the MSE for the test data will then increase, as shown by the violet spikes in the plot. In overall, the MSE increases with λ and for the maximum polynomial degree $p = 25$ the becomes less apparent for higher λ values. The grid search in two dimensions was applied to find the polynomial degree and hyperparameter minimizing the test MSE.

Using the optimal parameters found so far, the final terrain fit is presented in Fig. 44. This fit shows the main features of the mountains as well as smaller details without falling into the problem of either under- or overfitting. It is created with ridge regression, 8-fold cross-validation with polynomial degree 18 and $\lambda = 1.194\times10^{-4}$, which were the parameters found to be optimal from the grid search shown in Fig. 43, yielding the smallest value of the test MSE= 0.137 from all values considered so far.

In order to summarize all methods applied to the terrain data, the R^2 scores were estimated for both the test and train datasets for $p = 10$ as functions of λ and shown in Fig.45.In all three cases, the test R^2 is smaller than the corresponding training R^2. The distinction is small for a given polynomial degree and would be more apparent for the region of overfitting. Both OLS and Ridge give almost coinciding scores for $\lambda < 10$. The ridge score starts slowly degrading with $\lambda > 10$. In contrast to ridge, the LASSO fit degrades fast even for relatively low values of $10^{-2} - 10^{-1}$.

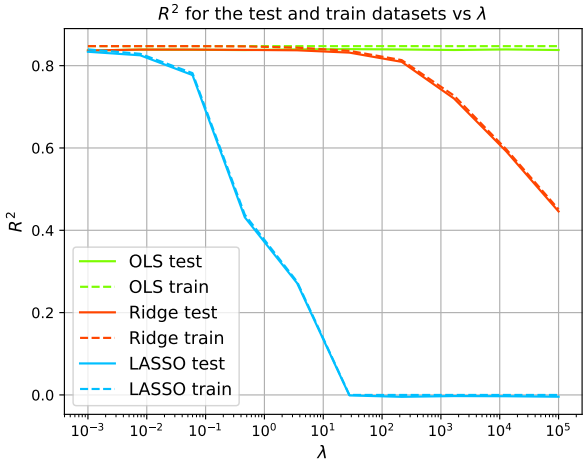

FIG. 45.— R^2 scores for the test and training datasets for OLS, Ridge and LASSO for polynomial degree $p = 10$ as functions of λ.

7. DISCUSSION OF THE RESULTS

7.1. *Discussion of the Franke's data*

The first method tested in this project is the ordinary least squares regression with no resampling applied. As it was demonstrated in Fig.5 and 6, polynomial degrees up to 4 result in a oversimplification of the model and particularly high MSE values and almost 0 values of the R^2 score ($p = 0, 1$), *i.e.* underfitting. The polynomial degrees up to $p = 5$ provide us with the models which are equally unable to reproduce the details of the function for both the training and testing stages. Here, we might expect the increasing noise to have larger effect for the higher polynomial degrees, and this is exactly the behavior demonstrated in Fig.10. Training on the data with particularly big noise (variation in the function value from point to point), if high polynomial degrees are considered, would give us a model attempting to repro-

duce these random variations of the function shape. This model is expected to fail greatly while attempting to reproduce the random features it was trained on for the test set where these features are no longer present (this is the case for $\alpha = 0.1$). The trends for R^2 scores for OLS and no resampling are correlated with the corresponding trends for the MSE, increasing up to ≈ 1 and showing how the fit is improved with a higher polynomial degree.

The problem of overfitting is demonstrated clearly for the OLS with bootstrap for p up to 13 (see Fig.8 and Fig.9). The training MSE decreases monotonously to 0 as the model gains additional complexity and adapts to the training data better. On the contrary, the test MSE demonstrates considerable increase as the predictability of the model decreases with higher complexity. As it might be seen from Fig.9, this is solely due to the rapid increase of the test variance. Increasing complexity reduces the test bias, decreasing from its maximum value, corresponding to almost 0 variance value, to 0 for maximum variance. Insignificant fluctuations of the bias (as at $p = 13$) are observed up to even higher polynomial degrees ($p = 30$), but there is no clear tendency for the test bias to increase, it remains at ≈ 0 values with the increasing p value. These fluctuations can be altered by choosing different random sequence seed, so it is assumed to be related to random splitting into the test and train data and further resampling on each bootstrap iteration.

Other factors leading to fast overfitting are the number of data point and percentage of the data used for the test of the model. As it is shown in Fig.11, the same tendency to decrease to 0 is present for the training set regardless how many data points were attributed to this dataset. However, the test MSE is affected greatly, and even relatively low polynomial degrees (*e.g.* $p = 6$) result in a dramatic overfitting. In this case, less statistics are used to test the trained model, the stronger the dependency on a particular choice of the data points in the test set. This dependence is alleviated by increasing the number of data points included in both the test and training subsets. Similarly, for the highest test/train fraction, the amount of data points in the training set decreases, and, for a given polynomial degree, the test data automatically become overfitted. This current polynomial degree is too high for the decreased amount of training points. This might also be an explanation for the previous case: the smaller N, the less points are attributed to the training set, the more overfitted the prediction on the test set would be for a given p.

Substituting the bootstrap with the 5-fold cross-validation allowed us to achieve good fitting scores by analogy with the bootstrap case. In the case of 5-folds, the overfitting problem at higher polynomial degrees becomes more noticeable, as compared to 8 and 10 folds (see Fig.13). This might also be related to how many data points are attributed to the training set. For 5-folds, 80% of data would be used for the training as compared to 90% for the 10-fold case. Therefore, for a given polynomial degree (*e.g.* $p = 17$) one might expect overfitting for the test set for a smaller number of data points used for the training. As it was mentioned in (1), high-fold test MSE results are characterized by a relatively high variance, since the training datasets are so similar to each other (becoming more severe for $k = N$). On the contrary, low-k results might suffer from an increased bias. To address this issue, one has to perform the bias-variance decomposition, which was particularly challenging (this is the reason the bias-variance study is performed with bootstrap here). The number of data points was, presumably, high enough to soften the issue of an increased bias for the 5-fold case.

All above-mentioned steps were also repeated with the ridge regression method. The same overfitting problem is quite clear for higher polynomial degrees due to monotonously increasing variance for the test dataset, as shown in Fig.14 and Fig.15. Here, $\lambda = 0.01$ was a good choice to address this issue. Further, the test MSE fluctuates over the same level from $\lambda \approx 10^5$ to $\lambda \approx 10^0$ where it swaps with the training MSE which becomes slightly larger than the test MSE at higher values of λ (see Fig.16). The test variance seems to be almost unaffected by λ, whilst all changes in the test MSE are due to the increasing bias, as shown in Fig.17. The region where both training MSE, test MSE and test bias starts increasing coincides with the beginning of the gradual converging of regression parameters to 0 (see Fig.18. As all parameters β are getting closer to 0, the model can no longer be considered well trained and able to perform reliable test fit as well. These values of λ should be disregarded while searching for the optimal fit. The same analysis was performed with the 5-fold cross-validation. It is interesting to notice that overfitting becomes apparent for the higher polynomial degrees as compared to the bootstrap. This might be related to the way the data points are sampled from the training set on each iteration. The same data points might appear more than once. This problem is eliminated for the cross-validation, and since the total number of data points, test/train fraction and noise were kept the same for the bootstrap and cross-validation case, this might be the reason for the faster overfitting in bootstrap. Similarly, the test MSE for $p = 5$ remains at almost the same level for $\lambda < 1$ (slightly higher than the training MSE) and grows fast together with the training MSE for higher λ. This is practically the same behavior as observed with the bootstrap resampling. Fig.21 demonstrates explicitly how fast the fit degrades with the increasing λ for different polynomial degrees. The general tendency is an overall increase of the test MSE for higher λ and lower degree of a polynomial resulting in a sharp overfitting. For the lowest $\lambda = 0.001$, the polynomial degree for which sharp overfitting occurs is $p = 18$.

Similar calculations were performed for the LASSO regression method. Again, Fig.22 and Fig.23 demonstrate overfitting after $p = 8$ (there is two local minima at $p = 8$ and $p = 11$). However, in contrast to ridge regression with $\lambda = 0.01$, the LASSO regression with the same λ results in a slower decrease of the test bias and slower increase of the test variance (the variance exceeds the bias at higher polynomial degrees $p > 13$). However, the dependence of the test MSE, train MSE, test bias and variance on λ is quite similar to that observed for ridge. The test MSE fluctuates at slightly higher level than train MSE below $\lambda \approx 10^{-2}$. At higher λ the train-

ing MSE becomes greater and this correlates with the fast convergence of regression coefficients to 0 at $\lambda \approx 10^{-3}$. This convergence is significantly more abrupt than one observed for ridge, and it can be seen how dramatically the fit is worsened with the increasing λ for all polynomial degrees up to 20, as shown in Fig.29. Comparing the 5-fold cross-validation results for LASSO with $\lambda = 0.01$ (see Fig.27 with the corresponding results for ridge shows that the overfitting becomes apparent for $p > 11$ and smaller value of λ should be used to achieve the same performance as for ridge.

To sum up all obtained results, let us analyse the test MSE and R^2 scores obtained for all three methods with all resampling methods for a $p = 5$ model. For OLS (see Table 1), the MSE obtained with no resampling is slightly lower (R^2 is higher) than the corresponding estimate for bootstarp with $N_{boot} = 100$. However, the former does not provide a good estimate of the MSE as it only depends on the redistribution of the data into the test and train subsets performed only once. In contrast, both bootstrap and cross-validation provide more reliable estimates averaged over the number of iterations. The test MSE for the 5-fold cross-validation is slightly smaller than that for the bootstrap case. In overall worse performance of bootstrap might be due to repetition of the same variables while resampling. For the ridge regression (see Table 2) the smallest test MSE was achieved without resampling. Again, this value is not as representative as those for bootstrap and cross-validation. As compared to the OLS results, all MSE are smaller for ridge in accordance with the theorem of Theobald (5) (there exists $\lambda > 0$ which yiels MSE($\beta(\lambda)$) <MSE($\beta(0)$)). Performance of the cross-validation is again slightly better that for the bootstrap. The result for the LASSO regression (see Table 3) for cross-validation is again smaller than for the OLS and quite close to ridge, however, for the bootstrap the test MSE is slightly smaller than for both OLS and ridge. This is the best value obtained with resampling techniques. However, with increasing polynomial degree, LASSO calculations take a sufficient CPU run time. In case of the present work we limit ourselves to searching for an optimal parameter on the interval $10^{-5}\lambda < 10^{5}$. Decreasing λ below this limit increases both the number of iterations for LASSO regression to converge and computational time.

Comparing all results, we might conclude that the resampling gives more reliable MSE estimates, so either bootstrap or cross-validation should be applied. For the polynomial degree $p = 5$ the best score was achieved for LASSO regression method in the combination with bootstrap. This combination of methods is recommended in this work for the Franke's function and $p = 5$. However, for higher polynomial degrees we might expect better performance for ridge combined with the cross-validation.

7.2. *Discussion of the terrain data*

After being tested for the simpler Franke's function, the program was applied to build up a fit for the coastal line of the real landscape. The fit performed with the OLS regression for different polynomial degrees demonstrates clearly how the model ability to reproduce the separated mountain peaks improves with the polynomial degree. However, the spikes in the fit, accompanying overfitting, is hard to trace by eye based on the fit surfaces. This was resolved again by studying the test and train MSE as a function of the polynomial degree (see Fig.32). It is interesting to notice, that the MSE remains at almost the same level for $p = 17 - 21$ and has a local minimum at $p = 17$. Overfitting seems to be apparent after a relatively high polynomial degree $p > 24$. The similar trend is reflected by the R^2 score reaching the highest value $R^2 \approx 0.85$ for $p = 17, 18$. The R^2 score is plotted together with the corresponding score obtained with no resampling technique included (see Fig.33. The latter demonstrates significant fluctuations over the cross-validation result, having multiple local minima and maxima. With 8 folds this problem is eliminated, since the score is averaged for 8 different combinations of the test and training data picked from the original dataset.

Similar analysis for the ridge regression method with $\lambda = 0.01$ results in the MSE and R^2 scores which are quite close to the OLS values. The fit surfaces for the polynomial degree up to $p = 16$ do not demonstrate any significant deviations for the OLS fit surfaces (see Fig.34). Similarly, the behavior of the test and train MSE is quite similar: overfitting becomes clear for $p > 24$, the local minimum is at $p = 21$. Analogous analysis for the LASSO regressiom method, even for a relatively high value of $\lambda = 0.01$, was more challenging. With the increasing dimension of the design matrix (higher polynomial degree), calculation of the MSE takes increasingly big run times. For this reason, the test and train MSE are shown up to $p = 16$ only. The test MSE increases slightly as compared to the training MSE for polynomial degrees above $p = 12$. The drawbacks of choosing $\lambda = 0.01$ is, however, clearly demonstrated in the fit surfaces in Fig.37: The fit is barely able to reproduce the separated details of the landscape and gives the shape similar to what might be produced for lower polynomial degree for ridge. Moreover, the procedure was not able to converge for the values $\lambda < 0.0001$ with a used number of iterations. The number of iterations should be increased by, at least, two orders of magnitude to go below $\lambda = 0.0001$ for the example case, and this noticeably increases the run time.

The way the quality of the fit degrades with increasing λ is demonstrated in Fig.40 and 41. The degrading is more dramatic for the LASSO regression: the ridge fit for $\lambda = 100$ is still able to capture the general shape as for LASSO the fit degrades to the surface with $z = const$.

One might judge on the quality of the fit and compare performance of all models on the base scores, collected in Tables 4.5 and 6. Here we fix the polynomial degree $p = 10$ in order avoid complications with run times for higher polynomial degrees with LASSO. For both Ridge and LASSO, a parameter λ minimizing the test mean squared error were found by analogy with the Franke's function. There is no clear pattern of smaller test MSE for all three methods without resampling as compared to 8-fold cross-validation. This pattern is not expected

and broken by the smaller MSE obtained with 8-fold cross-validation for OLS as compared to the case with no resampling. For both Ridge and LASSO the MSE estimates smaller than for OLS were achieved with 8-fold cross-validation. Moreover, if comparing the cases with resampling, the best value of $R^2 \approx 0.84$ is achieved for ridge with $\lambda \approx 0.004$. This method combined with the 8-fold cross-validation results in the most reliable (due to cross-validation) results in the smallest value of the test MSE and, therefore, can be recommended for for performing a grid search and the fit of real terrain data.

8. PERSPECTIVES FOR FURTHER IMPROVEMENTS

One of the major desired improvements is related to the organisation of interaction between different functions in the written program. The transfer of necessary valuables from one function to another results in a great amount of input parameters (e.g. the bootstrap function that takes $X_{train}, X_{test}, z_{train}, z_{test}$, λ, design matrix and an option for the regression method. The amount of variables could have been reduced if splitting into the train and test samples was performed within the function, but, in this case, it would lead to an additional "redistribution" while studying MSE dependence on λ. Another desired modification is to make the program more general, *i.e.* able to combine the main functions responsible for the Franke's function and the terrain data. This could have been easily performed with the given set of self-made functions, however, the calculations on the terrain data were kept separately in order to avoid additional confusion and branching over different user-defined options. In overall, the interaction between different identities (in our case functions) could be considerably improved by implementing classes, *e.g.* data class, fit (regression) class, resampling class and so on. This is planned to be performed in the next project.

9. CONCLUSIONS

The following project presents a comparison of the OLS, ridge and LASSO regression methods tested with the bootstrap resampling technique, k-fold cross-validation and no resampling included. All methods were applied to the analytical Franke's function with the added noise and the real terrain data. Both the Ridge and OLS-based models performed well on both sets of data, demonstrating relatively low MSE for the test dataset and high R^2 scores. In addition, simplicity of the ridge regression allows fast search of the parameter λ yielding MSE($\beta(\lambda)$) <MSE($\beta(0)$). Applying the resampling methods allowed us to obtain more reliable information on the MSE scores as compared to the singular distribution of the data into the test and train datasets. The 8-fold cross-validation was used to search for the optimal parameter λ and polynomial degree for the fit of the terrain data, yielding the MSE score smaller than for the corresponding OLS score. Relatively good scores were obtained for the LASSO regression applied to the Franke's function. However, the CPU time required for these calculations is considerably larger than the ridge and OLS run times. In addition, the grid search of the optimal parameters for LASSO is complicated by rapid increase of the run time with the decreasing λ. The LASSO regression was found to be the worst of all tested methods in the sense of timing performance, despite relatively good scores for both the Franke's and terrain datasets. The ridge regression method would be recommended for application to the large-scale terrain fitting with high complexity of the model involved.

10. APPENDIX A: LINK TO ALL PROGRAMS

Link to the project in Github

11. APPENDIX B: STUDY OF THE MEAN SQUARED ERROR DEPENDENCE ON NOISE, NUMBER OF DATA POINTS AND TRAIN/TEST DATASET CHOICE FOR RIDGE AND LASSO REGRESSION

The following Appendix summarizes the main results on the MSE dependence on the amplitude of the added noise, number of data points and relative fractions of the data attributed to the test and train datasets. This was performed for both the ridge and LASSO regression by analogy with the OLS study (see subsec.5.2). Practically, the same tendencies are observed: for both ridge and LASSO, larger amplitudes of the noise result in, on average, higher MSE for higher polynomial degrees. Since the irreducible error is proportional to the squared amplitude, it is natural to expect the MSE to be in general higher for the higher value of α. However, the fast increase of the MSE for $\alpha = 0.1$ for higher polynomial orders is mainly due to the fast increase of the variance between the original data and predicted values. This is also an expedited effect accompanying adding a significant random distortion (noise) to the original data with sharp variation of the z value from point to point. This effect appears to be more mild for the LASSO regression for the same value of hyperparameter λ as used for ridge. The situation might change significantly for the different values of λ, here, the results are shown for one reference case.

Similar behaviour is observed for the decreasing number of the data points: the test MSE tends to be significantly higher for the polynomial degree higher than $p = 5$ than for the case of higher statistics for (x_i, y_i). Indeed, the less data points are in the training dataset for a given polynomial degree, the more excessive it might become for further predictions on the test set, *i.e.* we might expect overfitting. This might be also explained by the fact that both variables x_i and y_i are picked randomly while forming both the train and test sets. The larger the number of data points in overall, the more data points will be drawn from the distributions of x and y and the less statistics we have for both subsets, the less would be the dependence of the obtained scored on the particular set of data points, the less the variance for the test data. This effect is equally strong for the Ridge and LASSO regression methods.

Finally, the same study of the MSE was performed for the different relative amount of the data points attributed to the test and the train subsets. As the amount

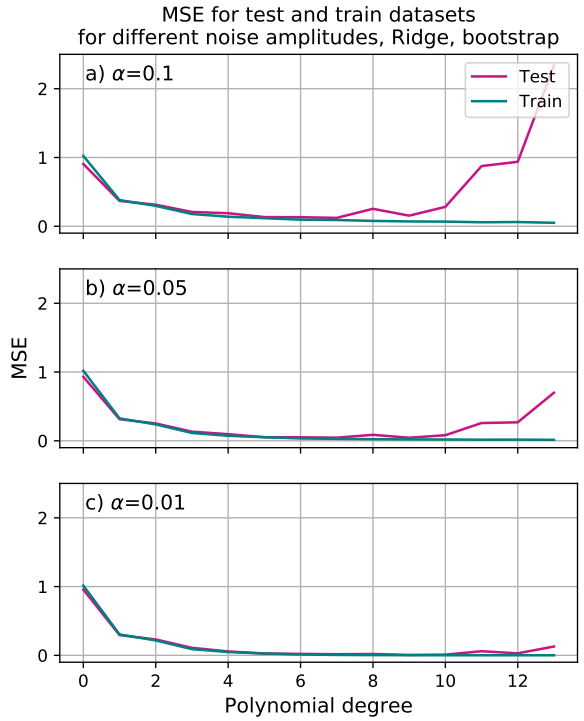

FIG. 46.— Mean squared error for the test and train datasets produced with 100 bootstraps for three selected values of $\alpha = 0.1, 0.05, 0.01$ with the ridge regression method, $\lambda = 0.01$.

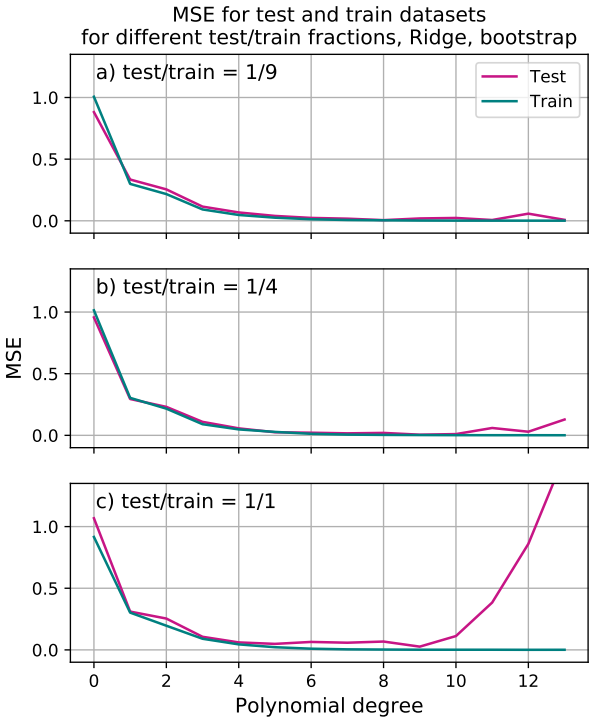

FIG. 48.— Mean squared error for the test and train datasets produced with 100 bootstraps for three different fractions of the test data - 10%,20% and 50% with the ridge regression method, $\lambda = 0.01$.

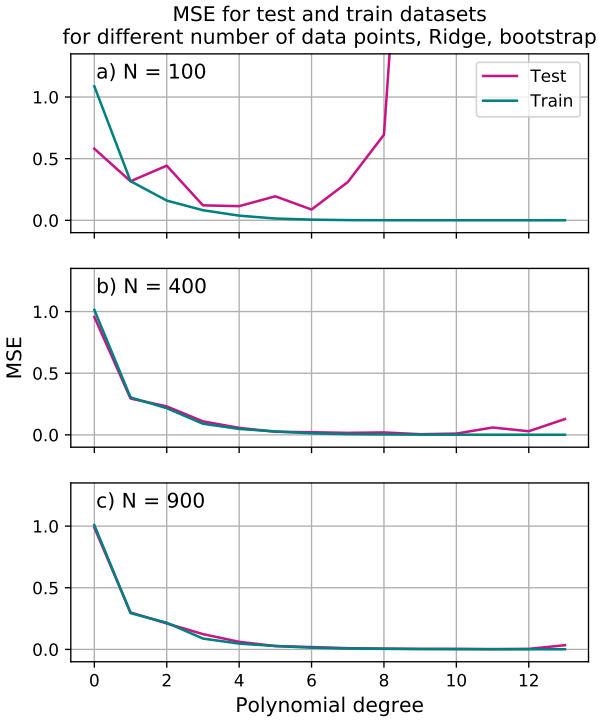

FIG. 47.— Mean squared error for the test and train datasets produced with 100 bootstraps for $N = 100, 400, 900$ with the ridge regression method, $\lambda = 0.01$.

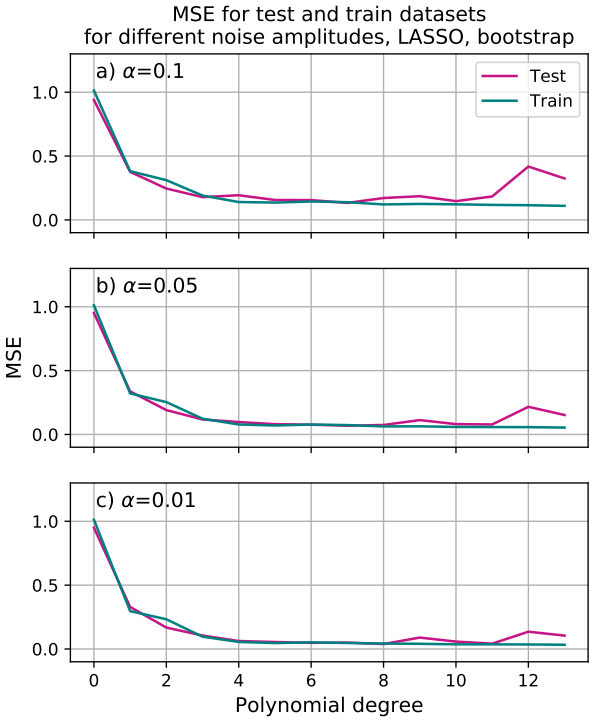

FIG. 49.— Mean squared error for the test and train datasets produced with 100 bootstraps for three selected values of $\alpha = 0.1, 0.05, 0.01$ with the LASSO regression method, $\lambda = 0.01$.

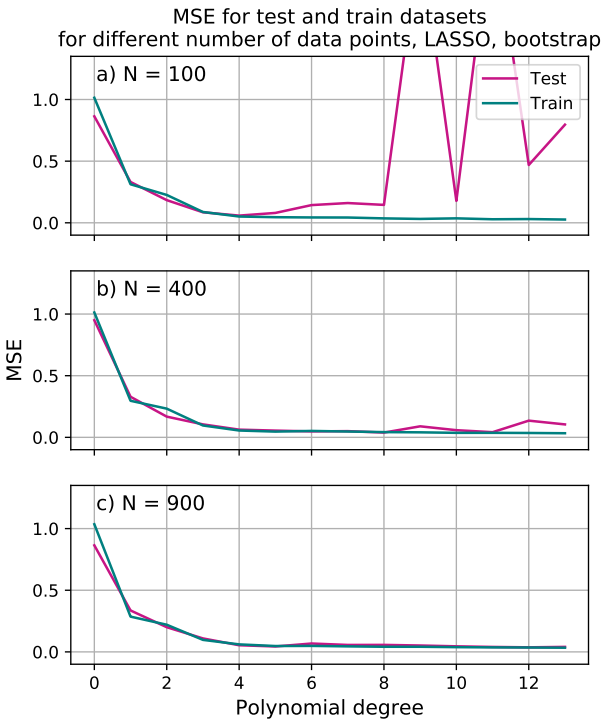

FIG. 50.— Mean squared error for the test and train datasets produced with 100 bootstraps for $N = 100, 400, 900$ with the LASSO regression method, $\lambda = 0.01$.

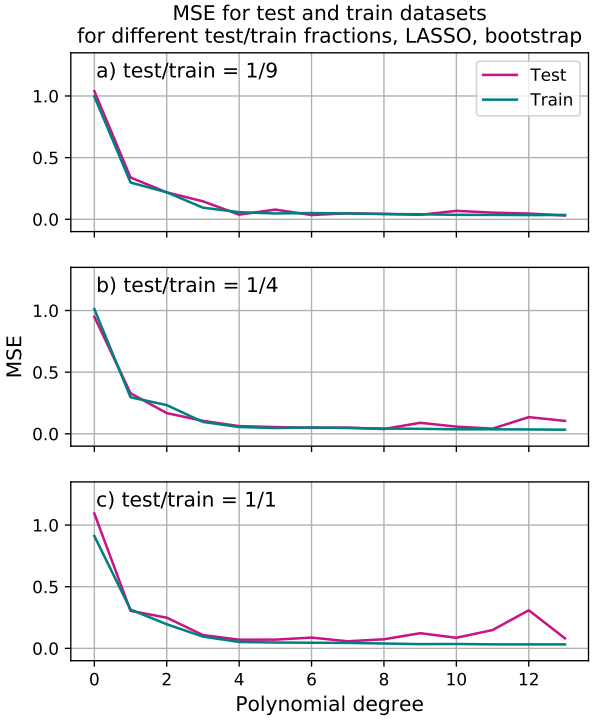

FIG. 51.— Mean squared error for the test and train datasets produced with 100 bootstraps for three different fractions of the test data - 10%,20% and 50% with the LASSO regression method, $\lambda = 0.01$.

of data decreases in the training set, the given complexity of the model becomes excessive and we might expect overfitting to happen for the remaining data in the test set. This problem appears to be equally present for the ridge and LASSO regressions.

12. APPENDIX C: FITTED TERRAIN DATA FOR OLS, RIDGE AND LASSO REGRESSION METHODS WITH NO RESAMPLING INCLUDED

The following Appendix summarizes the results on the fitting of the real data with no 8-fold cross-validation included. In overall, both three regression methods demonstrate the same tendencies for the increasing polynomial degree as in case of 8-fold cross-validation (see Fig.53, 54, 57). The difference between cases with and without resampling is minor and can not be clearly traced without studying the MSE and R^2 scores in detail. The comparison of R^2 scores for the test and train sets is demonstrated in sec.6. The corresponding MSE for no resampling for the same polynomial degree are displayed in Fig.52, 55, 56. Similarly to the R^2 scores, one might notice significant variability as one moves from lower to higher polynomial degrees. Instead of having one polynomial degree for a minimum MSE, multiple local minima might be observed. The same will be observed for each of k iterations in the 8-fold cross validation. However, for the latter, the MSE and R^2 scores will be averaged over all 8 iterations, resulting in a more smooth dependency of the test MSE on the complexity. The behaviour of the MSE shown for all three regression methods provides a good illustration for a need to use the resampling method in order to obtain more reliable fit of the real data.

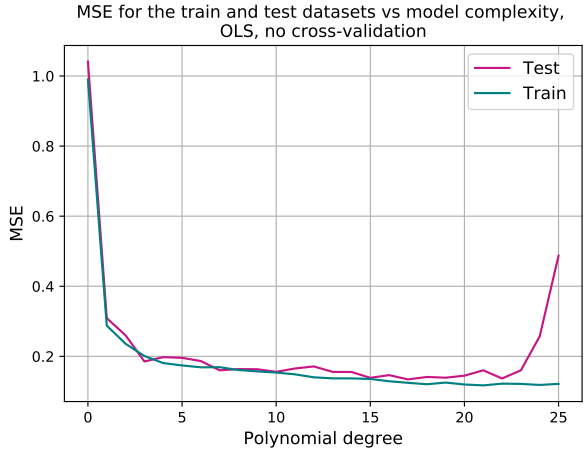

FIG. 52.— The mean squared error for the test and train datasets obtained with OSL and no resampling for $p = 10$.

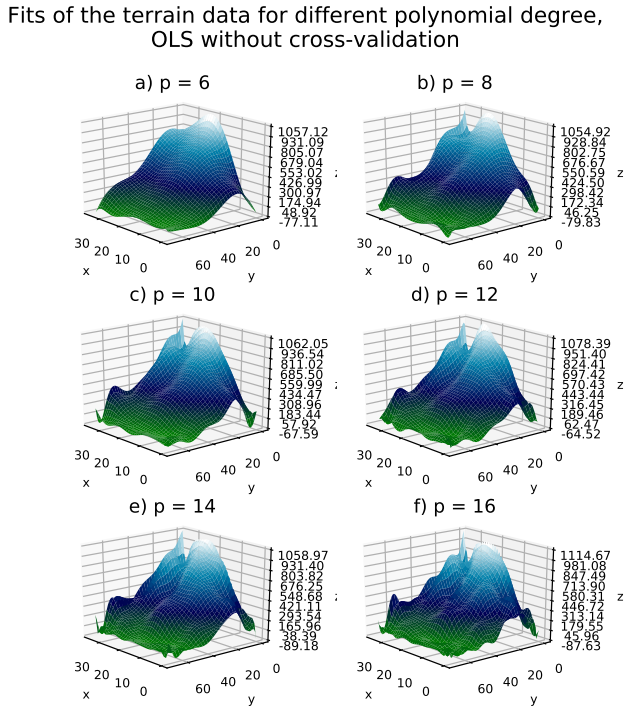

FIG. 53.— The fitted surfaces for the terrain data obtained for polynomial degrees $p = 6, 8, 10, 12, 14, 16$ and OLS with no resampling of the data.

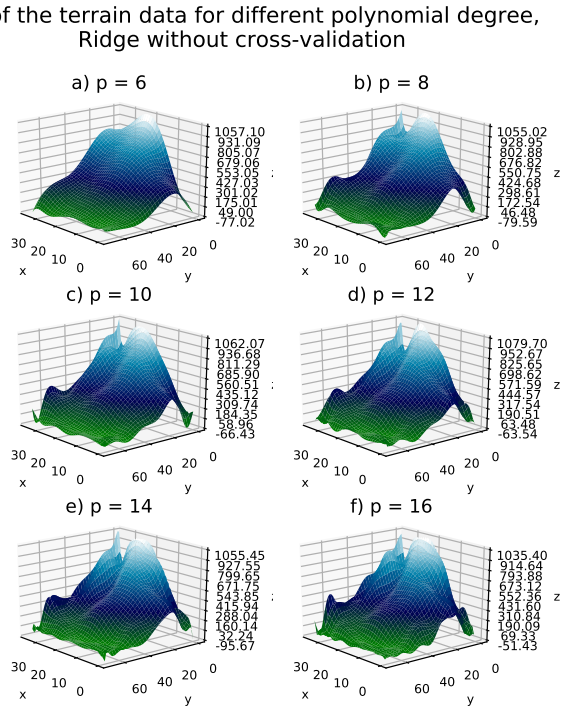

FIG. 54.— The fitted surfaces for the terrain data obtained for polynomial degrees $p = 6, 8, 10, 12, 14, 16$ and Ridge with no resampling of the data.

REFERENCES

[1]T. Hastie, R. Tibshirani, J. Friedman// The Elements of Statistical Learning, Second Edition, Springer Science+Business Media, New-York, ISBN:978-0-387-84858-7, 2016.

[2]H.J. Morten // Applied Data Analysis and Machine Learning, FYS-STK4155 - lecture notes fall 2020.

[3]B. D. Ripley// Pattern Recognition and Neural Networks, Cambridge University Press, ISBN:9780511812651, 1996.

[4]U.S. Geological Survey, https://earthexplorer.usgs.gov/

[5]Wessel N. van Wieringen// Lecture notes on ridge regression, Amsterdam Public Health research institute, Amsterdam, 2020.

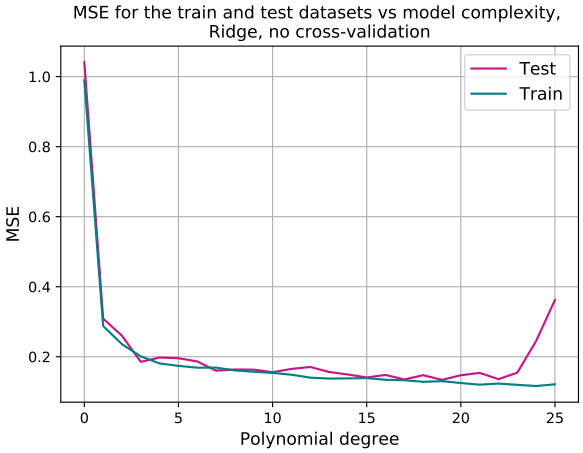

FIG. 55.— The mean squared error for the test and train datasets obtained with Ridge and no resampling for $p = 10$ and $\lambda = 0.01$.

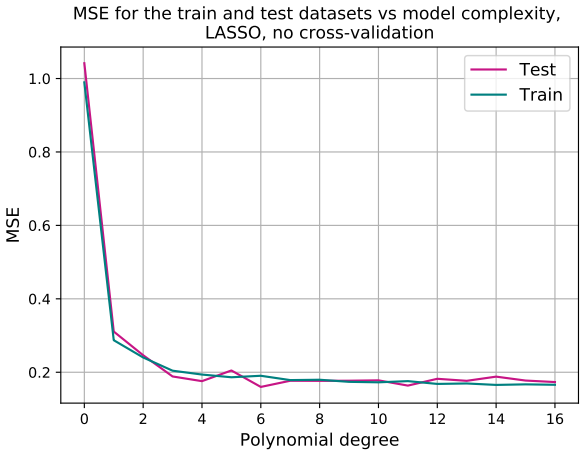

FIG. 56.— The mean squared error for the test and train datasets obtained with LASSO and no resampling for $p = 10$ and $\lambda = 0.01$.

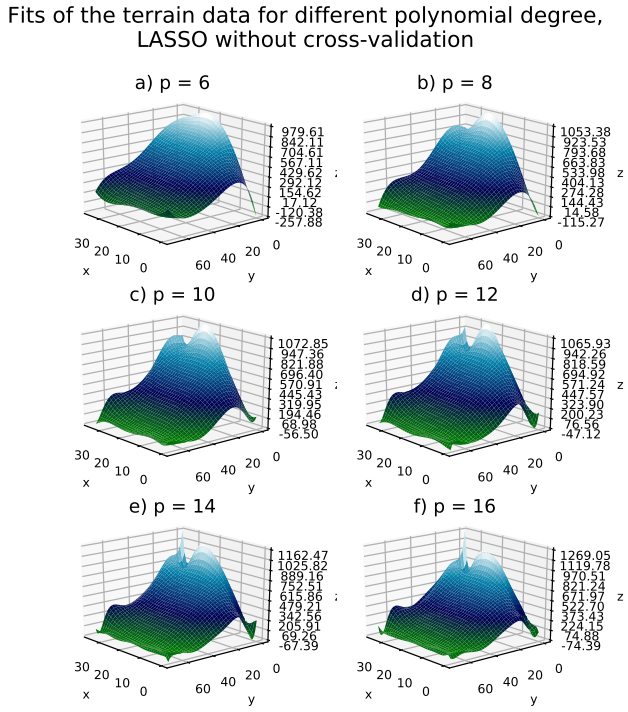

FIG. 57.— The fitted surfaces for the terrain data obtained for polynomial degrees $p = 6, 8, 10, 12, 14, 16$ and LASSO with no re-sampling of the data.