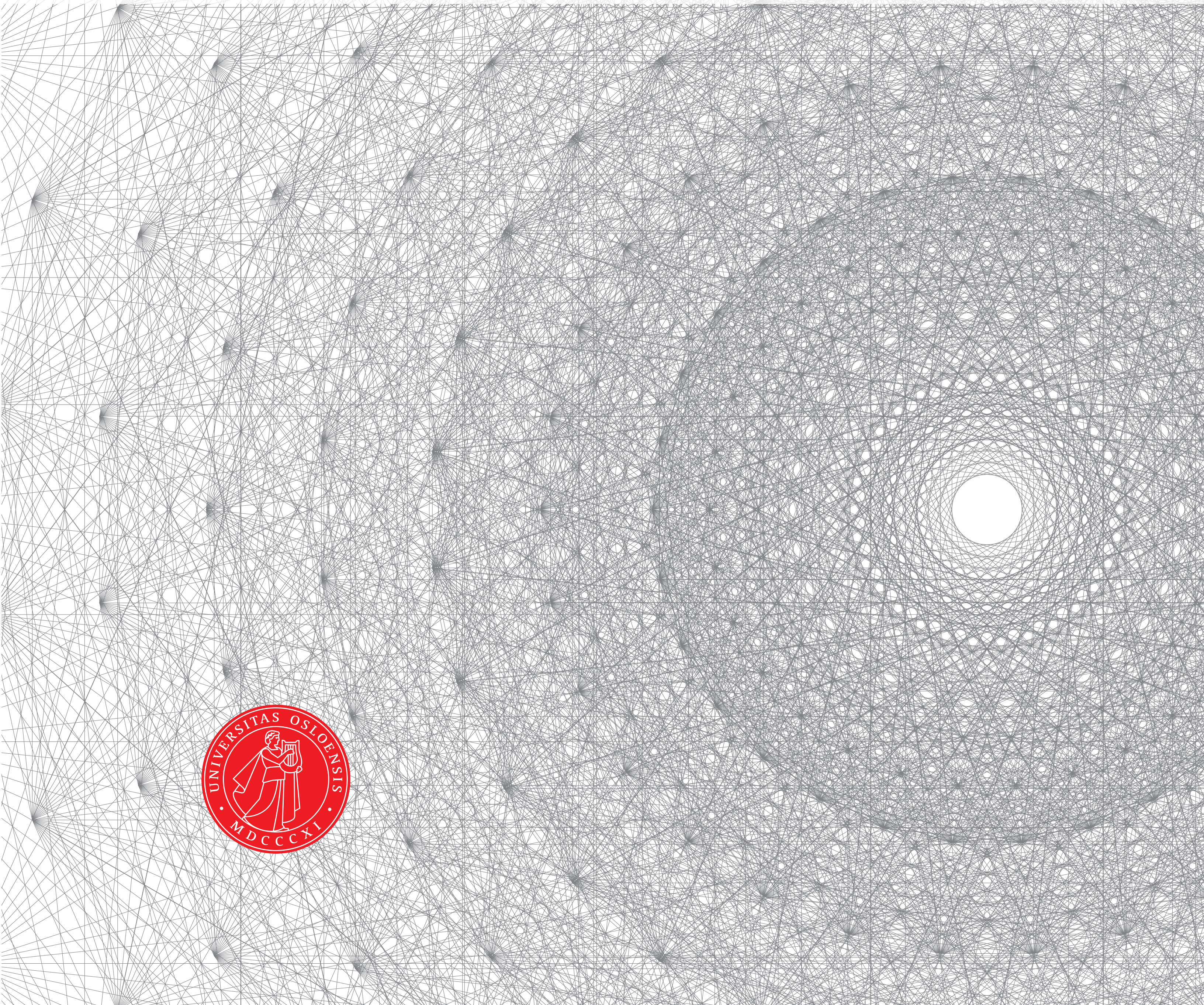
UiO : Department of Mathematics
University of Oslo

Monotone Regression

in high (and lower) dimensions

Solveig Engebretsen

Master's Thesis, Spring 2016



Cover design by Martin Helsø

The front page depicts a section of the root system of the exceptional Lie group E_8, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Abstract

In this thesis, we first present an overview of monotone regression, both in the classical setting and in the high dimensional setting. High dimensional data means that the number of covariates, p, exceeds the number of observations, n. It is often reasonable to assume a monotone relationship between a predictor variable and the response, especially in medicine and biology. The monotone regression methods for the high dimensional data setting that are considered are the liso regression method and the monotone splines lasso regression method (to our knowledge, the only two methods). Both these methods are special forms of penalised regression. The performances of these two high dimensional methods in the classical setting are studied and compared to the performances of existing methods for monotone regression developed for the classical setting, known as MonBoost, scam and scar. The two high dimensional methods work well also in the classical setting, but they do not outperform the existing methods. The two methods can still be useful in the classical setting, since they can be used in situations where the monotonicity directions of the effects are not known, in contrast to the existing methods and also perform automatic variable selection. Furthermore, we investigate the robustness of the monotone splines lasso method to the number of interior knots used to fit the monotone splines and find that it is very robust. In addition, two new methods for fitting a partially linear model where the non-linear covariates are assumed to have a monotone effect on the response are developed. These two methods can be used in the setting where $p > n$ as well as in the classical setting. To our knowledge, no such methods have been developed in the past. The first method, PLAMM-1, is a straight forward extension of the monotone splines lasso method to the partially linear setting. The second method developed, PLAMM-2, is a method with two penalties, one on the linear parameters and one on the non-linear parameters. In this last case, estimation has to be performed iteratively, and we prove convergence of the iterative scheme. The estimation, selection and prediction performances of the methods are investigated by simulation experiments in different settings. Through the simulation experiments, the methods are shown to work well in both the classical settings and in the high dimensional setting where the number of observations is not too small. We also apply the partially linear monotone model to a medical dataset where clinical covariates enter the linear part and genomic covariates are assumed to have a monotone effect on the outcome.

Acknowledgments

First and most importantly, I want to thank my supervisor Ingrid K. Glad. Thank you for your tremendous encouragement, support and optimism. Thanks for always taking time to meet with me for guidance and discussion and for introducing me to this interesting field.

I would also like to thank the study hall *Parameterrommet* for great discussions, cups of tea, social events and group training classes.

In addition, I want to thank Sjur Reppe for providing the bone mineral data, making it possible for me to try out the methods developed on a real data set.

I am very grateful to my family for always being there for me. A special thanks to Andreas for being a great team player with your help, love and support.

Contents

List of abbreviations

1CTP	Carboxy-terminal telopeptide of type 1 collagen.
Ad. lasso	Adaptive lasso.
Ad. liso	Adaptive liso.
Ad. MS-lasso	Adaptive monotone splines lasso.
age	Proportion of owner units built prior to 1940.
AGL	Adaptive group lasso.
AIC	Akaike information criterion.
APLAMM-1	Adaptive partially linear additive monotone method 1.
APLAMM-2	Adaptive partially linear additive monotone method 2.
BIC	Bayesian information criterion.
BMD	Bone mineral density.
BMI	Body mass index.
crime	Crime rate by town.
FP	False positives.
GCV	Generalised cross-validation.
GL	Group lasso.
gMDL	g-prior minimum description length.
indus	Proportion of non-retail business acres per town, serves as a measure of amount of industry.
LAND	Linear and non-linear discoverer.
Lasso	Least absolute shrinkage and selection operator.
Liso	Lasso isotone.
MS-lasso	Monotone splines lasso.

MSE	Mean squared error.
NOX	Concentration of nitrogen oxides.
PE	Prediction error.
PLAMM-1	Partially linear additive monotone method 1.
PLAMM-2	Partially linear additive monotone method 2.
PTH	Parathyroid hormone values.
rad	Index of accessibility to radial highways.
Scam	Shape constrained additive models.
Scar	Shape constrained additive regression.
SNR	Signal to noise ratio.
tax	The cost of public services.
TP	True positives.
vitD	Vitamin D.
zn	Proportion of a town's residential land zoned into lots greater than 25 000 square feet.

1 Introduction

In this thesis, we work with methods for monotone regression. Especially in biology, it is often reasonable to assume monotonicity. Methods for monotone regression in both higher and lower dimensions are presented. Tutz and Leitenstorfer (2007) write that "It is surprising that most of the literature on monotonic regression focuses on the case of unidimensional covariate x and metrically scaled, continuous response variable y". To our knowledge, not much work has been done in the multidimensional setting, and we will try to give an overview of the methods available for monotonic regression in the multidimensional setting. We will especially consider the methods developed by Tutz and Leitenstorfer (2007), Pya and Wood (2014) and Chen and Samworth (2015). These are all methods developed for the classical regression setting, where the number of observations is larger than the number of covariates. In the high dimensional data setting, we will study the liso regression method by Fang and Meinshausen (2012) and the monotone splines lasso regression method by Bergersen et al. (2014).

In our times, there is almost no limit to how much data we can measure and store. Large data sets might have to be distributed on clusters of computers in order to be processed and analysed, which asks for special care in the design of statistical methods used. A special kind of large data sets are those usually referred to as high dimensional data, which are characterised by the fact that the number of parameters p is a lot greater than the number of observations n, so $p >> n$. For such data, the problem is not necessarily the size and storage, but the fact that most conventional methods do not give unique solutions when $p > n$. We need regression methods that can analyse these high dimensional data. An example of high dimensional data is genomic data, where gene expressions are treated as possible explanatory variables for some disease. Another example from astronomy is galaxy spectra with flux measurements at thousands of different wavelengths which are used to explain a physical condition of the galaxies, for instance redshift, as in Lee and Izbicki (2016).

Before we can go into the methods for monotone regression, we need to introduce the theory and the methods for linear regression. In this thesis, we will mostly consider penalisation methods for analysing high dimensional data. We will especially consider the lasso penalty (Tibshirani, 1996) with variations. These methods use a penalty on the parameter estimates to impose restrictions so that the methods can be used in the high dimensional data setting. The type of penalty used depends on the aim of the method, for instance preserving monotonicity.

Additive penalisation methods have been developed, allowing for non-linear effects in the high dimensional setting. In a recent paper, Bergersen et al. (2014) developed a method for fitting a regression model where the effects are monotone and non-linear. This is done by fitting functions that describe the effect of the covariate on the response by monotone splines and using a proper penalty term encouraging monotonicity. We will use this method throughout the thesis and consider different properties. We study robustness to the number of interior knots used to fit the monotone splines and whether or not the method is applicable in the classical setting ($p < n$).

We also develop two new methods for fitting additive partially linear models. Additive partially linear models are models where some of the covariates are assumed to have a linear effect on the response, and some are assumed to have a non-linear effect on the response. We consider the special case where the non-linear effects are assumed to be monotone, and use monotone splines to fit the model. The first method we develop, PLAMM-1, is a straight forward extension of the monotone splines lasso method to the partially linear setting. The second method, PLAMM-2, is an iterative procedure with two penalties – one on the linear parameters and one on the non-linear parameters. We also prove convergence of the iterative method.

The thesis is organised as follows: in section 2, basic theory and linear regression methods for high dimensional data are presented. In section 3, we consider methods for fitting additive models and give an overview of existing methods for monotone regression in the classical setting. In section 4, two methods for fitting a monotone regression model in the high dimensional setting are described. These methods are the liso method (Fang and Meinshausen, 2012) and the monotone splines lasso method (Bergersen et al., 2014). To our knowledge, these are the only two existing methods for performing monotone regression in the high dimensional setting. The methods are also illustrated with a simulation example. We study robustness of the monotone splines lasso to the number of interior knots used to fit the estimated functions. In section 5, we evaluate and compare the performance of the monotone regression methods in the low dimensional data case where $p < n$. We also apply the methods to the Boston housing data, which is a data set with house value and different explanatory variables. In section 6, we present existing methods for additive partially linear models in the high dimensional data setting and we develop two new methods for additive partially linear models using monotone splines lasso, so that the effects of the non-linear covariates are restricted to being monotone. The two methods are also tried out on a data set with bone mineral density as response and clinical and genomic variables as covariates.

2 Linear regression in higher dimensions

Before we can start with the monotone regression methods, we need to look into the simple linear regression methods.

We consider observed data $(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_n}, y_n)$, where $\boldsymbol{x_i} \in \mathbb{R}^p$, so p is the number of parameters, and n is the number of observations. The standard linear model is

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i \quad (i = 1, \dots, n), \tag{2.1}$$

where x_{ij} is the value of covariate j for the ith observation and ϵ_i, $i = 1, \dots, n$, are assumed to be independent and identically distributed normal random variables with mean 0 and variance σ^2. The parameters $\beta_j \in \mathbb{R}$ are the regression coefficients. Let $\boldsymbol{y} = (y_1, y_2, \dots, y_n)'$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)'$, $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \dots, \epsilon_n)'$ and

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix}.$$

Then the linear model can also be written on matrix form as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

To estimate the linear model, the standard approach is the method of ordinary least squares. The ordinary least squares solution, $\hat{\boldsymbol{\beta}}$, is given as the minimiser of the residual sum of squares, so

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Differentiating with respect to $\boldsymbol{\beta}$, we see that the solution is given by

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

when the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is invertible. When $p > n$, which is the case with high dimensional data, $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is singular, so we can not use the ordinary least squares estimates. We will in this section consider some of the statistical methods used to fit this linear model in cases with high dimensional data.

2.1 High dimensional data and penalised regression

In high dimensional data, there are more parameters than observations, so $p > n$, or even $p >> n$. Thus there are more unknown parameters than equations, and the ordinary least squares method can no longer be used to fit the data to the linear regression model. The minimiser of the residual sum of squares is no longer unique when there are more parameters than observations.

The methods presented in this section can also be useful in the classical setting, to reduce the mean squared error. Including many covariates will often give a high model complexity, and a large variance in the parameter estimates. Leaving out relevant covariates will increase the bias. So there is a bias-variance trade-off. We do not want the fitted model to be too close to the training data (the data used to fit the model), because this will make the fitted model useless for generalisation and prediction (Hastie et al. (2011), section 2.9). To prevent this overfitting and/or to make the problem feasible in the high dimensional setting, we have to eliminate some of the parameters by using a subset selection method and/or use a complexity regularisation on the parameters. This will lead to more bias, but less variance and might improve the predictions. Subset selection methods retain a subset of the variables and discard the other. Since this is a discrete process, subset selection methods can suffer from high variability. Shrinkage methods, which shrink the size of the fitted parameters, do not suffer as much from high variability as subset selection methods (Hastie et al. (2011), section 3.4).

For an L_q regularisation (Vidaurre et al., 2013), the parameter constraint $||\boldsymbol{\beta}||_q < s$ for some s, where $||\cdot||_q$ is the L_q norm, is used. This constraint leads to shrinkage of the estimates. Minimising the residual sum of squares with the L_q regularisation is the same as considering the problem

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_q^q, \tag{2.2}$$

where there is a one-to-one correspondence between s and λ. The parameter λ is called the penalty parameter and it controls the level of regularisation. In the classical setting where $p < n$, the solution will converge towards the least squares solution as $\lambda \to 0$. The larger λ, the closer to zero will the parameter estimates be. Note that the intercept is not penalised, since that would make the other parameter estimates dependent on the origin of $\mathbf{y}$. Normally the data is centered, if not, a non-penalised intercept is included in $\boldsymbol{\beta}$. The data is normally also standardised so that the penalty does not depend on the measurement scale of the covariates.

For the case with L_1 regularisation, some of the regression coefficients can be set to exactly zero. This is due to the non-differentiability of the L_1 norm at zero. Hence L_1 regularisation methods perform variable selection as well as shrinking the estimates. This is illustrated in section 2.4.

2.2 Selection of the penalty parameter

A criterion is needed for selecting the optimal penalty parameter in for instance equation (2.2). There are several approaches for estimating the optimal penalty parameter. Two famous criteria for choosing the penalty parameter are AIC (Akaike information criterion) and BIC (Bayesian information criterion). The

AIC value for our model with normal noise is given by (Hastie et al., 2011, section 7.5)

$$\text{AIC}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + 2\frac{\text{df}(\lambda)}{n}\hat{\sigma}^2,$$

where $\hat{y}_i$ is the predicted value of y_i using the fitted model considered, $\text{df}(\lambda)$ is the degrees of freedom for the fitted model with penalty parameter λ, n is the number of observations and $\hat{\sigma}$ is an estimate of the variance of ϵ. The higher complexity of the model, the larger will the degrees of freedom for the model be.

The BIC value for our model with normally distributed noise is given by (Hastie et al., 2011, section 7.7)

$$\text{BIC}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \frac{\log(n)}{n}\text{df}(\lambda)\hat{\sigma}^2.$$

The AIC criterion and the BIC criterion both work as a balance between good fit (low value of the first term) and complexity (the more complexity, the higher value of the last term). The model chosen by the AIC criterion or the BIC criterion is the model with the smallest AIC or BIC value, respectively. Typically, lower complexity leads to less variance, but more modelling bias, and higher complexity leads to more variance, but less modelling bias. So AIC and BIC try to find the optimal balance between bias and variance. Another popular method for selecting the optimal penalty parameter is the method of cross-validation, which will be treated in the next section.

2.2.1 Cross-validation

One commonly used approach for selecting the penalty parameter is the method of cross-validation. Cross-validation is a method for estimating the prediction error of a model. In a K-fold cross-validation scheme, the data is split randomly into K equally sized (or roughly equal) folds. For each value of λ (or for each model considered), the kth fold is left out, and the model is fitted by using the remaining data in the other $K-1$ folds. This fitted model is used to predict the response for the observations in fold k. This is repeated for $k = 1, \ldots, K$. Let $\kappa(k)$ be the set of indices for observations in fold k. Then an estimate of the prediction error for each value of λ, $\text{PE}(\lambda)$, is given by (Hastie et al., 2011, section 7.10)

$$\text{PE}(\lambda) = \sum_{k=1}^{K}\sum_{i\in\kappa(k)}(y_i - \hat{y}_i)^2,$$

where $\hat{y}_i$ is the predicted value of y_i. This can be used as a model selection criterion. For each model considered (each value of λ), we calculate the cross-validated prediction error, and then we choose the model (λ) with the smallest value of the estimated prediction error.

2.3 Ridge regression

The ridge method is a shrinkage method for fitting the linear regression model, applicable also in a high dimensional data setting. Ridge regression minimises the residual sum of squares with an L_2 penalisation. So the estimated coefficients are the solution to equation (2.2) with $q = 2$ (Hastie et al. (2011), section 3.4)

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

The penalty parameter λ can be chosen by for instance the method of cross-validation. The ridge method shrinks the coefficients with an amount controlled by λ, but it does not perform variable selection (no coefficients are estimated to exactly zero). This makes it harder to see which coefficients are important for the response, and it also makes it harder to interpret the final model. However, the ridge regression method is better at prediction than the lasso regression method (see section 2.4) in some situations, for instance when the covariates are correlated (Vidaurre et al., 2013).

2.4 Lasso regression

A different method for fitting the linear regression model which can be used in the high dimensional setting is the least absolute shrinkage and selection operator (lasso) (Tibshirani, 1996). Lasso is a method which minimises the residual sum of squares, subject to an L_1 constraint. So we want to solve problem (2.2) with $q = 1$, that is

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \text{argmin}_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2,$$

under the restriction

$$\sum_{j=1}^{p} |\beta_j| \leq s.$$

The corresponding Lagrangian is

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \text{argmin}_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{2.3}$$

where there is a one-to-one correspondence between s and λ. The penalty parameter λ is often chosen by a cross-validation method to minimise the prediction error.

The lasso method is based on an assumption of sparsity. That is, it assumes that a model with few covariates is more realistic than a model which includes small contributions of all the variables. This means that only a few variables are important. Since the lasso method uses an L_1 regularisation, it both shrinks the

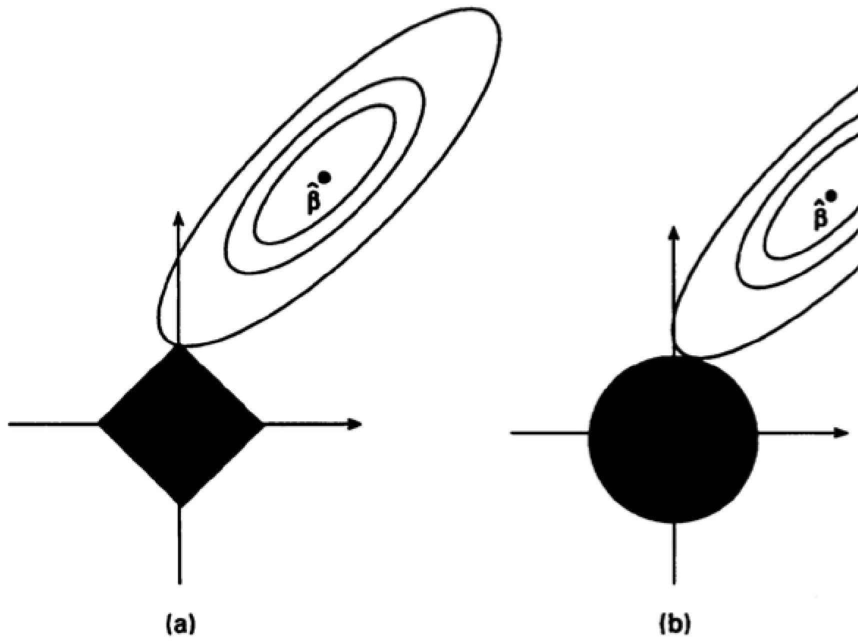


Figure 1: Sketch of contour curves for the residual sum of squares, with the constraint region for a) the L_1 penalisation (corresponding to lasso regression) and b) the L_2 penalisation (corresponding to ridge regression). The figure is from Tibshirani (1996).

parameter estimates and performs variable selection. The fact that lasso performs variable selection makes it a very attractive method, since the traditional methods for variable selection are not satisfying in the high dimensional setting. Forward stepwise regression is based on univariate analysis and is thus very biased, and best subset selection methods are too computationally demanding when there are many parameters (Vidaurre et al., 2013). As mentioned in section 2.3, the ridge method does not perform variable selection. This is also illustrated in Figure 1. In Figure 1, a two-dimensional situation is considered, so there are two parameters, β_1 and β_2. The residual sum of squares has elliptical contours centered at the ordinary least squares estimate (which minimises the residual sum of squares). With an L_1 penalty, the constraint region for the parameters is a diamond with corners at the origin, given by $|\beta_1| + |\beta_2| \leq s$. This is due to the fact that the L_1 penalty is not differentiable at the origin. If the contour curve for the residual sum of squares intersects with the constraint region at a corner, variable selection is obtained. For the L_2 penalty, the constraint region for the parameters is a disc given by $\beta_1^2 + \beta_2^2 \leq s$. The disc does not have corners, so variable selection is not obtained (Hastie et al., 2011, section 3.4).

In Zou and Hastie (2005), three limitations of the lasso method are discussed.

The first one is that in the situation where $p > n$, the lasso method selects at most n variables. This is clearly a limitation, since there might be more than n important variables. Since ridge does not leave out any variables, it clearly does not have this limitation. The second limitation is that if we have groups of covariates being highly correlated, the lasso tends to select any one of these covariates and leave the rest out. Again, this is avoided with ridge regression, since ridge regression does not perform variable selection. The parameter estimates of correlated variables are shrunken towards each other with ridge regression (Hastie et al. (2011), section 3.4). The last one is that when the data are correlated, ridge regression often performs better than lasso regression in prediction, as also noted in section 2.3. In the following sections, improvements and variations of the lasso penalty are considered.

2.4.1 Adaptive lasso

The optimal λ for the lasso method is often found by cross-validation to optimise the prediction error. The λ which is optimal for prediction error is often too low for the purpose of correct variable selection (that is, selecting the true underlying model). This may cause the lasso method to select too many variables (Bühlmann and van de Geer (2013), section 2.5). The adaptive lasso method (Zou, 2006) is an improvement of the lasso method, which attempts to correct for the overestimation, using a two-stage approach. Let $\hat{\boldsymbol{\beta}}_{\text{init}}$ be an initial guess of the parameter values, typically the parameter estimates from a lasso estimation, that is, the solution to equation (2.3). Then, as in Bühlmann and van de Geer (2013) (section 2.8), the adaptive lasso estimates, $\hat{\boldsymbol{\beta}}_{\text{adapt}}$, are given by

$$\hat{\boldsymbol{\beta}}_{\text{adapt}} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{p} \frac{|\beta_j|}{|\hat{\beta}_{\text{init},j}|},$$

where for instance cross-validation can be used to estimate the optimal λ for this problem. When $\hat{\beta}_{\text{init},j} = 0$, $1/|\hat{\beta}_{\text{init},j}|$ will be infinitely large, so we have the property

$$\hat{\beta}_{\text{init},j} = 0 \Rightarrow \hat{\beta}_{\text{adapt},j} = 0.$$

Hence there are never more parameters in the adaptive lasso than in the one-stage lasso approach. Furthermore, if the initial parameter value, $\hat{\beta}_{\text{init},j}$, is small, then $1/|\hat{\beta}_{\text{init},j}|$ will be large, so the penalty will be large. Similarly, if $\hat{\beta}_{\text{init},j}$ is large, the penalty will be small. Since a large $\hat{\beta}_{\text{init},j}$ indicates that parameter number j is of importance (keep in mind that the covariates are standardised), the parameters are penalised according to the relative importance from the initial estimation, and the method can be used to reduce the number of irrelevant variables.

2.4.2 Group lasso

In some cases, some of the covariates in the linear model might be grouped, so that the parameter vector $\boldsymbol{\beta}$ contains grouped parameters. This can for instance happen when one of the explanatory variables is a factor. The index set $\{1, \dots, p\}$ of $\boldsymbol{\beta}$ is partitioned into groups $\mathscr{G}_1, \mathscr{G}_2, \dots, \mathscr{G}_k$, where k is the number of groups/factors. If covariate number j is just a simple numerical covariate, then its group will contain only one parameter. Let $|\mathscr{G}_i|$ denote the number of parameters in group i, and let

$$\boldsymbol{\beta}_{\mathscr{G}_j} = \{\beta_r; r \in \mathscr{G}_j\}.$$

When the variables are grouped, we often do not want one of the variables to be taken out of the model, while the other variables are kept. So we have to use a method which ensures that either all the parameters within a group are set to zero, or that all are non-zero. This is done by using the group lasso penalty (Yuan and Lin, 2006)

$$\sum_{j=1}^{k} m_j ||\boldsymbol{\beta}_{\mathscr{G}_j}||_2 \leq s,$$

where m_j is a weight chosen to balance group sizes, typically $m_j = \sqrt{|\mathscr{G}_j|}$ (Bühlmann and van de Geer (2013), section 4.2). The parameter estimates are then found by

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{k} m_j ||\boldsymbol{\beta}_{\mathscr{G}_j}||_2,$$

where again there is a one-to-one correspondence between s and λ. Depending on the size of the regularisation parameter, the estimated coefficients will satisfy either $\boldsymbol{\beta}_{\mathscr{G}_j} = 0$ in all components, or that all components in $\boldsymbol{\beta}_{\mathscr{G}_j} \neq 0$. This is due to the non-differentiability of the square root function at the origin (note that the penalty term is a square root function of the parameters).

If all the groups consist of singletons (so that the covariates are not really grouped), then the group lasso penalty is equal to the L_1 penalty, and the problem is reduced to the same problem as in equation (2.3).

2.4.3 Cooperative lasso

The cooperative lasso is an enhancement of the group lasso, which can be used to obtain sign-coherent parameter estimates within a group when fitting a linear regression model. To obtain the sign-coherent group lasso variable selection, the cooperative lasso penalty (Chiquet et al., 2012) is used

$$||\boldsymbol{\beta}||_{\text{coop}} = \sum_{j=1}^{k} ||\boldsymbol{\beta}_{\mathscr{G}_j}^{+}||_2 + ||\boldsymbol{\beta}_{\mathscr{G}_j}^{-}||_2,$$

where $\boldsymbol{\beta}^{+}_{\mathscr{G}_j} = \max(\boldsymbol{\beta}_{\mathscr{G}_j}, 0)$ and $\boldsymbol{\beta}^{-}_{\mathscr{G}_j} = \max(-\boldsymbol{\beta}_{\mathscr{G}_j}, 0)$. The parameter estimates are then found by

$$\hat{\boldsymbol{\beta}}_{\text{coop}} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_{\text{coop}},$$

where, as before, λ controls the regularity. This penalisation scheme favours sign-coherent solutions, in the sense that it penalises more on sign-incoherent solutions. When the penalty parameter goes to zero, sign-coherence is no longer guaranteed (Chiquet et al., 2012). So if the regularisation parameter is small, the solution might be sign-incoherent.

The cooperative penalty was originally proposed for multi-task learning with related networks with Gaussian Graphical Models (Chiquet et al., 2011). When genes interact, there is a statistical dependence between the expressions of the genes. Gaussian Graphical Models can be used to model multiple dependence patterns. An example of a related network is gene interactions from data measured on slightly different stem cells (wild and mutant).

Another application of the cooperative lasso that is mentioned in Chiquet et al. (2012) is when there are groups of ordered categorical variables. To obtain monotonicity, the variables can be given quantitative values reflecting the order. However, there might not be a reasonable/known numerical difference between the levels. The cooperative lasso can be used in this situation by biasing the effects of the levels on the response variable towards a monotone solution, without having to quantify the levels. Another application is when there is redundancy in the variables and sign-coherence is expected. One example is redundant noisy measurements of the same unobserved variable. Then sign-coherence is expected, since all the measurements should be positively correlated with the unobserved variable. Sign-coherence is also expected when predicting closely related responses.

The cooperative lasso penalty can also be used to obtain a monotone spline, and will be used later in the monotone splines lasso regression method.

3 Monotone regression

Instead of the linear model (2.1), the more general additive model (Hastie and Tibshirani, 1986) is considered

$$Y_i = \beta_0 + \sum_{j=1}^{p} g_j(x_{ij}) + \epsilon_i \quad (i = 1, \ldots, n), \tag{3.1}$$

where the g_js are unknown smooth functions to be estimated. A natural approach is to fit the functions g_j by splines, so that each g_j is a linear combination of spline basis functions. This brings us back to a linear problem, which there are methods for solving. In the setting where there are more parameters than observations, a group lasso procedure can be used, where for each covariate, there is a group of spline basis function coefficients used to represent g_j. This is done in for instance Huang et al. (2010).

3.1 Regression splines

A spline is a function consisting of piecewise polynomials joined at points called knots. These piecewise polynomials are the spline basis functions. Let m be the number of spline basis functions, and let $s_j(x)$, $j = 1, \ldots, m$, denote the spline basis functions. Then a spline is given by

$$g(x) = \sum_{j=1}^{m} \beta_j s_j(x),$$

where β_j are the basis coefficients, and the spline is a linear combination of the basis functions. If the spline basis functions are polynomials of degree l, then the spline has $l-1$ continuous derivatives at the knots (Hastie et al., 2011, section 5.2). If the spline has K knots, then the number of basis functions needed to fit the spline is $m = K + l$ (Ramsay, 1988). Examples of splines are piecewise constant functions, piecewise linear functions which are continuous at the knots, piecewise quadratic functions which are continuous at the knots and have continuous first derivatives at the knots and so on. One of the most common spline bases is the B-spline basis. In this thesis, we will mostly work with I-splines, see section 4.2.

Splines are often used to fit a general function. Fitting a spline means finding the β_js which give the best fit to the function. To fit a spline, we need to select the order of the spline, the number of knots and the placement of the knots. The knots are often placed at the quantiles of the variable. In the statistical setting, the function we want to fit is normally unknown, but we have some observations which we use to fit the spline function. Then β_j could be found as the least squares solution.

3.2 Monotone regression overview

It is often reasonable to assume that the relationship between the explanatory variable and the response is monotonically increasing or decreasing. For example, it is common to assume that the relationship between some measure of cognitive performance of children and age is a monotonically increasing function, and it is not plausible that this relationship is linear (Bollaerts et al., 2006). In medicine, we often have monotone relationships between two variables, for example between the amount of exercise and serum cholesterol level (Schell and Singh, 1997).

Many of the methods developed for monotone regression are developed for the univariate case, with only one predictor variable. Barlow and Brunk (1972) use isotonic step functions to fit regression models in the one dimensional setting. Dette et al. (2006) also develop a method for unidimensional monotone regression. There the inverse of the monotone regression function is estimated first, and then inverted to find the estimate of the monotone regression function. In the multidimensional setting, the inverse would not be unique, and this method is thus not trivial to generalise to the multidimensional setting. He and Shi (1998) develop a method for univariate monotone regression using monotone B-spline smoothing and Meyer (2008) develops a method for shape-restricted regression splines using I-splines, for the one-dimensional case.

It is more challenging, but of course more relevant, to consider multiple regression, as we rarely have only one predictor variable. Ramsay (1988) develops a method for monotone regression using I-splines, which can also be used in the multivariate setting. Bacchetti (1989) developed a method for additive isotonic regression. In Bacchetti (1989), each function is fitted by an isotonic step function, and the method is based on an iterative cyclic optimisation scheme starting with an initial guess for all the functions, and then iteratively updating cyclically one by one keeping the other functions at their currently best guess and minimising the loss with respect to the current function by a unidimensional isotonic regression method. This is repeated until convergence is obtained. Tutz and Leitenstorfer (2007) use the ideas of Ramsay (1988) in combination with monotone boosting. Pya and Wood (2014) use P-splines to fit a regression model where some of the functions are fit by functions with shape constraints, and the rest have no shape constraint. Chen and Samworth (2015) also develop a method for regression with different shape constraints on the functions. The method in Chen and Samworth (2015) is based on using different basis functions with different constraints on the parameters, depending on what shape restriction is imposed. Meyer (2013) developed a method very similar to the method in Chen and Samworth (2015). All these methods are developed for lower dimensional regression. In the high dimensional data setting, with $p > n$, we will consider the only two available methods to our knowledge, namely the liso regression method (Fang and Meinshausen, 2012) and the monotone splines lasso (Bergersen et al., 2014). This is done in section 4. In this section, the monotone regression methods developed by Tutz and Leitenstorfer (2007), Pya and Wood (2014) and Chen and Samworth (2015) are considered.

3.2.1 MonBoost

The method developed in Tutz and Leitenstorfer (2007) is called the MonBoost method. MonBoost estimates the model given in equation (3.1), where some of the functions g_j are restricted to being monotone. Tutz and Leitenstorfer (2007) only consider monotonically increasing functions, but it is easy to see how the algorithm can also be used for monotonically decreasing functions. One could also have a mixture of the two.

Let g_j be approximated by a basis expansion, where the basis functions are the I-spline basis functions of order two, given on closed form in section 4.2. These are monotonically increasing basis functions. A sufficient condition for monotonicity is then that all the basis coefficients are of the same sign. Since we only consider monotonically increasing functions, we want a solution where all the basis coefficients are nonnegative. The basis expansion is given by

$$\tilde{g}_j(x) = \sum_{k=1}^{m} \beta_{jk} I_k^{(l)}(x),$$

where $\tilde{g}_j(x)$ is the approximation of g_j. In Ramsay (1988) and Bergersen et al. (2014), a small number of knots is used. In MonBoost, many interior knots are used, and boosting is used to avoid overfitting. In Tutz and Leitenstorfer (2007), sigmoidal basis functions are also an option, in addition to the I-spline basis functions.

The concept of boosting is to combine many weak learners (in classification, a weak learner is one that is only slightly better than random guessing) to obtain a good predictor. Componentwise boosting is used, so that each weak learner only changes the contribution of one variable. Here, the contribution of one basis function is updated in each iteration. The more iterations, the closer will the model be fitted to the training data. Thus, we need a stopping criterion for determining when we should stop. This stopping criterion should estimate when there is an optimal balance between a good fit and complexity. In MonBoost, both AIC and the g-prior minimum description length (gMDL) are implemented. gMDL is a hybrid between AIC and BIC, see Tutz and Leitenstorfer (2007) or Bühlmann and Yu (2006) for more details. It is also possible to regularise by using a shrinkage parameter which shrinks the learner for each iteration. In MonBoost, this is done by using a ridge regression estimate as the weak learner, with a quite large value of λ. We will give the algorithm for MonBoost as in Tutz and Leitenstorfer (2007) in the one dimensional case. Remember that

$$\tilde{g}_j(x) = \sum_{k=1}^{m} \beta_{jk} I_k^{(l)}(x).$$

Let M denote the number of iterations. The algorithm is given in Algorithm 1, where $\mathbf{y}$ is the vector with observed responses, and $\mathbf{x}$ is the vector with observations of the predictor. Since the estimated functions are constructed by ensuring

Algorithm 1 MonBoost

```
Initialise:
    Standardise y to have mean zero, so $\hat{\boldsymbol{\mu}}^{(0)} = (\bar{y}, \ldots, \bar{y})$.
Iteration:

    for l=1 to M do
        $\mathbf{u}^{(l)} = \mathbf{y} - \hat{\boldsymbol{\mu}}^{(l-1)}$                ▷ Compute the current residuals
        for k=1 to m do
            Compute the ridge estimators $\hat{\beta}_k$ with
            tuning parameter $\lambda$ for the model
                $$\mathbf{u}^{(l)} = \beta_k I_k(\mathbf{x}) + \boldsymbol{\epsilon}.$$
        end for
        From the subset of components that fulfil the constraint
        $\hat{\beta}_k^{(l)} = \hat{\beta}_k^{(l-1)} + \hat{\beta}_k \geq 0$, choose the component $\hat{\gamma}^{(l)}$ which
        minimises $||\mathbf{u}^{(l)} - \hat{\beta}_k I_k(x)||^2$.
        if $\hat{\beta}_k^{(l)} \leq 0$ for all k then
            stop iteration.
        else
            $\hat{\gamma}^{(l)} = k$.
        end if
        Set
            $$\hat{\beta}_k^{(l)} = \begin{cases} \hat{\beta}_k^{(l-1)} + \hat{\beta}_k, & \text{if } j = \hat{\gamma}^{(l)}, \\ \hat{\beta}_k^{(l-1)}, & \text{otherwise,} \end{cases}$$
        and
            $$\hat{\boldsymbol{\mu}}^{(l)} = \hat{\boldsymbol{\mu}}^{(l-1)} + \hat{\beta}_{\hat{\gamma}^{(l)}} I_{\hat{\gamma}^{(l)}}(\mathbf{x}).$$
    end for
```

that all the estimated parameters are positive, the estimated function will necessarily be monotone. If there are no shape constraints on the function, we do not need to consider only the subset of positive estimated parameters in the algorithm. If the function is monotonically decreasing, only the subset of negative estimated parameters is considered. The method is easily extended to the multivariate setting. Instead of only considering the m basis functions as candidates for update in the algorithm, all p sets of m basis functions are considered. An R-implementation of this algorithm is available, but it does not have the option of a monotonically decreasing function.

In Tutz and Leitenstorfer (2007), they only consider applications in the classical setting. However, the algorithm would also work when $p > n$. There might be a problem with the computational aspect, in that one would need many iterations before obtaining a good fit.

3.2.2 Shape constrained additive models (Scam)

The method developed in Pya and Wood (2014) estimates the model given in equation (3.1), where the functions have different optional shape constraints. The model is fitted by using P-splines. P-splines are B-splines with a difference penalty on adjacent B-spline coefficients. See Eilers and Marx (1996) for more details on P-splines. The shape constraints on the functions have to be known a priori to use this method. Among these constraints are the linear, monotonically increasing and monotonically decreasing. Consider first the one dimensional setting, where

$$Y = g(x) + \epsilon.$$

The function $g(x)$ is approximated by a B-spline. Let B_k denote the spline basis functions and γ_k denote the basis coefficients. Then we have

$$\tilde{g}(x) = \sum_{k=1}^{m} \gamma_k B_k(x),$$

where m is the number of basis functions, and $\tilde{g}$ is the spline approximation of g. A sufficient condition for the function $\tilde{g}$ to be monotonically increasing is that $\gamma_k \geq \gamma_{k-1}$. A reparametrisation is used, so that

$$\boldsymbol{\gamma} = \boldsymbol{\Sigma}\tilde{\boldsymbol{\beta}},$$

where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_m)'$, $\tilde{\boldsymbol{\beta}} = (\beta_1, \exp(\beta_2), \dots, \exp(\beta_m))'$ and $\boldsymbol{\Sigma}$ is such that $\Sigma_{ij} = 0$ if $i < j$ and $\Sigma_{ij} = 1$ if $i \geq j$. Then this reparametrisation ensures that the fitted function is monotonically increasing. Let $\mathbf{Z}$ denote the matrix with the $\mathbf{x}$ observations represented in the B-spline basis. Then we have

$$\tilde{g}(\mathbf{x}) = \mathbf{Z}\boldsymbol{\Sigma}\tilde{\boldsymbol{\beta}}.$$

The reparametrisations necessary for other shape constraints are listed in Table 1 in Pya and Wood (2014).

To control the wiggliness of $\tilde{g}(x)$, Pya and Wood (2014) introduce a penalty term, penalising the squared differences between adjacent β_k. The penalty is given as $||\mathbf{D}\boldsymbol{\beta}||_2^2$, where $\mathbf{D}$ is such that all elements are zero, except from $D_{i,i+1} = -D_{i,i+2} = 1$ for $i = 1, \dots, m-2$. Note that the penalty is on the $\boldsymbol{\beta}$ and not on the $\tilde{\boldsymbol{\beta}}$.

In the multidimensional setting, it is assumed that all the functions have mean zero, for unique identification of the functions. Let each shape constrained function be represented by a model matrix on the form $\tilde{g}_j(\mathbf{x}_j) = \mathbf{Z}_j\mathbf{\Sigma}_j\tilde{\boldsymbol{\beta}}_j = \boldsymbol{M}_j\tilde{\boldsymbol{\beta}}_j$, where $\mathbf{x}_j$ are the observed values of covariate j. Let $\mathbf{M}$ denote the matrix with all the $\mathbf{M}_j$ and $\tilde{\boldsymbol{\beta}}$ the vector with all the $\tilde{\boldsymbol{\beta}}_j$. If there are in addition linear covariates, the design matrix with the linear covariates and the linear parameters are also included in $\mathbf{M}$ and the parameter vector $\tilde{\boldsymbol{\beta}}$. There are no penalties on the linear parameters. In a similar manner, functions with no shape constraints can also be added to the model, given as B-spline approximations, so that we have a design matrix with the observations represented in the B-spline basis and a parameter vector for the covariates with no shape constraints. The penalty term is on the form $\boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}_\lambda\boldsymbol{\beta}$, where $\mathbf{S}_\lambda = \sum_{j=1}^{p} \lambda_j \mathbf{S}_j$ and $\mathbf{S}_j = \mathbf{D}_j^{\mathrm{T}}\mathbf{D}_j$. The parameters λ_j are smoothing parameters. Given the λ_j, the solution, $\hat{\boldsymbol{\beta}}$, is given as the minimiser of

$$\hat{\boldsymbol{\beta}} = \mathrm{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{M}\tilde{\boldsymbol{\beta}}||_2^2 + \boldsymbol{\beta}^{\mathrm{T}}\mathbf{S}_\lambda\boldsymbol{\beta}.$$

This is solved by a Newton-Raphson like scheme. The smoothing parameters λ_j are estimated by the AIC criterion or the generalised cross-validation (GCV). See Pya and Wood (2014) for the algorithm for solving the problem and details on the GCV.

This scheme is implemented in the R-package *scam*, and we will refer to this method as scam.

3.2.3 Shape constrained additive regression (Scar)

The method developed in Chen and Samworth (2015) also estimates the model in equation (3.1), where each function g_j is assumed to satisfy one out of nine possible shape constraints. It is assumed that it is a priori known which shape constraint each function satisfies. Among these nine constraints are the linear shape constraint, monotonically increasing and monotonically decreasing. A table with the different shape constraints available and their label is given in Table 1. As before, all the functions are assumed to have mean zero, for unique identification of the functions.

Assume that the first d_1 parameters have linear effects, and that each of the remaining covariates belongs to one of the labels 2-9 in Table 1. Denote the label of covariate j by l_j. Let $\mathbf{X}$ be the design matrix of the observations, and let $\mathbf{X}_{(i)j}$ be the corresponding order statistics. Let $s_{0j}(x_j) = x_j$ for $j = 1, \ldots, d_1$. Then let

$$s_{ij} = \begin{cases} I(\mathbf{X}_{(i),j} \leq x_j) - I(\mathbf{X}_{(i),j} \leq 0), & \text{if } l_j = 2, \\ I(x_j \leq \mathbf{X}_{(i),j}) - I(0 < \mathbf{X}_{(i),j}), & \text{if } l_j = 3, \\ (x_j - \mathbf{X}_{(i),j})I(\mathbf{X}_{(i),j} \leq x_j) + \mathbf{X}_{(i),j}I(\mathbf{X}_{(i),j} \leq 0), & \text{if } l_j = 4 \text{ or } l_j = 5, \\ (\mathbf{X}_{(i),j} - x_j)I(x_j \leq \mathbf{X}_{(i),j}) - \mathbf{X}_{(i),j}I(0 \leq \mathbf{X}_{(i),j}), & \text{if } l_j = 6, \\ (\mathbf{X}_{(i),j} - x_j)I(\mathbf{X}_{(i),j} \leq x_j) - \mathbf{X}_{(i),j}I(\mathbf{X}_{(i),j} \leq 0), & \text{if } l_j = 7 \text{ or } l_j = 9, \\ (x_j - \mathbf{X}_{(i),j})I(x_j \leq \mathbf{X}_{(i),j}) + \mathbf{X}_{(i),j}I(0 \leq \mathbf{X}_{(i),j}), & \text{if } l_j = 8, \end{cases}$$

Shape constraint	Label	Shape constraint	Label
Linear	1	Monotone increasing	2
Monotone decreasing	3	Convex	4
Convex increasing	5	Convex decreasing	6
Concave	7	Concave increasing	8
Concave decreasing	9		

Table 1: Shape constraints supported by the scar method from Chen and Samworth (2015).

where I is the indicator function. The functions s_{ij} are the basis functions. As is seen from the basis functions, the fitted monotone functions will be step functions. Then introduce the vector of weights, $\mathbf{w}$, as

$$\mathbf{w} = (w_{00}, \ldots, w_{0d_1}, w_{1(d_1+1)}, \ldots, w_{n(d_1+1)}, \ldots, w_{1p}, \ldots, w_{np})'.$$

These weights satisfy

$$\begin{cases} w_{ij} \geq 0, & \text{for every } i = 1, \ldots, n \text{ and every j with } l_j \in \{2, 3, 5, 6, 8, 9\}, \\ w_{ij} \geq 0, & \text{for every } i = 2, \ldots, n \text{ and every j with } l_j \in \{4, 7\}. \end{cases}$$

The solution is given by the $\mathbf{w}$ minimising

$$\hat{\mathbf{w}} = \text{argmin}_{\mathbf{w}} ||\mathbf{y} - (w_{00} + \sum_{j=1}^{d_1} w_{0j} s_{0j}(\mathbf{x}_j) + \sum_{j=(d_1+1)}^{p} \sum_{i=1}^{n} w_{ij} s_{ij}(\mathbf{x}_j))||_2^2,$$

where $\mathbf{y}$ is the observed response and $\mathbf{x}_j$ are the observations of covariate j. The algorithm for solving the problem is given in Chen and Samworth (2015). It is implemented in the R-package *scar*, and we will refer to this method as scar.

4 Monotone regression when $p > n$

In this section we present and work with methods for monotone regression in the high dimensional data setting. The methods that are considered are the liso method and the monotone splines lasso method. A simulation experiment is included for illustration of the methods. In addition, we study the robustness of the monotone splines lasso method to the number of interior knots used.

4.1 Liso regression

The most common method for modelling monotone relationships is to use isotonic regression, which produces step functions instead of smooth functions. For high dimensional data, there has been developed a method, lasso isotone (liso), which combines isotonic regression with lasso. It is defined as the minimisation of the liso loss, L_λ (Fang and Meinshausen, 2012), with respect to $(g_1, g_2, \ldots, g_p)$. The liso loss L_λ is given by

$$L_\lambda(\beta_0, g_1, \ldots, g_p) = \frac{1}{2}||\mathbf{y} - \beta_0 - \sum_{j=1}^{p} g_j(\mathbf{X}^{(j)})||_2^2 + \lambda \sum_{j=1}^{p} \Delta(g_j),$$

where $\mathbf{X}^{(j)}$ is the jth column of $\mathbf{X}$, and the g_js are bounded, univariate and monotonically increasing functions. If g_j is monotonically decreasing, the same method/algorithm is used, but with reversed sign on the observed covariates. The $\Delta(g_j)$ denotes the total variation in g_j

$$\Delta(g_j) = \sup_{x \in \mathbb{R}} g_j(x) - \inf_{x \in \mathbb{R}} g_j(x).$$

The residual error only considers the value of g_j at the observed points. Thus for optimality, the bounds for g_j should be at the extremal observed value of the covariate. Outside the interval between the smallest and the largest observed value of the covariate, the function should be flat. Any interpolation function between the points minimising L_λ will be an optimal solution. Therefore, for simplicity, right-continuous step functions are normally used, with knots at the observation points. To perform the fitting of the liso method, we need to know a priori if the covariates are monotonically increasing or monotonically decreasing. The liso method can be improved by an adaptive procedure similar to what is done in section 2.4.1. This improved method is called adaptive liso, and it can be used without prior knowledge about the monotonicity directions of the functions. So the adaptive liso has the advantage over the lower dimensional methods for monotone regression that it does not need to be provided the monotonicity directions. Let $\hat{g}_j^{\text{init}}$ for $j = 1, \ldots, p$ be initial liso fits for the functions. Let then

$$w_j = \begin{cases} \infty, & \text{if } \Delta\hat{g}_j^{\text{init}} = 0, \\ \frac{1}{\Delta\hat{g}_j^{\text{init}}}, & \text{otherwise,} \end{cases}$$

for $j = 1, \ldots, p$. The adaptive liso fit is then given by

$$(\hat{g_1}, \hat{g_2}, \ldots, \hat{g_p}) = \text{argmin}_{(g_1, g_2, \ldots, g_p)} \frac{1}{2} ||\mathbf{y} - \beta_0 - \sum_{j=1}^{p} g_j(\mathbf{X}^{(j)})||_2^2 + \lambda \sum_{j=1}^{p} w_j \Delta(g_j).$$

Even though the adaptive liso does not have to be provided the monotonicity directions, it does have the disadvantage of not guaranteeing a monotone fit, but it shrinks the estimated functions towards monotone functions.

4.2 Monotone splines lasso regression

The monotone splines lasso method is a recently developed method for monotone regression in high dimensions by Bergersen et al. (2014). With this method, the fitted functions are smooth monotone functions. In applications, it is often more reasonable to assume that the true underlying function is smooth (rather than a step function as in liso). To apply the monotone splines lasso method, the monotonicity directions do not need to be known a priori. Consider again the model in equation (3.1), and assume that the functions g_j can be approximated by m spline basis functions of order l, so that

$$\tilde{g}_j(x) = \sum_{k=1}^{m} \beta_{jk} I_k^{(l)}(x),$$

where $I_k^{(l)}$ are the basis functions, β_{jk}, $k = 1, \ldots, m$, are the basis coefficients for covariate j in the spline basis and $\tilde{g}_j$ is a spline approximation of g_j. If there is no relationship between the response y and covariate j, then $\beta_{jk} = 0$ for $k = 1, 2, \ldots, m$. The spline basis functions used are the integrated spline basis functions, I-spline basis functions. I-splines are integrals of M-splines, where M-splines are positive splines. Since the integral of a positive function is monotonically increasing, this ensures that the I-spline basis functions are monotonically increasing. This again means that since the I-spline basis functions are monotonically increasing, $\tilde{g}_j$ will be monotone as long as for each j, all the coefficients β_{jk}, $k = 1, \ldots, m$, have the same sign. So β_{jk}, $k = 1, 2, \ldots, m$, are either all nonnegative, all nonpositive or all zero.

The covariates are assumed to be transformed to $[0, 1]$ (without loss of generality). Define the knot sequence $0 = t_1 = t_2 = \ldots = t_l < \ldots < t_{l+K+1} = \ldots = t_{K+2l} = 1$. K is the number of internal knots, placed evenly at the quantiles of the data. As also noted in section 3.1, the order, l, controls the continuity of the fitted spline for each covariate. If $l = 1$, then the fitted function is continuous at the internal knots. If $l = 2$, then the fitted function is continuous at the internal knots, and has continuous first derivatives at the internal knots (Ramsay, 1988). The number of spline basis functions needed to fit the function, m, is $K + l$. The

I-spline basis functions of order $l = 2$ are given by (Tutz and Leitenstorfer, 2007)

$$I_k^2(x) = \begin{cases} 0, & \text{if } x \leq t_k, \\ \frac{(x-t_k)^2}{(t_{k+1}-t_k)(t_{k+2}-t_k)}, & \text{if } t_k < x \leq t_{k+1}, \\ 1 - \frac{(t_{k+2}-x)^2}{(t_{k+2}-t_k)(t_{k+2}-t_{k+1})}, & \text{if } t_{k+1} < x \leq t_{k+2}, \\ 1, & \text{if } x > t_{k+2}. \end{cases}$$

We will use these I-spline basis functions of order two. The I-spline basis functions of order two are plotted in Figure 2, where six interior knots are used, equidistantly placed in [0,1]. As in Bergersen et al. (2014), the I-spline basis functions are centered so that $\mathrm{E}[\tilde{g}_j(x)] = 0$, to ensure unique identification of the functions. Let $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_p)$ be the $n \times pm$ design matrix with the covariates represented in the I-spline basis, where $\mathbf{Z}_j$ is the $n \times m$ design matrix for covariate j represented in the I-spline basis. Let $\boldsymbol{\beta}$ be the corresponding vector of basis coefficients. Then consider the minimisation problem

$$\hat{\boldsymbol{\beta}} = \mathrm{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{Z}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_{\mathrm{coop}},$$

where λ controls the regularisation as before, and a cooperative lasso penalty is used to ensure that the estimated coefficients for each covariate are sign-coherent. In Bergersen et al. (2014), λ is chosen by cross-validation. The estimated function with the monotone splines lasso method is then

$$\hat{\tilde{g}}_j(x) = \sum_{k=1}^{m} \hat{\beta}_{jk} I_k^{(l)}(x).$$

Since the cooperative penalty has the variable selection property, the monotone splines lasso method can perform variable selection. Then the whole group of parameters for covariate j will be set to zero, so $\hat{\beta}_{jk} = 0$ for $k = 1, \ldots, m$. If the covariate j is selected by the method, all the parameters within one group will be nonnegative or nonpositive, as long as the penalty parameter is large enough (see section 2.4.3).

The monotone splines lasso can also be improved by an adaptive procedure in a similar way as described in section 2.4.1. This improved method is called adaptive monotone splines lasso. Let $\hat{\boldsymbol{\beta}}_j^{\mathrm{init}}$ be the initial fit for the basis coefficients for covariate j, for $j = 1, \ldots, p$. The adaptive monotone splines lasso estimates are then given by

$$\hat{\boldsymbol{\beta}} = \mathrm{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{Z}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{p} w_j (||\boldsymbol{\beta}_j^+||_2 + ||\boldsymbol{\beta}_j^-||_2),$$

where $\boldsymbol{\beta}_j$ are the m basis coefficients corresponding to covariate j, and

$$w_j = \begin{cases} \infty, & \text{if } ||\hat{\boldsymbol{\beta}}_j^{\mathrm{init}}||_2 = 0, \\ \frac{1}{||\hat{\boldsymbol{\beta}}_j^{\mathrm{init}}||_2}, & \text{otherwise.} \end{cases}$$

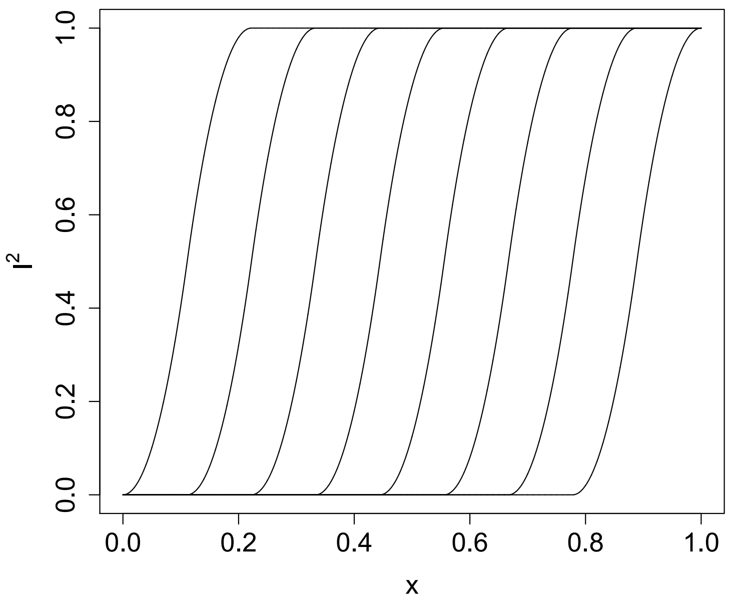


Figure 2: I-spline basis functions of order two with six interior knots placed equidistantly in [0,1].

4.3 Simulation example in the high dimensional setting

To demonstrate how the monotone regression methods work in the high dimensional setting, we perform a simulation experiment similar to the simulation experiment in Bergersen et al. (2014). There are n observations and p covariates.

We draw random variables $\mathbf{x} = (x_{11}, x_{12}, \dots, x_{1n}, \dots, x_{p1}, \dots, x_{pn})$, where the x_{ij} are drawn from a standard normal distribution truncated to $[0, 1]$.

The purpose of the simulation experiment is to compare estimation and variable selection for different methods. As in Bergersen et al. (2014), the methods considered are the monotone splines lasso, the adaptive monotone splines lasso, the adaptive liso, lasso and the adaptive lasso. Bergersen et al. (2014) also looked at BS-lasso, which is an adaptive group lasso method using B-splines, see Huang et al. (2010). However, we do not consider this method in this thesis, since we will only work with monotone regression methods.

For the monotone splines lasso and the adaptive monotone splines lasso, we use the R-package *mslasso* developed by Bergersen as described in Bergersen et al. (2014). Six evenly distributed internal knots are used, placed at the quantiles of the covariates. For the adaptive liso, we use the R-package *scoop*. For the lasso and the adaptive lasso, we use the R-package *glmnet*. For all the methods, a 10-fold cross-validation scheme is used to find the optimal penalisation parameter.

The goals of these methods are different. The monotone splines lasso method and the liso method fit general monotone functions. Monotone splines lasso does this by fitting smooth functions and liso does this by fitting step functions. Lasso fits linear functions. All these methods are developed for high dimensional data.

As in Bergersen et al. (2014), the following functions are considered

$$g_1(x) = -\exp x^2,$$

$$g_2(x) = -\log(x + 0.1),$$

$$g_3(x) = 2\tanh(20x^2) + 0.5\exp(x^3),$$

and

$$g_4(x) = \frac{2\exp(10x - 5)}{1 + \exp(10x - 5)}.$$

Then let

$$y_i = g_1(x_{i1}) + g_2(x_{i2}) + g_3(x_{i3}) + g_4(x_{i4}) + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$, and σ is chosen to control the signal to noise ratio (SNR), so $\sigma = \text{sd}(g(\mathbf{X}_{\mathscr{A}}))/\text{SNR}$, where $g(\mathbf{X}_{\mathscr{A}}) = \sum_{j=1}^{4} g_j(x_{ij})$, and $\mathscr{A}$ denotes the set of true (active) covariates. sd is the standard deviation. The functions are centered since the monotone splines lasso method and the liso method assume that the expected values of the functions are zero. Let $n = 50$ and $p = 1000$, so that there are 996 noise covariates.

We simulate 100 times, and if we get a non-monotonic fit for monotone splines lasso (too small regularisation parameter, see section 2.4.3), the results for this one simulation are discarded. The ratio of the number of times each true covariate is selected, the mean number of true covariates selected (true positives, TP), the mean number of false covariates selected (false positives, FP) and the mean squared errors (MSE) for the fitted functions with the different methods are given in Table 2. MS-lasso is the monotone splines lasso, Ad. MS-lasso is the adaptive monotone splines lasso, Ad. liso is the adaptive liso and Ad. lasso is the adaptive lasso. The mean squared errors are estimated by considering the value of the true functions and the value of the fitted functions at the observation points. We see from Table 2 that the monotone splines lasso method was best at selecting the true covariates, but it also selected many false covariates. The adaptive monotone splines lasso method selected fewer true covariates, but it decreased the number of false positives substantially. The adaptive monotone splines lasso method was better than adaptive liso in recovering the true model, both when considering true and false positives. From Table 2, we observe that the adaptive monotone splines method has the smallest estimation error among all the methods.

The estimated functions with monotone splines lasso and adaptive monotone splines lasso are given in Figure 3. The estimated functions with adaptive liso are given in Figure 4 and the estimated functions with lasso and adaptive lasso are given in Figure 5. Note that in Figure 3, 200 curves are drawn, because we have chosen to draw the estimated functions with monotone splines lasso and adaptive monotone splines lasso in the same plot. This is done to make it easier to compare these methods. Similarly, in Figure 5, the estimated functions with lasso and adaptive lasso are given in the same plot. This gives an optical impression of higher variability, so to get a correct impression of the variability, it is necessary to separate the plots by colour. From Figure 3, it is apparent that the functions fitted by adaptive monotone splines lasso are much closer to the true function than the functions fitted by monotone splines lasso. This coincides with the mean squared errors. From Figure 5 it is also clear that the adaptive lasso method is better than the lasso method in recovering the true function. All of the monotone regression methods seem to be good at recovering the shapes of the functions.

The results from the simulation in this section were not identical to the simulation results in Bergersen et al. (2014). This is not only due to randomness in the observations (noise), but also because a different design matrix was used in the simulations. In Appendix A, results from simulation experiments with different random design matrices for each simulation run are given. In addition, simulation experiments with a design matrix having good variable selection results for the monotone splines lasso and a matrix having bad variable selection results for the monotone splines lasso are performed.

From the simulations and the discussion in Appendix A, we find that the liso regression method and the monotone splines lasso regression method are

Selection

	g_1	g_2	g_3	g_4
MS-lasso	0.95 (0.22)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. MS-lasso	0.79 (0.41)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. liso	0.73 (0.45)	1.0 (0)	0.99 (0.10)	1.0 (0)
Lasso	0.86 (0.35)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. lasso	0.74 (0.44)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
MS-lasso	3.95 (0.22)	24.48 (9.39)		
Ad. MS-lasso	3.79 (0.41)	2.18 (3.82)		
Ad. liso	3.72 (0.47)	3.35 (1.83)		
Lasso	3.86 (0.35)	37.17 (17.25)		
Ad. lasso	3.74 (0.44)	11.25 (3.70)		

Estimation

	g_1	g_2	g_3	g_4
MS-lasso	0.067 (0.031)	0.038 (0.023)	0.093 (0.020)	0.096 (0.047)
Ad. MS-lasso	0.036 (0.025)	0.015 (0.011)	0.045 (0.020)	0.047 (0.032)
Ad. liso	0.045 (0.030)	0.047 (0.031)	0.052 (0.021)	0.036 (0.024)
Lasso	0.12 (0.035)	0.082 (0.029)	0.12 (0.015)	0.17 (0.064)
Ad. lasso	0.079 (0.038)	0.051 (0.016)	0.11 (0.0070)	0.088 (0.043)

Table 2: Ratio of the number of times the different covariates are selected, ratio of the total true (TP) and false positives (FP) and mean squared errors (MSE) for the estimated functions in the simulation experiment considered in section 4.3, where SNR ≈ 4, $p = 1000$ and $n = 50$. Standard deviations are given in parenthesis.

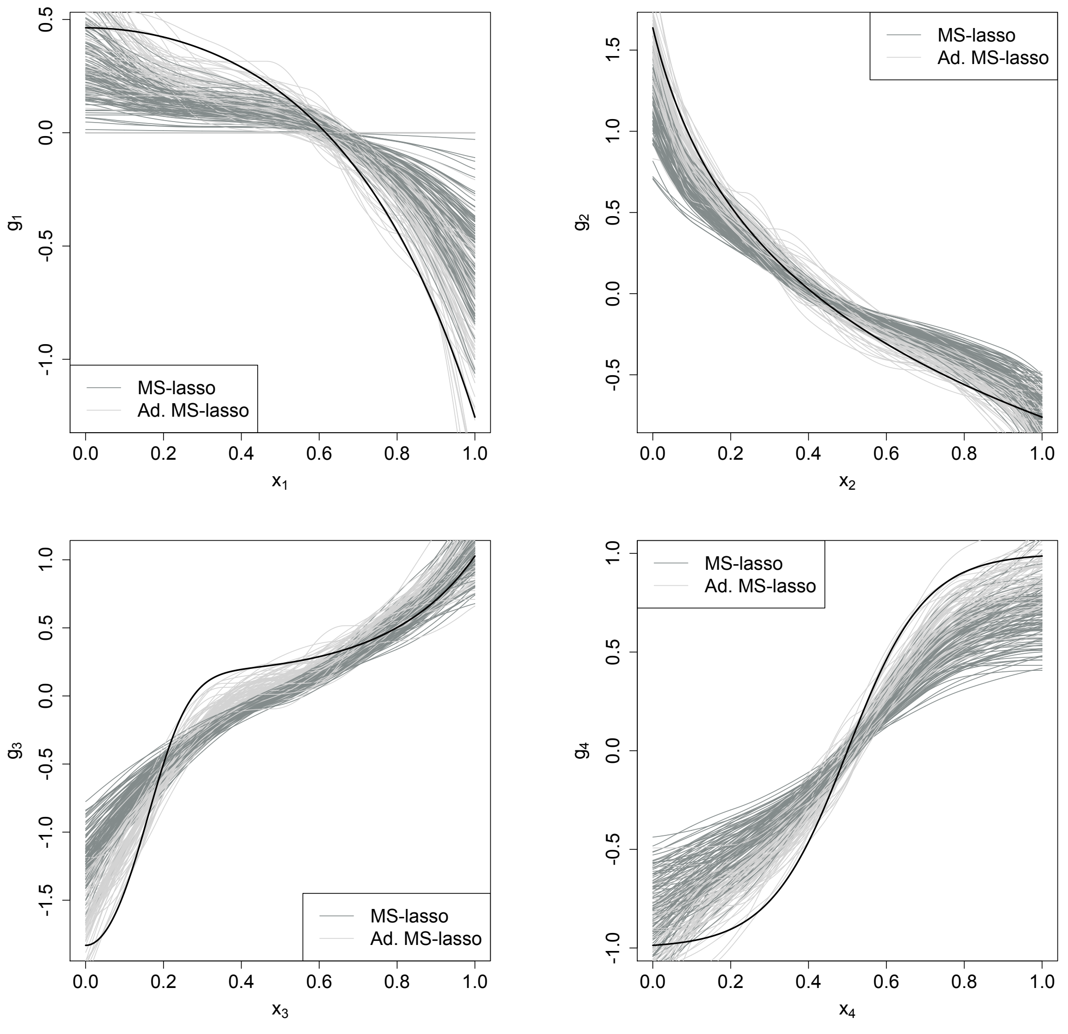


Figure 3: Estimated functions in the simulation considered in section 4.3 with $n = 50$, $p = 1000$ and SNR ≈ 4 with the monotone splines lasso (grey) and the adaptive monotone splines lasso (light grey). The true function is given in black.

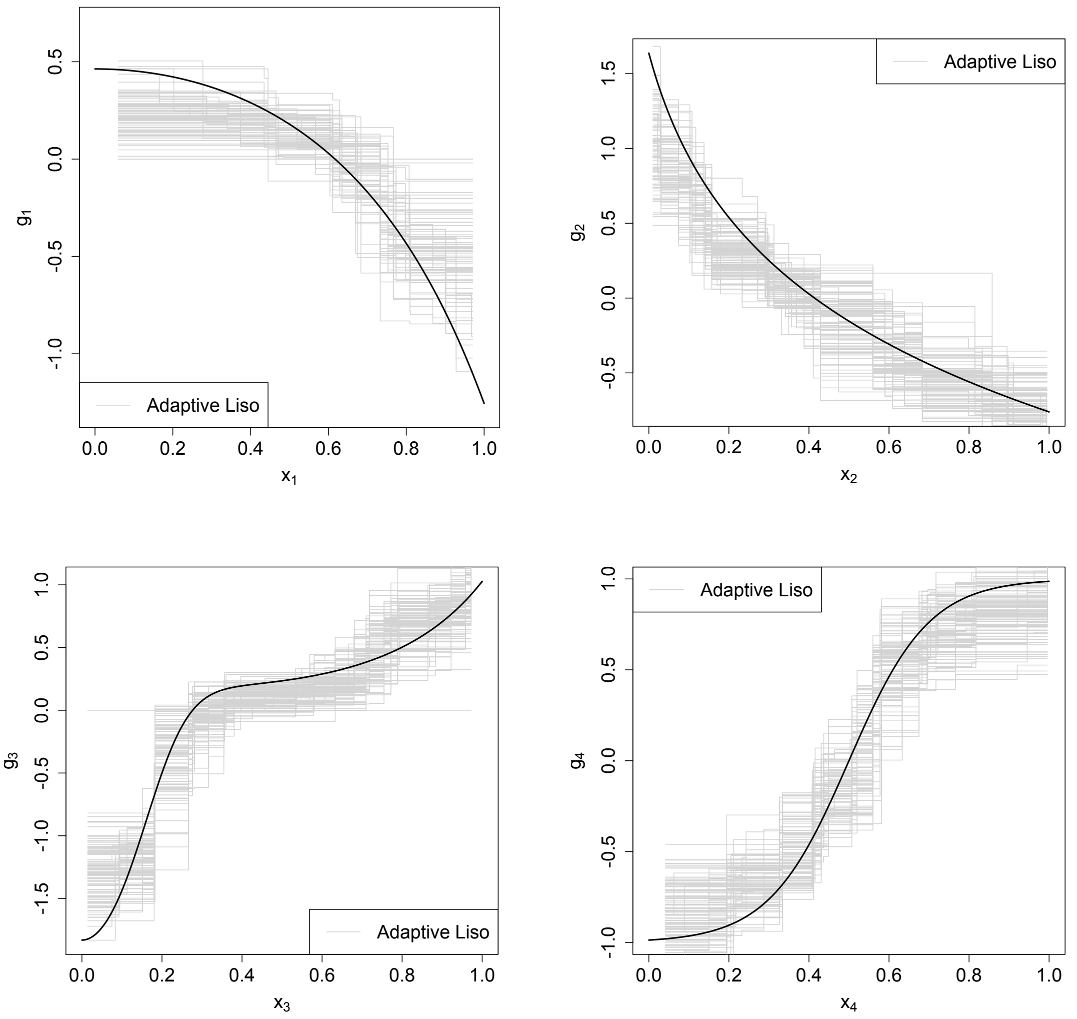


Figure 4: Estimated functions in the simulation considered in section 4.3 with $n = 50$, $p = 1000$ and SNR ≈ 4 with adaptive liso (grey). The true function is given in black.

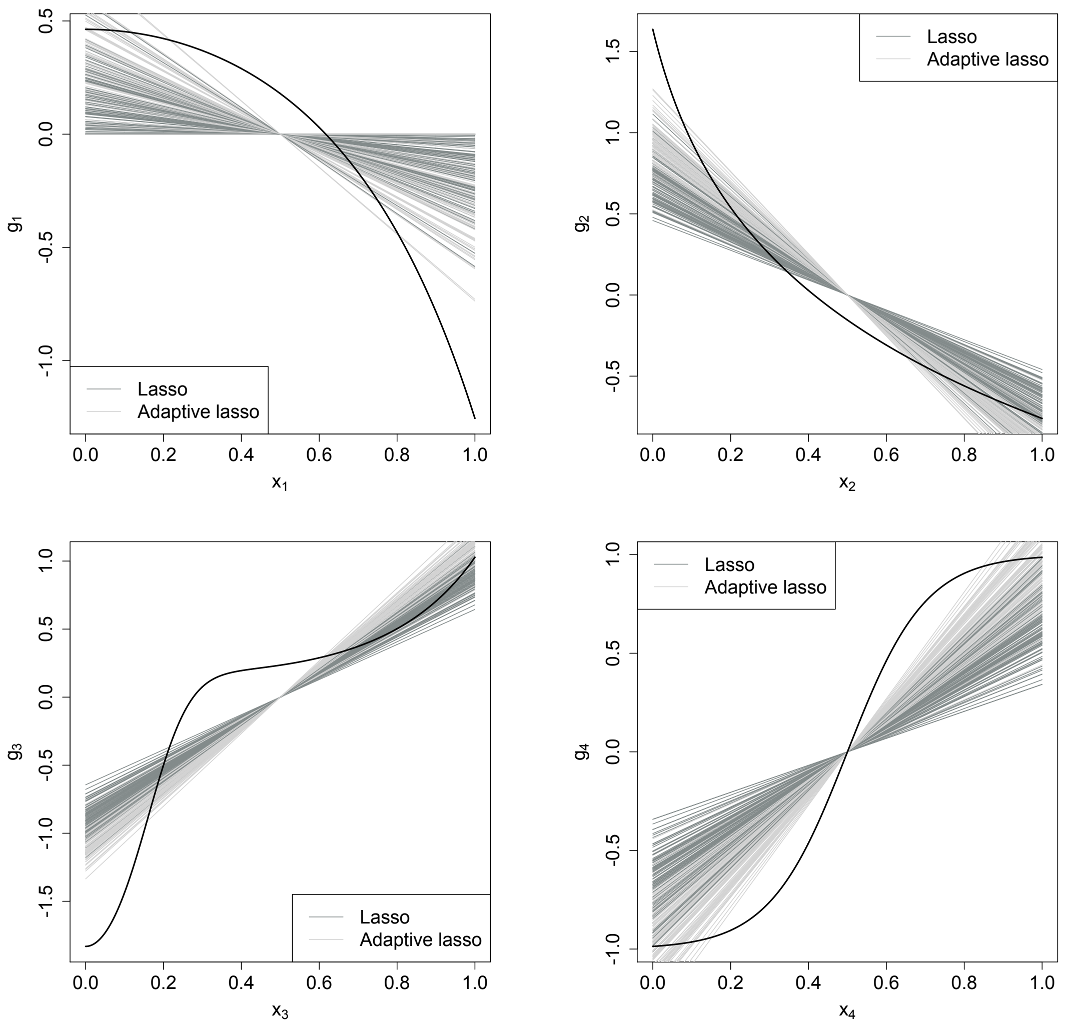


Figure 5: Estimated functions in the simulation considered in section 4.3 with $n = 50$, $p = 1000$ and SNR ≈ 4 with the lasso method (grey) and the adaptive lasso method (light grey). The true function is given in black.

sensitive to the location of the observations. Clustering of observations will give good function estimation in the local area of these points, but not on the whole interval. To detect an effect of a variable, it is of course advantageous to have observations in the area where the function is steepest, especially if it is quite flat elsewhere.

We conclude that the results are in general better when the random x_{ij} are drawn from a normal distribution with mean 0.5 and standard deviation 1, truncated to $[0, 1]$. Since we find a symmetric distribution more reasonable, the random variables in all the following simulation experiments in this thesis will be drawn from this distribution. In addition, a different design matrix will be drawn in each simulation, to cover more situations and to give a more fair comparison.

4.4 Importance of the number of knots in monotone splines lasso

In the simulation with monotone splines lasso in section 4.3, six interior knots placed evenly at the quantiles of the data were used to fit the splines to the functions. We wish to study the importance of the number of knots used to fit the functions. A greater flexibility for the function is obtained the more knots used to fit it. However, the more data points there are to estimate each curve, the better will each curve estimate be. With more knots, there are more spline basis functions to fit, and thus less data points for each spline basis fit. In addition, if too many knots are used, the estimated functions might overfit. Ramsay (1988) argues that it is more important to fit each curve well, since there is little to gain from having many knots if the function is poorly estimated between the knots. He also states that in practice, there is often enough flexibility in the curve with a single interior knot, so that we do not need a large number of knots. Although splines in general are sensitive to the placement and number of interior knots, Meyer (2008) argues that when there are shape-restrictions (such as monotonicity), then the restricted regression splines are robust to the number of knots. Delecroix and Thomas-Agnan (2000) state that the number of knots recommended in an I-spline procedure (as with monotone splines lasso) is usually much smaller than the number of observations.

We will check if the fits are actually robust to the number of knots used, since inference and method comparison is problematic if the method is not robust to the number of knots (Meyer, 2008). Since we are already in an over-parametrised situation with lack of information, we do not want to use a model selection criterion for selecting the number of knots when using monotone splines lasso, so it is important that the model fit is robust to the number of knots used. Following Ramsay (1988) and Delecroix and Thomas-Agnan (2000), we will not test for many interior knots.

A simulation experiment similar to the previous simulation experiments is performed, where the number of internal knots, K, vary. We consider $K = 4, 5, 6, 7$ and 8. As before, I-spline basis functions of order two are used. There are $n = 50$ observations, $p = 1000$ parameters and SNR ≈ 4. To select the optimal tuning parameter λ, 10-fold cross-validation is used. Let

$$y_i = g_1(x_{i1}) + g_2(x_{i2}) + g_3(x_{i3}) + g_4(x_{i4}) + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$, and σ is chosen to control the signal to noise ratio (SNR). The g-functions are as in section 4.3. The x_{ij} are drawn from a normal distribution with mean 0.5 and standard deviation 1, truncated to $[0, 1]$.

The aim is to compare variable selection and estimation when varying the number of interior knots. We simulate 100 times drawing new observations for each simulation. The number of true and false covariates selected and the mean squared errors for monotone splines lasso are given in Table 3. The number of true and false covariates selected and the mean squared errors for adaptive monotone splines lasso are given in Table 4. We see that there are no big differences between the performance with the different number of knots for neither monotone splines lasso nor adaptive monotone splines lasso. The selection performance of the methods is very similar for the different number of interior knots. The estimation errors are also very similar. The only notable difference is that the number of false covariates selected by adaptive monotone splines lasso seems to be decreasing with the number of interior knots. This is probably due to the fact that the more interior knots you have, the more degrees of freedom are used to include a covariate. However, the differences are small, and this effect is not present for monotone splines lasso. Looking closer at one specific simulation, we get the estimated mean squared errors given in Table 5. The fitted functions for g_1 and g_4 for this simulation are given in Figure 6. We see from Figure 6 that the fitted functions are very close to each other even though different number of spline basis functions are used to fit them. From Table 5 we also have that the mean squared errors are quite similar, and there is no systematic difference. So the monotone splines lasso (and the adaptive monotone splines lasso) is robust to the number of interior knots used, and it should be safe to use this method without estimating the number of interior knots.

Selection

	g_1	g_2	g_3	g_4
4 interior knots	0.87 (0.34)	0.98 (0.14)	0.99 (0.10)	0.99 (0.10)
5 interior knots	0.87 (0.34)	0.98 (0.14)	0.99 (0.10)	0.99 (0.10)
6 interior knots	0.88 (0.33)	0.98 (0.14)	0.99 (0.10)	0.99 (0.10)
7 interior knots	0.87 (0.34)	0.96 (0.20)	0.99 (0.10)	0.99 (0.10)
8 interior knots	0.88 (0.33)	0.97 (0.17)	0.99 (0.10)	0.99 (0.10)
	TP	FP		
4 interior knots	3.83 (0.45)	15.66 (10.04)		
5 interior knots	3.83 (0.45)	15.05 (9.60)		
6 interior knots	3.84 (0.44)	15.23 (9.59)		
7 interior knots	3.81 (0.49)	14.65 (9.52)		
8 interior knots	3.83 (0.45)	14.90 (9.83)		

Estimation

	g_1	g_2	g_3	g_4
4 interior knots	0.069 (0.044)	0.077 (0.054)	0.10 (0.048)	0.11 (0.069)
5 interior knots	0.070 (0.043)	0.079 (0.053)	0.10 (0.048)	0.11 (0.069)
6 interior knots	0.072 (0.046)	0.077 (0.052)	0.10 (0.049)	0.11 (0.066)
7 interior knots	0.074 (0.049)	0.078 (0.052)	0.11 (0.063)	0.11 (0.068)
8 interior knots	0.075 (0.049)	0.078 (0.049)	0.11 (0.053)	0.11 (0.064)

Table 3: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions, with different number of interior knots for monotone splines lasso in the simulation experiment considered in section 4.4. Standard deviations are given in parenthesis.

Selection

	g_1	g_2	g_3	g_4
4 interior knots	0.72 (0.45)	0.94 (0.24)	0.99 (0.11)	0.99 (0.11)
5 interior knots	0.76 (0.43)	0.94 (0.25)	0.99 (0.11)	0.99 (0.11)
6 interior knots	0.73 (0.45)	0.94 (0.23)	0.99 (0.11)	0.98 (0.15)
7 interior knots	0.70 (0.46)	0.93 (0.25)	0.99 (0.11)	0.98 (0.15)
8 interior knots	0.72 (0.45)	0.94 (0.24)	0.98 (0.15)	0.99 (0.11)

	TP	FP
4 interior knots	3.64 (0.61)	1.19 (2.97)
5 interior knots	3.67 (0.62)	0.82 (2.37)
6 interior knots	3.64 (0.63)	0.69 (2.01)
7 interior knots	3.60 (0.66)	0.37 (1.16)
8 interior knots	3.63 (0.64)	0.34 (1.11)

Estimation

	g_1	g_2	g_3	g_4
4 interior knots	0.041 (0.030)	0.042 (0.041)	0.044 (0.035)	0.072 (0.063)
5 interior knots	0.042 (0.029)	0.041 (0.040)	0.046 (0.036)	0.071 (0.069)
6 interior knots	0.044 (0.029)	0.046 (0.041)	0.048 (0.040)	0.068 (0.055)
7 interior knots	0.045 (0.031)	0.048 (0.043)	0.051 (0.043)	0.073 (0.061)
8 interior knots	0.045 (0.033)	0.049 (0.043)	0.048 (0.036)	0.073 (0.065)

Table 4: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions, with different number of interior knots for adaptive monotone splines lasso in the simulation considered in section 4.4. Standard deviations are given in parenthesis.

Estimation				
MS-lasso				
	g_1	g_2	g_3	g_4
4 interior knots	0.051	0.077	0.063	0.030
5 interior knots	0.048	0.071	0.054	0.026
6 interior knots	0.049	0.080	0.059	0.029
7 interior knots	0.046	0.072	0.048	0.022
8 interior knots	0.048	0.079	0.051	0.027
Ad. MS-lasso				
	g_1	g_2	g_3	g_4
4 interior knots	0.034	0.047	0.020	0.0093
5 interior knots	0.043	0.032	0.017	0.0097
6 interior knots	0.039	0.033	0.017	0.0084
7 interior knots	0.035	0.029	0.014	0.0063
8 interior knots	0.036	0.034	0.015	0.0087

Table 5: Mean squared errors for the different functions for one simulation with monotone splines lasso and adaptive monotone splines lasso with different number of interior knots for the simulation considered in section 4.4.

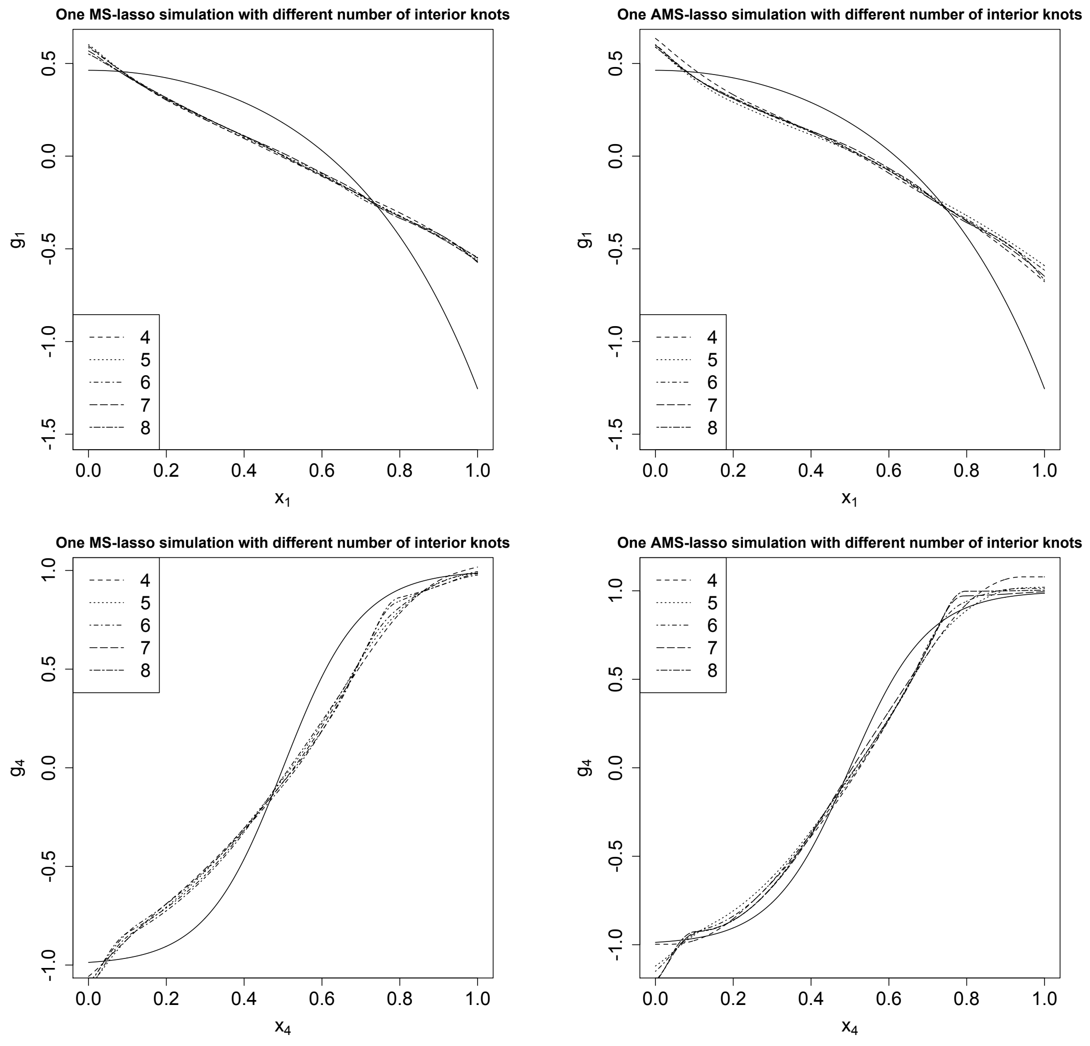


Figure 6: Estimated functions for g_1 and g_4 with different number of interior knots with monotone splines lasso (left) and adaptive monotone splines lasso (right) in the simulation considered in section 4.4. The true function is drawn in a solid line.

5 Comparison of monotone methods when $p < n$

We intend to study the performance of the monotone splines lasso method described in section 4.2 in the classical setting, where there are less parameters than observations (so $p < n$), to see if the method can also be used in this setting.

This is done by performing simulation experiments, where data is simulated in a similar manner as in section 4.3 and Bergersen et al. (2014), with n observations and p parameters.

We draw random variables $\mathbf{v} = (v_1, v_2, \ldots, v_n), \mathbf{u} = (u_1, u_2, \ldots, u_n)$ and $\mathbf{w} = (w_{11}, w_{12}, \ldots, w_{1n}, \ldots, w_{p1}, \ldots, w_{pn})$, where u_i, v_i and w_{ij} are drawn from a normal distribution with mean 0.5 and standard deviation 1, truncated to $[0, 1]$. We choose to draw from this distribution rather than the one used in Bergersen et al. (2014), since we find it more realistic to work with a symmetric distribution, as noted in the end of section 4.3.

We let

$$x_{ij} = \frac{w_{ij} + tu_i}{1 + t} \text{ for } j \in \mathscr{A} \ ,$$

and

$$x_{ij} = \frac{w_{ij} + tv_i}{1 + t} \text{ for } j \notin \mathscr{A} \ ,$$

where $\mathscr{A}$ is the set of true covariates. The dependence between the covariates is controlled by t, and the covariates are independent when $t = 0$. This is the same simulation set up as in Bergersen et al. (2014), except from the fact that the random variables are drawn from a slightly different distribution.

As in section 4.3, we let

$$y_i = g_1(x_{i1}) + g_2(x_{i2}) + g_3(x_{i3}) + g_4(x_{i4}) + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$, and σ is chosen to control the SNR. The g-functions are as in section 4.3. The functions are centered as in section 4.3, since the monotone regression methods assume that the expected values of the functions are zero.

We wish to compare estimation and variable selection for different methods. The methods that are compared are the monotone splines lasso, the adaptive monotone splines lasso, the adaptive liso, scam, scar and classical linear regression using ordinary least squares. We use the R-function *lm* to fit the linear model. We do not use the liso method for comparison, since it needs prior knowledge about the monotonicity directions of the functions, while adaptive liso does not. Scam and scar also need to be provided the monotonicity directions, but since there are no alternative versions of these methods which do not need the monotonicity directions, these methods will still be used for comparison. To estimate the optimal penalisation parameter for monotone splines lasso and adaptive liso, a 10-fold cross-validation scheme is used. For the monotone splines lasso, six evenly distributed internal knots placed at the quantiles of the observed data and I-spline basis functions of order two are used, as before. Ten interior knots are used to fit the functions with scam, and no shape constraints are put on the noise covariates. The smoothing parameters for scam are chosen by the default GCV option in the implementation. With scar, it is not possible to have no shape constraints on the functions, so the noise covariates are fitted by a linear function.

In the classical linear regression setting, linear functions are fitted, but not general monotone functions. The ordinary least squares, scam and scar assume that $p < n$.

5.1 Variable selection and estimation in simulation experiments

We intend to study and compare the performance of the monotone splines lasso method to the performances of the other methods in the classical setting. We let $n = 80$ and $p = 7$. We simulate 100 times, drawing new x_{ij} for each simulation. If one of the simulation runs with the monotone splines lasso method does not give a monotonic fit, the results from this one simulation run are discarded. Since $p = 7$, there are four true covariates and three false (noise) covariates. The number of true covariates selected, the number of false covariates selected and the mean squared errors from the estimated functions to the true functions in the observed points are recorded. The performance is studied in a situation with large noise, SNR ≈ 2, and with less noise, SNR ≈ 4. Both dependent ($t = 1$) and independent ($t = 0$) covariates are also considered.

5.1.1 Strong signal and independent covariates

The first situation considered is the situation with a strong signal and independent covariates, so we let SNR ≈ 4 and $t = 0$. The number of true positives, the number of false positives and the estimation errors for the estimated functions are given in Table 6. Lin. mod is the ordinary least squares fit. Since the scar, scam and linear regression method do not perform variable selection, we need a criterion for whether or not a covariate is selected. The implementation of scam and *lm* both provide p-values, so we choose to record the covariates which have an effect on the response at significance level 0.05. The implementation of scar does not provide any p-values, and therefore we do not record the selected variables of scar, but we can still compare the mean squared errors of the fitted functions.

The fitted functions with the monotone splines lasso and the adaptive monotone splines lasso are given in Figure 7. The fitted functions with the adaptive liso are given in Figure 8. The fitted functions with scam are given in Figure 9. The fitted functions with scar are given in Figure 10 and the fitted functions with the linear method are given in Figure 11.

We see from Table 6 that all the methods select all the true covariates in each simulation. The adaptive monotone splines lasso method is the only method which does not select any false covariates. Hence in variable selection, the adaptive monotone splines method seems to perform the best. The scam method selects the most false covariates, followed by the linear method. Adaptive liso selected slightly more false covariates than monotone splines lasso in this simulation. Note however that the variable selection criterion for scam and the linear method are not data driven, and the comparison is thus not completely fair.

Considering the estimation error, we find that the estimated functions with scam are a lot closer to the true functions than the estimated functions with any of the other methods. Scar and adaptive liso have quite similar mean squared errors, and they have smaller estimation errors than the adaptive monotone splines lasso. The adaptive monotone splines lasso has smaller estimation errors than monotone splines lasso. We see from Figures 7, 8, 9 and 10 that all the monotone regression methods are good at recovering the true shape of the functions. We also observe that the estimated functions with scam are the most accurate.

The reason why adaptive liso performs better than monotone splines lasso in estimation error is that it fits the function at all observation points, and the mean squared error is measured by using the observation points. Since there are relatively many observations, we will have a good fit here. Plots of the observations, the true function and the fitted function for adaptive liso and monotone splines lasso for g_2, from one simulation, are given in Figure 12. The fits for the monotone splines lasso and the adaptive monotone splines lasso are almost inseparable in the figure. We see from the figure that all the estimated functions seem to be capturing the shape of the true function well. The mean squared errors for the fitted functions in Figure 12, calculated by using the observation points, are 0.013 for the fitted function with adaptive liso, 0.026 with monotone splines lasso and 0.019 with adaptive monotone splines lasso. Using instead the overall mean squared error (estimated by evaluating the function at equidistant points from 0 to 1 with a distance of 0.0001 between the points) from the fitted function to the true function, we get that the mean squared error for adaptive liso is 0.067, the mean squared error for monotone splines lasso is 0.021 and the mean squared error for adaptive monotone splines lasso is 0.015, showing that adaptive liso was not better at estimating the function, it was just closer to the true function at the observation points. The mean squared errors for the estimated g_2 for the full function, averaging over all the 100 simulations, are 0.036 for the adaptive liso, 0.030 for the monotone splines lasso and 0.024 for the adaptive monotone splines lasso.

In Figure 7, the clear benefits in using the adaptive monotone splines lasso instead of the monotone splines lasso is no longer apparent. There is a benefit, but the difference is less extreme than it was in the high dimensional setting. We can see that the adaptive monotone splines lasso fits are a little closer to the true function for g_3, but the difference is not at all as obvious as in the higher dimensional setting, comparing with for instance Figure 3. We did however see from Table 6 that the estimation errors are smaller for the adaptive monotone splines lasso method than for the monotone splines lasso method. However, the differences in the high dimensional setting were more extreme, while here the adaptive monotone splines lasso only has slightly smaller estimation errors than monotone splines lasso. As already noted, the adaptive monotone splines method also performed better at variable selection in this setting than the monotone splines lasso.

Comparing the monotone splines lasso with the scam method, which is constructed for monotone regression in the classical setting, we see that the monotone splines lasso method is not as good in estimating the functions, but it has less false covariates, when including all the variables significant at significance level 0.05 for the scam method. It should be noted, however, that the 0.05 level is arbitrarily set, while the variable selection for the monotone splines lasso is guided by the penalty parameter λ, which is chosen by using cross-validation. So the variable selection criterion for the monotone splines lasso is data driven. This makes the comparison not completely fair for scam. It should also be noted that the scam method is fed with more information about the functions, in that it is provided with the monotonicity directions.

Selection

	g_1	g_2	g_3	g_4
MS-lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. MS-lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. liso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Scam	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Lin. mod	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)

	TP	FP
MS-lasso	4.0 (0)	0.08 (0.27)
Ad. MS-lasso	4.0 (0)	0 (0)
Ad. liso	4.0 (0)	0.09 (0.32)
Scam	4.0 (0)	0.30 (0.54)
Lin. mod	4.0 (0)	0.17 (0.38)

Estimation

	g_1	g_2	g_3	g_4
MS-lasso	0.027 (0.017)	0.027 (0.017)	0.044 (0.017)	0.036 (0.017)
Ad. MS-lasso	0.032 (0.024)	0.021 (0.017)	0.025 (0.013)	0.034 (0.021)
Ad. liso	0.016 (0.011)	0.020 (0.0088)	0.020 (0.0080)	0.017 (0.0073)
Scam	0.0065 (0.0058)	0.0052 (0.0046)	0.010 (0.0063)	0.0078 (0.0058)
Scar	0.020 (0.0096)	0.022 (0.0098)	0.024 (0.0094)	0.023 (0.011)
Lin. mod	0.038 (0.013)	0.039 (0.014)	0.12 (0.020)	0.040 (0.0096)

Table 6: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions in the simulation considered in section 5.1.1, where $n = 80$, $p = 7$, SNR ≈ 4 and $t = 0$. Standard deviations are given in parenthesis.

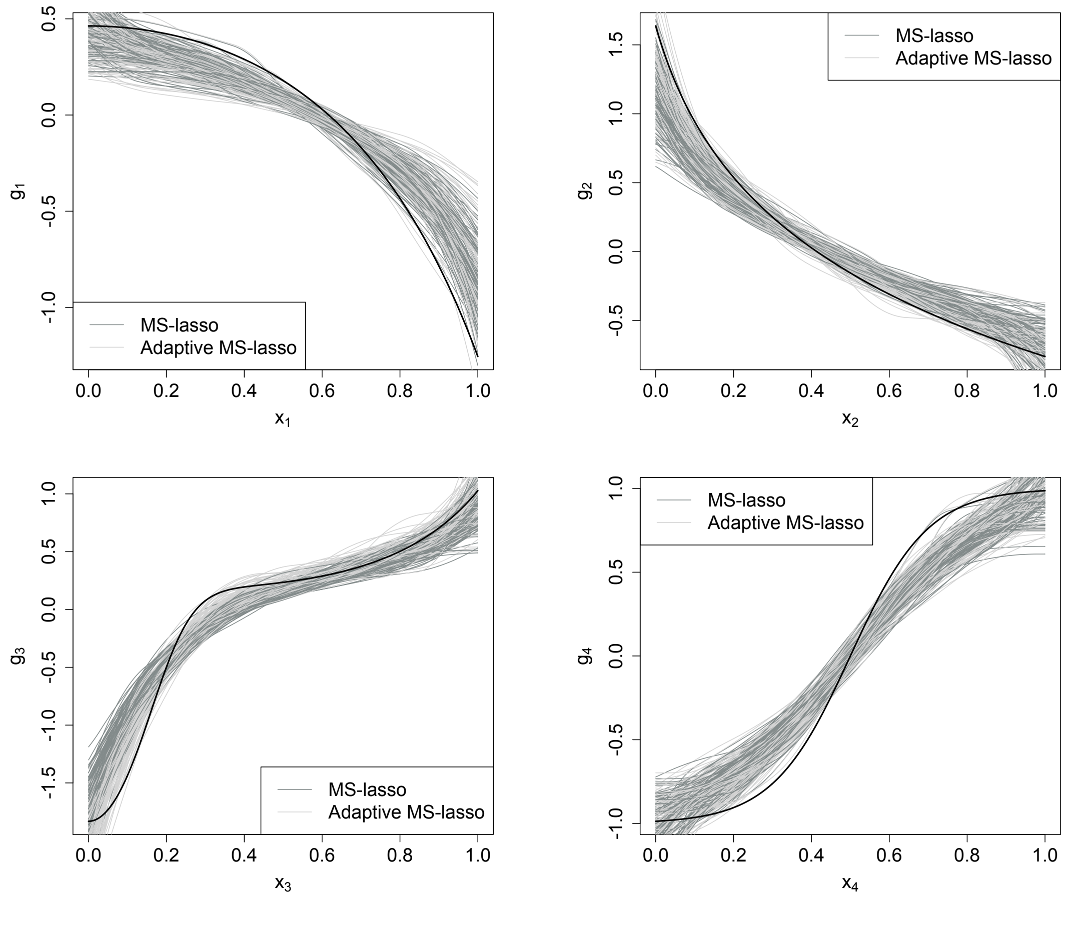


Figure 7: Estimated functions in the simulation considered in section 5.1.1 with $n = 80$, $p = 7$, SNR ≈ 4 and $t = 0$ with the monotone splines lasso (grey) and the adaptive monotone splines lasso (light grey). The true function is given in black.

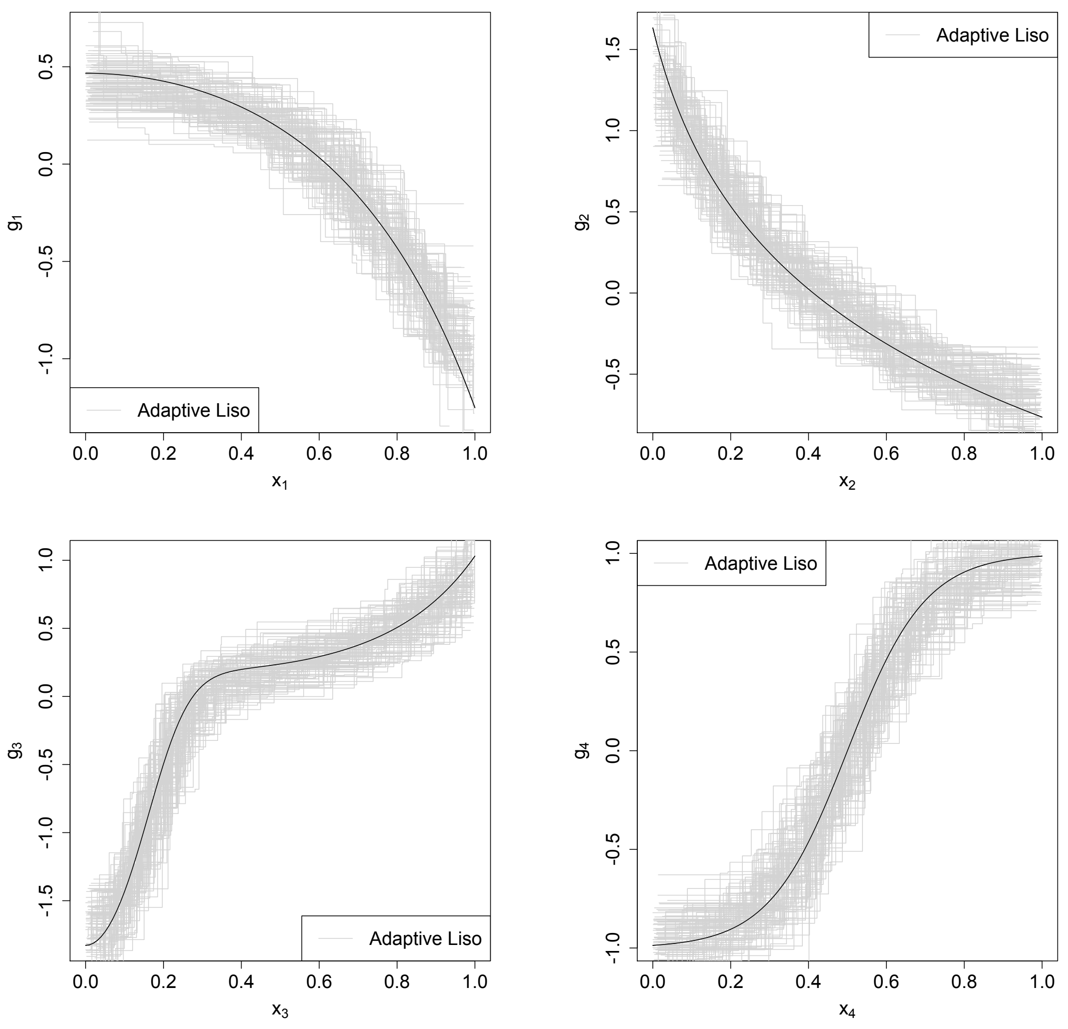


Figure 8: Estimated functions in the simulation considered in section 5.1.1 with $n = 80$, $p = 7$, SNR ≈ 4 and $t = 0$ with the adaptive liso (grey). The true function is given in black.

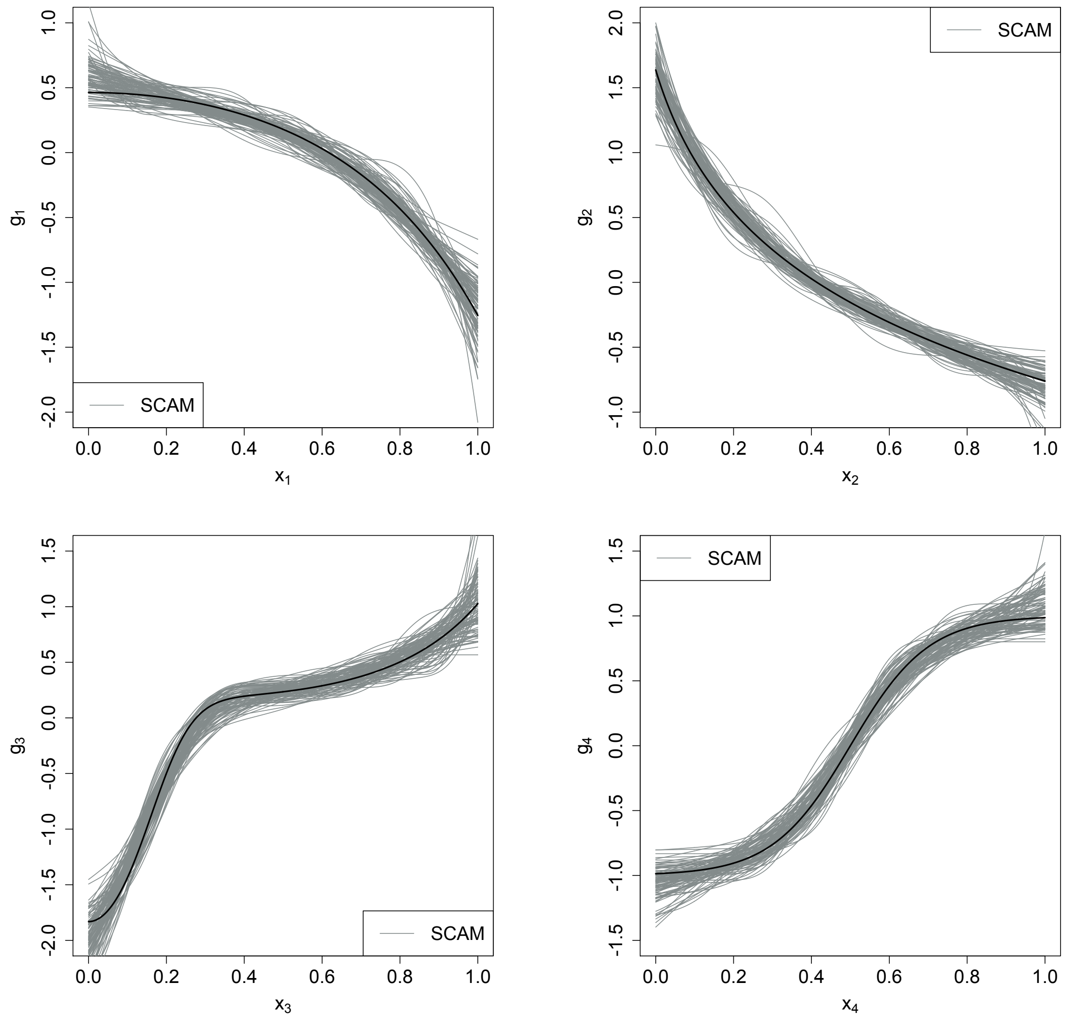


Figure 9: Estimated functions in the simulation considered in section 5.1.1 with $n = 80$, $p = 7$, SNR ≈ 4 and $t = 0$ with the scam method (grey). The true function is given in black.

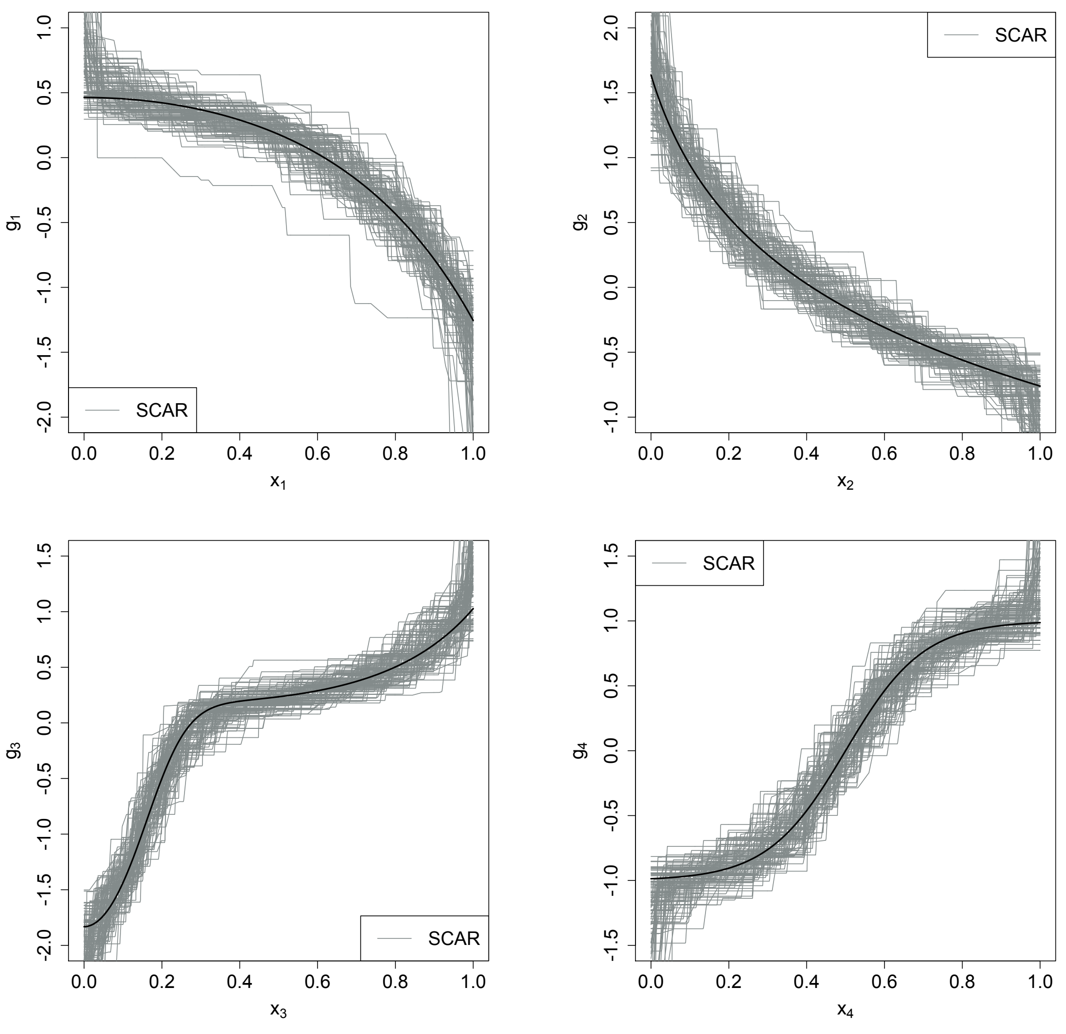


Figure 10: Estimated functions in the simulation considered in section 5.1.1 with $n = 80$, $p = 7$, SNR ≈ 4 and $t = 0$ with the scar method (grey). The true function is given in black.

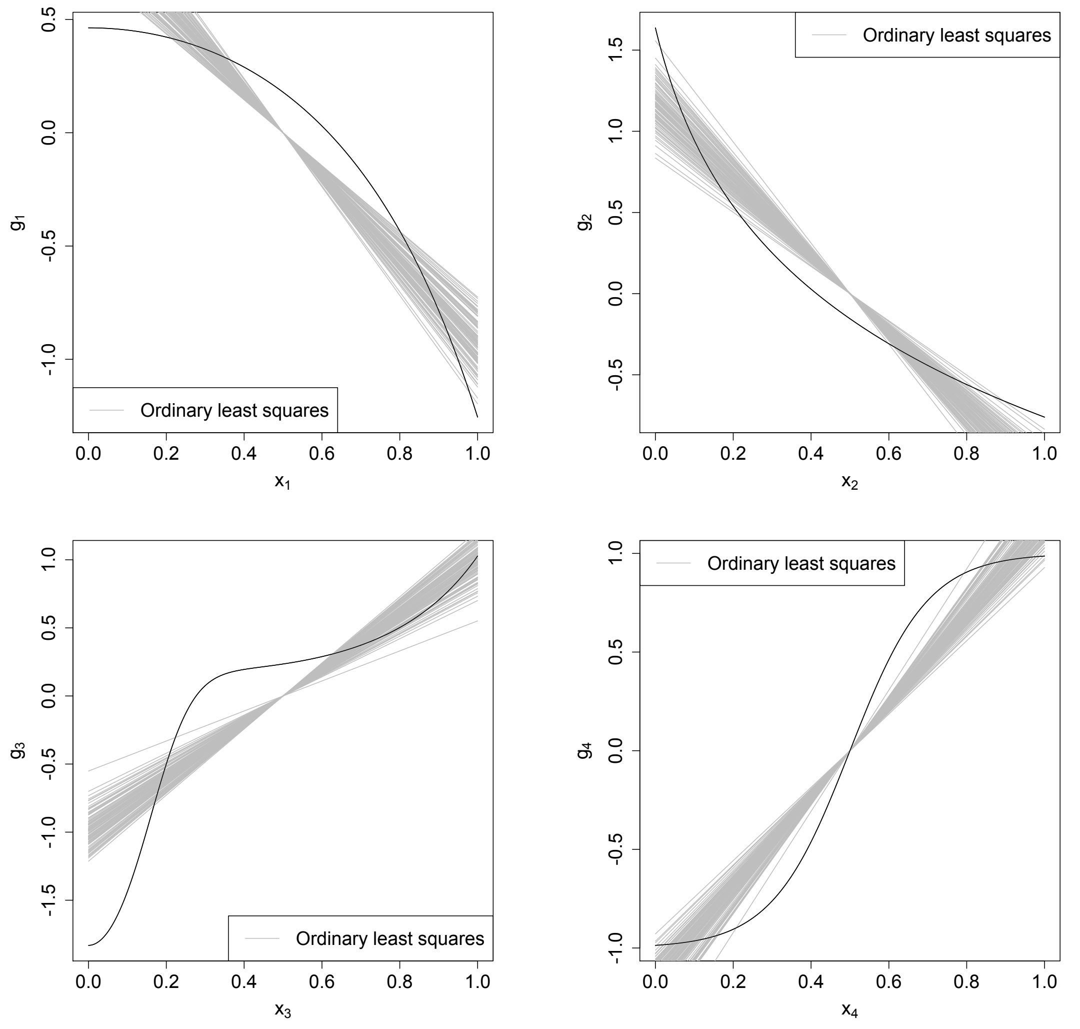


Figure 11: Estimated functions in the simulation considered in section 5.1.1 with $n = 80$, $p = 7$, SNR ≈ 4 and $t = 0$ with the ordinary least squares (grey). The true function is given in black.

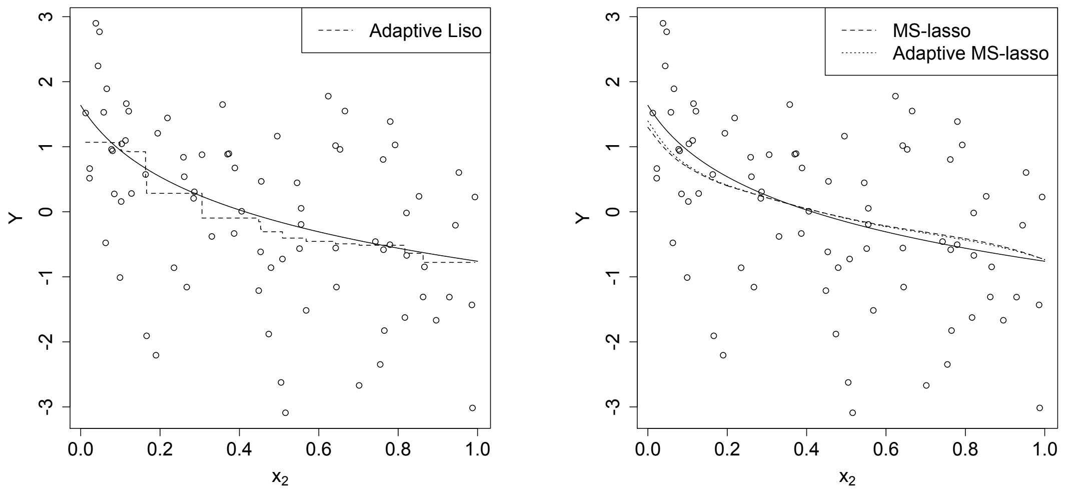


Figure 12: Observations with fitted functions and the true function for adaptive liso (left) and monotone splines lasso (right) for the simulation experiment considered in section 5.1.1. In the left figure, the true function is given in a solid line and the function fitted with adaptive liso is given in a dashed line. In the right figure, the true function is given in a solid line, the estimated function with monotone splines lasso is dashed and the estimated function with adaptive monotone splines lasso is dotted.

5.1.2 Weak signal and dependent covariates

We now wish to see what happens when the covariates are dependent and the signal is weaker. We thus simulate the situation with dependent covariates and a strong signal, so SNR ≈ 4 and $t = 1$. We also simulate the case with a weaker signal and independent covariates, so SNR ≈ 2 and $t = 0$. The ratio of true covariates selected, the ratio of false covariates selected and the mean squared errors for the estimated functions are reported. The results for SNR ≈ 4 and $t = 1$ are given in Table 9. The results for SNR ≈ 2 and $t = 0$ are given in Table 10.

We see from Table 9 that when it comes to variable selection when we have SNR ≈ 4 and dependent covariates ($t = 1$), adaptive liso seems to be the best method. It selects all the true covariates, and has few false covariates. Adaptive monotone splines lasso selects no false covariates, but it has problems selecting x_1. Monotone splines lasso performed better than adaptive monotone splines lasso, in that it was much better at selecting the true covariates, while still selecting few false covariates. However, monotone splines lasso performed slightly worse than adaptive liso in this simulation, both in selecting the true covariates, and false covariates. The linear method and scam select too many false covariates, but this may also be due to the selection criterion we have chosen to use (p-values smaller than 0.05), which does not penalise with the number of observations. If we had instead used a selection criterion penalising with n to select the covariates for the linear method and scam, like BIC, this might have been avoided. When it comes to estimation error, scam outperforms all the other methods, followed by scar. Adaptive liso here too performs better in estimation than both monotone splines lasso and adaptive monotone splines lasso, but note that the mean squared errors are measured in the observation points. Adaptive monotone splines lasso performed slightly better than monotone splines lasso in estimation.

In Table 10, we see that when we have SNR ≈ 2 and $t = 0$, all the methods are good at selecting the true covariates, except the adaptive monotone splines lasso, which has problems selecting x_1. However, adaptive monotone splines lasso outperforms the other methods when it comes to false covariates. Adaptive liso selects the most false covariates. Monotone splines lasso and scam also select a lot more false covariates than the linear method. It is reasonable to assume that in most scenarios, selecting the true covariates is more important than not selecting false covariates, so that the adaptive monotone splines lasso performs worst in this simulation. If avoiding selection of false covariates is the most important, the adaptive monotone splines lasso method is the best in this situation. When considering estimation error (measured in the observation points), we see that the estimation errors are quite similar for adaptive liso and adaptive monotone splines lasso, while the estimation errors for monotone splines lasso and the linear method are larger. Again we find that scam outperforms all the other methods in estimation error. Scar has the largest estimation error among all the methods.

If n is increased to 150 in the case with SNR ≈ 4 and $t = 1$, we get the results in Table 11. All the methods manage to capture the true model well here, except from the linear method, which selects too many false variables. Scam also selects relatively many false covariates. This is due to the fact that the p-values get very small with growing n, as noted earlier. All the methods select all the true covariates, but adaptive monotone splines lasso is the only method which selects no false covariates and is thus the best at variable selection. Adaptive liso also selects almost no false covariates. In estimation error, scam again has the smallest estimation error among all methods, followed by scar and adaptive liso (measured in the observation points). We also see that the difference in estimation error between adaptive monotone splines lasso and monotone splines lasso is very small.

5.2 Prediction performance

To compare the methods, their prediction performances are also studied. This is done by generating 500 new observations, $(\mathbf{X}^{\text{new}}, \mathbf{y}^{\text{new}})$, from the same distribution as the training data, in the setting where SNR ≈ 4 and $t = 0$, and estimating the prediction error, PE, as

$$\text{PE} = \frac{1}{500}\sum_{i=1}^{500}(y_i^{\text{new}} - \hat{y}_i^{\text{new}}),$$

where $\hat{y}_i^{\text{new}}$ are the predicted values of y_i^{new}, using the fitted models. For the linear method and scam, the full fitted model is used for prediction, and not only the significant covariates. We draw 100 such sets of size 500, and estimate the mean prediction error over all the sets. In Table 7, the prediction errors in the setting with $n = 80$ and $p = 7$ are given. The true underlying model is the same as before, so there are three noise covariates. The prediction errors for the different methods are also estimated in the setting where $n = 200$ and $p = 20$. The results for this setting are given in Table 8. We see from Table 7 that when there are not many noise covariates, the scam method is best at prediction. Second best is the adaptive liso, but monotone splines lasso and adaptive monotone splines lasso perform better than scar. The linear method is also better at prediction than scar, but clearly worse than the other monotone methods. From Table 8, we see that when there are more noise covariates, the prediction errors were smallest for the adaptive monotone splines lasso, followed by monotone splines lasso. Adaptive liso and scam had the same prediction errors. Scar and the linear method had larger prediction errors than all the other methods. So the prediction error is better with monotone splines lasso when there are many noise covariates, but when there are few noise covariates, scam performs better. In both settings, adaptive monotone splines lasso had slightly smaller prediction errors than monotone splines lasso.

Prediction error		
MS-lasso	Ad. MS-lasso	Ad. liso
0.26 (0.070)	0.25 (0.070)	0.22 (0.059)
Scam	Scar	Lin. mod
0.20 (0.066)	0.47 (0.86)	0.38 (0.051)

Table 7: Prediction error for the different methods when $n = 80$, $p = 7$, SNR ≈ 4 and $t = 0$ for the simulation considered in section 5.2. Standard deviations are given in parenthesis.

Prediction error		
MS-lasso	Ad. MS-lasso	Ad. liso
0.15 (0.021)	0.14 (0.021)	0.16 (0.033)
Scam	Scar	Lin. mod
0.16 (0.029)	0.22 (0.13)	0.35 (0.029)

Table 8: Prediction error for the different methods when $n = 200$, $p = 20$, SNR ≈ 4 and $t = 0$ for the simulation considered in section 5.2. Standard deviations are given in parenthesis.

Selection

	g_1	g_2	g_3	g_4
MS-lasso	0.99 (0.10)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. MS-lasso	0.88 (0.33)	0.99 (0.10)	0.99 (0.10)	1.0 (0)
Ad. liso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Scam	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Lin. mod	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
MS-lasso	3.99 (0.10)	0.09 (0.29)		
Ad. MS-lasso	3.86 (0.38)	0 (0)		
Ad. liso	4.0 (0)	0.07 (0.26)		
Scam	4.0 (0)	0.23 (0.49)		
Lin. mod	4.0 (0)	0.53 (0.72)		

Estimation

	g_1	g_2	g_3	g_4
MS-lasso	0.020 (0.010)	0.026 (0.016)	0.031 (0.015)	0.028 (0.014)
Ad. MS-lasso	0.022 (0.014)	0.023 (0.018)	0.020 (0.016)	0.032 (0.021)
Ad. liso	0.016 (0.011)	0.020 (0.0088)	0.020 (0.0080)	0.017 (0.0073)
Scam	0.0031 (0.0022)	0.0026 (0.0025)	0.0039 (0.0039)	0.0030 (0.0027)
Scar	0.0095 (0.0063)	0.010 (0.0053)	0.011 (0.0051)	0.011 (0.0054)
Lin. mod	0.018 (0.0085)	0.016 (0.011)	0.059 (0.020)	0.024 (0.0089)

Table 9: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions in the simulation considered in section 5.1.2, where $n = 80$, $p = 7$, SNR ≈ 4 and $t = 1$. Standard deviations are given in parenthesis.

Selection

	g_1	g_2	g_3	g_4
MS-lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. MS-lasso	0.89 (0.31)	0.99 (0.10)	1.0 (0)	1.0 (0)
Ad. liso	0.99 (0.10)	1.0 (0)	1.0 (0)	1.0 (0)
Scam	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Lin. mod	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
MS-lasso	4.0 (0)	0.36 (0.54)		
Ad. MS-lasso	3.88 (0.35)	0.011 (0.10)		
Ad. liso	3.99 (0.10)	0.67 (0.91)		
Scam	4.0 (0)	0.27 (0.51)		
Lin. mod	4.0 (0)	0.15 (0.36)		

Estimation

	g_1	g_2	g_3	g_4
MS-lasso	0.038 (0.029)	0.039 (0.029)	0.058 (0.030)	0.049 (0.030)
Ad. MS-lasso	0.037 (0.028)	0.040 (0.039)	0.044 (0.027)	0.052 (0.038)
Ad. liso	0.036 (0.025)	0.045 (0.024)	0.049 (0.026)	0.043 (0.025)
Scam	0.023 (0.018)	0.017 (0.015)	0.034 (0.023)	0.026 (0.018)
Scar	0.051 (0.028)	0.055 (0.031)	0.063 (0.031)	0.059 (0.034)
Lin. mod	0.042 (0.021)	0.043 (0.020)	0.13 (0.024)	0.044 (0.015)

Table 10: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions in the simulation considered in section 5.1.2, where $n = 80$, $p = 7$, SNR ≈ 2 and $t = 0$. Standard deviations are given in parenthesis.

Selection

	g_1	g_2	g_3	g_4
MS-lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. MS-lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. liso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Scam	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Lin. mod	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
MS-lasso	4.0 (0)	0.08 (0.27)		
Ad. MS-lasso	4.0 (0)	0 (0)		
Ad. liso	4.0 (0)	0.01 (0.10)		
Scam	4.0 (0)	0.21 (0.54)		
Lin. mod	4.0 (0)	0.92 (0.69)		

Estimation

	g_1	g_2	g_3	g_4
MS-lasso	0.012 (0.0060)	0.014 (0.0066)	0.017 (0.0064)	0.017 (0.0065)
Ad. MS-lasso	0.015 (0.0086)	0.010 (0.0081)	0.0077 (0.0044)	0.018 (0.0066)
Ad. liso	0.0054 (0.0023)	0.0061 (0.0023)	0.0061 (0.0023)	0.0056 (0.0016)
Scam	0.0012 (0.00086)	0.0012 (0.0011)	0.0018 (0.0013)	0.0014 (0.00097)
Scar	0.0046 (0.0020)	0.0055 (0.0022)	0.0053 (0.0018)	0.0059 (0.0019)
Lin. mod	0.016 (0.0057)	0.013 (0.0053)	0.060 (0.014)	0.023 (0.0055)

Table 11: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions in the simulation considered in section 5.1.2, where SNR ≈ 4, $t = 1$, $p = 7$ and $n = 150$. Standard deviations are given in parenthesis.

5.3 A situation where a linear model is not sufficient

The linear model can sometimes serve as a rough approximation to a monotone effect, but it is not always so. In some settings, it might be harder for a linear model to detect the monotone effect of a covariate. We consider a simulation experiment with such data, using the monotone splines lasso, the adaptive monotone splines lasso, ordinary least squares, lasso and adaptive lasso for the analysis. This simulation experiment is kept in the classical setting, where $n = 150$ and $p = 20$. The observations are drawn as before, with $t = 1$, so that the covariates are dependent. We let

$$y_i = g_1(x_{1i}) + g_2(x_{2i}) + g_3(x_{3i}) + g_4(x_{4i}) + g_5(x_{5i}) + \epsilon_i,$$

where g_1, g_2, g_3 and g_4 are as in section 4.3, ϵ_i are random noise variables from a normal distribution with mean 0 and variance chosen so that SNR ≈ 4. The last function g_5 is given as

$$g_5(x) = 2\,\mathrm{erf}\,(5x),$$

where erf is the error function.

This function is not well fitted by a linear function, and we thus expect the monotone splines lasso to capture the effect of this function better than the linear methods. As before, we let the ordinary least squares select the variables which are significant at level 0.05. The results are given in Table 12. We see from the table that the monotone splines lasso method in fact was best at selecting x_5. The lasso method selected x_5 more times than the adaptive monotone splines lasso. The ordinary least squares method performed the worst at selecting x_5 (estimating the effect to be significant). Even though lasso performed better than adaptive monotone splines lasso in selecting x_5, we note that lasso and adaptive lasso select a lot more false covariates than all the other methods. Lasso selects almost eight false covariates (on average) and the adaptive lasso selects four. The adaptive monotone splines lasso method selects no false covariates, and the monotone splines lasso selects very few in comparison with the lasso and the adaptive lasso.

The estimated functions of g_5 are given in Figure 13. As noted, the shape of the true function is not well captured by a linear function, and we see that the estimated functions with monotone splines lasso and adaptive monotone splines lasso are a lot closer to the true function. Note also that again we have chosen to draw the estimated functions with monotone splines lasso and adaptive monotone splines lasso in the same plot, and the estimated functions with lasso and adaptive lasso in the same plot. This causes an optical effect of larger variance for these methods, since 200 curves are drawn in these plots, while for the ordinary least squares method, only 100 curves are drawn.

Selection

	g_1	g_2	g_3	g_4
MS-lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. MS-lasso	0.98 (0.15)	0.99 (0.10)	1.0 (0)	1.0 (0)
Lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. Lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Lin. mod	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)

	g_5	TP	FP
MS-lasso	0.96 (0.20)	4.96 (0.20)	0.45 (0.73)
Ad. MS-lasso	0.82 (0.39)	4.79 (0.41)	0 (0)
Lasso	0.86 (0.35)	4.86 (0.35)	7.71 (2.45)
Ad. Lasso	0.72 (0.45)	4.72 (0.45)	4.08 (1.90)
Lin. mod	0.44 (0.50)	4.44 (0.50)	0.90 (1.01)

Estimation

	g_1	g_2	g_3
MS-lasso	0.012 (0.0062)	0.017 (0.0091)	0.015 (0.0069)
Ad. MS-lasso	0.012 (0.0083)	0.014 (0.012)	0.0071 (0.0056)
Lasso	0.013 (0.0035)	0.011 (0.0044)	0.062 (0.015)
Ad. Lasso	0.014 (0.0042)	0.012 (0.0049)	0.059 (0.014)
Lin. mod	0.015 (0.0053)	0.013 (0.0059)	0.059 (0.014)

	g_4	g_5
MS-lasso	0.015 (0.0076)	0.011 (0.0063)
Ad. MS-lasso	0.015 (0.011)	0.011 (0.0084)
Lasso	0.028 (0.0089)	0.036 (0.013)
Ad. Lasso	0.024 (0.0064)	0.036 (0.013)
Lin. mod	0.024 (0.0059)	0.036 (0.012)

Table 12: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions in the simulation considered in section 5.3, where SNR ≈ 4, $t = 1$, $p = 20$ and $n = 150$. Standard deviations are given in parenthesis.

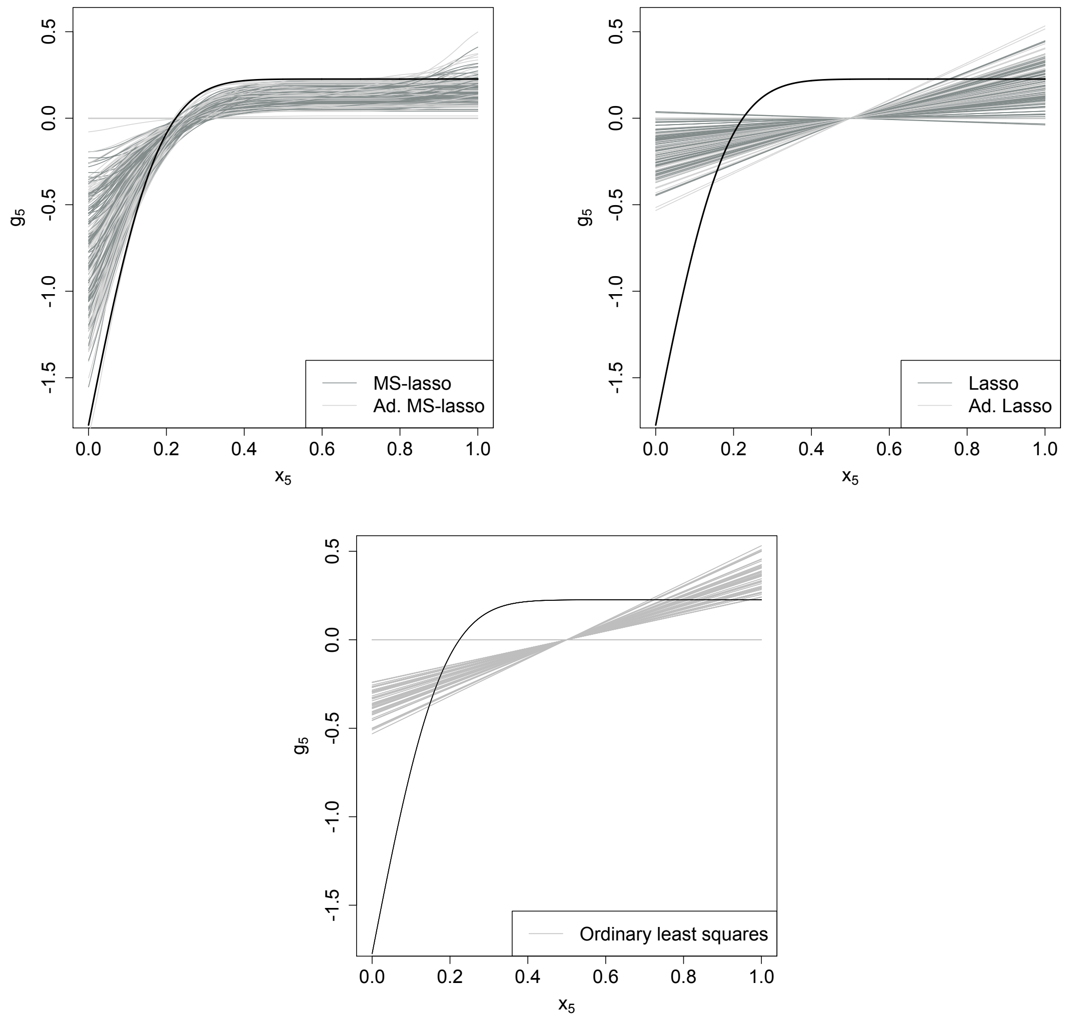


Figure 13: Estimated functions in the simulation considered in section 5.3 with $n = 150$, $p = 20$, SNR ≈ 4 and $t = 1$ with the monotone splines lasso, the adaptive monotone splines lasso, lasso, adaptive lasso and ordinary least squares. The true function is given in black.

5.4 Boston housing data

We try out the different methods for monotone regression in the classical setting using the Boston housing data set. The data consists of a response variable which is the house value, and different explanatory variables. This data set is from Harrison and Rubinfeld (1978) and is available in the R-library *MASS*. It consists of $n = 506$ observations, 13 covariates and the house value (response). We will consider the explanatory variables crime (crime rate by town), zn (proportion of a town's residential land zoned into lots greater than 25 000 square feet), indus (proportion of non-retail business acres per town, serves as a measure of amount of industry), NOX (a pollution variable representing air quality as the concentration of nitrogen oxides), the mean number of rooms, age (the proportion of owner units built prior to 1940), distance to employment centres, rad (index of accessibility to radial highways), tax (the cost of public services), pupil-teacher ratio and the proportion of the population that is considered as lower status. We will not consider the covariate which gives the proportion of the population being black, since this covariate is expected to have a parabolic effect (Harrison and Rubinfeld, 1978). In addition, an indicator variable for whether or not it is a riverside location is given in the data set, but we will only consider numerical covariates. We will thus consider eleven predictors.

The methods that are used are monotone splines lasso, adaptive monotone splines lasso, adaptive liso, scam and scar. The data set was also used with adaptive liso in Fang and Meinshausen (2012).

For the scam and scar, we have to provide the monotonicity directions. In Harrison and Rubinfeld (1978), they propose that crime should have a negative effect on the house value, zn should have a positive effect, indus should have a negative effect, the mean number of rooms should have a positive effect, the distance should have a negative effect, rad should have a positive effect, tax should have a negative effect and the pupil-teacher ratio should have a negative effect. In addition, we assume that NOX and the proportion of the population having lower status will have a negative effect on the house values. For age, we do not know the monotonicity direction, so with scam we do not use any shape constraints on age, while we set a linear shape constraint on age with scar.

For monotone splines lasso and adaptive monotone splines lasso, I-splines of order two with six interior knots are used. The optimal tuning parameters for monotone splines lasso, adaptive monotone splines lasso and adaptive liso are estimated by 10-fold cross-validation. For scam, eight interior knots are used to fit the functions and the smoothing parameters are chosen by the default GCV option.

The estimated effects of the different covariates are given in Figures 14 and 15. All the variables are centered. Crime is selected by all the methods except from scar. Scar does not perform variable selection, but the estimated function is constant, which means that it has no estimated effect. The fit with adaptive liso is not monotone, which is also noted in Fang and Meinshausen (2012).

Only scam estimates a non-constant effect of zn, and it has a small, linear and positive effect. The effect of zn with scam is not significant on a 0.05 significance level.

Indus is only selected by scam, but the effect is not significant. The estimated effect is largest and decreasing for small values of industry, and then flattens out to a constant function.

NOX is selected by all the methods, except from the adaptive monotone splines lasso. The estimated effects are decreasing, but the estimated function with adaptive liso is not monotone.

The mean number of rooms is selected by all the methods and has an increasing effect. The estimated effects are quite similar, with a steeper increase the more rooms.

Age is selected by scam and scar. The estimated effect with scar is small, positive and linear (remember that a linear shape constraint was used), while the estimated effect with scam is small and non-linear. With scam, the effect is not significant.

Distance is selected by all the methods, and it has a decreasing effect on the house value. For the estimated functions with monotone splines lasso, adaptive monotone splines lasso, adaptive liso and scam, the effect is largest for smaller values of distance.

Rad is selected by monotone splines lasso and scam. The estimated effect with scam is quite large and positive. The estimated effect with monotone splines lasso is positive, but small, and only visible for low values of rad. The estimated effect with scam is also largest for smaller values of rad, then flattens out, before increasing again for higher values of rad. Note that the estimated effect with scam seems to be very affected by one extreme observation.

Tax is selected by monotone splines lasso, adaptive liso and scam and has a negative effect. The effect is largest for lower values of tax for monotone splines lasso and adaptive liso, while it is linear with scam.

The pupil-teacher ratio is selected by all the methods except from the adaptive monotone splines lasso, and has a decreasing effect. The estimated effect with monotone splines lasso is quite similar to the effect of scam, while the estimated effects with adaptive liso and scar are somewhat smaller.

The proportion of lower status is selected by all the methods, and the estimated effect is decreasing. The effect is higher for lower values of the proportion of lower status. The estimated effects of proportion of lower status are very similar for all the methods. The scam method would not run with a monotonicity constraint on the proportion of lower status, so this function is fitted without any constraints, and we see that the estimated function is not monotone. The estimated function is decreasing, before hitting a breakpoint, and then it starts to increase. This might be because many people can only afford the cheapest homes, and there are only a limited amount of the cheapest homes.

The estimated models with the different methods are quite similar, but scam and monotone splines lasso select quite many variables. The adaptive monotone splines lasso only selects four variables. The estimated functions with the different methods are also quite similar, but the estimated effects with scam are in general larger. From the fitted functions, it seems like scam is very sensitive to extreme/influential observations. This is especially prominent in the estimated effects of rad and tax.

We know from the previous simulation experiments that the adaptive monotone splines lasso is good at not selecting false covariates. We therefore trust that all covariates selected by the adaptive monotone splines lasso method are important covariates for explaining the house value. We thus believe that crime, the mean number of rooms, distance and proportion of lower status should be kept in the final model. NOX is selected by all the methods except from the adaptive monotone splines lasso. However, the estimated function with adaptive liso is very non-monotone and not very reasonable, and the effect with monotone splines lasso is very small, so we do not believe that this covariate is important for predicting the house value. Tax is selected by all methods except from the adaptive monotone splines lasso and scar, but the estimated effects with monotone splines lasso and adaptive liso are very small, and scam seems to be very affected by one influential observation, so we do not think that tax is important for house value. The pupil-teacher ratio is selected by all the methods except adaptive monotone splines lasso and might have an important effect on the house value. However, from the scale of the estimated effect, it seems to be less important than the other four covariates that we have concluded have an important effect. Since we have a relatively large n compared to the number of covariates, we choose to trust the adaptive monotone splines lasso, and we conclude that the variables important for explaining the house value are crime, mean number of rooms, distance and the proportion of lower status.

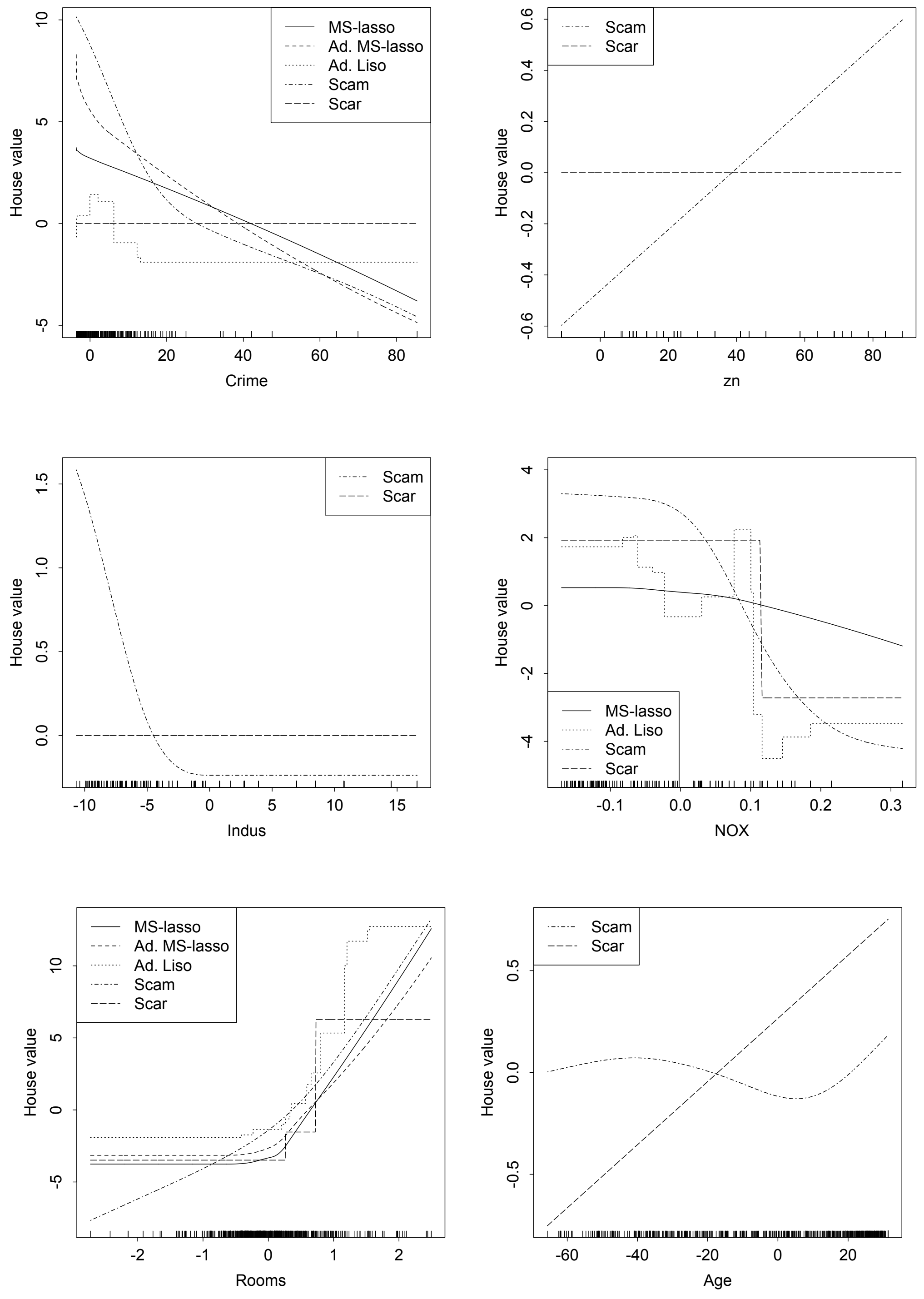


Figure 14: Estimated functions for the Boston housing data, considered in section 5.4. The observed values of the covariates are given at the bottom of the plots.

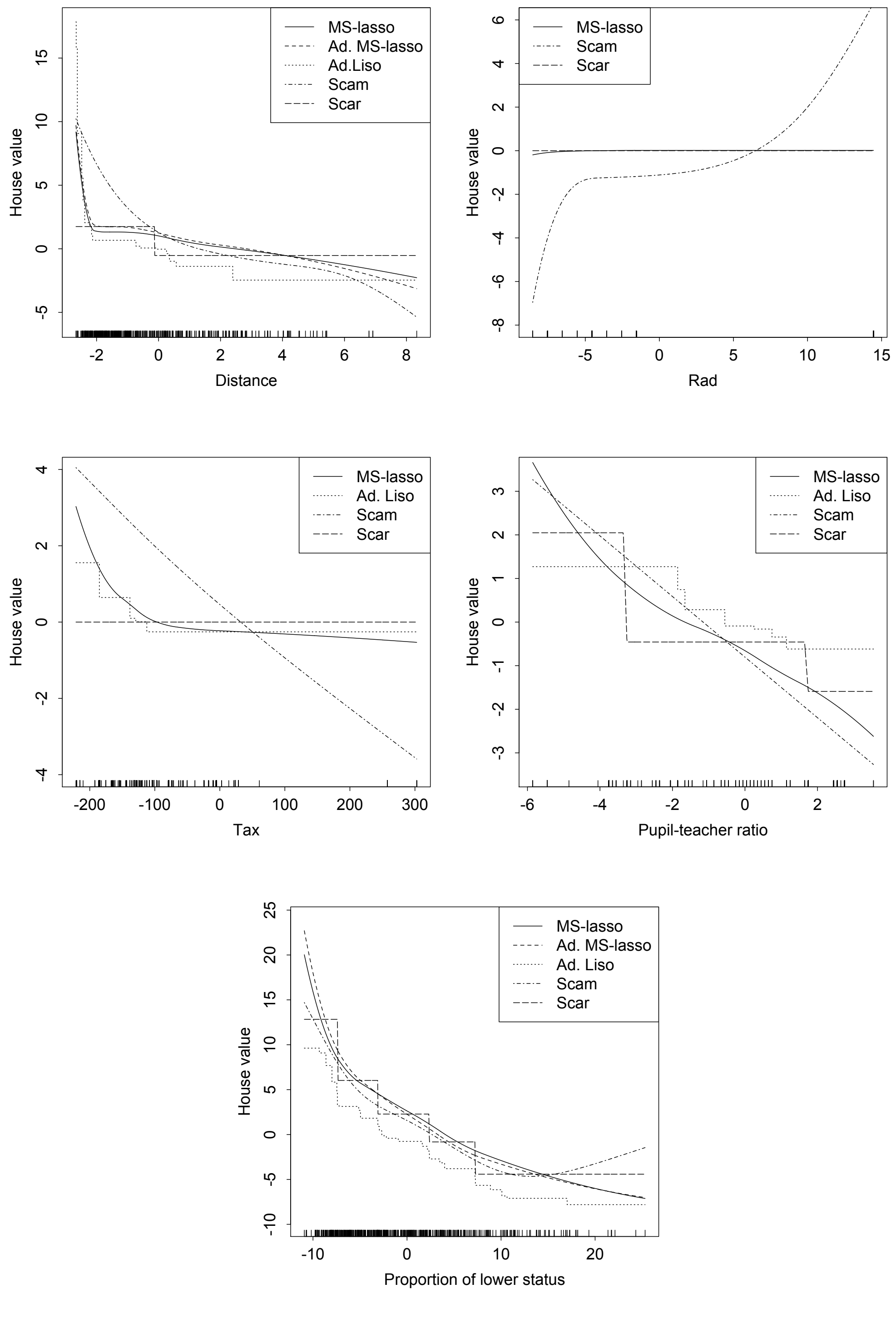


Figure 15: Estimated functions for the Boston housing data, considered in section 5.4. The observed values of the covariates are given at the bottom of the plots.

5.5 Conclusion on the performance of monotone splines lasso

We have seen that in a setting with independent covariates and not too much noise, the adaptive monotone splines method works well in lower dimensions, while monotone splines lasso selects too many false covariates. Even though there is some gain in estimation accuracy in using an adaptive step, the benefit in function estimation when using adaptive monotone splines lasso instead of monotone splines lasso is smaller than it was in the high dimensional setting. The estimation errors for the estimated functions with the monotone splines lasso and the adaptive monotone splines lasso are a lot larger than the estimation errors using the scam method, which is developed for the classical setting.

When the covariates are dependent, adaptive monotone splines lasso may select too few covariates. Here, both adaptive liso and monotone splines lasso perform better than the adaptive monotone splines lasso. The scam method selects more false covariates, but is much better at capturing the true shape of the functions. Note again that the selection criterion for scam is not data driven. When we increase the number of observations, adaptive monotone splines lasso does no longer have the problem of selecting too few true covariates, but it still has estimation errors that are a lot larger than the estimation errors with scam. In both scenarios with dependent covariates, adaptive monotone splines lasso had slightly smaller estimation errors than the monotone splines lasso, and it had the best selection performance among all the methods in the setting with many observations ($n = 150$, $p = 7$).

With more noise, adaptive monotone splines lasso again might select too few variables, while adaptive liso, monotone splines lasso and scam might select too many (noise) variables. The adaptive monotone splines lasso method performed worse than monotone splines lasso in selection, but it had slightly smaller mean squared errors. Scam also here outperforms all the other methods in estimation.

We conclude that if the covariates are independent, adaptive monotone splines lasso perform well in the classical setting. If there are dependent covariates and few observations, then adaptive monotone splines lasso might detect too few covariates, and monotone splines lasso, scam and adaptive liso will perform better in variable selection. With more noise, none of the monotone regression methods performed as well as the linear method in variable selection, but they all performed quite well, with adaptive monotone splines lasso selecting too few variables, and monotone splines lasso, adaptive liso and scam selecting too many.

Comparing the monotone splines lasso to the adaptive monotone splines lasso, we had that the prediction performance was better for the adaptive monotone splines lasso than for the monotone splines lasso. The estimation errors were slightly smaller for the adaptive monotone splines lasso than for the monotone splines lasso, but the difference in estimation is not at all as large as for the high dimensional setting. The adaptive step decreases the number of false covariates, but it may also give too few true covariates.

We have seen that the adaptive monotone splines lasso does not outperform the scam method in the classical setting. The monotone splines lasso method has the advantage of performing automatic variable selection, which scam does not do. It also has the advantage that it does not need to be provided the monotonicity directions of the functions, which scam does. This also makes the comparisons a little unfair for monotone splines lasso, since scam is provided with more information about the functions. In all the simulation experiments, scam selected all the true covariates, while adaptive monotone splines lasso had some problems in selecting x_1. Adaptive monotone splines lasso had fewer false positives than scam, but note that we decided to record all covariates significant at the 0.05 level for the scam method, while it would be better to use an information criterion considering all possible models, and then selecting the best in terms of the information criterion used. Monotone splines lasso has the disadvantage that it does not guarantee a monotonic fit. The estimated functions with scam will always be monotone. We also noted that scam outperformed the adaptive monotone splines lasso in estimation error in all the simulation experiments. In the Boston housing data example, we saw that scam is sensitive to influential observations. In the simulation experiments, we found that the scar method in most settings (all except from the one where we had more noise) had smaller estimation errors than the adaptive monotone splines lasso, but the difference was not as large as with the scam method. In addition, note that scar was also provided the monotonicity direction of the functions before the analysis. When considering prediction error, scam performed better than both monotone splines lasso and adaptive monotone splines lasso when we did not have many noise covariates, but when we increased the number of noise covariates, monotone splines lasso and adaptive monotone splines lasso performed better than scam. Monotone splines lasso and adaptive monotone splines lasso had smaller prediction errors than scar in both settings.

6 Partially linear monotone models

In additive partially linear models, some covariates are assumed to have a linear effect on the response, and some are assumed to have a non-linear effect on the response. In the classical setting, we have seen that the methods scam and scar (described in sections 3.2.2 and 3.2.3) can be used to fit an additive partially linear model, where there are different shape constraints on the non-linear functions. There are also some existing methods which can be used for fitting additive partially linear models in the high dimensional setting.

Liu et al. (2011) developed a method for fitting additive partially linear models with variable selection for the linear covariates. In a paper by Du et al. (2012), a method for fitting the additive partially linear model with variable selection in both the linear and the non-linear covariates was developed. Wei (2012) develops a method for additive partially linear models with grouped linear covariates performing variable selection in the linear covariates. These methods all assume that it is a priori known which covariates will (potentially) have a linear effect, and which covariates will (potentially) have a non-linear effect. In this section, we will first look closer at the additive partially linear models and methods for fitting additive partially linear models, which can be used in the high dimensional data setting. Then we will develop two new methods for fitting additive partially linear models where the non-linear effects are assumed/restricted to be monotone, using I-splines.

Let $\mathbf{y}$ be the observations. Let $\mathbf{X}$ denote the design matrix for the covariates assumed to potentially have a linear effect on the response, and $\mathbf{Z}$ the design matrix for the covariates assumed to potentially have a non-linear effect on the response. Let d_1 be the number of covariates with linear effect, and d_2 the number of covariates with non-linear effect. Then $d_1 + d_2 = p$, where p is the total number of covariates. $\mathbf{X}$ is an $n \times d_1$ matrix and $\mathbf{Z}$ is an $n \times d_2$ matrix. Then the additive partially linear model is

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \sum_{k=1}^{d_2} g_k(\mathbf{Z}^{(k)}) + \boldsymbol{\epsilon},$$

where $\mathbf{Z}^{(k)}$ is the kth column of $\mathbf{Z}$ and $g_1, g_2, \ldots, g_{d_2}$ are unknown smooth functions. Again it is assumed that $\mathrm{E}[g_k(x)] = 0$ for all k, for unique identification.

To fit the model, let the covariates in $\mathbf{Z}$ be transformed to the interval $[0, 1]$, and let the g_ks be approximated by spline functions. In Liu et al. (2011), the g_ks are fit by functions from $\mathscr{S}_n$, where $\mathscr{S}_n$ is the space of polynomial splines on $[0, 1]$ of order l. Let $\mathscr{G}_n$ be the space of functions $h(\mathbf{z}) = h_1(z_1) + h_2(z_2) + \ldots + h_{d_2}(z_{d_2})$ with all $h_k \in \mathscr{S}_n$ and $\sum_{i=1}^{n} h_k(z_{ik}) = 0$. Then, if we are in the classical setting, we want a $\tilde{g} \in \mathscr{G}_n$ and $\hat{\boldsymbol{\beta}}$ so that

$$\tilde{g}, \hat{\boldsymbol{\beta}} = \mathrm{argmin}_{g,\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - g(\mathbf{Z})||_2^2,$$

where

$$g(\mathbf{Z}) = \begin{pmatrix} g_1(z_{11}) + g_2(z_{12}) + \ldots + g_{d_2}(z_{1d_2}) \\ g_1(z_{21}) + g_2(z_{22}) + \ldots + g_{d_2}(z_{2d_2}) \\ \vdots \\ g_1(z_{n1}) + g_2(z_{n2}) + \ldots + g_{d_2}(z_{nd_2}) \end{pmatrix},$$

and z_{ik} is the ith observation of covariate k. Consider the situation with K interior knots, and let $b_{j,k}(z)$ be the basis functions of degree l for covariate k. Then all $h \in \mathscr{G}_n$ can be written as

$$h(\mathbf{z}) = \boldsymbol{\gamma}'\mathbf{b}(\mathbf{z}),$$

where

$$\begin{aligned}\mathbf{b}(\mathbf{z}) = (&b_{11}(z_1), b_{21}(z_1), \ldots, b_{K+l,1}(z_1), b_{12}(z_2), b_{22}(z_2), \ldots, \\ &b_{K+l,2}(z_2), \ldots, b_{1,d_2}(z_{d_2}), b_{2,d_2}(z_{d_2}), \ldots, b_{K+l,d_2}(z_{d_2}))',\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{\gamma} = (&\gamma_{11}, \gamma_{21}, \ldots, \gamma_{K+l,1}, \gamma_{12}, \gamma_{22}, \ldots, \\ &\gamma_{K+l,2}, \ldots, \gamma_{1,d_2}, \gamma_{2,d_2}, \ldots, \gamma_{K+l,d_2})'.\end{aligned}$$

So an equivalent problem is to find $\hat{\boldsymbol{\gamma}}$ and $\hat{\boldsymbol{\beta}}$ so that

$$\hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\gamma},\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \boldsymbol{\gamma}'\mathbf{b}(\mathbf{Z})||_2^2,$$

where

$$\begin{aligned}\mathbf{b}(\mathbf{Z}) = (&b_{11}(\mathbf{Z}^{(1)}), \ldots, b_{K+l,1}(\mathbf{Z}^{(1)}), b_{12}(\mathbf{Z}^{(2)}), \ldots, \\ &b_{K+l,2}(\mathbf{Z}^{(2)}), \ldots, b_{1,d_2}(\mathbf{Z}^{(d_2)}), \ldots, b_{K+l,d_2}(\mathbf{Z}^{(d_2)}))'.\end{aligned}$$

This is a linear regression problem, which can be solved by ordinary linear regression if the number of parameters is small enough. If not, subset selection and/or regularisation is needed. Note that when counting the number of parameters, we have to take into account the number of basis functions used to represent the non-linear covariates.

In Liu et al. (2011) they use a general penalisation term on the linear component, so the estimates are given by

$$\hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\gamma},\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \boldsymbol{\gamma}'\mathbf{b}(\mathbf{Z})||_2^2 + \sum_{j=1}^{d_1} p_{\lambda_j}(|\beta_j|),$$

with the lasso penalisation as a special case.

In Du et al. (2012), a method for simultaneous variable selection in the linear and the non-linear part is proposed. With an initial guess for $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}_0$, a first guess for $\mathbf{g} = (g_1, g_2, \ldots, g_{d_2})$ is found by

$$\hat{\mathbf{g}} = \text{argmin}_g ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - g(\mathbf{Z})||_2^2 + \lambda \sum_{k=1}^{d_2} w_j ||g_k||_q^q, \tag{6.1}$$

where $\hat{\boldsymbol{\beta}}_0$ is inserted for $\boldsymbol{\beta}$, $||g_k||_q^q = \sum_{i=1}^{n} |g_k(z_{ik})|^q$ and w_j are optional weights. With an initial guess for $\mathbf{g}$, $\hat{\mathbf{g}}_0$, a first guess for $\boldsymbol{\beta}$ is found by

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - g(\mathbf{Z})||_2^2 + \sum_{j=1}^{d_1} p_{\lambda_j}\left(|\beta_j|\right), \tag{6.2}$$

where $\hat{\mathbf{g}}_0$ is inserted for $\mathbf{g}$.

So the estimates are obtained as follows: start with an initial guess for $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}_0$. Then the first guess for g, $\hat{\mathbf{g}}_0$, is found by plugging $\hat{\boldsymbol{\beta}}_0$ into equation (6.1). The next guess for $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}_1$, is found by plugging $\hat{\mathbf{g}}_0$ into equation (6.2). This new estimate of $\boldsymbol{\beta}$ is then used to obtain a new estimate of $\mathbf{g}$. The updating is continued until convergence is reached. With appropriate penalisation terms, the method will then perform variable selection both in the linear and the non-linear parameters.

In Wei (2012), a method for variable selection and estimation in an additive partially linear model with grouped linear covariates is considered, with variable selection in the grouped linear covariates. The method developed uses splines to approximate the functions for the non-parametric components. As before, there are d_1 linear covariates, and these are divided into k groups, $\mathscr{G}_1, \mathscr{G}_2, \ldots, \mathscr{G}_k$. Let $\boldsymbol{\beta}_{\mathscr{G}_j}$ denote the vector of linear parameters for group j.

Let each of the non-parametric components be represented by m spline basis functions $s_l(x)$ for $l = 1, \ldots, m$. Then $g_j(x) = \sum_{l=1}^{m} \gamma_{jl} s_l(x)$ for $j = 1, \ldots, d_2$, where d_2 is the number of covariates with non-linear effect on the response. The spline basis functions are centered for unique identification of the functions. Let $\boldsymbol{\gamma}$ be the vector of basis coefficients for the non-linear components. Let $\mathbf{Z}'$ be the design matrix for the covariates with non-linear effects on the response, represented in the spline basis. The estimated parameters are then given by

$$\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}} = \text{argmin}_{\boldsymbol{\beta},\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2 + \lambda \sum_{j=1}^{k} m_j ||\boldsymbol{\beta}_{\mathscr{G}_j}||_2,$$

where m_j are weights, typically adjusting for group size, but can also be weights for an adaptive scheme.

Although this was not done in Wei (2012), we see that this method can easily be extended to include variable selection in the non-linear components as well, by including the d_2 groups of basis coefficients in the penalty term. Let $\boldsymbol{\gamma}_j = (\gamma_{j1}, \gamma_{j2}, \ldots, \gamma_{jm})$ for $j = 1, \ldots, d_2$. Then consider

$$\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}} = \text{argmin}_{\boldsymbol{\beta},\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2 + \lambda(\sum_{j=1}^{k} m_j ||\boldsymbol{\beta}_{\mathscr{G}_j}||_2 + \sum_{j=1}^{d_2} w_j ||\boldsymbol{\gamma}_j||_2),$$

where w_j are weights for the groups of basis coefficients, typically adjusting for the number of basis coefficients. This will give variable selection in both the linear covariates and the non-linear covariates.

6.1 Fitting the partially linear model using monotone splines

In some situations, it might be reasonable to assume that the non-linear functions in an additive partially linear model are monotone. The ideas of monotone splines lasso can be used to develop methods for fitting a partially linear model in which some covariates are assumed to have a linear effect on the response, and some are assumed to have a general monotone effect on the response. The method should perform variable selection simultaneously in both the linear and the non-linear variables. We assume that we know a priori which covariates have a linear effect on the response, and which covariates have a monotone non-linear effect.

As before, let d_1 be the number of covariates with a linear effect and let d_2 be the number of covariates with a non-linear monotone effect. Let $\mathbf{X}$ be the design matrix for the covariates with linear effect, so $\mathbf{X}$ is an $n \times d_1$ matrix. Let $\mathbf{Z}$ be the design matrix for the covariates with non-linear monotone effect, so $\mathbf{Z}$ is an $n \times d_2$ matrix. All the covariates are transformed to the interval $[0, 1]$ to avoid scaling dependency in the penalty terms that are used for variable selection. Let $\mathbf{y}$ be the observed responses. The model is then

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \sum_{j=1}^{d_2} g_j(\mathbf{Z}^{(j)}) + \boldsymbol{\epsilon},$$

where $\mathbf{Z}^{(j)}$ is the jth column of $\mathbf{Z}$ and the g-functions are unknown smooth monotone functions. $\boldsymbol{\epsilon}$ is a vector of random noise. For identifiability it is assumed that $\text{E}[g_j(x)] = 0$ for all j. I-splines of order l with K interior knots are used to represent the monotone functions. The number of basis splines needed to fit each function is then $m = K + l$. So g_j is approximated by

$$\tilde{g}_j(x) = \sum_{k=1}^{m} \gamma_{jk} I_k^{(l)}(x),$$

where $I_k^{(l)}$ are the I-spline basis functions of order l as given in section 4.2 for $l = 2$, γ_{jk} are the coefficients for covariate j in the spline basis and $\tilde{g}_j$ is a spline approximation of g_j. The I-spline basis functions are centered to ensure that $\mathrm{E}[\tilde{g}_j(x)] = 0$ for unique identification of the functions.

Let $\mathbf{Z}' = (\mathbf{Z}'_1, \ldots, \mathbf{Z}'_{d_2})$ be the design matrix for the covariates with non-linear monotone effect on the response, represented in the I-spline basis. $\mathbf{Z}'$ is then an $n \times d_2 m$ matrix and $\mathbf{Z}'_j$ is the $n \times m$ design matrix for covariate j. There is then a grouping of the variables, where the first d_1 groups are singletons, and the last d_2 groups are groups of group size m, corresponding to the m basis functions for each covariate.

6.2 PLAMM-1

The first method for fitting our additive partially linear model that we propose is a natural extension of the monotone splines lasso method, where the cooperative penalty is used on all the parameters in the model. This was also mentioned (but not carried out) in Bergersen et al. (2014). Let now $\mathbf{X}' = (\mathbf{X}, \mathbf{Z}')$ be the design matrix with the linear covariates as the first d_1 columns, and the monotone non-linear covariates represented in the I-spline basis in the last $d_2 \cdot m$ columns. Then $\mathbf{X}'$ is an $n \times (d_1 + d_2 \cdot m)$ matrix. Let $\boldsymbol{\phi} = (\boldsymbol{\beta}, \boldsymbol{\gamma})$, where $\boldsymbol{\beta}$ is the vector with linear parameters, and $\boldsymbol{\gamma}$ is the vector of basis coefficients. Then $\boldsymbol{\phi}$ is a vector with $d_1 + d_2 \cdot m$ entries. Let $\mathscr{G}_j$, $j = 1, \ldots, d_1, d_1 + 1, \ldots, d_1 + d_2$, denote the groups of the covariates. Then $\boldsymbol{\phi}_{\mathscr{G}_j} = \beta_j$ for $j = 1, \ldots, d_1$ and $\boldsymbol{\phi}_{\mathscr{G}_j} = (\gamma_{j1}, \ldots, \gamma_{jm})$ for $j = d_1 + 1, \ldots, d_1 + d_2$. Consider the problem

$$\hat{\boldsymbol{\phi}} = \mathrm{argmin}_{\boldsymbol{\phi}} ||\mathbf{y} - \mathbf{X}'\boldsymbol{\phi}||_2^2 + \lambda ||\boldsymbol{\phi}||_{\mathrm{coop}}. \tag{6.3}$$

For the covariates with linear effects there is only one parameter, so the corresponding group consists of a singleton. The penalty term is

$$||\boldsymbol{\phi}||_{\mathrm{coop}} = \sum_{j=1}^{d_1+d_2} m_j ||\boldsymbol{\phi}^+_{\mathscr{G}_j}||_2 + m_j ||\boldsymbol{\phi}^-_{\mathscr{G}_j}||_2,$$

where, as before, $\boldsymbol{\phi}^+_{\mathscr{G}_j} = \max(\boldsymbol{\phi}_{\mathscr{G}_j}, 0)$ and $\boldsymbol{\phi}^-_{\mathscr{G}_j} = \max(-\boldsymbol{\phi}_{\mathscr{G}_j}, 0)$. Weights on the penalty terms are now used, since the group sizes are not equal. The standard group lasso weights are used so that for the linear terms, $m_j = 1$, and for the non-linear terms, $m_j = \sqrt{m}$, where m is the number of I-spline basis functions for each covariate. Note that for the linear terms

$$m_j ||\boldsymbol{\phi}^+_{\mathscr{G}_j}||_2 + m_j ||\boldsymbol{\phi}^-_{\mathscr{G}_j}||_2 = \sqrt{|\beta_j|^2} = |\beta_j|,$$

which is the lasso penalty. So we have an L_1 penalty for the linear parameters and a cooperative lasso penalty (see section 2.4.3) for the $\boldsymbol{\gamma}$ parameters, with a common penalty parameter λ. Solving equation (6.3) will then give us an additive partially linear method with monotone splines lasso performing variable selection simultaneously in the linear part and the non-linear part. We call this method

PLAMM-1, which stands for partially linear additive monotone method 1. As before, λ is a tuning parameter controlling the regularisation, and can be chosen by for instance cross-validation.

6.2.1 Properties

To study the properties of PLAMM-1, it is first assumed that each g-function can be represented exactly as a monotone function in the basis with m I-spline basis functions, so that

$$g_j = \sum_{k=1}^{m} \gamma_{jk} I_k^{(l)} \text{ for } j = 1, \ldots, d_2.$$

Theorem 2 in Chiquet et al. (2012) says that under appropriate conditions, the estimators for the coefficients will be asymptotically unbiased and have the property of exact support recovery. That is,

$$\hat{\boldsymbol{\phi}} \xrightarrow{P} \boldsymbol{\phi} \text{ and } \mathrm{P}(\mathscr{S}(\hat{\boldsymbol{\phi}}) = \mathscr{S}) \to 1,$$

where $\mathscr{S}(\hat{\boldsymbol{\phi}})$ is the estimated support, so it is the set of covariates selected by the method, and $\mathscr{S}$ is the true support, so it is the set of the true covariates. This means that under appropriate assumptions, the estimated parameters are consistent, and the method will, with probability converging to one, select the true model. Since the parameter estimates are consistent under these assumptions, the linear parameters can be estimated with arbitrary small error, by increasing n. Since we have assumed that the non-linear functions can be represented exactly in the I-spline basis with m spline basis functions, these functions can also be estimated with arbitrary small error by increasing n. The assumptions for which these properties hold are given in (A1)-(A5) in Chiquet et al. (2012). We have to assume that $\mathbf{X}'$ and $\mathbf{Y}$ have finite fourth order moments and that the covariance matrix of $\mathbf{X}'$, $E[\mathbf{X}'\mathbf{X}'^T]$, is invertible. It is also assumed that in the sign-incoherent groups, all the coefficients are non-zero. In addition, two variants of the irrepresentable condition for the cooperative lasso are needed, which can be found in (A4)-(A5) in Chiquet et al. (2012). The irrepresentable conditions for lasso are conditions guaranteeing exact support recovery for the lasso (Zhao and Yu, 2006).

6.2.2 Adaptive scheme for PLAMM-1

The method developed can be extended with an adaptive step. Let $\hat{\boldsymbol{\phi}}^{\text{init}}$ be the initial fit obtained from solving equation (6.3). Then let

$$w_j = \begin{cases} \infty, & \text{if } ||\hat{\boldsymbol{\phi}}^{\text{init}}_{\mathscr{G}_j}||_2 = 0, \\ 1/||\hat{\boldsymbol{\phi}}^{\text{init}}_{\mathscr{G}_j}||_2, & \text{otherwise.} \end{cases}$$

The adaptive solution is then

$$\hat{\boldsymbol{\phi}} = \text{argmin}_{\boldsymbol{\phi}} ||\mathbf{y} - \mathbf{X}'\boldsymbol{\phi}||_2^2 + \lambda \sum_{j=1}^{d_1+d_2} m_j w_j ||\boldsymbol{\phi}^+_{\mathscr{G}_j}||_2 + m_j w_j ||\boldsymbol{\phi}^-_{\mathscr{G}_j}||_2,$$

where, as before, m_j are weights typically balancing for group size. Standard values are $m_j = \sqrt{m}$ for the non-linear terms and $m_j = 1$ for the linear terms. This adaptive method is called APLAMM-1.

6.2.3 Simulation experiment

In order to study the performances of PLAMM-1 and APLAMM-1, we perform simulation experiments similar to the previous simulation experiments, but in a partially linear setting. There are n observations, d_1 covariates with a (potential) linear effect on the response and d_2 covariates with a (potential) non-linear effect on the response. Random variables x_{ij} and z_{ik} are drawn from a normal distribution with mean 0.5 and standard deviation 1, truncated to $[0, 1]$, where $j = 1, \dots, d_1$ and $k = 1, \dots, d_2$. We then let

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + g_1(z_{i1}) + g_2(z_{i2}) + g_3(z_{i3}) + g_4(z_{i4}) + \epsilon_i,$$

where $\epsilon_i \sim \text{N}(0, \sigma^2)$, and σ is chosen to control the signal to noise ratio (SNR). $(\beta_1, \beta_2, \beta_3, \beta_4) = (2, 2, -2, -2)$ and the g-functions are as in section 4.3. The g-functions are centered, since this assumption is made about the functions for unique identification. I-splines of order two are used to estimate the g-functions, with six interior knots evenly distributed at the quantiles of the observed data. The penalty parameter λ is chosen by 10-fold cross-validation.

A simple setting We start with a simple setting, where $d_1 = 10$, $d_2 = 10$, $n = 50$ and SNR ≈ 4. There are thus six linear noise covariates and six non-linear noise covariates. We simulate 100 times and new random variables are drawn for each simulation. If one of the simulations results in an estimated function which is not monotone, then the results for this one simulation are discarded. The results for PLAMM-1 and APLAMM-1 with the standard $\sqrt{m}$ group weights on the penalty for the non-linear groups and weights 1 on the penalty for the linear parameters, are given in Table 13. We see from Table 13 that the true model is captured well in this setting. We observe the clear benefit in the adaptive step in that the number of false covariates decreases. The estimation errors for the functions are quite similar for PLAMM-1 and APLAMM-1, but the estimated linear parameters are closer to their true values with the adaptive scheme. Note that for g_2, β_2, β_3 and β_4, it seems like APLAMM-1 has selected the corresponding covariates in more simulations than PLAMM-1, which is impossible. This is due to the fact that the results for some simulations for APLAMM-1 are discarded, since they did not provide a monotone fit.

From Table 13 we see that the methods select more of the non-linear covariates than the linear covariates, both true and false. We also see that the estimates for the linear parameters are way too small (in absolute value). Since there is a penalty on the size of the linear parameters, we do expect them to be shrunken, but these estimated parameters are very small. When we performed the simulation, 12% of the simulations resulted in a non-monotone fit for one or more of the non-linear covariates with PLAMM-1, while for the adaptive method, 24% of the simulations resulted in a non-monotone fit for one or more of the non-linear covariates. This indicates that the penalty for the non-linear terms is too small in comparison to the penalty for the linear terms. We therefore investigate whether a different penalty scheme will give us a more fair balance between the linear and the non-linear covariates. This is done by adjusting the weights in the penalty term. Remember that the penalty term is given by

$$||\boldsymbol{\phi}||_{\text{coop}} = \sum_{j=1}^{d_1+d_2} m_j ||\boldsymbol{\phi}^{+}_{\mathscr{G}_j}||_2 + m_j ||\boldsymbol{\phi}^{-}_{\mathscr{G}_j}||_2.$$

We let the weights m_j for the penalty on the linear covariates be 1. We simulate with the weights for the non-linear penalty terms being $m^{0.6}, m^{0.7}, m^{0.8}$ and $m^{0.9}$, keeping the linear weights at 1. Note that $m^{0.5}$ was what we used above. The results for PLAMM-1 and APLAMM-1 with group weights $m^{0.6}, m^{0.7}, m^{0.8}$ and $m^{0.9}$ are given in Appendix B, in Tables B1, B2, B3 and B4, respectively. From Table B1, we see that with weights $m^{0.6}$ on the non-linear penalties, the methods select more linear covariates in comparison to the results with $\sqrt{m}$ given in Table 13, but we still see that the methods seem to favour the non-linear covariates since they select more false non-linear covariates than false linear covariates. The estimated linear parameters are closer to the truth, but they are still too small. From Table B2 where we have weights $m^{0.7}$, the favouring of the non-linear covariates is no longer that clear, even though we have some more false non-linear covariates than false linear covariates. We see that less of the true non-linear covariates are selected than the true linear covariates. The estimated linear parameters are larger than they were with weights $m^{0.6}$. With penalty weight $m^{0.7}$, 6% of the simulations with PLAMM-1 resulted in one or more non-monotone fits, while 16% of the simulations with APLAMM-1 resulted in a non-monotone fit. From Table B3, we see that when we have weights $m^{0.8}$ on the non-linear penalty terms, the methods select a little more false linear covariates than false non-linear covariates, and more true linear covariates than true non-linear covariates. The estimates of the linear parameters are even better than they were for $m^{0.7}$. The number of simulations resulting in a non-monotone fit was 5% for PLAMM-1 and 14% for APLAMM-1. From Table B4 we see that with weights $m^{0.9}$ for the non-linear penalty terms, the linear covariates are favoured even more than they were with $m^{0.8}$. We therefore conclude that weights $m^{0.7}$ or $m^{0.8}$ give the best balance among the linear and non-linear covariates in this simulation experiment. Since we want to avoid a non-monotone fit, we choose to use $m^{0.8}$ in all the following simulation experiments.

Looking closer at the results from the simulation with weights $m^{0.8}$ in Table B3, we see that both methods perform well. With APLAMM-1, we get a large reduction in the false positives, and the estimated linear parameters are closer to their true values than for PLAMM-1. The mean squared errors for the estimated functions are slightly larger for APLAMM-1, and APLAMM-1 has more problems in selecting g_1. However, due to the substantial improvement in the false positives, we conclude that APLAMM-1 performed better in this setting than PLAMM-1, so we conclude that there is a benefit in using an adaptive scheme.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	0.99 (0.11)	0.99 (0.11)	1.0 (0)	1.0 (0)
APLAMM-1	0.99 (0.11)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	3.97 (0.18)	3.54 (1.49)		
APLAMM-1	3.99 (0.11)	0.39 (0.87)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.020 (0.019)	0.025 (0.022)	0.035 (0.021)	0.033 (0.029)
APLAMM-1	0.024 (0.025)	0.028 (0.026)	0.032 (0.020)	0.033 (0.028)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	0.99 (0.11)	0.99 (0.11)	0.98 (0.15)	0.98 (0.15)
APLAMM-1	0.99 (0.11)	1.0 (0)	0.99 (0.11)	1.0 (0)
	TP	FP		
PLAMM-1	3.93 (0.45)	1.24 (1.11)		
APLAMM-1	3.97 (0.23)	0.092 (0.29)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.51 (0.30)	1.55 (0.35)	-1.54 (0.34)	-1.56 (0.32)
APLAMM-1	1.67 (0.32)	1.68 (0.37)	-1.65 (0.36)	-1.68 (0.35)

Table 13: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.2.3, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 50$. Standard deviations are given in parenthesis.

6.3 PLAMM-2

Since we saw that the PLAMM-1 method developed in section 6.2 had trouble with the balance between the penalties for the non-linear covariates and the linear covariates, we develop a method for fitting the additive partially linear model with monotone splines lasso using two penalty terms with different penalty parameters, so we have one penalisation term for the linear coefficients and one penalisation term for the non-linear monotone basis coefficients. We call this method PLAMM-2 (partially linear additive monotone method 2), both because it is our second method (method 2), and because it has two penalty terms instead of one. As before, let $\boldsymbol{\beta}$ be the linear parameters corresponding to $\mathbf{X}$ and let $\boldsymbol{\gamma}$ be the vector of basis coefficients corresponding to $\mathbf{Z}'$. To perform variable selection in both the linear and the monotone non-linear terms, we use an L_1 penalty for the linear parameters and a cooperative lasso penalty (see section 2.4.3) for the $\boldsymbol{\gamma}$ parameters. If the linear variables are grouped, a group lasso penalty term on the linear parameters can be used instead. We want to solve

$$\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}} = \text{argmin}_{\boldsymbol{\beta},\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2 + \lambda_1 ||\boldsymbol{\beta}||_1 + \lambda_2 ||\boldsymbol{\gamma}||_{\text{coop}}.$$

This is done in a similar way as in Du et al. (2012), as explained in the beginning of this section. Start with an initial guess for $\boldsymbol{\beta}$, and denote it by $\hat{\boldsymbol{\beta}}^{(0)}$. A first estimate for $\boldsymbol{\gamma}$, $\hat{\boldsymbol{\gamma}}^{(1)}$, is then found by

$$\hat{\boldsymbol{\gamma}}^{(1)} = \text{argmin}_{\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^{(0)} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2 + \lambda_2 ||\boldsymbol{\gamma}||_{\text{coop}}.$$

This estimate for $\boldsymbol{\gamma}$ is then used to find a first estimate for $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}^{(1)}$, by

$$\hat{\boldsymbol{\beta}}^{(1)} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\hat{\boldsymbol{\gamma}}^{(1)}||_2^2 + \lambda_1 ||\boldsymbol{\beta}||_1.$$

We continue updating using $\hat{\boldsymbol{\beta}}^{(k)}$ to find $\hat{\boldsymbol{\gamma}}^{(k+1)}$ by

$$\hat{\boldsymbol{\gamma}}^{(k+1)} = \text{argmin}_{\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^{(k)} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2 + \lambda_2 ||\boldsymbol{\gamma}||_{\text{coop}}.$$

Then $\hat{\boldsymbol{\gamma}}^{(k+1)}$ is used to find $\hat{\boldsymbol{\beta}}^{(k+1)}$ by

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\hat{\boldsymbol{\gamma}}^{(k+1)}||_2^2 + \lambda_1 ||\boldsymbol{\beta}||_1.$$

This is repeated until convergence is reached in both $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\gamma}}$, in terms of change in squared distance between the current and the previous estimate. So we continue until $||\hat{\boldsymbol{\beta}}^{(k+1)} - \hat{\boldsymbol{\beta}}^{(k)}||_2^2 < \epsilon_t$ and $||\hat{\boldsymbol{\gamma}}^{(k+1)} - \hat{\boldsymbol{\gamma}}^{(k)}||_2^2 < \epsilon_t$ for some tolerance level ϵ_t. The penalty parameters λ_1 and λ_2 are computed for each iteration, for instance by cross-validation.

6.3.1 Convergence of PLAMM-2

We investigate the convergence property of PLAMM-2, where the penalty parameters λ_1 and λ_2 are treated as fixed constants. Our algorithm follows the same idea as von Neumann's alternating projection algorithm, introduced in von Neumann (1950), which is also used in for instance Xu and Zikatanov (2002). Different applications of the algorithm of alternating projections are given in Deutsch (1992). In von Neumann (1950), it is shown that if we have an algorithm where we perform alternating projections onto two closed linear spaces, then the result will converge to the projection onto the intersection of the two closed linear spaces. The result is given in Theorem 13.7 on p. 55 in von Neumann (1950). The theorem is restated here without proof

Theorem 6.1 (von Neumann (1950))**.** *Let P_A denote the projection operator onto the set A. Assume that M and N are closed linear subspaces. If $E = P_M$ and $F = P_N$, then the sequence Σ_1 of operators $E, FE, EFE, FEFE, \ldots$ has a limit G. The sequence Σ_2: $F, EF, FEF, \ldots$ has the same limit G, and $G = P_{M \cap N}$.*

Let M be the column space of $(\mathbf{X}, \mathbf{Z}')$, under the constraint

$$||\boldsymbol{\beta}||_1 \leq t_1,$$

for some $t_1 > 0$, and $\boldsymbol{\gamma}' = \boldsymbol{\gamma}$, where $\boldsymbol{\beta}$ are the coefficients for the possible linear combinations of the columns of $\mathbf{X}$ and $\boldsymbol{\gamma}'$ are the coefficients for the possible linear combinations of the columns of $\mathbf{Z}'$. $\boldsymbol{\gamma}'$ is a constant vector in M, so that when projecting onto M, $\boldsymbol{\gamma}'$ is not changed.

Projecting onto M then gives the solution to

$$\text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2,$$

under the constraint

$$||\boldsymbol{\beta}||_1 \leq t_1,$$

for some $t_1 > 0$, where $\boldsymbol{\gamma}$ is any given vector in $\mathbb{R}^{d_2 \cdot m}$, satisfying $||\boldsymbol{\gamma}||_{\text{coop}} \leq t_2$.

Correspondingly, N is the column space of $(\mathbf{X}, \mathbf{Z}')$, where the coefficients for the linear combinations of the columns of $\mathbf{X}$ are restricted to being held constant, and the coefficients for the linear combinations of the columns of $\mathbf{Z}'$ are restricted to

$$||\boldsymbol{\gamma}||_{\text{coop}} \leq t_2,$$

for some $t_2 > 0$.

Projecting onto N thus gives the solution to

$$\text{argmin}_{\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2,$$

under the constraint

$$||\boldsymbol{\gamma}||_{\text{coop}} \leq t_2,$$

for some $t_2 > 0$, where $\boldsymbol{\beta}$ is any given vector in $\mathbb{R}^{d_1}$, satisfying $||\boldsymbol{\beta}||_1 \leq t_1$.

The projection onto the intersection $M \cap N$ is thus the solution to our problem

$$\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}} = \text{argmin}_{\boldsymbol{\beta},\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2 + \lambda_1 ||\boldsymbol{\beta}||_1 + \lambda_2 ||\boldsymbol{\gamma}||_{\text{coop}}, \tag{6.4}$$

where there is a one-to-one correspondence between λ_1 and t_1 and a one-to-one correspondence between λ_2 and t_2. This is seen from the fact that the intersection $M \cap N$ contains the column space of $(\mathbf{X}, \mathbf{Z}')$ with linear combination coefficients $(\boldsymbol{\beta}, \boldsymbol{\gamma})'$, where the linear part $\boldsymbol{\beta}$ is restricted by $||\boldsymbol{\beta}||_1 \leq t_1$ and the non-linear part $\boldsymbol{\gamma}$ is restricted by $||\boldsymbol{\gamma}||_{\text{coop}} \leq t_2$, and the solution we are seeking is the projection onto this set.

If the sets M and N satisfied the assumptions of Theorem 6.1, we would know from Theorem 6.1 that our iterative method converges to the desired solution. However, to use the theorem, M and N need to be linear, and it is easy to see that they are not: Let $\mathbf{a}_1 = (\boldsymbol{\beta}_1, \boldsymbol{\gamma}_1)^{\text{T}}$ be an element in M such that $||\boldsymbol{\beta}_1||_1 = t_1$. For M to be linear, it needs to be closed under scalar multiplication, so that for any $c \in \mathbb{R}$, $c\mathbf{a}_1$ is also an element in M. Let c be any scalar larger than 1, so $c > 1$, and let $\mathbf{a}_2 = c\mathbf{a}_1$. Denote the linear parameters of $\mathbf{a}_2$ by $\boldsymbol{\beta}_2$. Then we have that

$$||\boldsymbol{\beta}_2||_1 = \sum_{j=1}^{d_1} |\boldsymbol{\beta}_{2j}| = c \sum_{j=1}^{d_1} |\boldsymbol{\beta}_{1j}| = c||\boldsymbol{\beta}_1||_1 = ct_1 > t_1,$$

so $\mathbf{a}_2 \notin M$, and M is thus not linear. Showing that N is not linear can be done in exactly the same way.

In Bauschke and Borwein (1993), convergence of the von Neumann algorithm for two arbitrary closed convex subsets of a Hilbert space is considered. Under the assumption that H is a Hilbert space and M and N are closed, convex non-empty subsets of H, Corollary 3.4 in Bauschke and Borwein (1993) gives us conditions under which the von Neumann algorithm converges. We will restate Corollary 3.4 here without proof, as a theorem.

Theorem 6.2 (Bauschke and Borwein (1993))**.** *If H is finite-dimensional and the interior of $M \cap N \neq \emptyset$, then the alternating von Neumann sequence converges (in norm).*

Let $H = \mathbb{R}^{d_1 + d_2 \cdot m}$. Then H is finite-dimensional, since d_1 and d_2 are finite. It is also a Hilbert space, since $\mathbb{R}^n$ is a Hilbert space for any finite n. We obviously have that both M and N are subsets of H. Since M only has restrictions on the linear parameters, and N only has restrictions on the non-linear parameters, the interior of the intersection between M and N is clearly not empty, so $M \cap N \neq \emptyset$. If we can show that M and N are closed, convex sets, Theorem 6.2 tells us that we have convergence of our iterative algorithm. We start by giving the definition of a convex set. The definition is as in Dahl (2009), section 1.4.

Definition 6.1. A set $C \subseteq \mathbb{R}^n$ is convex if $(1-\alpha)\mathbf{a} + \alpha\mathbf{b} \in C$ whenever $\mathbf{a}, \mathbf{b} \in C$ and $\alpha \in [0, 1]$.

Lemma 6.3. *M is a closed, convex set.*

Proof. M contains an arbitrary constant linear combination of the column space of $\mathbf{Z}'$, given by the basis coefficients $\boldsymbol{\gamma} \in \mathbb{R}^{d_2 \cdot m}$, satisfying $||\boldsymbol{\gamma}||_{\text{coop}} \leq t_2$ and the linear combinations of the column space of $\mathbf{X}$ with basis coefficients $\boldsymbol{\beta}$, satisfying $||\boldsymbol{\beta}||_1 \leq t_1$. Since the restriction area for $\boldsymbol{\beta}$ is closed and the set of possible values for $\boldsymbol{\gamma}$ is closed, M is a closed set. It remains to show that M is convex. Let $\mathbf{a}_1$ and $\mathbf{a}_2$ be two elements in M. Denote the linear parts of $\mathbf{a}_1$ and $\mathbf{a}_2$ by $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ and their non-linear parts by $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$, respectively. Let $\mathbf{a} = (1-\alpha)\mathbf{a}_1 + \alpha\mathbf{a}_2$, for some $\alpha \in [0,1]$. If we can show that $\mathbf{a} \in M$, then we know that M is convex. Denote the linear part of $\mathbf{a}$ by $\boldsymbol{\beta}$ and the non-linear part of $\mathbf{a}$ by $\boldsymbol{\gamma}$. Then we have that $\boldsymbol{\beta} = (1-\alpha)\boldsymbol{\beta}_1 + \alpha\boldsymbol{\beta}_2 \in \mathbb{R}^{d_1}$ and $\boldsymbol{\gamma} = (1-\alpha)\boldsymbol{\gamma}_1 + \alpha\boldsymbol{\gamma}_2 \in \mathbb{R}^{d_2 \cdot m}$. We need to show that $||\boldsymbol{\beta}||_1 \leq t_1$ and $||\boldsymbol{\gamma}||_{\text{coop}} \leq t_2$. We have

$$||\boldsymbol{\beta}||_1 = ||(1-\alpha)\boldsymbol{\beta}_1 + \alpha\boldsymbol{\beta}_2||_1 \leq (1-\alpha)||\boldsymbol{\beta}_1||_1 + \alpha||\boldsymbol{\beta}_2||_1 \leq (1-\alpha)t_1 + \alpha t_1 = t_1,$$

and

$$||\boldsymbol{\gamma}||_{\text{coop}} = ||(1-\alpha)\boldsymbol{\gamma}_1 + \alpha\boldsymbol{\gamma}_2||_{\text{coop}} \leq (1-\alpha)||\boldsymbol{\gamma}_1||_{\text{coop}} + \alpha||\boldsymbol{\gamma}_2||_{\text{coop}} \leq (1-\alpha)t_2 + \alpha t_2 = t_2,$$

where we have used the triangle inequality. Hence, we have that $\mathbf{a} \in M$, and M is convex. □

Lemma 6.4. *N is a closed, convex set.*

The proof of Lemma 6.4 is identical to the proof of Lemma 6.3. We thus omit it. We now have all the tools we need to prove that our iterative procedure converges.

Theorem 6.5 (Convergence of PLAMM-2)**.** *Let d_1 and d_2 be finite. Then PLAMM-2 converges to the solution of equation* (6.4).

Proof. The result follows directly from Theorem 6.2, using Lemma 6.3 and Lemma 6.4. Theorem 6.2 says that the method converges if the interior of $M \cap N$ is non-empty, and M and N are both closed and convex sets. The interior of $M \cap N$ is obviously non-empty, as also noted above, since it contains all vectors with the linear part $\boldsymbol{\beta}$ satisfying $||\boldsymbol{\beta}||_1 \leq t_1$ and the non-linear part $\boldsymbol{\gamma}$ satisfying $||\boldsymbol{\gamma}||_{\text{coop}} \leq t_2$. From Lemma 6.3 we have that M is closed and convex, and from Lemma 6.4 we have that N is closed and convex. So we can apply Theorem 6.2, and we thus have that our method converges to the solution of equation (6.4). □

6.3.2 Adaptive scheme for PLAMM-2

PLAMM-2 can easily be extended with an adaptive step, in the same manner as the other adaptive methods. Let $\hat{\boldsymbol{\beta}}^{\text{init}}$ and $\hat{\boldsymbol{\gamma}}^{\text{init}}$ be the initial parameter estimates from equation (6.4). Then let

$$w_j = \begin{cases} \infty, & \text{if } |\hat{\beta}_j^{\text{init}}| = 0, \\ 1/|\hat{\beta}_j^{\text{init}}|, & \text{otherwise,} \end{cases}$$

for $j = 1, \ldots, d_1$ and

$$w_j = \begin{cases} \infty, & \text{if } ||\hat{\boldsymbol{\gamma}}_{\mathscr{G}_j}^{\text{init}}||_2 = 0, \\ 1/||\hat{\boldsymbol{\gamma}}_{\mathscr{G}_j}^{\text{init}}||_2, & \text{otherwise,} \end{cases}$$

for $j = d_1 + 1, \ldots, d_1 + d_2$, where $\boldsymbol{\gamma}_{\mathscr{G}_j} = (\gamma_{j1}, \ldots, \gamma_{jm})$, the m basis coefficients corresponding to each non-linear covariate j. We then estimate the parameters as the solution to

$$\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}} = \text{argmin}_{\boldsymbol{\beta},\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2 + \lambda_1 \sum_{j=1}^{d_1} w_j |\beta_j| + \lambda_2 \sum_{j=d_1+1}^{d_1+d_2} w_j ||\boldsymbol{\gamma}_{\mathscr{G}_j}^{+}||_2 + w_j ||\boldsymbol{\gamma}_{\mathscr{G}_j}^{-}||_2.$$

This is solved by an iterative scheme, similar to the non-adaptive method. We use $\hat{\boldsymbol{\beta}}^{(k)}$ to find $\hat{\boldsymbol{\gamma}}^{(k+1)}$ by

$$\hat{\boldsymbol{\gamma}}^{(k+1)} = \text{argmin}_{\boldsymbol{\gamma}} ||\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}^{(k)} - \mathbf{Z}'\boldsymbol{\gamma}||_2^2 + \lambda_2 \sum_{j=d_1+1}^{d_1+d_2} w_j ||\boldsymbol{\gamma}_{\mathscr{G}_j}^{+}||_2 + w_j ||\boldsymbol{\gamma}_{\mathscr{G}_j}^{-}||_2.$$

Then $\hat{\boldsymbol{\gamma}}^{(k+1)}$ is used to find $\hat{\boldsymbol{\beta}}^{(k+1)}$ by

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \text{argmin}_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}'\hat{\boldsymbol{\gamma}}^{(k+1)}||_2^2 + \lambda_1 \sum_{j=1}^{d_1} w_j |\beta_j|.$$

As before, start with an initial guess for $\boldsymbol{\beta}$, and continue updating until convergence is reached in both the linear and the non-linear parameters. The penalty parameters λ_1 and λ_2 are tuning parameters computed by for instance cross-validation. This adaptive extension is called APLAMM-2.

6.3.3 Simulation experiment

To compare PLAMM-2 to the PLAMM-1 method developed in section 6.2, we perform a simulation experiment with the same set-up as in section 6.2.3, using PLAMM-2 and APLAMM-2. So we have $d_1 = 10$, $d_2 = 10$ and $n = 50$. Our initial guess for $\boldsymbol{\beta}$ is $\mathbf{0}$. We have an SNR of 4 and independent covariates. The tuning parameters λ_1 and λ_2 for PLAMM-2 and APLAMM-2 are chosen by 10-fold cross-validation. The results with PLAMM-2 and APLAMM-2 are given in Table 14. We see that with PLAMM-2, the true model is captured quite well, but it selects quite many false linear positives. For APLAMM-2, there are less false positives, but the method has problems selecting g_1. There is a benefit in estimation in the adaptive step, both in the mean squared errors and the estimated linear parameters, but due to the relatively large reduction in true non-linear positives, there is no clear benefit in the adaptive step.

Comparing Table 14 to Table B3 and Table B4, which shows the results for PLAMM-1 and APLAMM-1 where the weights on the non-linear penalty terms are $m^{0.8}$ and $m^{0.9}$, respectively, we see that PLAMM-2 penalises the non-linear covariates even more since for PLAMM-2, the number of false linear covariates is larger. As noted, the selection of the true non-linear covariates is not so good for APLAMM-2. So in this scenario, APLAMM-1 with weights $m^{0.8}$ on the non-linear groups performed better in recovering the true model. The estimated linear parameters are closer to the true parameters for PLAMM-2 than for PLAMM-1, but the mean squared errors for the fitted functions are larger. Note however that with PLAMM-2, we get no non-monotone fits, and for APLAMM-2, only 3% of the simulations gave one or more non-monotone fits for the non-linear functions. This is a lot better than for PLAMM-1 with weights $m^{0.8}$, where 5% of the simulations resulted in one or more non-monotone fits, and 14% resulted in one of more non-monotone fits for the adaptive method. Note also that $m^{0.8}$ was chosen because it had the best performance in this particular setting, so the comparison is not completely fair.

We also try increasing n to $n = 80$ and see if the methods then perform better. The results with $n = 80$ and weights $m^{0.8}$ on the non-linear penalty terms for PLAMM-1 and APLAMM-1 are given in Table 15. The results for PLAMM-2 and APLAMM-2 with $n = 80$ are given in Table 16. It is seen from Table 15 that with PLAMM-1, the true model is captured well now. All the true covariates are selected, but we have some false covariates. With APLAMM-1, we also have that all the true covariates are selected, but we have almost no false covariates. We get small estimation errors for the fitted functions with both methods, but the estimated linear covariates are a bit too small. Again, we do shrink the parameters with the penalty, but the estimated linear parameters seem to be shrunken more than the estimated functions, since the mean squared errors for the functions are not that large. The estimated linear parameters are closer to the true value for APLAMM-1 than for PLAMM-1. We conclude that APLAMM-1 clearly performs better than PLAMM-1 in this setting. From Table 16 we see that PLAMM-2 also captures the true model well, but it has more false linear covariates. The estimation errors for the fitted functions are larger for PLAMM-2 than for PLAMM-1, but the estimated linear coefficients are much closer to the true parameter values with PLAMM-2. APLAMM-2 has larger mean squared errors for the fitted functions than APLAMM-1, but the estimated linear parameters are closer to the true values for APLAMM-2. However, APLAMM-2 has more false covariates, and we therefore conclude that APLAMM-1 performed the best also in this setting. For PLAMM-2, we also see that there is a clear benefit in the adaptive step in this setting. The number of true positives is almost

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-2	0.97 (0.17)	0.99 (0.10)	1.0 (0)	1.0 (0)
APLAMM-2	0.67 (0.47)	0.90 (0.31)	0.99 (0.10)	0.97 (0.17)
	TP	FP		
PLAMM-2	3.96 (0.20)	0.43 (0.90)		
APLAMM-2	3.53 (0.72)	0.010 (0.10)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-2	0.061 (0.038)	0.069 (0.049)	0.089 (0.051)	0.086 (0.052)
APLAMM-2	0.045 (0.031)	0.061 (0.045)	0.069 (0.052)	0.085 (0.065)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-2	4.0 (0)	3.50 (1.90)		
APLAMM-2	4.0 (0)	1.57 (1.30)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-2	1.77 (0.37)	1.79 (0.40)	-1.78 (0.38)	-1.81 (0.38)
APLAMM-2	1.89 (0.33)	1.93 (0.40)	-1.92 (0.40)	-1.94 (0.37)

Table 14: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.3.3, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 50$. Standard deviations are given in parenthesis.

the same, while there is a large reduction in false positives. The estimated mean squared errors are slightly smaller with APLAMM-2, and the estimated linear parameters are closer to the true values. For PLAMM-1, 2% of the simulations gave a non-monotone fit, while 6% of the simulations with APLAMM-1 resulted in a non-monotone fit. For PLAMM-2, none of the simulations gave a non-monotone fit, while for APLAMM-2, 3% of the simulations gave a non-monotone fit. So all the methods perform well in this simulation setting.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	1.38 (1.30)		
APLAMM-1	4.0 (0)	0.011 (0.10)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.017 (0.015)	0.018 (0.015)	0.031 (0.017)	0.021 (0.012)
APLAMM-1	0.022 (0.021)	0.018 (0.016)	0.024 (0.015)	0.024 (0.013)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	1.69 (1.18)		
APLAMM-1	4.0 (0)	0.032 (0.18)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.75 (0.20)	1.71 (0.22)	-1.76 (0.21)	-1.70 (0.21)
APLAMM-1	1.86 (0.20)	1.80 (0.23)	-1.86 (0.21)	-1.79 (0.23)

Table 15: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.3.3, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 80$. The weights for the penalties on the non-linear groups are $m^{0.8}$. Standard deviations are given in parenthesis.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-2	0.99 (0.10)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-2	4.0 (0)	0.38 (0.75)		
APLAMM-2	3.99 (0.10)	0 (0)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-2	0.033 (0.023)	0.030 (0.019)	0.051 (0.023)	0.043 (0.022)
APLAMM-2	0.040 (0.032)	0.024 (0.020)	0.032 (0.022)	0.041 (0.023)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-2	4.0 (0)	3.21 (1.58)		
APLAMM-2	4.0 (0)	1.34 (1.39)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-2	1.85 (0.23)	1.87 (0.24)	-1.87 (0.23)	-1.81 (0.24)
APLAMM-2	1.96 (0.22)	1.99 (0.23)	-1.97 (0.22)	-1.93 (0.23)

Table 16: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.3.3, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 80$. Standard deviations are given in parenthesis.

6.4 Comparison with scam

In the simple setting with $d_1 = 10$, $d_2 = 10$ and $n = 50$, it is possible to compare with the scam method (see section 3.2.2). We do not compare with the scar method, since we would have to put constraints on the non-linear noise covariates, and we do not know what constraints are reasonable for these non-linear noise covariates. The scam method is a method for fitting a regression model with different shape constraints on the functions. We thus let the first d_1 have a linear constraint, and for the d_2 covariates with non-linear monotone effect, we give the monotonicity directions for the first four true covariates, and we let the remaining covariates have a general non-linear effect, since we do not know which monotonicity direction to put on the noise covariates in the simulation experiment. The scam procedure uses B-spline smoothing, so one would generally use quite many interior knots and smooth the fitted functions. However, since we then quickly get into trouble with having more parameters than observations, we use four interior knots to estimate all the non-linear functions. For scam, we report the variables as selected if they are significant at level 0.05.

The results with scam are given in Table 17. Comparing Table 17 to Tables B3 and 14, we see that scam is good at selecting the true linear covariates, as were all the other methods. Only APLAMM-1 with weights $m^{0.8}$ selected fewer false linear positives, but the scam method selects quite many non-linear false positives compared to the other methods. Only PLAMM-1 has more false non-linear positives. Note that the selection criterion for scam is not data driven, and the comparison is thus not completely fair. Scam selects more true positives than APLAMM-1 (which we in the previous section concluded performed best in variable selection out of the other four methods in this simulation experiment), but it selects more false positives, both linear and non-linear. Since capturing the true covariates is often more important than avoiding false covariates, we conclude that scam performed slightly better than APLAMM-1 in selection.

In estimation of the functions, we see that scam has smaller estimation errors than PLAMM-2 and APLAMM-2, but larger estimation errors than PLAMM-1 and APLAMM-1. However, considering the estimated linear parameters, scam outperforms all the other methods. We conclude that in capturing and estimating the true model, scam seems to perform best in this lower dimensional setting, but again note that scam was given more information than the other methods, since it was provided the monotonicity directions of the four functions with monotone effect on the response.

6.4.1 Prediction performance

In order to compare the methods, we also consider the prediction performances in the setting $n = 50$, $d_1 = 10$ and $d_2 = 10$ for the different methods. This is done in a similar manner as in section 5.2. We generate 500 new observations from the

Scam method for the additive partially linear model			
Selection			
g_1	g_2	g_3	g_4
0.94 (0.24)	1.0 (0)	1.0 (0)	1.0 (0)
β_1	β_2	β_3	β_4
1.0 (0)	1.0 (0)	1.0 (0)	0.99 (0.10)
Non-linear TP	Non-linear FP	Linear TP	Linear FP
3.94 (0.24)	0.64 (0.89)	3.99 (0.10)	0.48 (0.75)
Estimation			
g_1	g_2	g_3	g_4
0.035 (0.021)	0.026 (0.022)	0.087 (0.043)	0.045 (0.028)
Estimated parameters			
β_1	β_2	β_3	β_4
1.97 (0.41)	2.00 (0.42)	-2.02 (0.40)	-1.98 (0.44)

Table 17: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.4 with the scam method, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 50$. Standard deviations are given in parenthesis.

Prediction error		
PLAMM-1	APLAMM-1	PLAMM-2
0.49 (0.12)	0.48 (0.18)	0.74 (0.20)
APLAMM-2	Scam	
0.77 (0.34)	0.77 (0.26)	

Table 18: Prediction error for the different methods when $n = 50$, $d_1 = 10$, $d_2 = 10$ and SNR ≈ 4 for the simulation considered in section 6.4.1. Standard deviations are given in parenthesis.

same distribution as the data used to fit the model with the different methods, and then we estimate the mean prediction error of these 500 observations. We repeat this 100 times, and we report the mean prediction error of these 100 repetitions. The estimated prediction errors for the different methods are given in Table 18. The predictions with scam are based on the full fitted model, and not only the significant variables. We see that the prediction error for APLAMM-1 is the smallest among all the methods, but the prediction error for PLAMM-1 is almost as good. The prediction errors for the other methods are a lot larger. The scam method and APLAMM-2 obtained the same prediction error, while PLAMM-2 obtained a slightly smaller prediction error.

We also record the prediction errors in the setting where we increase the number of observations to $n = 80$. Here we use six interior knots for the non-linear functions with scam, since we have more observations. The results are given in Table 19. We see that APLAMM-1 performs the best also in this setting, followed by PLAMM-1. APLAMM-2 performs better than scam, and scam has a smaller prediction error than PLAMM-2. So PLAMM-1 and APLAMM-1 have smaller prediction errors than the scam method, while PLAMM-2 and APLAMM-2 perform roughly equally well as the scam method in prediction. Note that the improvements on prediction error in the adaptive step are not large in this setting, for neither PLAMM-1 nor PLAMM-2.

Prediction error				
PLAMM-1	APLAMM-1	PLAMM-2	APLAMM-2	Scam
0.36 (0.066)	0.34 (0.071)	0.43 (0.084)	0.40 (0.084)	0.42 (0.099)

Table 19: Prediction error for the different methods when $n = 80$, $d_1 = 10$, $d_2 = 10$ and SNR ≈ 4 for the simulation considered in section 6.4.1. Standard deviations are given in parenthesis.

6.5 High dimensional setting

In order to study the performances of PLAMM-1 and PLAMM-2 in the high dimensional data setting, we perform simulation experiments in a more complicated situation, with more covariates with no effect on the response, having $p >> n$. Let $d_1 = 500$, $d_2 = 500$, $n = 50$ and SNR ≈ 4.

The true underlying model is the same as before, so we have 496 linear noise covariates and 496 non-linear noise covariates. We simulate 100 times drawing new random variables for each simulation. All the tuning parameters are chosen by 10-fold cross-validation. We record the number of true covariates selected (true positives) and the number of false covariates selected (false positives). If one of the fitted functions from a simulation is non-monotone, the results from this simulation is discarded. The mean squared error from the estimated function to the true function in the observed points and the mean of the estimated linear coefficients are reported. The results with PLAMM-1 and APLAMM-1 for this simulation are given in Table 20. Here the weights $\sqrt{m}$ is used on the penalty for the non-linear groups. We see from Table 20 that the selection performance is quite bad, especially when it comes to selecting the linear covariates. The methods select almost no linear covariates, neither true nor false. They are better at selecting the non-linear covariates, but also here the methods are far from selecting the true model. The estimation errors are quite good for g_1 and g_2 both for PLAMM-1 and APLAMM-1. For g_3 and g_4, the estimation errors are very large. The estimation error is smaller for APLAMM-1 than for PLAMM-1. We see that the estimated linear covariates are a lot smaller (in absolute value) than the truth for PLAMM-1. The estimated linear parameters are slightly better for the APLAMM-1, but note that these estimates are not based on many simulations, since the linear covariates were rarely selected.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	0.20 (0.40)	0.35 (0.48)	0.50 (0.50)	0.60 (0.49)
APLAMM-1	0.11 (0.32)	0.22 (0.42)	0.41 (0.49)	0.49 (0.50)
	TP	FP		
PLAMM-1	1.65 (1.24)	8.0 (9.51)		
APLAMM-1	1.23 (1.11)	2.6 (3.78)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.12 (0.062)	0.15 (0.097)	0.29 (0.16)	0.24 (0.13)
APLAMM-1	0.081 (0.053)	0.069 (0.062)	0.19 (0.13)	0.12 (0.079)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	0.06 (0.24)	0.03 (0.17)	0.08 (0.27)	0.04 (0.20)
APLAMM-1	0.011 (0.11)	0.022 (0.15)	0 (0)	0.011 (0.11)
	TP	FP		
PLAMM-1	0.21 (0.52)	0.18 (0.57)		
APLAMM-1	0.044 (0.26)	0.022 (0.15)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	0.16 (0.15)	0.57 (0.42)	-0.26 (0.37)	-0.27 (0.40)
APLAMM-1	0.71	0.63	NA	-2.17

Table 20: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.5, where SNR ≈ 4, $d_1 = 500$, $d_2 = 500$ and $n = 50$. Standard deviations are given in parenthesis.

Choice of weights We saw in the simulation experiment that PLAMM-1 selected almost no linear covariates. Therefore, the weights must be imbalanced. To compensate for this, we need to penalise the non-linear terms even more in comparison to the linear terms. This was also noted in section 6.2.3, where we concluded that weights $m^{0.8}$ on the non-linear penalty terms gave a good balance in the simulation setting considered.

As noted earlier, the standard group lasso penalty (Yuan and Lin, 2006) multiplies the parameter norms with the square root of the group size to penalise the larger groups more. However, in standard group lasso, there is only one parameter for each covariate, while there are here m parameters for each non-linear covariate. These parameters have additive effects, since the I-spline basis functions of order two, $I_k^2(x)$, take the value 1 for $x > t_{k+2}$. This may be the reason why we do not get fair penalty balance, and why it was not possible for us to find similar additive partially linear methods with variable selection by using groups of spline basis functions in the literature.

In Simon and Tibshirani (2012), they argue that when there are correlated features within a group (as with I-splines), a different choice of penalty is better. They claim that the standard penalty used in Yuan and Lin (2006) is designed for uncorrelated features with orthonormal design matrices, even though a lot of the applications have been used for general problems with correlated features. Simon and Tibshirani (2012) introduce the standardised group lasso, where the penalty is given by

$$\sum_{j=1}^{J} |\mathscr{G}_j|^{0.5} ||\mathbf{X}^{(j)}\boldsymbol{\beta}^{(j)}||_2,$$

where J is the number of groups, $\boldsymbol{\beta}^{(j)}$ is the vector of parameters corresponding to group j and $\mathbf{X}^{(j)}$ is the design matrix for the covariates in group j.

Unfortunately, it is not possible to use this penalty with the currently available implementation of cooperative lasso which is used for the implementation of monotone splines lasso, since it takes as input a vector of weights with one scalar weight for each group, and it is not in general possible to write $||\mathbf{X}^{(j)}\boldsymbol{\beta}^{(j)}||_2$ as $w||\boldsymbol{\beta}^{(j)}||_2$ for any $w \in \mathbb{R}$.

Solving the group lasso problem by first orthonormalising the design matrices within each group, running the standard group lasso and then transforming back to the coefficients in the original basis turns out to give the same results as the standardised lasso when the group sizes are smaller than n for every group. However, if we use this approach and orthonormalise the design matrix for one of the covariates represented in the I-spline basis, then the basis functions would no longer be monotone, and using the cooperative lasso penalty will no longer solve our problem. We thus stick with our ad hoc penalisation scheme with $m^{0.8}$.

The results for PLAMM-1 and APLAMM-1 when $n = 50$, $d_1 = 500$, $d_2 = 500$ and $m_j = m^{0.8}$ for the non-linear groups of covariates, where again m is the number of I-spline basis functions, and $m_j = 1$ for the linear covariates, are given in Table 21. Comparing Table 20 to Table 21, we see that with weights that penalise the non-linear covariates more, we get a huge improvement. With this weighting scheme, the methods select a lot more of the linear covariates, and estimate them better. We also have slightly more true non-linear covariates, and less false non-linear covariates. The mean squared errors for the fitted functions for the non-linear covariates are quite similar for PLAMM-1, while for APLAMM-1, the estimation errors are somewhat larger when we penalise the non-linear covariates more. We see that the balance between the linear and the non-linear covariates is more fair, but it may seem like the methods now favour the linear covariates a little more than the non-linear covariates. Even though the methods perform much better in selection with this weighting scheme, they still perform quite badly. They are not close to selecting the true underlying model.

PLAMM-2 The results for PLAMM-2 and APLAMM-2 in this high dimensional setting are given in Table 22. We see from Table 22 that PLAMM-2 and APLAMM-2 favour the linear covariates over the non-linear covariates, since they include more of the false linear covariates. This coincides with what we have seen in the previous simulation experiments. Since including a false covariate with a non-linear effect costs more in terms of degrees of freedom, it is desirable that the method rather selects a false linear covariate, than a false non-linear covariate. Comparing Table 22 to Table 20, we see that PLAMM-2 and APLAMM-2 perform much better in selection. They select a lot more of the linear covariates and estimates them better. PLAMM-2 does however have slightly larger mean squared errors for the non-linear functions than PLAMM-1, and APLAMM-2 has slightly larger mean squared errors for the non-linear functions than APLAMM-1, but the differences are not big. Even though they perform better than PLAMM-1 and APLAMM-1, they still perform quite badly. They are not close to selecting the true underlying model.

Comparing Table 22 to Table 21, we see that the performances of PLAMM-2 and APLAMM-2 and the performances of PLAMM-1 and APLAMM-1, respectively, with weights $m^{0.8}$ on the non-linear penalty terms are quite similar. PLAMM-2 selects more true non-linear covariates, more false covariates, both linear and non-linear, and less true linear covariates than PLAMM-1. The estimation errors for the fitted non-linear functions and the estimated linear parameters are quite similar. Comparing APLAMM-2 to APLAMM-1 with weights $m^{0.8}$, we see that APLAMM-2 selects more true non-linear covariates, more true linear covariates and more false covariates, both linear and non-linear. The mean squared errors for the fitted functions are quite similar, but the linear parameters are estimated a little better with APLAMM-1. It is not clear that there is a benefit in the adaptive step for PLAMM-2 here. APLAMM-2 has fewer true positives, but the number of false positives decreases substantially. The estimation is better for APLAMM-2 than for PLAMM-2. With APLAMM-2, only 9% of the simulations

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	0.18 (0.39)	0.41 (0.49)	0.51 (0.50)	0.59 (0.49)
APLAMM-1	0.11 (0.32)	0.29 (0.46)	0.38 (0.49)	0.47 (0.50)
	TP	FP		
PLAMM-1	1.69 (1.29)	4.66 (4.88)		
APLAMM-1	1.25 (1.20)	1.89 (2.29)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.11 (0.067)	0.17 (0.10)	0.28 (0.15)	0.29 (0.15)
APLAMM-1	0.084 (0.037)	0.13 (0.094)	0.19 (0.12)	0.19 (0.13)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	0.47 (0.50)	0.48 (0.50)	0.48 (0.50)	0.52 (0.50)
APLAMM-1	0.38 (0.49)	0.38 (0.49)	0.39 (0.49)	0.46 (0.50)
	TP	FP		
PLAMM-1	1.95 (1.33)	12.08 (9.60)		
APLAMM-1	1.61 (1.30)	6.53 (6.22)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	0.80 (0.59)	0.84 (0.55)	-0.89 (0.55)	-0.76 (0.49)
APLAMM-1	1.35 (0.66)	1.31 (0.68)	-1.44 (0.57)	-1.26 (0.66)

Table 21: Ratio of the number of times the different covariates are selected, ratio of the total true and false covariates selected, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.5, where SNR ≈ 4, $d_1 = 500$, $d_2 = 500$, $n = 50$ and the weights are the group size to the power 0.8. Standard deviations are given in parenthesis.

resulted in a non-monotone fit, while for APLAMM-1 with weights $m^{0.8}$ on the non-linear groups, 21% of the simulations resulted in a non-monotone fit. So even though PLAMM-2 and APLAMM-2 select more false covariates, they are at least as good as PLAMM-1 and APLAMM-2 in selecting the true covariates, and since PLAMM-2 and APLAMM-2 are more robust to non-monotone fits, we conclude that PLAMM-2 or APLAMM-2 performed the best in this setting, but all methods perform quite badly.

Increased number of observations We try increasing the number of observations, so that $n = 150$. The results from this simulation experiment for PLAMM-1 and APLAMM-1 with weights $\sqrt{m}$ on the non-linear terms are given in Table B5 in Appendix B. The true model now seems to be captured very well, with all the true covariates selected in every simulation. We have small estimation errors, especially for the estimated functions, but the estimated linear parameters are also a lot closer to the truth than they were with $n = 50$ (Table 20). For PLAMM-1, we see that the number of false non-linear covariates selected is very large. For APLAMM-1, the number of false non-linear covariates selected has decreased substantially, but there are still a lot more false non-linear covariates than false linear covariates. This is in agreement with the previous results – the methods seem to favour the non-linear covariates. In addition, in the simulation experiment with 150 observations, almost half of the simulations for APLAMM-1 resulted in one or more non-monotone fits for the non-linear covariates. Again, this coincides with what we saw in the previous simulation experiments in section 6.2.3 and section 6.5, and the discussion in section 6.5. We need to penalise the non-linear parameters more, since the penalisation parameter is not large enough for us to get a sign-coherent solution, see section 2.4.3.

We tried using weights $m_j = m^{0.8}$ for the penalties on the non-linear groups with $n = 150$. The results are given in Table B6 in Appendix B. We see that we now have a more fair balance between the linear and non-linear covariates, with less false non-linear covariates than what we saw in Table B5 with $m_j = \sqrt{m}$, and more false linear covariates. The mean squared errors for the estimated functions are larger when the non-linear parameters are penalised more, but the estimated linear parameters are closer to the true values. Even though increasing the weights to $m^{0.8}$ for the non-linear parameters gave a better balance between the linear and non-linear covariates, this was not large enough to avoid the problem with obtaining non-monotone fits. In fact, more than half of the simulations with APLAMM-1 now resulted in one or more non-monotone fits.

In Table B7, the results for PLAMM-2 and APLAMM-2 for this setting are given. These simulations were very computationally intensive, and the results are therefore only based on 50 simulations. We see that the results with PLAMM-2 and APLAMM-2 are also very good in this setting. All the true covariates are selected in every simulation. PLAMM-2 has quite many false covariates, but

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-2	0.28 (0.45)	0.39 (0.49)	0.56 (0.50)	0.66 (0.48)
APLAMM-2	0.14 (0.35)	0.30 (0.46)	0.40 (0.49)	0.47 (0.50)
	TP	FP		
PLAMM-2	1.89 (1.29)	8.92 (11.12)		
APLAMM-2	1.31 (1.08)	2.01 (3.33)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-2	0.12 (0.078)	0.17 (0.13)	0.26 (0.13)	0.29 (0.16)
APLAMM-2	0.074 (0.054)	0.10 (0.074)	0.18 (0.13)	0.21 (0.14)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-2	0.42 (0.50)	0.45 (0.50)	0.46 (0.50)	0.51 (0.50)
APLAMM-2	0.36 (0.48)	0.41 (0.49)	0.38 (0.49)	0.49 (0.50)
	TP	FP		
PLAMM-2	1.84 (1.60)	13.99 (13.81)		
APLAMM-2	1.65 (1.49)	7.80 (7.71)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-2	0.78 (0.60)	0.79 (0.48)	-0.84 (0.61)	-0.92 (0.59)
APLAMM-2	1.23 (0.72)	1.29 (0.57)	-1.34 (0.67)	-1.36 (0.69)

Table 22: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.5, where SNR ≈ 4, $d_1 = 500$, $d_2 = 500$ and $n = 50$. Standard deviations are given in parenthesis.

APLAMM-2 has substantially decreased the number of false covariates. The estimation errors and the estimated coefficients are slightly better with APLAMM-1 than with APLAMM-2. APLAMM-2 also had the problem of obtaining non-monotone fits in this simulation setting, but it had fewer non-monotone fits than APLAMM-1.

We conclude that there is a huge benefit in estimation and selection in the adaptive step, both for PLAMM-1 and PLAMM-2. APLAMM-1 with weights $m^{0.8}$ performed the best in this setting. However, if obtaining monotone fits is important, then PLAMM-1 with weights $m^{0.8}$ performed the best.

6.5.1 Slightly simpler setting

Since one of the benefits of the methods developed is that they can be used in the high dimensional data setting (as opposed to scam), we also include a simulation experiment in a slightly simpler high dimensional data setting. We let $d_1 = 100$, $d_2 = 100$, $n = 80$ and the signal to noise ratio be 4. The true underlying model is the same as before. The results with PLAMM-1 and APLAMM-1 where the weights for the penalties on the non-linear groups are $m^{0.8}$, are given in Table 23, and the results for PLAMM-2 and APLAMM-2 are given in Table 24. All methods perform very well in this setting. APLAMM-1 clearly performs the best among all the methods. It selects all the true covariates (both linear and non-linear) and it selects very few false covariates. It has the smallest estimation errors among all the methods, and only APLAMM-2 is better at estimating the linear parameters.

In Table 23, the benefit in the adaptive step is very clear. Both PLAMM-1 and APLAMM-1 select all the true covariates, but the number of false positives is substantially decreased with the adaptive step. The mean squared errors for the estimated functions are a lot smaller with the adaptive method, and the estimated linear parameters are closer to their true values.

In Table 24, the benefit in the adaptive step is not so clear. APLAMM-2 has problems selecting g_1. The number of false positives decreases with the adaptive step, the estimation errors for the functions are clearly smaller, and the estimated linear parameters are closer to the true values.

None of the simulations with PLAMM-1 resulted in a non-monotone fit. For APLAMM-1, 31% resulted in one or more non-monotone fits. For PLAMM-2, no simulations resulted in a non-monotone fit, while for APLAMM-2, 3% resulted in one or more non-monotone fits.

The prediction errors are also estimated, and they are given in Table 25. We see that the estimated prediction error for APLAMM-1 is the smallest among all the methods. Second best is the prediction error for PLAMM-1. The estimated prediction error for APLAMM-2 is smaller than the estimated prediction error for PLAMM-2. So in prediction error, there is a benefit in the adaptive step for both methods.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	10.19 (6.52)		
APLAMM-1	4.0 (0)	0.17 (0.51)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.029 (0.021)	0.034 (0.027)	0.053 (0.029)	0.041 (0.020)
APLAMM-1	0.024 (0.018)	0.020 (0.017)	0.026 (0.018)	0.028 (0.019)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0(0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	13.1 (5.95)		
APLAMM-1	4.0 (0)	0.42 (1.01)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.54 (0.30)	1.62 (0.27)	-1.59 (0.27)	-1.59 (0.30)
APLAMM-1	1.78 (0.26)	1.86 (0.22)	-1.82 (0.24)	-1.82 (0.27)

Table 23: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.5.1, where SNR ≈ 4, $d_1 = 100$, $d_2 = 100$ and $n = 80$. The weights for the penalties on the non-linear groups are $m^{0.8}$. Standard deviations are given in parenthesis.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-2	0.98 (0.14)	0.99 (0.10)	1.0 (0)	1.0 (0)
APLAMM-2	0.93 (0.26)	0.98 (0.14)	1.0 (0)	0.99 (0.10)
	TP	FP		
PLAMM-2	3.97 (0.17)	4.14 (4.78)		
APLAMM-2	3.90 (0.31)	0.072 (0.33)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-2	0.051 (0.035)	0.051 (0.039)	0.078 (0.057)	0.070 (0.053)
APLAMM-2	0.044 (0.029)	0.035 (0.034)	0.048 (0.049)	0.057 (0.058)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-2	4.0 (0)	17.6 (11.3)		
APLAMM-2	4.0 (0)	5.85 (5.75)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-2	1.58 (0.30)	1.59 (0.29)	-1.60 (0.31)	-1.60 (0.31)
APLAMM-2	1.88 (0.25)	1.91 (0.27)	-1.90 (0.27)	-1.91 (0.26)

Table 24: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.5.1, where SNR ≈ 4, $d_1 = 100$, $d_2 = 100$ and $n = 80$. Standard deviations are given in parenthesis.

Prediction error	
PLAMM-1	APLAMM-1
0.50 (0.096)	0.35 (0.061)
PLAMM-2	APLAMM-2
0.67 (0.31)	0.53 (0.24)

Table 25: Prediction error for the different methods when $n = 80$, $d_1 = 100$, $d_2 = 100$ and SNR ≈ 4 for the simulation considered in section 6.5.1. Standard deviations are given in parenthesis.

6.6 Conclusion on the performance of the methods

We have seen that in a setting with enough observations and not too many noise covariates ($n = 80$, $d_1 = 10$, $d_2 = 10$, four true non-linear covariates and four true linear covariates), all the methods developed perform well both in selecting the true underlying model and in estimation. There is a clear benefit in the adaptive step for both PLAMM-1 and PLAMM-2. APLAMM-1 performed the best among all methods, and there was no big problem in not obtaining monotone fits. Weights $m^{0.8}$ were used on the penalty terms for the non-linear groups with PLAMM-1 and APLAMM-1. This is because we note that the weights for the linear and the non-linear parameters are imbalanced, and by testing (trial and error) different weighting schemes, we conclude that $m^{0.8}$ gives the best balance. With less observations ($n = 50$), APLAMM-2 might select too few true covariates, while APLAMM-1 might result in non-monotone fits. For PLAMM-1 there is an improvement in the adaptive step, while for PLAMM-2, there is no clear improvement in using the adaptive scheme. We conclude that APLAMM-1 performed the best in this setting, but it did result in some non-monotone fits. Comparing with the scam method in this setting when $n = 50$, we saw that none of our methods performed as well as the scam method, but they still performed well. Again, scam is given more prior information than the other methods. In prediction, the methods we developed performed better than or as well as the scam method in these two settings.

In a high dimensional setting with many noise covariates and few observations (here we had $n = 50$, $d_1 = 500$, $d_2 = 500$, four true non-linear covariates and four true linear covariates), none of the methods performed satisfactorily. PLAMM-1 and PLAMM-2 select many false covariates and quite few true covariates. APLAMM-1 and APLAMM-2 select fewer false covariates, but also less true covariates. With PLAMM-1 it is hard to find an optimal balance between the non-linear terms and the linear terms, while still ensuring that we do not get a non-monotone fit. Our ad hoc weighting scheme using $m^{0.8}$ on the non-linear covariates worked well in the scenario with $n = 50$, $d_1 = 10$ and $d_2 = 10$, the scenario with $n = 80$, $d_1 = 10$ and $d_2 = 10$, $n = 80$, $d_1 = 100$ and $d_2 = 100$, and the scenario with $n = 50$, $d_1 = 500$ and $d_2 = 500$, but not in the scenario where $n = 150$, $d_1 = 500$ and $d_2 = 500$.

When increasing the number of observations to $n = 150$, APLAMM-1 and APLAMM-2 perform very well in estimation and selection. We observe a huge benefit in the adaptive step, in that the number of false covariates decreases substantially, and the estimation errors and estimated parameters improve. However, as noted, with the adaptive methods there are many non-monotone fits. APLAMM-1 performed slightly better than APLAMM-2 in estimation and selection, while APLAMM-2 had fewer non-monotone fits.

The simulations are also performed in a slightly simpler high dimensional setting, with $n = 80$, $d_1 = 100$ and $d_2 = 100$. All methods perform well here, but APLAMM-1 clearly performed the best. It did however result in quite many non-monotone fits.

In general, PLAMM-1 performs better than PLAMM-2 in estimating the effects of the non-linear covariates, while PLAMM-2 estimates the linear parameters better. We conclude that APLAMM-1 seems to be the best method among all the methods developed, especially if it is not important to obtain monotone fits. For PLAMM-2, the benefit in the adaptive step is not clear, since APLAMM-2 has more problems selecting g_1. Comparing prediction errors of the methods to the prediction errors of their adaptive versions, we have that in the simpler setting where $d_1 = 10$, $d_2 = 10$ and $n = 50$ and $n = 80$, the prediction performance for the methods and their adaptive versions are quite similar. In the high dimensional setting where $n = 80$, $d_1 = 100$ and $d_2 = 100$, there is a clear benefit in the adaptive step for the prediction errors. In prediction, we also had that APLAMM-1 performed better than APLAMM-2. If we are more interested in obtaining good predictions than estimating the true effects of the covariates, then it might not be so important to obtain monotone functions, and we would recommend using APLAMM-1.

6.7 Linear or non-linear

The methods PLAMM-1 and PLAMM-2 developed for additive partially linear models with monotone effects of the non-linear covariates and all the other existing methods that we have considered, assume that we a priori know which covariates have a linear effect on the response, and which covariates have a non-linear effect on the response. However, as noted in Lian et al. (2015), such knowledge is rarely available, especially in a high dimensional setting. If you do not have any prior knowledge about the effect of the covariates, one possibility is to let the continuous covariates have a non-linear effect, and code the discrete variables by dummy variables with linear effects, as is done in Du et al. (2012) and the data example in section 6.8. With this coding of the discrete variables, there is one constant effect for each level. There are no assumptions about linearity of the variables, since there are no assumptions about the levels. The methods are thus very flexible. In Liu et al. (2011), they first let the categorical variables have a linear effect and the continuous variables have a non-linear effect. Then they see which of the non-linear fitted functions resemble a linear function, and then refit with these covariates as covariates with linear effects. However, this is too cumbersome to do in the high dimensional data setting, as also noted in Lian et al. (2015).

In Zhang et al. (2011), a method for separating the covariates into covariates with linear effects and general non-linear effects, LAND (Linear and Non-linear Discoverer), was developed. LAND distinguishes linear and non-linear terms, performs variable selection and estimates the functions and the linear parameters. They focus on the classical setting. The underlying model is as before, but we now include the intercept term β_0, so

$$Y_i = \beta_0 + \sum_{k=1}^{d_1} \beta_k x_{ik} + \sum_{k=1}^{d_2} g_k(z_{ik}) + \epsilon_i,$$

where d_1 is the number of linear components, d_2 is the number of non-linear components, $\mathbf{X}$ is the design matrix for the covariates with linear effects and $\mathbf{Z}$ is the design matrix for the covariates with non-linear effects. The x_{ik} and z_{ik} are the matrix elements of $\mathbf{X}$ and $\mathbf{Z}$. Without loss of generality, the covariates are scaled to $[0, 1]$. Since it is not assumed that it is a priori known which covariates will have a linear effect, which will have a non-linear effect and which are irrelevant for the response, we instead write the model as

$$Y_i = \beta_0 + \sum_{k=1}^{d_1+d_2} g_k(x_{ik}) + \epsilon_i,$$

where now $\mathbf{X} = (\mathbf{X}, \mathbf{Z})$. The functions g_k can be linear, general non-linear or identically zero. The functions g_k are assumed to lie in $\mathscr{H}_k$, where $\mathscr{H}_k = \{g :$

g, g' absolutely continuous, $g'' \in L^2[0,1]\}$, where $L^2[0,1]$ is the space of square integrable functions over $[0,1]$. Then the space $\mathscr{H}_k$ has the following orthonormal decomposition: $\mathscr{H}_k = \{1\} \otimes \mathscr{H}_{0k} \otimes \mathscr{H}_{1k}$, where $\{1\}$ is the mean space, $\mathscr{H}_{0k}$ is the linear subspace and $\mathscr{H}_{1k}$ is the non-linear subspace. That is, $\mathscr{H}_{0k} = \{g_k : g_k''(x) = 0\}$ and $\mathscr{H}_{1k} = \{g_k : \int_0^1 g_k(x)dx = 0, \int_0^1 g_k'(x)dx = 0, g_k'' \in L^2[0,1]\}$. Since the function space can be decomposed into a linear part and a non-linear part, every function $g_k \in \mathscr{H}_k$ can be correspondingly decomposed into a linear and a non-linear part (Zhang et al., 2011) as

$$g_k(x) = \beta_{0k} + \beta_k(x_k - \frac{1}{2}) + g_{1k}(x),$$

where $\beta_k(x_k - \frac{1}{2})$ is the linear part and g_{1k} is the non-linear part of g_k. Let $g(\mathbf{x}_i) = \beta_0 + \sum_{k=1}^{d_1+d_2} g_k(x_{ik})$. Then $g \in \mathscr{H}$, where

$$\mathscr{H} = \otimes_{k=1}^{d_1+d_2} \mathscr{H}_k = \{1\} \otimes \left\{\otimes_{k=1}^{d_1+d_2} \mathscr{H}_{0k}\right\} \otimes \left\{\otimes_{k=1}^{d_1+d_2} \mathscr{H}_{1k}\right\} = \{1\} \otimes \mathscr{H}_0 \otimes \mathscr{H}_1.$$

If the estimated $\beta_k \neq 0$ and the estimated $g_{1k} = 0$, then covariate k is estimated to have a linear effect. If the estimated $g_{1k} \neq 0$, then covariate k is estimated to have a non-linear effect. If the estimated $\beta_k = 0$ and the estimated $g_{1k} = 0$, then covariate k is not selected (estimated to having no effect on the response). The estimated g is given as the solution to

$$\text{argmin}_{g \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - g(\mathbf{x}_i))^2 + \lambda_1 \sum_{k=1}^{d_1+d_2} w_{0k} ||\mathscr{P}_{0k} g||_{\mathscr{H}_0} + \lambda_2 \sum_{k=1}^{d_1+d_2} w_{1k} ||\mathscr{P}_{1k} g||_{\mathscr{H}_1},$$

where $\mathscr{P}_{0k} g$ is the projection of g onto $\mathscr{H}_{0k}$, $\mathscr{P}_{1k} g$ is the projection of g onto $\mathscr{H}_{1k}$, $\mathbf{x}_i$ is the vector of observed covariates for observation i, λ_1 and λ_2 are tuning parameters used to control the regularisation of the linear and the non-linear terms, respectively, and w_{0k} and w_{1k} are optional weights, which can for instance be used for an adaptive scheme.

Even though the non-linear functions in LAND are general functions and not necessarily monotone, LAND could be used to separate our covariates into linear and non-linear covariates before performing the analysis with the methods developed in this thesis. However, there is no openly available implementation of the LAND procedure. Lian et al. (2015) also develops a method for separating the covariates into linear and non-linear effects, which can be used in the high dimensional data setting. The idea behind this method is similar to the LAND (Lian et al., 2015). There is no openly available implementation of this method either, but Lian et al. (2015) links to a non-existing page with an R-code for the method. We have unsuccessfully tried to get in touch with the authors of these two methods, asking for implementations of the methods.

6.8 Illustration of the methods using bone mineral density data

We now try out the methods developed in this section on a real data set. The methods scam and scar (see sections 3.2.2 and 3.2.3) are not feasible in the setting here, since this data set has more variables than observations. We will study a bone mineral density data set, from Reppe et al. (2010). Parts of this data set was also used for illustration in Bergersen et al. (2014). The data set consists of 84 post menopausal women who have had a transiliacal bone biopsy. The data set contains various measurements for these women. These are measurements of the bone mineral density (BMD) which is the response variable of interest, gene expressions for 22 815 genes, age, body mass index (BMI), parathyroid hormone values (PTH), vitamin D values (vitD) and carboxy-terminal telopeptide of type 1 collagen (1CTP) from blood samples. For one of the women, the vitamin D value was missing, so we deleted this one observation and worked with the remaining 83 women.

This data is used to fit an additive partially linear model where the response is the BMD. The gene expressions are assumed to have a general monotone effect on the BMD, while the clinical variables are coded as dummy variables with one constant effect on BMD for each level. With this coding, there are no assumptions about linearity on the clinical variables, since we have made no assumptions about the levels. This is a choice we make to be open to all sorts of effects from these variables, but it would also be possible to let the clinical variables have a linear effect on BMD (since they are all numerical variables, and not categorical). For simplicity, only the 100 genes with the largest empirical variance are considered. The idea of selecting the genes with the largest empirical variance is based on the implicit assumption that the response will vary most with the covariates varying the most (Hastie et al. (2011), section 3.4). This is also done in the study in Zhou et al. (2011).

To code the clinical variables as dummy variables, they have to be categorised. Age, PTH, vitamin D and 1CTP are categorised by data quantiles. BMI is categorised by the international classification from WHO [1]. So we fit a partially linear additive model where the clinical variables are coded as dummy variables with linear effects on BMD, and the gene expressions are assumed to have monotone effects on BMD.

First the clinical variables are categorised. Since there was only one of the women who had a BMI low enough to be classified as underweight, we only work with two categories for BMI, normal and overweight. A person is classified as overweight if the BMI is higher than or equal to 25. So we make a new BMI variable by

[1] `http://apps.who.int/bmi/index.jsp?introPage=intro_3.html`

$$\text{BMI} = \begin{cases} 0, & \text{if BMI} < 25, \\ 1, & \text{otherwise.} \end{cases}$$

All the other clinical variables are divided into three categories based on the quantiles of the observations, so we make new age, PTH, vitD and 1CTP variables by

$$\text{Age} = \begin{cases} \text{low}, & \text{if Age} < 57, \\ \text{medium}, & \text{if } 57 \leq \text{Age} \leq 68, \\ \text{high}, & \text{otherwise.} \end{cases}$$

$$\text{PTH} = \begin{cases} \text{low}, & \text{if PTH} < 3.6, \\ \text{medium}, & \text{if } 3.6 \leq \text{PTH} \leq 4.8, \\ \text{high}, & \text{otherwise.} \end{cases}$$

$$\text{vitD} = \begin{cases} \text{low}, & \text{if vitD} < 66, \\ \text{medium}, & \text{if } 66 \leq \text{vitD} \leq 96, \\ \text{high}, & \text{otherwise.} \end{cases}$$

$$\text{1CTP} = \begin{cases} \text{low}, & \text{if 1CTP} < 3.3, \\ \text{medium}, & \text{if } 3.3 \leq \text{1CTP} \leq 4.1, \\ \text{high}, & \text{otherwise.} \end{cases}$$

The regression model is fitted by PLAMM-1, APLAMM-1, PLAMM-2 and APLAMM-2. I-splines of order two, with six interior knots placed evenly at the quantiles of the data, are used. To select the penalty parameters, a 10-fold cross-validation scheme is used. For comparison, the regression model is also fitted by a group lasso regression method, using the R-package *grpreg*. For the group lasso regression, every gene expression is a group consisting of a singleton, and the clinical variables are represented by groups of dummy variables for each clinical variable. Note that with the group lasso method, the gene expressions are forced to have linear effects, so it is not directly comparable to the other methods.

Our clinical variables are coded as grouped dummy variables. Since there is no reason to assume sign-coherence, the dummy variables for each clinical variable are not grouped in PLAMM-1 and APLAMM-1. They are thus all singletons. We let the weights on the penalty terms be group size to the power of 0.8. To still encourage grouping of the dummy variables for the same clinical variable for APLAMM-1, the weights for the adaptive scheme are calculated as if the variables were grouped, so that if one of the dummy variables for a level of a clinical variable is estimated to zero and another is estimated to non-zero in the initial fit, they will both be included as candidates for the adaptive fit. Explained more thoroughly: let $\hat{\beta}_{1_{\text{init}}}$ and $\hat{\beta}_{2_{\text{init}}}$ be two parameter estimates from an initial fit, corresponding to two different levels of the same clinical variable. Then both their weights, w, for the adaptive scheme will be

$$w = \frac{\sqrt{2}}{\sqrt{\hat{\beta}_{1_{\text{init}}}^2 + \hat{\beta}_{2_{\text{init}}}^2}},$$

so that if $\hat{\beta}_{1_{\text{init}}} = 0$ and $\hat{\beta}_{2_{\text{init}}} \neq 0$, the variable corresponding to β_1 will still have finite weight in the adaptive scheme.

We avoid this drawback with PLAMM-2 and APLAMM-2, since we can use the group lasso penalty when fitting the linear part. So with PLAMM-2 and APLAMM-2, the dummy variables corresponding to the same clinical variable are grouped.

The estimated parameters for the clinical variables are given in Table 26. In the table, GL is the group lasso regression method and AGL is the adaptive group lasso regression method. The parameter $\beta^{\text{BMI}}_{\text{overweight}}$ is the difference in bone mineral density between a woman with a BMI value classified as overweight and a woman with a BMI value classified as normal (or underweight), when all other covariates are kept the same.The parameter $\beta^{\text{Age}}_{\text{medium}}$ is the difference in bone mineral density between a woman in the medium age group and a woman in the lower age group. The parameter $\beta^{\text{Age}}_{\text{high}}$ is the difference in bone mineral density between a woman in the high age group and a woman in the lower age group. The parameters $\beta^{\text{PTH}}_{\text{medium}}$, $\beta^{\text{PTH}}_{\text{high}}$, $\beta^{\text{vitD}}_{\text{medium}}$, $\beta^{\text{vitD}}_{\text{high}}$, $\beta^{\text{1CTP}}_{\text{medium}}$ and $\beta^{\text{1CTP}}_{\text{high}}$ are similarly the difference in bone mineral density between a woman being in the corresponding group, compared to a woman being in the lower group.

We see from Table 26 that the only clinical variable selected by all the methods is the PTH. The estimated parameters from all the different methods give the estimated effect that the higher level of PTH, the lower bone mineral density. Vitamin D is selected by all the methods except from APLAMM-1. From the estimated parameters, it is seen that if one has a medium or high value of vitamin D, the bone mineral density is lower than if one has a low value of vitamin D. There does however not seem to be a big difference between bone mineral density between the women having medium and high values of vitamin D. This is in contrast to what one would expect, and there might be a confounding effect. It might for instance be the case that women who are aware of their low bone mineral density start taking vitamin D supplements. Vitamin D is used for treatment and prevention of osteoporosis. However, Reid et al. (2014) present an extensive study where they conclude that there is very little evidence of an overall benefit of vitamin D supplements on bone mineral density. BMI is selected by all the methods except from the adaptive group lasso. The effect is positive for all the methods selecting BMI, so that the higher BMI, the higher bone mineral density. This is as expected. Age is selected by all the methods except from APLAMM-1 and the adaptive group lasso. The estimated effect of age is negative for all the methods selecting age. The higher age, the lower bone mineral density, again as one should expect. The estimated difference in bone mineral density for the medium age group and the low age group is quite small. The estimated difference in bone mineral density for the medium age group and the high age group is a lot larger. 1CTP is only selected by PLAMM-1 and PLAMM-2, and not by their adaptive versions. The sets of genes selected by the methods are very similar. PLAMM-1 selects four genes, and APLAMM-1 selects two of these genes. PLAMM-2 selects four genes. These four genes are the same genes that were selected by PLAMM-1. APLAMM-2 selects two of these genes – the same two genes as selected by APLAMM-1. The group lasso regression method selects seven genes, among which four of them are the four genes selected by PLAMM-1 and PLAMM-2. The adaptive group lasso selects two genes – the same two genes that are selected by APLAMM-1 and APLAMM-2. In Table 26, the monotonicity directions for the estimated monotone functions and the estimated parameters for the gene expressions for group lasso and adaptive group lasso are also given. $+$ denotes that the estimated function is monotonically increasing, and $-$ denotes that the estimated function is monotonically decreasing.

The genes that are selected by all the methods are AFFX-M27830_M_at and 210170_at (PDLIM3). The estimated effects of these genes with all the methods are given in Figure 16 and Figure 17, respectively. The gene AFFX-M27830_M_at was also selected by the monotone splines lasso regression analysis on this data set in Bergersen et al. (2014) and in Reppe et al. (2010). We see that the estimated functions for both the genes seem to be quite linear, but we see some curvature in the estimated functions with the monotone methods. The estimated functions with PLAMM-1 and PLAMM-2 are very similar. The adaptive versions are more different, but they are still quite similar. The two genes that are selected only by PLAMM-1, PLAMM-2 and the group lasso method are 221491_x_at and 235009_at. The estimated effects of these two genes on BMD are given in Figure 18 and Figure 19, respectively. From Figure 18, we see from the estimated monotone functions that the gene 221491_x_at seems to have almost no effect for lower gene expressions, but after reaching a threshold, the effect of the gene is large and steeply decreasing. This effect does not seem to be fitted well by a linear function. From Figure 19, we see that the effect of the gene 235009_at does not seem so important as the effect of the other genes, considering the size of the effects (scale on y-axis). The estimated function with PLAMM-2 is very small – it is almost flat. With PLAMM-1, there seems to be no effect of the gene for low gene expressions, but after reaching a threshold, the estimated function seems quite linear (and increasing). Note that all the estimated functions are monotone.

Estimated parameters

	PLAMM-1	APLAMM-1	PLAMM-2	APLAMM-2	GL	AGL
$\hat{\beta}^{\text{Age}}_{\text{medium}}$	0	0	-0.17	-0.064	-0.021	0
$\hat{\beta}^{\text{Age}}_{\text{high}}$	-0.39	0	-0.42	-0.11	-0.045	0
$\hat{\beta}^{\text{BMI}}_{\text{overweight}}$	0.68	0.64	0.80	0.67	0.50	0
$\hat{\beta}^{\text{PTH}}_{\text{medium}}$	-0.32	-0.36	-0.61	-0.68	-0.53	-0.71
$\hat{\beta}^{\text{PTH}}_{\text{high}}$	-0.56	-0.67	-0.91	-1.1	-0.73	-1.1
$\hat{\beta}^{\text{vitD}}_{\text{medium}}$	-0.14	0	-0.45	-0.41	-0.19	-0.39
$\hat{\beta}^{\text{vitD}}_{\text{high}}$	-0.14	0	-0.46	-0.39	-0.16	-0.35
$\hat{\beta}^{\text{1CTP}}_{\text{medium}}$	0	0	-0.045	0	0	0
$\hat{\beta}^{\text{1CTP}}_{\text{high}}$	-0.10	0	-0.22	0	0	0

Estimated monotonicity direction

	PLAMM-1	APLAMM-1	PLAMM-2	APLAMM-2	GL	AGL
AFFX-M278-30_M_at	+	+	+	+	0.50	0.60
210170_at	−	−	−	−	-0.31	-0.23
221491_x_at	−	0	−	0	-0.070	0
235009_at	+	0	+	0	0.067	0
209480_at	0	0	0	0	0.051	0
236203_at	0	0	0	0	0.0061	0
210305_at	0	0	0	0	0.080	0

Table 26: Estimated parameters with the different methods for the bone mineral density data from the data example in section 6.8. The monotonicity directions for the selected genes are given for the monotone methods, and the estimated linear parameters for the gene expressions are given for group lasso and adaptive group lasso. GL is the group lasso method and AGL is the adaptive group lasso method.

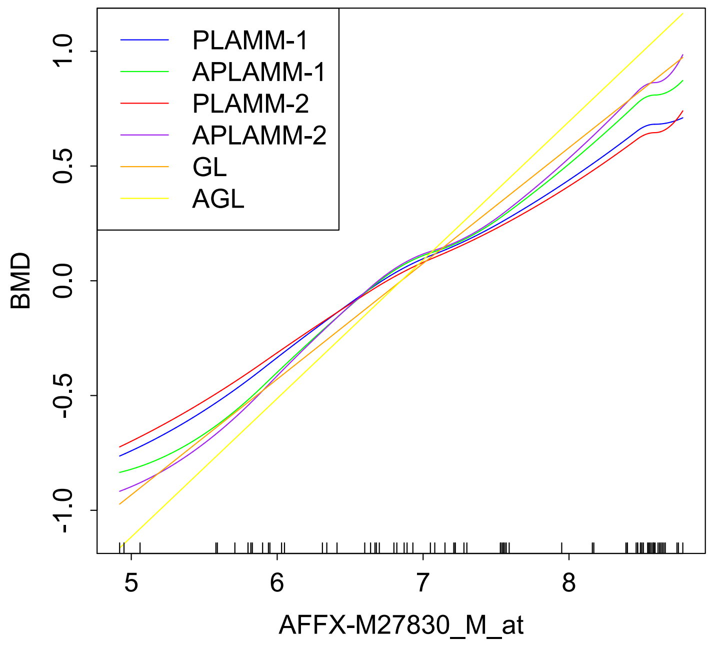


Figure 16: Estimated effect of the gene AFFX-M27830_M_at in the bone mineral data example from section 6.8.

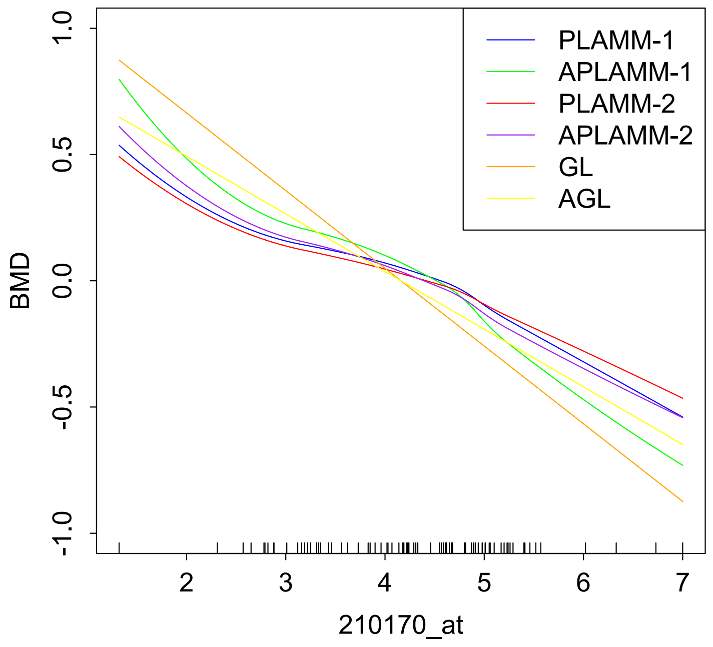


Figure 17: Estimated effect of the gene 210170_at in the bone mineral data example from section 6.8.

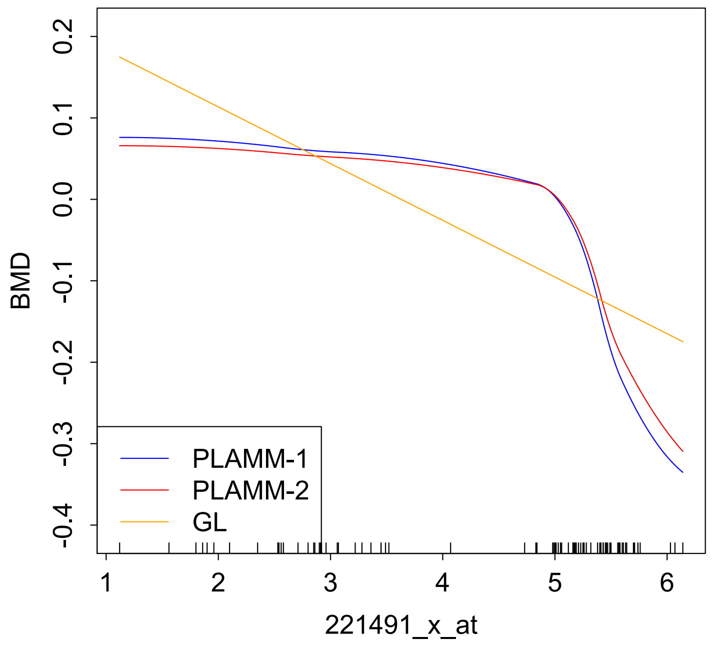


Figure 18: Estimated effect of the gene 221491_x_at in the bone mineral data example from section 6.8.

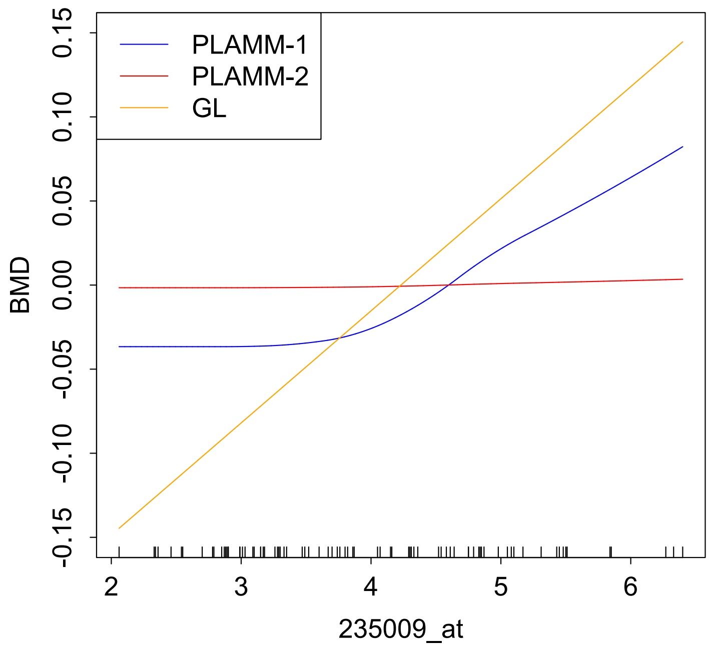


Figure 19: Estimated effect of the gene 235009_at in the bone mineral data example from section 6.8.

6.8.1 Increased number of genes

The model is also fitted with all the methods in a situation with more genes included. The set-up is the same as before, but the genes considered are the 500 genes with the highest empirical variance.

Now the sets of genes selected by the methods differ more. This is not surprising, especially since genes are often highly correlated. PLAMM-1 selects nine genes and APLAMM-1 selects five of these genes. PLAMM-2 selects 24 genes, which is a lot more than PLAMM-1. APLAMM-2 selects six of these. Among the 24 genes selected by PLAMM-2 are eight of the genes selected by PLAMM-1. All these methods have three selected genes in common. All the estimated functions for all the methods were monotone.

The group lasso selects 51 genes, which is a lot more than all the other methods. Among these are eleven of the genes selected by PLAMM-2 and six of the genes selected by PLAMM-1. The adaptive group lasso selects 15 genes. Among these are five of the genes selected by APLAMM-2 and three of the genes selected by APLAMM-1. The three genes that are selected by all the partially linear methods are also selected by the group lasso method, but the adaptive group lasso method only selects two of them.

The estimated functions for the genes that are selected by all the partially linear methods are given in Figures 20, 21 and 22. In Figure 20, we again have some sort of threshold effect, where the effect of the gene expression of gene 206881_s_at seems to be quite small, until we hit a breakpoint, and the slope gets steeper. The estimated effect is a lot larger with APLAMM-2 than with the other methods. In Figure 21, we see that the effect of the gene expression of the gene 223869_at seems to be quite linear, with some flattening for midrange values. For the gene expression in Figure 22, we see that the gene 205959_at also seems to have a threshold effect, but the effect is more extreme than what we had for the gene 206881_s_at in Figure 20. This shape does not resemble a linear function, which might be the reason why adaptive group lasso was the only method which did not detect an effect of this gene. Note that the shape of the estimated function resembles the shape of the function in Figure 13, a function which was not well captured by a linear method.

In the setting where only 100 genes were considered, the genes AFFX-M27830-_M_at and 210170_at (PDLIM3) were selected by all the methods. In the setting with 500 genes, the gene AFFX-M27830_M_at is selected by all the methods except from APLAMM-2, while the gene 210170_at is only selected by PLAMM-1, APLAMM-1 and PLAMM-2.

The estimated parameters for the clinical variables with the methods in the case with 500 genes are given in Table 27. We see that the clinical covariates selected by the different methods and the estimates are quite similar to the situation with 100 genes, comparing with Table 26. Age is not selected by APLAMM-2 and the group lasso method, as opposed to what we had earlier. BMI is now

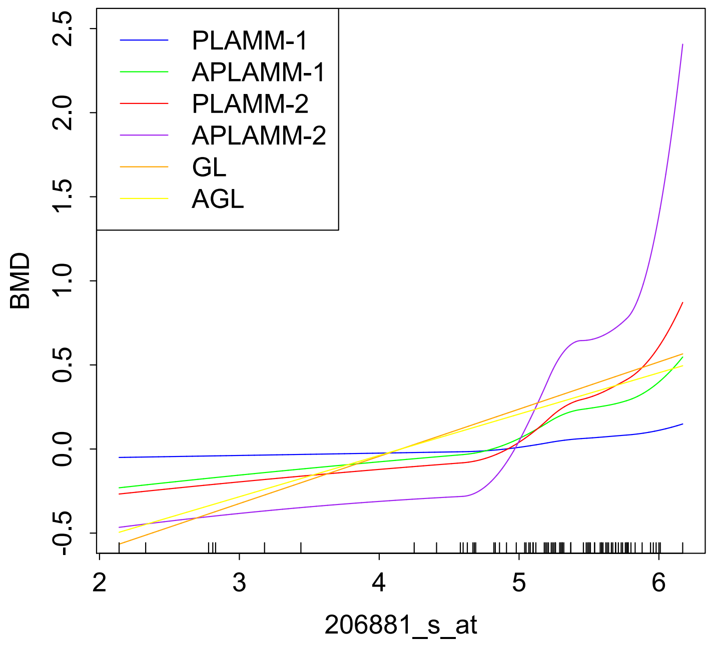


Figure 20: Estimated effect of the gene 206881_s_at in the bone mineral data example from section 6.8.1.

selected by all the methods, while adaptive group lasso did not select BMI in the situation where we only considered 100 genes. The estimated effects of PTH, vitamin D and 1CTP are very similar for the two situations.

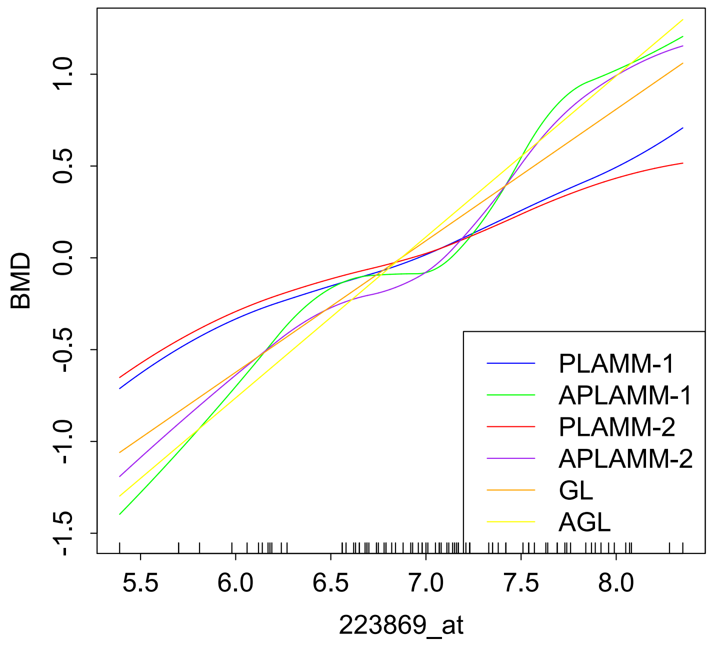


Figure 21: Estimated effect of the gene 223869_at in the bone mineral data example from section 6.8.1.

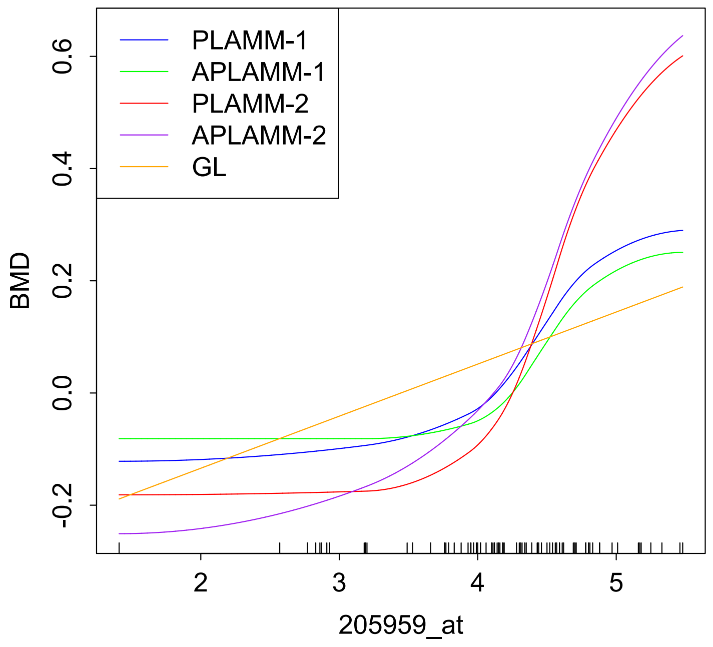


Figure 22: Estimated effect of the gene 205959_at in the bone mineral data example from section 6.8.1.

Estimated parameters

	PLAMM-1	APLAMM-1	PLAMM-2	APLAMM-2	GL	AGL
$\hat{\beta}^{\text{Age}}_{\text{medium}}$	0	0	-0.018	0	0	0
$\hat{\beta}^{\text{Age}}_{\text{high}}$	-0.38	0	-0.15	0	0	0
$\hat{\beta}^{\text{BMI}}_{\text{overweight}}$	0.49	0.61	0.57	0.60	0.20	0.11
$\hat{\beta}^{\text{PTH}}_{\text{medium}}$	0	-0.29	-0.79	-0.72	-0.95	-0.82
$\hat{\beta}^{\text{PTH}}_{\text{high}}$	-0.30	-0.61	-0.98	-1.0	-1.0	-1.2
$\hat{\beta}^{\text{vitD}}_{\text{medium}}$	0	0	-0.54	-0.57	-0.44	-0.41
$\hat{\beta}^{\text{vitD}}_{\text{high}}$	-0.073	0	-0.58	-0.73	-0.40	-0.45
$\hat{\beta}^{\text{1CTP}}_{\text{medium}}$	0	0	-0.053	0	0	0
$\hat{\beta}^{\text{1CTP}}_{\text{high}}$	0	0	-0.15	0	0	0

Table 27: Estimated parameters with the different methods for the bone mineral density data from the data example in section 6.8.1. GL is the group lasso method and AGL is the adaptive group lasso method.

7 Concluding remarks

In this thesis, we have tried to give an overview of methods developed for monotone regression in both high dimensional and classical settings. The linear model is a simple model with strong restrictions. Sometimes it is not sufficient to work with a linear model, and a model which is a lot more flexible is the general additive model, which assumes that the effects of each covariate is a general function. However, it is often reasonable to assume that the effect of a covariate is monotone, especially in the life sciences. Then it is desirable to have methods which incorporate this restriction/prior information to the model estimation. In this thesis, different methods for monotone regression have been presented and studied. We have especially worked with monotone regression in the high dimensional setting, in particular the liso regression method and the monotone splines lasso method. In addition, two new regression methods for fitting a partially linear model with monotone effects have been developed, building on ideas from the monotone splines lasso method.

We have investigated robustness properties of the monotone splines lasso method to the number of interior knots used to fit the monotone splines, and concluded that the method is in fact robust. This is good news, because it means that one does not have to use information in the data to select the number of knots to use. This is especially important in the high dimensional data setting, where there is in general lack of information.

In section 5, we studied the performance of the liso method and the monotone splines lasso method in the classical setting. We found that both methods perform well in the classical setting in terms of selection and estimation, but they did not outperform the scam method, which is a method constructed for monotone (among other options) regression in the classical setting. The need for an adaptive step with the monotone splines lasso is less apparent in the low dimensional setting than it was in the high dimensional setting. When it comes to prediction performance in the classical setting, we found that the monotone splines lasso and the adaptive monotone splines lasso methods performed better than scam in a situation with many noise covariates, but with fewer noise covariates, scam outperformed all the other methods. Even though the monotone splines lasso method did not outperform the scam method, it is still a contribution to the existing methods. It has the property of automatic variable selection, the estimated functions are smooth and it does not have to be provided the monotonicity directions of the functions. To our knowledge, there are no other existing methods with these properties.

In section 6, we worked with the partially linear model where the non-linear functions were assumed to be monotone. Two methods were developed. The first one, PLAMM-1, is a straight forward extension of the monotone splines lasso to the partially linear setting. This method worked well in a simple classical setting, but it did not have a fair penalty balance between the linear and the non-linear covariates. Due to the problems with the balance between the linear and the non-linear covariates, weights penalising more with larger group size (group size to the power 0.8) are used instead of the standard group lasso weights (square root of group size). A two-penalty method was also developed, PLAMM-2, with different penalty parameters on the linear and the non-linear parameters, to avoid the problem of not having a fair balance. The two-penalty problem was solved by an iterative scheme, and convergence of this iterative method was proven. The two methods were also extended with adaptive versions. In the classical setting, PLAMM-2 also performed well. These methods have the advantage that they can be used in the high dimensional data setting. To our knowledge, there exists no other methods for partially linear additive models with monotone effects of the non-linear covariates that can be used in the high dimensional setting. For the scam method, the number of observations quickly becomes too small, since splines are used to fit the functions.

In the high dimensional setting with 50 observations and many noise covariates, none of the methods performed well. Increasing the number of observations to 150 gave a huge improvement in both selection and estimation, both for PLAMM-1 and PLAMM-2. Both methods had problems in providing non-monotone fits, and APLAMM-1 more frequently resulted in a non-monotone fit than APLAMM-2. In a simpler high dimensional data setting, all the methods perform well, while APLAMM-1 clearly performs the best.

PLAMM-2 is in general better at estimating the linear parameters, while PLAMM-1 is in general better at estimating the monotone functions (smaller mean squared errors). The estimated functions with PLAMM-1 were more frequently non-monotone functions than the estimated functions with PLAMM-2, while PLAMM-2 had a higher total number of false covariates selected.

In the classical setting, the methods were compared to the scam method. All of the methods performed well in the classical setting, but the scam method performed the best. Note however that the scam method is given more prior information about the functions, since it is provided the monotonicity directions of the true functions. The prediction performances of the methods were also studied and compared, and the results show that APLAMM-1 was best at prediction, closely followed by PLAMM-1, while PLAMM-2, APLAMM-2 and the scam method have larger and quite similar prediction errors.

We recommend using APLAMM-1. It had the best selection and estimation performance in most settings. It also performed the best in prediction. Conveniently, it is also more computationally efficient than PLAMM-2 and APLAMM-2. The only problem with APLAMM-1 is that the estimated functions might not be monotone. If it is important that the estimated functions are monotone, then one of the other methods should be used instead. Note that the analysis on the bone mineral data set resulted in no non-monotone fits.

7.1 Further work

The weights $m^{0.8}$ on the penalties for the non-linear groups that we have chosen to use for PLAMM-1 and APLAMM-1 were found by trial and error and are quite ad hoc. It would be interesting to try the methods with the weights suggested by Simon and Tibshirani (2012), given in section 6.5. This was not done in this thesis, since the currently available functions for fitting a model with a cooperative penalty do not work with these weights, and reprogramming such a complex optimisation algorithm would be far beyond the scope of this thesis.

With PLAMM-2, the two penalty parameters λ_1 and λ_2 are estimated by cross-validation. They are therefore re-estimated in each iteration. It would be interesting to study whether or not a different selection procedure for the penalty parameters leads to different results. The iteration stops when the estimated model parameters converge, but since the estimated parameters depend on λ_1 and λ_2, a selection method which estimates the penalty parameters before the model estimation might give better results. One could for instance use a generalised cross-validation (GCV) approach. However, it is not straight forward to find GCV expressions for these methods.

Another aspect is the computational efficiency. PLAMM-2 is very computationally demanding, and it will probably converge faster if the penalty parameters were estimated before the model fitting. In most of our simulation runs, PLAMM-2 has not taken many iterations before converging, but each iteration takes a long time, and therefore it is desirable to reduce the number of iterations. In addition, the computation of each fit will be less time demanding if we do not have to compute the cross-validation error for each iteration. The methods from *scoop* which are used for the cooperative lasso penalty are very slow. It could also be that a different iterative algorithm would have a faster convergence rate. In general, improving the computational efficiency of the methods is a topic for further work.

As mentioned in the thesis, it would also be interesting to try out a method which separates the covariates into linear and non-linear covariates and use this before or integrated in the analysis with PLAMM-1 or PLAMM-2. It would also be interesting to develop a method for separation into linear and monotone covariates which utilises the monotonicity property of the functions.

References

Bacchetti, P. (1989). Additive isotonic models. *Journal of the American Statistical Association*, 84(405):289–294.

Barlow, R. and Brunk, H. (1972). The isotonic regression problem and its dual. *Journal of the American Statistical Association*, 67(337):140–147.

Bauschke, H. H. and Borwein, J. M. (1993). On the convergence of von Neumann's alternating projection algorithm for two sets. *Set-Valued Analysis*, 1(2):185–212.

Bergersen, L. C., Tharmaratnam, K., and Glad, I. K. (2014). Monotone splines lasso. *Computational Statistics & Data Analysis*, 77:336–351.

Bollaerts, K., Eilers, P. H., and Mechelen, I. (2006). Simple and multiple P-splines regression with shape constraints. *British Journal of Mathematical and Statistical Psychology*, 59(2):451–469.

Bühlmann, P. and van de Geer, S. (2013). *Statistics for High-Dimensional Data (Springer Series in Statistics)*. Springer.

Bühlmann, P. and Yu, B. (2006). Sparse boosting. *The Journal of Machine Learning Research*, 7:1001–1024.

Chen, Y. and Samworth, R. J. (2015). Generalized additive and index models with shape constraints. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

Chiquet, J., Grandvalet, Y., and Ambroise, C. (2011). Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553.

Chiquet, J., Grandvalet, Y., and Charbonnier, C. (2012). Sparsity with sign-coherent groups of variables via the cooperative-lasso. *The Annals of Applied Statistics*, 6(2):795–830.

Dahl, G. (2009). An introduction to convexity. *Lecture notes, University of Oslo*.

Delecroix, M. and Thomas-Agnan, C. (2000). *Spline and Kernel Regression under Shape Restrictions*, pages 109–133. John Wiley & Sons, Inc.

Dette, H., Neumeyer, N., Pilz, K. F., et al. (2006). A simple nonparametric estimator of a strictly monotone regression function. *Bernoulli*, 12(3):469–490.

Deutsch, F. (1992). The method of alternating orthogonal projections. In *Approximation Theory, Spline Functions and Applications*, pages 105–121. Springer.

Du, P., Cheng, G., and Liang, H. (2012). Semiparametric regression models with additive nonparametric components and high dimensional parametric components. *Computational Statistics & Data Analysis*, 56(6):2006–2017.

Eilers, P. H. and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science*, 11:89–102.

Fang, Z. and Meinshausen, N. (2012). Lasso isotone for high-dimensional additive isotonic regression. *Journal of Computational and Graphical Statistics*, 21(1):72–91.

Harrison, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102.

Hastie, T. and Tibshirani, R. (1986). Generalized additive models. *Statistical Science*, 1(3):297–310.

Hastie, T., Tibshirani, R., and Friedman, J. (2011). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer.

He, X. and Shi, P. (1998). Monotone B-spline smoothing. *Journal of the American statistical Association*, 93(442):643–650.

Huang, J., Horowitz, J. L., and Wei, F. (2010). Variable selection in nonparametric additive models. *The Annals of Statistics*, 38(4):2282–2313.

Lee, A. B. and Izbicki, R. (2016). A spectral series approach to high-dimensional nonparametric regression. *Electronic Journal of Statistics*, 10(1):423–463.

Lian, H., Liang, H., and Ruppert, D. (2015). Separation of covariates into nonparametric and parametric parts in high-dimensional partially linear additive models. *Statistica Sinica*, 25:591–607.

Liu, X., Wang, L., and Liang, H. (2011). Estimation and variable selection for semiparametric additive partial linear models. *Statistica Sinica*, 21(3):1225–1248.

Meyer, M. C. (2008). Inference using shape-restricted regression splines. *The Annals of Applied Statistics*, 2(3):1013–1033.

Meyer, M. C. (2013). Semi-parametric additive constrained regression. *Journal of nonparametric statistics*, 25(3):715–730.

Pya, N. and Wood, S. N. (2014). Shape constrained additive models. *Statistics and Computing*, 25(3):543–559.

Ramsay, J. O. (1988). Monotone regression splines in action. *Statistical Science*, 3(4):425–441.

Reid, I. R., Bolland, M. J., and Grey, A. (2014). Effects of vitamin D supplements on bone mineral density: a systematic review and meta-analysis. *The Lancet*, 383(9912):146–155.

Reppe, S., Refvem, H., Gautvik, V. T., Olstad, O. K., Høvring, P. I., Reinholt, F. P., Holden, M., Frigessi, A., Jemtland, R., and Gautvik, K. M. (2010). Eight genes are highly associated with BMD variation in postmenopausal Caucasian women. *Bone*, 46(3):604–612.

Schell, M. J. and Singh, B. (1997). The reduced monotonic regression method. *Journal of the American Statistical Association*, 92(437):128–135.

Simon, N. and Tibshirani, R. (2012). Standardization and the group lasso penalty. *Statistica Sinica*, 22(3):983.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Tutz, G. and Leitenstorfer, F. (2007). Generalized smooth monotonic regression in additive modeling. *Journal of Computational and Graphical Statistics*, 16(1):165–188.

Vidaurre, D., Bielza, C., and Larranaga, P. (2013). A survey of L1 regression. *International Statistical Review*, 81(3):361–387.

von Neumann, J. (1950). Functional operators, volume 2: The geometry of orthogonal spaces.(am-22).

Wei, F. (2012). Group selection in high-dimensional partially linear additive models. *Brazilian Journal of Probability and Statistics*, 26(3):219–243.

Xu, J. and Zikatanov, L. (2002). The method of alternating projections and the method of subspace corrections in Hilbert space. *Journal of the American Mathematical Society*, 15(3):573–597.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zhang, H. H., Cheng, G., and Liu, Y. (2011). Linear or nonlinear? Automatic structure discovery for partially linear models. *Journal of the American Statistical Association*, 106(495):1099–1112.

Zhao, P. and Yu, B. (2006). On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563.

Zhou, S., Rütimann, P., Xu, M., and Bühlmann, P. (2011). High-dimensional covariance estimation based on Gaussian graphical models. *The Journal of Machine Learning Research*, 12:2975–3026.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Appendices

A Robustness to the distribution of the design matrix

The results from the simulation in section 4.3 were not identical to the results in the simulation experiment in Bergersen et al. (2014). This is due to randomness in the observations, but also due to the fact that our simulations are run with a different design matrix. Consider again the situation where SNR$\approx$4, $n = 50$ and $p = 1000$, but where a different design matrix is drawn in each simulation. The x_{ij} are drawn from the standard normal distribution, truncated to $[0, 1]$, as in the simulation experiment in section 4.3 and the simulation experiment in Bergersen et al. (2014). We simulate 100 times. As before, if a simulation results in a non-monotonic fit for monotone splines lasso, the results from this one simulation are discarded. The ratio of the number of times each of the true covariates is selected, the number of true covariates selected, the number of false covariates selected and the mean squared errors for the fitted functions with the different methods are given in Table A1. The fitted functions with the monotone splines lasso method are given in Figure A1. The fitted functions with the adaptive liso method are given in Figure A2 and the fitted functions with the lasso method are given in Figure A3.

We see from Table A1 that the lasso and the adaptive lasso method are best at selecting the true covariates, followed by monotone splines lasso and adaptive monotone splines lasso. Adaptive liso performs the worst when it comes to selecting the true covariates. All the monotone regression methods seem to have a problem in selecting x_1. However, when it comes to selection of false covariates, the adaptive monotone splines lasso method outperforms all the other methods. Therefore, the adaptive monotone splines lasso method was the best at recovering the true model in this simulation.

From Figure A1, we again clearly see that the adaptive monotone splines lasso method is better at fitting the true function than the monotone splines lasso. From Figure A3 we also clearly see that the adaptive lasso method is better than the lasso method. All of the monotone regression methods are again good at recovering the shapes of the functions. From Table A1, we find that the adaptive monotone splines method has the smallest estimation error among all the methods in this simulation.

We observe that the results from this simulation are not as good for monotone splines lasso as the results in Bergersen et al. (2014) and the results from the simulation experiment in section 4.3. In this simulation experiment, new random x_{ij} were drawn for each simulation. In Bergersen et al. (2014) and section 4.3, the same design matrix was used in each simulation. The fact that we drew new x-observations for each simulation may be the reason why we did not get so good results with monotone splines lasso and adaptive monotone splines lasso for this simulation experiment as in Bergersen et al. (2014) and section 4.3. Monotone splines lasso and adaptive monotone splines lasso are sensitive to the placement of the x_{ij}, since clustering of the observations will give good function estimation in the local area of these points, but not on the whole interval. In addition, if we have a function which is flat in some intervals and steep in other intervals, we might not detect the effect of this function if there are not enough observations in the steep areas. This can for instance happen with the function g_1, which is steepest for x between approximately 0.8 and 1.0 and quite flat otherwise.

Linear methods are not so sensitive to the placements since they estimate one parameter which is a globally constant parameter, so it has the same value on the whole interval for x. However, it is advantageous for the linear methods to have observations at the endpoints of the interval, where the differences for the linear effects are largest. The differences for linear effects are proportional to the distance, and are therefore more apparent when there are observations at the endpoints. In addition, if the function is very flat in one interval, and steep in another, a linear function might not fit the function well, and a linear method might have trouble selecting a variable with such an effect if there are many observations in both types of intervals. An example of such a setting is given in section 5.3.

Liso also seems to be sensitive to the locations of the observations, since we get quite different results for adaptive liso in the simulation experiment in section 4.3 using the same design matrix in each simulation and the simulation experiment using a different design matrix for each simulation.

Drawing new x_{ij} in a random fashion for all simulations will cover more situations and give a more fair evaluation of the method, and we therefore choose to do this for all the other simulation experiments in this thesis.

Choosing an $\mathbf{X}$ from the previous simulation experiment which gave good variable selection with monotone splines lasso ("good x") and an $\mathbf{X}$ which gave bad variable selection with monotone splines lasso ("bad x") and simulating with these matrices 100 times drawing new $\mathbf{y}$-observations for each simulation, with the same noise in the simulations with both the design matrices, gives the results reported in Table A2. We see that monotone splines lasso and adaptive monotone splines lasso perform very differently with the different $\mathbf{X}$-matrices, even though we have the same noise. So monotone splines lasso is strongly dependent on the design matrix. Adaptive liso also performs worse with the "bad x", so it is also sensitive to the design matrix. Lasso on the other hand performs (roughly) equally well in the two scenarios, and thus seems not to be so sensitive to the placement of the covariates. We look closer at the distribution of the $\mathbf{X}$-matrices in the histogram for x_1 in Figure A4. These two matrices are surprisingly similar, given that the results were so different. However, we immediately see that the distribution for the observations giving good results seems to be more symmetric

Selection

	g_1	g_2	g_3	g_4
MS-lasso	0.81 (0.39)	0.97 (0.17)	0.97 (0.17)	0.99 (0.1)
Ad. MS-lasso	0.62 (0.49)	0.97 (0.18)	0.98 (0.15)	0.99 (0.11)
Ad. liso	0.36 (0.48)	0.81 (0.39)	0.90 (0.30)	0.94 (0.24)
Lasso	0.91 (0.29)	0.98 (0.14)	0.97 (0.17)	0.99 (0.10)
Ad. lasso	0.91 (0.29)	0.98 (0.14)	0.97 (0.17)	0.99 (0.10)

	TP	FP
MS-lasso	3.74 (0.52)	14.34 (9.34)
Ad. MS-lasso	3.55 (0.70)	0.76 (1.93)
Ad. liso	3.01 (0.98)	5.27 (3.76)
Lasso	3.85 (0.50)	25.6 (11.4)
Ad. lasso	3.85 (0.50)	4.93 (3.45)

Estimation

	g_1	g_2	g_3	g_4
MS-lasso	0.082 (0.050)	0.089 (0.060)	0.12 (0.069)	0.12 (0.079)
Ad. MS-lasso	0.047 (0.070)	0.052 (0.044)	0.054 (0.052)	0.064 (0.044)
Ad. liso	0.057 (0.037)	0.084 (0.072)	0.096 (0.092)	0.076 (0.076)
Lasso	0.062 (0.035)	0.084 (0.066)	0.27 (0.093)	0.18 (0.096)
Ad. lasso	0.036 (0.015)	0.042 (0.021)	0.17 (0.063)	0.073 (0.052)

Table A1: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions in the simulation experiment considered in Appendix A, where SNR $\approx$ 4, $p = 1000$ and $n = 50$. Standard deviations are given in parenthesis.

than the distribution for the observations giving bad results, and it has more observations at the right endpoint.

To avoid the overweight of observations at the left endpoint, the same simulation experiment is performed, but where the x_{ij} are drawn from a normal distribution with mean 0.5 and standard deviation 1, truncated to [0,1]. The results are given in Table A3. Comparing Table A1 where the random variables drawn have mean 0 to Table A3 where the random variables drawn have mean 0.5, we see that monotone splines lasso and adaptive liso perform better when the data are symmetrically distributed around 0.5, both in true positives and estimation. The improvement for monotone splines lasso is more apparent than the improvement for adaptive liso. The lasso performs worse in this scenario when it comes to true positives and false positives. We especially see that adaptive lasso has more problems in selecting x_1. Looking at g_1, we note that the fewer observations around the right endpoint, the better can the function be fitted by a straight line. This may be the reason why adaptive lasso now has more problems selecting x_1.

We also tried drawing x_{ij} from a uniform distribution on $[0, 1]$. The results from this simulation are given in Table A4. Comparing the results from Table A1 and Table A4, we see that the performance for the monotone splines lasso is equally good when the random variables are drawn from the uniform distribution and from the truncated standard normal distribution. The estimation errors are quite similar, the number of true positives are the same, while the number of false positives is slightly smaller when the observations are drawn from the uniform distribution. The adaptive monotone splines lasso method performed better when the random variables were drawn from the truncated standard normal distribution than when they were drawn from the standard uniform distribution, both when it comes to true positives and false positives. The estimation errors are very similar. The adaptive liso performs best when the variables are drawn from the uniform distribution, in that it selects more true covariates, less false covariates and the estimation errors are slightly smaller. The lasso and the adaptive lasso perform worse when the variables are drawn from the uniform distribution than when they are drawn from the truncated standard normal. Again this is probably due to the fact that there now are more observations around the right endpoint. The lasso and the adaptive lasso method perform quite similar when the variables are drawn from a standard uniform distribution as to when they are drawn from a normal distribution with mean 0.5 and standard deviation 1, truncated to [0,1].

Selection				
Good x				
	g_1	g_2	g_3	g_4
MS-lasso	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. MS-lasso	0.64 (0.48)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. liso	0.34 (0.48)	0.97 (0.17)	1.0 (0)	1.0 (0)
Lasso	0.99 (0.10)	1.0 (0)	1.0 (0)	0.99 (0.10)
Ad. lasso	0.99 (0.10)	1.0 (0)	1.0 (0)	0.99 (0.10)
	TP	FP		
MS-lasso	4.0 (0)	16.31 (6.99)		
Ad. MS-lasso	3.64 (0.48)	0.18 (1.14)		
Ad. liso	3.31 (0.53)	4.10 (2.05)		
Lasso	3.98 (0.20)	27.72 (11.72)		
Ad. lasso	3.98 (0.20)	4.14 (2.61)		
Bad x				
	g_1	g_2	g_3	g_4
MS-lasso	0.26 (0.44)	0.62 (0.49)	0.73 (0.45)	1.0 (0)
Ad. MS-lasso	0.16 (0.37)	0.41 (0.50)	0.59 (0.50)	1.0 (0)
Ad. liso	0 (0)	0.10 (0.30)	0.66 (0.48)	0.94 (0.24)
Lasso	0.96 (0.20)	1.0 (0)	1.0 (0)	1.0 (0)
Ad. lasso	0.96 (0.20)	1.0 (0)	0.99 (0.10)	1.0 (0)
	TP	FP		
MS-lasso	2.61 (1.13)	13.89 (11.39)		
Ad. MS-lasso	2.16 (1.13)	2.27 (3.19)		
Ad. liso	1.70 (0.70)	9.39 (6.12)		
Lasso	3.96 (0.20)	23.08 (11.52)		
Ad. lasso	3.95 (0.26)	5.10 (2.82)		

Table A2: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives for a fixed $\mathbf{X}$ in the simulation experiment considered in Appendix A. SNR ≈ 4, $p = 1000$ and $n = 50$.

Selection

	g_1	g_2	g_3	g_4
MS-lasso	0.88 (0.33)	0.98 (0.14)	0.99 (0.10)	0.99 (0.10)
Ad. MS-lasso	0.73 (0.45)	0.94 (0.23)	0.99 (0.11)	0.98 (0.15)
Ad. liso	0.46 (0.50)	0.78 (0.42)	0.88 (0.33)	0.96 (0.20)
Lasso	0.91 (0.29)	0.97 (0.17)	0.97 (0.17)	0.99 (0.10)
Ad. lasso	0.83 (0.38)	0.96 (0.20)	0.97 (0.17)	0.99 (0.10)

	TP	FP
MS-lasso	3.84 (0.44)	15.23 (9.59)
Ad. MS-lasso	3.64 (0.63)	0.69 (2.01)
Ad. liso	3.08 (0.90)	5.48 (3.61)
Lasso	3.84 (0.51)	29.6 (15.6)
Ad. lasso	3.75 (0.58)	10.2 (5.06)

Estimation

	g_1	g_2	g_3	g_4
MS-lasso	0.072 (0.046)	0.077 (0.052)	0.10 (0.049)	0.11 (0.066)
Ad. MS-lasso	0.044 (0.029)	0.046 (0.041)	0.048 (0.040)	0.068 (0.055)
Ad. liso	0.058 (0.044)	0.080 (0.054)	0.091 (0.078)	0.080 (0.073)
Lasso	0.096 (0.055)	0.094 (0.056)	0.19 (0.061)	0.13 (0.085)
Ad. lasso	0.056 (0.043)	0.048 (0.034)	0.14 (0.039)	0.071 (0.050)

Table A3: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions in the simulation experiment considered in Appendix A, where SNR $\approx$ 4, $p = 1000$, $n = 50$ and the x_{ij} are drawn from a normal distribution with mean 0.5 and standard deviation 1, truncated to [0,1]. Standard deviations are given in parenthesis.

Selection

	g_1	g_2	g_3	g_4
MS-lasso	0.87 (0.34)	0.93 (0.26)	0.97 (0.17)	0.97 (0.17)
Ad. MS-lasso	0.59 (0.49)	0.86 (0.35)	0.92 (0.27)	0.92 (0.27)
Ad. liso	0.49 (0.50)	0.77 (0.42)	0.89 (0.31)	0.97 (0.17)
Lasso	0.91 (0.29)	0.96 (0.20)	0.97 (0.17)	0.99 (0.10)
Ad. lasso	0.85 (0.36)	0.96 (0.20)	0.97 (0.17)	0.98 (0.14)

	TP	FP
MS-lasso	3.74 (0.69)	13.74 (11.27)
Ad. MS-lasso	3.29 (1.02)	0.89 (2.82)
Ad. liso	3.12 (0.90)	4.84 (3.75)
Lasso	3.83 (0.59)	26.39 (12.86)
Ad. lasso	3.76 (0.64)	9.64 (4.56)

Estimation

	g_1	g_2	g_3	g_4
MS-lasso	0.082 (0.053)	0.079 (0.057)	0.12 (0.086)	0.12 (0.072)
Ad. MS-lasso	0.042 (0.031)	0.042 (0.042)	0.062 (0.057)	0.071 (0.055)
Ad. liso	0.064 (0.048)	0.075 (0.065)	0.087 (0.085)	0.070 (0.067)
Lasso	0.091 (0.046)	0.095 (0.056)	0.19 (0.069)	0.13 (0.089)
Ad. lasso	0.050 (0.030)	0.053 (0.035)	0.14 (0.048)	0.071 (0.061)

Table A4: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives and mean squared errors for the estimated functions in the simulation experiment considered in Appendix A, where SNR $\approx$ 4, $p = 1000$, $n = 50$ and the x_{ij} are drawn from a standard uniform distribution. Standard deviations are given in parenthesis.

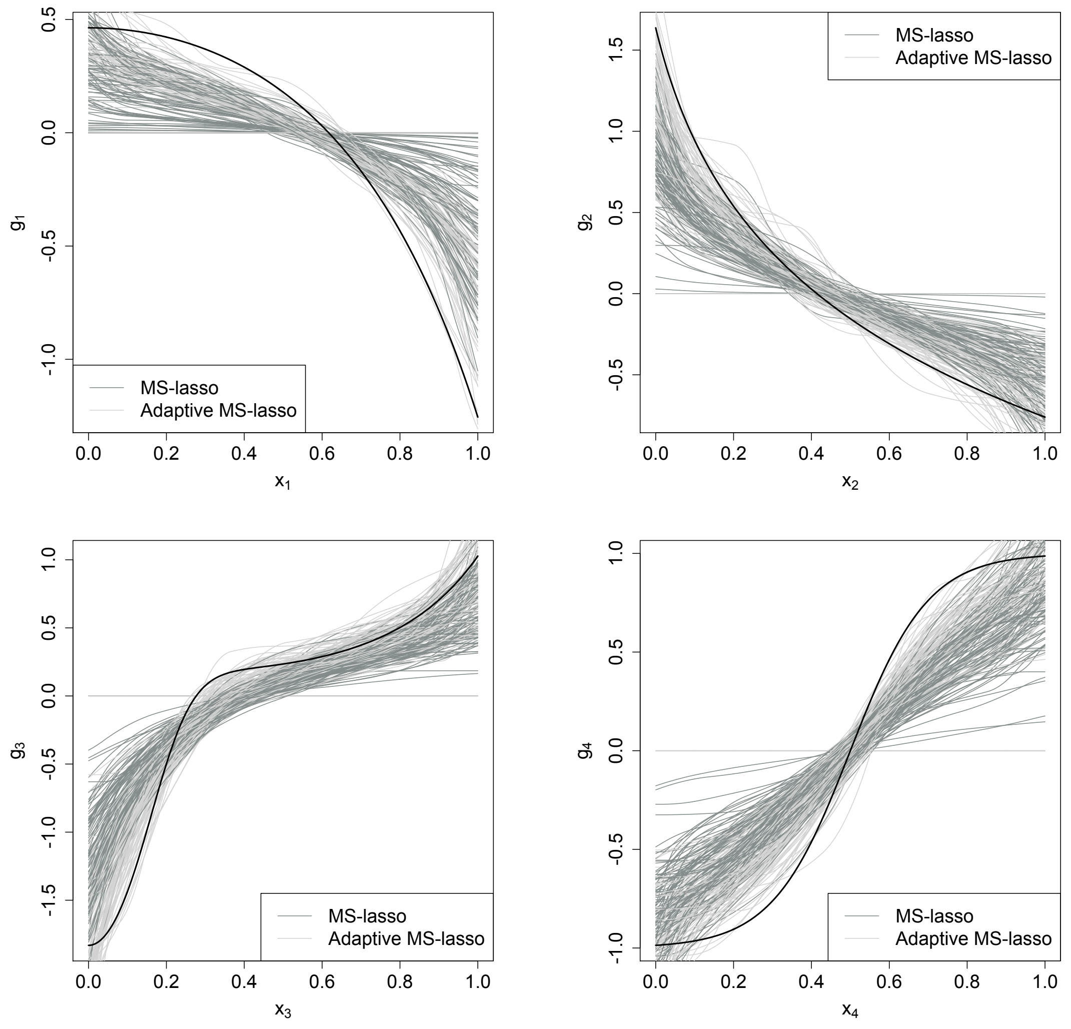


Figure A1: Estimated functions in the simulation considered in Appendix A with $n = 50$, $p = 1000$ and SNR ≈ 4 with the monotone splines lasso (grey) and the adaptive monotone splines lasso (light grey). The true function is given in black.

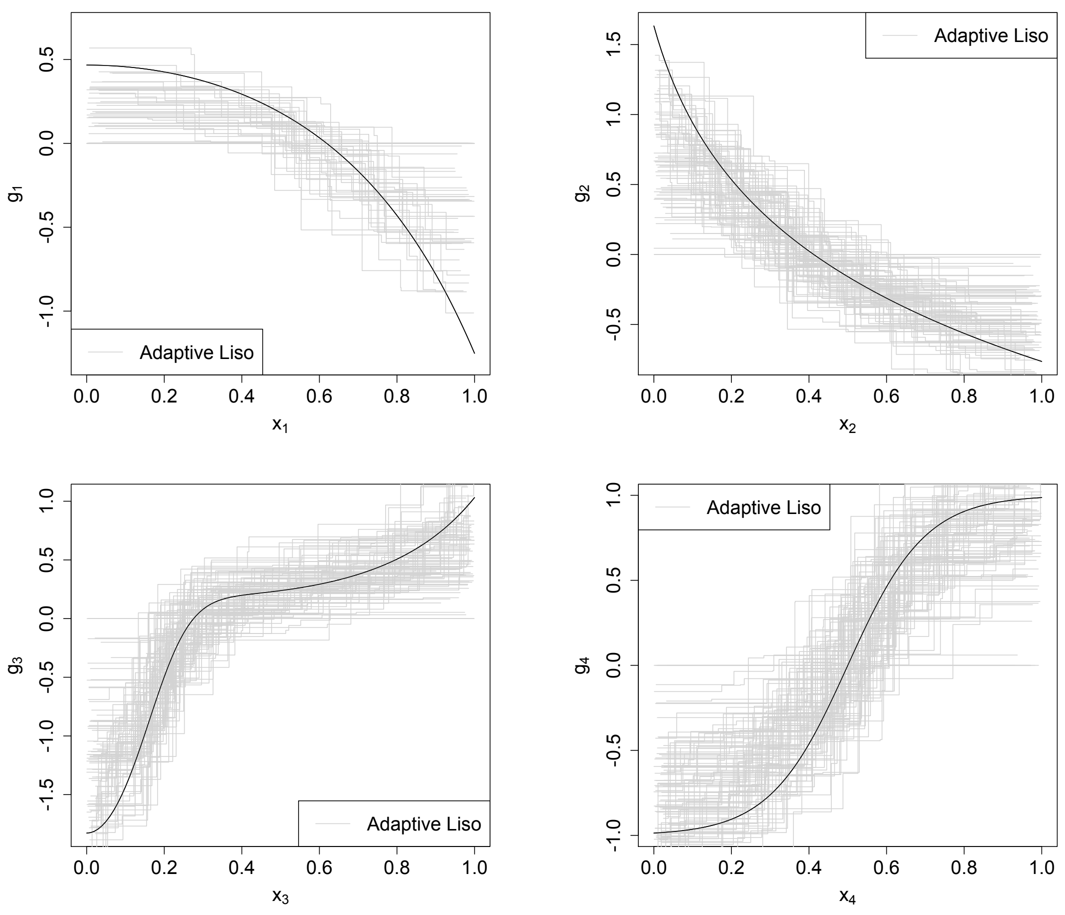


Figure A2: Estimated functions in the simulation considered in Appendix A with $n = 50$, $p = 1000$ and SNR ≈ 4 with adaptive liso (grey). The true function is given in black.

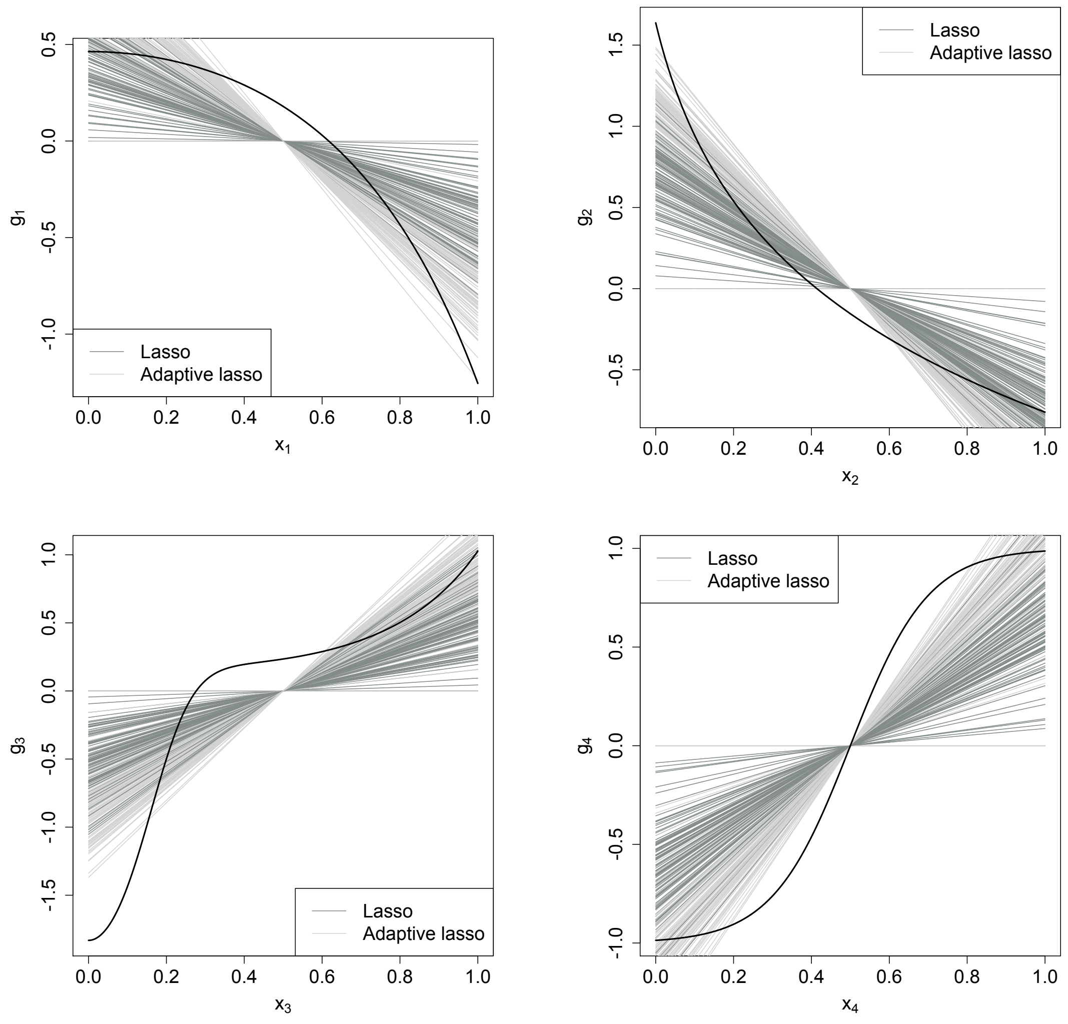


Figure A3: Estimated functions in the simulation considered in Appendix A with $n = 50$, $p = 1000$ and SNR ≈ 4 with the lasso method (grey) and the adaptive lasso method (light grey). The true function is given in black.

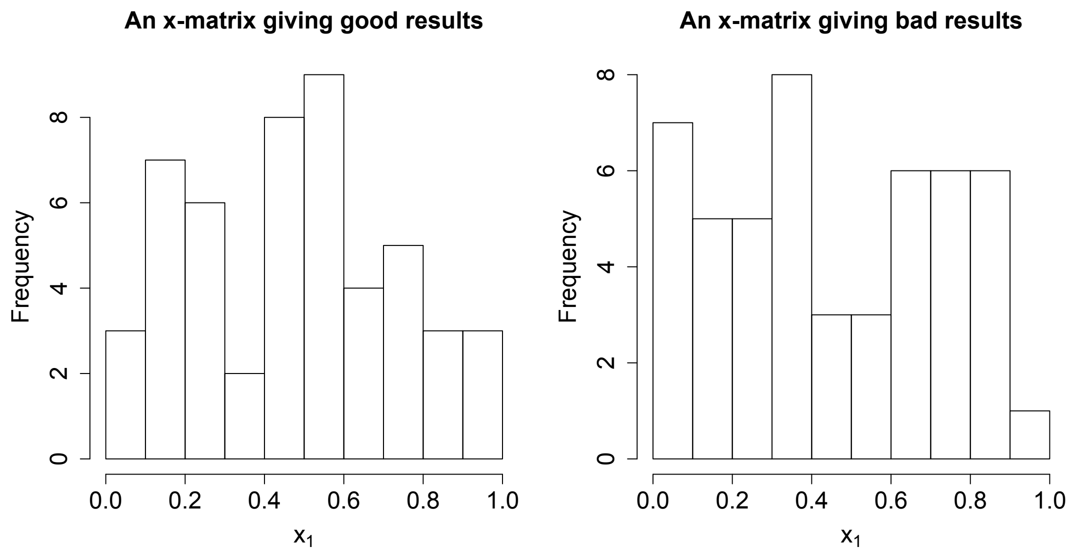


Figure A4: Histogram of observed values of x_1 for an $\mathbf{X}$ matrix giving good detection of x_1 and an $\mathbf{X}$ matrix giving bad detection of x_1 for the simulation experiment considered in Appendix A.

B Tables

Tables of results from different simulation experiments are given in this appendix. Tables B1, B2, B3 and B4 show the results from the simulation experiment in section 6.2.3 for PLAMM-1 and APLAMM-1 with weights $m^{0.6}$, $m^{0.7}$, $m^{0.8}$ and $m^{0.9}$, respectively. Table B5 gives the results for the PLAMM-1 and APLAMM-1 in the high dimensional setting (section 6.5, with $n = 150$). Table B6 gives the results for the same simulation experiment, but with weights $m^{0.8}$ on the penalties for the non-linear parameters. Table B7 gives the results in this simulation setting with PLAMM-2 and APLAMM-2.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	0.97 (0.18)	0.99 (0.10)	0.96 (0.20)	0.98 (0.15)
APLAMM-1	0.96 (0.19)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	3.89 (0.54)	2.82 (1.55)		
APLAMM-1	3.96 (0.19)	0.23 (0.59)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.022 (0.022)	0.031 (0.044)	0.039 (0.024)	0.035 (0.028)
APLAMM-1	0.026 (0.025)	0.034 (0.042)	0.036 (0.025)	0.039 (0.034)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	0.99 (0.10)	0.99 (0.10)	0.98 (0.15)	1.0 (0)
APLAMM-1	0.99 (0.11)	1.0 (0)	0.99 (0.11)	1.0 (0)
	TP	FP		
PLAMM-1	3.96 (0.33)	1.36 (1.16)		
APLAMM-1	3.98 (0.22)	0.12 (0.33)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.54 (0.30)	1.57 (0.35)	-1.56 (0.33)	-1.56 (0.36)
APLAMM-1	1.70 (0.32)	1.70 (0.39)	-1.69 (0.36)	-1.69 (0.39)

Table B1: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.2.3, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 50$. The weights for balancing group size are here $m^{0.6}$. Standard deviations are given in parenthesis.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	0.94 (0.24)	0.99 (0.11)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	2.12 (1.54)		
APLAMM-1	3.93 (0.26)	0.17 (0.49)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.025 (0.025)	0.029 (0.025)	0.042 (0.026)	0.038 (0.029)
APLAMM-1	0.031 (0.028)	0.033 (0.027)	0.040 (0.029)	0.047 (0.040)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	0.99 (0.11)	1.0 (0)	0.99 (0.11)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	1.56 (1.13)		
APLAMM-1	3.98 (0.22)	0.12 (0.33)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.57 (0.28)	1.59 (0.35)	-1.57 (0.35)	-1.60 (0.32)
APLAMM-1	1.73 (0.33)	1.73 (0.39)	-1.71 (0.36)	-1.74 (0.34)

Table B2: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.2.3, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 50$. The weights for balancing group size are here $m^{0.7}$. Standard deviations are given in parenthesis.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	0.91 (0.29)	0.99 (0.11)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	2.12 (1.54)		
APLAMM-1	3.90 (0.31)	0.15 (0.47)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.028 (0.027)	0.032 (0.028)	0.047 (0.029)	0.042 (0.032)
APLAMM-1	0.033 (0.028)	0.035 (0.030)	0.044 (0.034)	0.052 (0.045)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	0.99 (0.11)	1.0 (0)	0.99 (0.11)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	1.88 (1.19)		
APLAMM-1	3.98 (0.22)	0.21 (0.44)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.60 (0.30)	1.63 (0.36)	-1.60 (0.36)	-1.64 (0.32)
APLAMM-1	1.76 (0.32)	1.77 (0.39)	-1.75 (0.35)	-1.77 (0.34)

Table B3: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.2.3, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 50$. The weights for balancing group size are here $m^{0.8}$. Standard deviations are given in parenthesis.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	0.89 (0.32)	0.99 (0.11)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	1.49 (1.38)		
APLAMM-1	3.87 (0.33)	0.080 (0.27)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.030 (0.027)	0.034 (0.028)	0.052 (0.033)	0.046 (0.034)
APLAMM-1	0.034 (0.028)	0.037 (0.032)	0.046 (0.035)	0.056 (0.048)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	0.99 (0.11)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	2.25 (1.35)		
APLAMM-1	3.99 (0.11)	0.29 (0.55)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.65 (0.30)	1.68 (0.35)	-1.66 (0.35)	-1.69 (0.31)
APLAMM-1	1.79 (0.37)	1.80 (0.38)	-1.78 (0.35)	-1.81 (0.33)

Table B4: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.2.3, where SNR ≈ 4, $d_1 = 10$, $d_2 = 10$ and $n = 50$. The weights for balancing group size are here $m^{0.9}$. Standard deviations are given in parenthesis.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	44.98 (13.21)		
APLAMM-1	4.0 (0)	1.89 (5.73)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.019 (0.013)	0.018 (0.012)	0.032 (0.013)	0.024 (0.013)
APLAMM-1	0.0087 (0.0085)	0.0076 (0.0047)	0.012 (0.0076)	0.0095 (0.0071)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	2.72 (2.33)		
APLAMM-1	4.0 (0)	0.057 (0.23)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.44 (0.16)	1.47 (0.19)	-1.48 (0.17)	-1.45 (0.20)
APLAMM-1	1.83 (0.15)	1.84 (0.18)	-1.86 (0.16)	-1.83 (0.17)

Table B5: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.5, where SNR ≈ 4, $d_1 = 500$, $d_2 = 500$ and $n = 150$. Standard deviations are given in parenthesis.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	14.16 (0.37)		
APLAMM-1	4.0 (0)	0.41 (1.62)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-1	0.023 (0.013)	0.020 (0.012)	0.037 (0.014)	0.028 (0.014)
APLAMM-1	0.011 (0.0075)	0.0088 (0.0059)	0.013 (0.0085)	0.013 (0.010)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-1	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-1	4.0 (0)	25.74 (11.6)		
APLAMM-1	4.0 (0)	1.89 (6.64)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-1	1.67 (0.17)	1.67 (0.17)	-1.69 (0.17)	-1.67 (0.19)
APLAMM-1	1.90 (0.15)	1.90 (0.16)	-1.92 (0.14)	-1.90 (0.16)

Table B6: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.5, where SNR ≈ 4, $d_1 = 500$, $d_2 = 500$, $n = 150$ and the weights are the group size to the power 0.8. Standard deviations are given in parenthesis.

Non-linear				
Selection				
	g_1	g_2	g_3	g_4
PLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-2	4.0 (0)	29.34 (27.22)		
APLAMM-2	4.0 (0)	0 (0)		
Estimation (MSE)				
	g_1	g_2	g_3	g_4
PLAMM-2	0.024 (0.016)	0.022 (0.017)	0.036 (0.014)	0.031 (0.016)
APLAMM-2	0.016 (0.011)	0.011 (0.0068)	0.015 (0.0063)	0.021 (0.012)

Linear				
Selection				
	β_1	β_2	β_3	β_4
PLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
APLAMM-2	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
	TP	FP		
PLAMM-2	4.0 (0)	33.44 (22.41)		
APLAMM-2	4.0 (0)	3.33 (2.25)		
Estimated parameters				
	β_1	β_2	β_3	β_4
PLAMM-2	1.63 (0.21)	1.65 (0.18)	-1.64 (0.19)	-1.64 (0.16)
APLAMM-2	1.86 (0.18)	1.88 (0.17)	-1.89 (0.17)	-1.88 (0.16)

Table B7: Ratio of the number of times the different covariates are selected, ratio of the total true and false positives, mean squared errors for the estimated functions and estimated linear coefficients in the simulation considered in section 6.5, where SNR ≈ 4, $d_1 = 500$, $d_2 = 500$, $n = 150$. Standard deviations are given in parenthesis.

C R-code for monotone splines lasso

Included is an R-script for how to use the monotone splines lasso, with the functions for monotone splines lasso. These are slightly modified versions of the code from `http://folk.uio.no/glad/mslasso/`. This is part of the code used in the simulation experiments where we had four true covariates, see for instance section 4.3.

```
#Slightly modified version of the implementation in
#http://folk.uio.no/glad/mslasso/, since the cross-
   validation function
#there was not correct.

isplineDesign <- function(x,knot.vec){
  #function  for computation of I-Splines of degree 2,cf
     Ramsay 88
  #modified from http://www.stat.uni-muenchen.de/institut/
     lehrstuhl/semsto/software/

  I <- numeric()
  deg <- 2
  for(i in 1:(length(knot.vec)-deg)){
    if(x<knot.vec[i]){
      I[i] <- 0
    }
    if (x>knot.vec[i+2]) {
      I[i] <- 1
    }else{
      if(x>=knot.vec[i] && x<=knot.vec[i+1]){
        I[i] <- (x-knot.vec[i])^2 / ((knot.vec[i+1]-knot.
           vec[i])*(knot.vec[i+2]-knot.vec[i]))
      }
      if(x>=knot.vec[i+1] && x<=knot.vec[i+2]){
        I[i] <- 1-(knot.vec[i+2]-x)^2 / ((knot.vec[i+2]-
           knot.vec[i])*(knot.vec[i+2]-knot.vec[i+1]))
      }
    }
  }
  return(I)
}

monotone.basis <- function(X,k,spline,xl,xr,intercept){
  # Function which computes the value in the I-spline
```

```
      basis and the knot vector.
  n <- length(X)
  dx <- (xr-xl)/(k-1)
  num.knots <- k-2

  if(spline=="ispline"){
    deg <- 2
    knot.vec <- seq(xl-(deg-1)*dx,xr+(deg-1)*dx,by=dx)
    knot.vec[(deg+1):(num.knots+2)] <- quantile(unique(as.
        vector(X)), seq(0,1,length = (num.knots+2)))[-c(1,(
        num.knots+2))]
    B <- t(apply(as.matrix(X),1,"isplineDesign",knot.vec=
        knot.vec))
  }
  res <- list("B"=B, "knot.vec"=knot.vec)
  return(res)
}

monotone.splines <- function(Xf,num.knots){
  # Function giving you the design matrix in the I-spline
      basis.
  pf <- ncol(Xf)
  Z <- NULL
  for(j in 1:pf){
    x <- Xf[,j]
    spline <- monotone.basis(x, (num.knots+2), "ispline",
        xl = min(x), xr = max(x), FALSE)
    z1 <- spline$B
    Z=cbind(Z,z1)
  }
  Z <- scale(Z, center = TRUE, scale = FALSE)
  return(Z)
}


monotone.lasso <- function(Xf, Yf, familyf, num.knotsf,
    lambda = NULL){
  # Function for doing the ms-lasso fit
  groups <- as.vector(t(matrix(rep((1):(ncol(Xf)),(num.
      knotsf+2)),ncol(Xf),(num.knotsf+2))))
  Yc <- Yf-mean(Yf)
  Z <- monotone.splines(Xf, num.knotsf)
  #print("Fitting monotone lasso")
  coopfit <- coop.lasso(Z,Yc, groups, family = familyf,
     intercept = FALSE, lambda = lambda)
  msfit <- coopfit
  return(msfit)
}

adaptive.monotone <- function(Xf, Yf, familyf, num.knotsf,
    w, lambda = NULL){
 # Function for doing the adaptive ms-lasso fit.
 groups <- as.vector(t(matrix(rep((1):(ncol(Xf)),(num.
    knotsf+2)),ncol(Xf),(num.knotsf+2))))
 Yc <- Yf-mean(Yf)
 Z <- monotone.splines(Xf, num.knotsf)
 #print("Fitting adaptive monotone lasso")
 adaptivecoopfit <- coop.lasso(Z, Yc, groups, family =
    familyf, wk = w, intercept = FALSE, lambda = lambda)
 adaptivemsfit <- adaptivecoopfit
 return(adaptivemsfit)
}


cvmslasso <- function(Xfcv, Yfcv, K, method, num.knots,w,
   lambda){
  # Cross-validation function for ms-lasso and adaptive ms
     -lasso.
  # Must be called with a lambda sequence.
  error <- matrix(NA, nrow = K, ncol = 100)
  n = nrow(Xfcv)
  antf <- floor(n/K)

  if(n==(K*antf)){
    set.seed(1)
    fold <- sample(rep(1:K, antf))
  }else{
    fold <- sample(c(rep(1:K, antf), 1:(n-(K*antf))))
  }

  for(k in 1:K){
    test.ind <- which(fold == k)
    trainX <- Xfcv[-test.ind,]
    testX <- Xfcv[test.ind,]
    trainY <- Yfcv[-test.ind]
    testy <- Yfcv[test.ind]
    if(method == "monotone.lasso"){
      cv.fit <- monotone.lasso(trainX, trainY, "gaussian",
          num.knots, lambda)
      Ztest <- monotone.splines(testX, num.knots)
    }else if(method == "adaptive.monotone"){
      cv.fit = adaptive.monotone(trainX, trainY, "gaussian
         ", num.knots,w, lambda)
      Ztest <- monotone.splines(testX, num.knots)
    }
    error.k <- (testy-predict(cv.fit, newx = Ztest))^2
    error[k,] <- colSums(error.k)
  }
  MSE <- colMeans(error)
  return(MSE)
}


Ispline2 = function(x, tk, tk1, tk2){
        if(x < tk){
                return(0)
        }

        if(tk <= x  & x <= tk1){
                return((x-tk)^2/((tk1-tk)*(tk2-tk)))
        }

        if(tk1 <= x & x <= tk2){
                return (1-(tk2-x)^2/((tk2-tk)*(tk2-tk1)))
        }

        if(x>=tk2){
                return(1)
        }
}

library(scoop)
non.monotone <- NULL # Vector to store indicies of non-
   monotonic fits.

fit.ms <- monotone.lasso(x, y, familyf = "gaussian", num.
   knotsf = 6)  # Monotone lasso fit.
lambda.seq.ms <- fit.ms@lambda  # Extract lambda sequence
   used.
```

```
cv.ms <- cvmslasso(x, y, K = 10,
      method = "monotone.lasso",
      num.knots = 6, lambda = lambda.seq.ms)  #
         Crossvalidation error for these lambdas.
coef.ms <- fit.ms@coefficients
coef.ms <- coef.ms[cv.ms == min(cv.ms), ]  # Extract
   coefficients with smallest prediction error.

weights.ams <- rep(10000000, p)  # Vector with adaptive
   weights.
counter <- 1
for (i in 1:p){
# Check if the eight coefficients for each p are zero.
  if (sum(coef.ms[counter:(counter + 7)]) != 0){

    if(abs(sum(coef.ms[counter:(counter + 7)])) != sum(abs
       (coef.ms[counter:(counter + 7)]))){
    # Non-monotone fit
      non.monotone <- c(non.monotone, c(k, i))
    }

    if(i == 1){
      tp1.ms[k] <- 1
    }

    else if(i == 2){
      tp2.ms[k] <- 1
    }

    else if(i == 3){
      tp3.ms[k] <- 1
    }

    else if(i == 4){
      tp4.ms[k] <-  1
    }

    else{
      fp.ms[k] <- fp.ms[k] + 1
    }

      weights.ams[i] <- 1/sqrt(sum(coef.ms[counter:(
         counter + 7)]^2))  # Adaptive weights.
    }
    counter <- counter + 8  # Indices for next p.
}

fit.ams <- adaptive.monotone(x, y, familyf = "gaussian",
         num.knotsf = 6, w = weights.ams)  # Adaptive
             monotone fit.
lambda.seq.ams <- fit.ams@lambda  # Extract lambda values.
cv.ams <- cvmslasso(x, y, K = 10, method = "adaptive.
   monotone",
        num.knots = 6, w = weights.ams, lambda = lambda.seq
            .ams)  # Crossvalidation error for lambda.
coef.ams <- fit.ams@coefficients
coef.ams <- coef.ams[cv.ams == min(cv.ams), ]  # Extract
   coefficients with smallest prediction error.
```

D R-code for liso

This is an R-script for how to use adaptive liso. This is part of the code used in the simulation experiments where we had four true covariates, see for instance section 4.3.

```
library(liso)

# Liso cross-validation function.
cv.fit.liso <- do.call("cv.liso", c(list(x = x, y = y,
   monotone = FALSE)))
lambda.min.liso <- cv.fit.liso$optimlam
# Do the liso fit for the optimal lambda.
fit.liso <- do.call("liso.backfit", c(list(x = x, y = y,
lambda = lambda.min.liso, monotone = FALSE)))

weights.adaptive <- liso.covweights(fit.liso, signfind =
   TRUE)  # Adaptive weights.
# Cross-validation function for the adaptive fit.
cv.fit.adaptive.liso <- do.call("cv.liso", c(list(x = x, y
    = y,
monotone = FALSE, covweights = weights.adaptive)))
lambda.min.adaptive.liso <- cv.fit.adaptive.liso$optimlam
fit.liso.adaptive <- do.call("liso.backfit", c(list(x = x,
    y = y,
lambda = lambda.min.adaptive.liso, monotone = FALSE,
covweights = weights.adaptive))) # Do the liso fit for
    the optimal lambda.
# Extract the fitted parameters for the liso fit.
fit.liso.adaptive2 <- do.call("liso.backfit", c(list(x = x
    , y = y,
lambda = lambda.min.adaptive.liso, monotone = FALSE,
 givebeta = TRUE, covweights = weights.adaptive)))

# Check if fitted parameters are different from zero.
if(sum(fit.liso.adaptive2[1:(n - 1)]) != 0) {
  tp1[k] <- 1
}

if(sum(fit.liso.adaptive2[n:(2 * n - 1)]) != 0) {
  tp2[k] <- 1
}

if(sum(fit.liso.adaptive2[(2 * n):(3 * n - 1)]) != 0) {
  tp3[k] <- 1
}

if(sum(fit.liso.adaptive2[(3 * n):(4 * n - 1)]) != 0) {
  tp4[k] <- 1
}

sum.f <- 0  # Number of false positives.
for(i in 5:p) {
  if(sum(fit.liso.adaptive2[((i - 1) * n - (i - 2)):(i * (
      n - 1))]) != 0){
    sum.f <- sum.f + 1
  }
}

fp[k] <- sum.f
```

E R-code for lasso

This is an R-script for how to use lasso and adaptive lasso. This is part of the simulation experiments where we had four true covariates, see for instance section 4.3.

```
library(glmnet)
```

```
empty.fit <- NULL  # Vector to store the simulation
   numbers with no selected parameters.
cv.fit.lasso <- cv.glmnet(x, y, family = "gaussian",
   intercept = FALSE,
standardize = TRUE, alpha = 1)  # Lasso cross-validation
   fit.
lambda.min <- cv.fit.lasso$lambda.min
beta.lasso <- coef(cv.fit.lasso, s = lambda.min)
beta.lasso <- as.matrix(beta.lasso)
beta.lasso <- beta.lasso[-1]  # Remove intercept.

weights.adaptive <- rep(100000000, p)
weights.adaptive[beta.lasso != 0] <- 1/abs(beta.lasso[beta
   .lasso != 0])

# The adaptive lasso will not work if all estimated
   parameters are zero.
if (sum(beta.lasso) == 0) {
  empty.fit <- c(empty.fit, k)
}

else {
  #Adaptive lasso cross-validation fit.
  cv.fit.adaptive.lasso <- cv.glmnet(x, y, family = "
     gaussian",
                        intercept = FALSE, standardize =
                           TRUE,
                        alpha = 1, penalty.factor =
                           weights.adaptive,
                        exclude = which(beta.lasso == 0))
        lambda.adaptive.min <- cv.fit.adaptive.lasso$
           lambda.min
        beta.adaptive.lasso <- coef(cv.fit.adaptive.lasso,
            s = lambda.adaptive.min)
        beta.adaptive.lasso = beta.adaptive.lasso[-1]  #
           Remove intercept.
}
```

F R-code for lm, scam and scar

This is an R-script for how to perform the fits with the ordinary least squares, the scam and the scar. This is part of the simulation experiments with four true covariates and three false covariates, see section 5.

```
library(scam)
library(scar)

fit.lm <- lm(y ~ x)  # Ordinary least squares fit.
beta.ols <- summary(fit.lm)$coefficients[, 1]
pvalues.ols <- summary(fit.lm)$coefficients[, 4]

# True and false positives for lm.
if(pvalues.ols[1] < 0.05) {
  tp1.ols[k] <- 1
}
if(pvalues.ols[2] < 0.05) {
  tp2.ols[k] <- 1
}
if(pvalues.ols[3] < 0.05) {
  tp3.ols[k] <- 1
}
if(pvalues.ols[4] < 0.05) {
  tp4.ols[k] <- 1
}

for (i in 5:p) {
  if(pvalues.ols[i] < 0.05) {
    fp.ols[k] <- fp.ols[k] + 1
  }
}

x1  <- x[, 1]
x2  <- x[, 2]
x3  <- x[, 3]
x4  <- x[, 4]
x5  <- x[, 5]
x6  <- x[, 6]
x7  <- x[, 7]

# Fitted functions with scar.
fit.scar <- scar(x, y, shape = c("de", "de", "in", "in", "
    l", "l", "l"))

# Fitted functions with scam.
fit.scam <- scam(y ~ s(x1, k = 10, bs = "mpd") + s(x2, k =
    10, bs = "mpd")
           + s(x3, k = 10, bs = "mpi") + s(x4, k = 10, bs =
              "mpi") +
           s(x5, k = 10) + s(x6, k = 10) + s(x7, k = 10) -
              1)

# P-values for the covariates with scam.
pvalues.scam <- summary(fit.scam)[8]$s.pv

# True and false positives for scam.
if(pvalues.scam[1] < 0.05) {
  tp1.scam[k] <- 1
}
if(pvalues.scam[2] < 0.05) {
  tp2.scam[k] <- 1
}
if(pvalues.scam[3] < 0.05) {
  tp3.scam[k] <- 1
}
if(pvalues.scam[4] < 0.05) {
  tp4.scam[k] <- 1
}

for (i in 5:p) {
  if(pvalues.scam[i] < 0.05) {
    fp.scam[k] <- fp.scam[k] + 1
  }
}
```

G R-code for PLAMM-1 and APLAMM-1

This is an R-script with methods (functions) for PLAMM-1 and APLAMM-1. Some of these methods are equal to the methods in the monotone splines lasso and are thus called without being stated. This is part of the simulation experiments with four true non-linear covariates and four true linear covariates, see section 6.2.3.

```
#Appropriate modifications of the monotone splines lasso
    functions,
#for applications to the partially linear models.

part.lin.ms <- function(X, Z, Y, family = "gaussian", num.
    knotsf,
                              lambda = NULL, power = 0.8) {
  # Z matrix with non-linear effects, X matrix with linear
      effects.
  # num.knotsf is the number of interior knots.
  # power is the exponent for group size to balance for
      different group sizes.
  # lambda is an optional grid of tuning parameters.

  #Vector with group index for each column in the design
      matrix.
  groups <- as.vector(t(matrix(rep((1):(ncol(Z)), (num.
      knotsf + 2)),
          ncol(Z), (num.knotsf + 2))))
  groups <- c(groups, seq(ncol(Z) + 1, ncol(Z) + ncol(X),
      by = 1))
  Yc <- Y - mean(Y)
  Z_I <- monotone.splines(Z, num.knotsf)
  design.matrix <- cbind(Z_I, X)
  w <- (tabulate(groups))^power
  coop.fit <- coop.lasso(design.matrix, Yc, groups,
  wk = w, intercept = FALSE, lambda = lambda)
  return(coop.fit)
}

ad.part.lin.ms <- function(X, Z, Y, family = "gaussian",
    num.knotsf,
                                   w, lambda = NULL, power =
                                       0.8) {
  #Z matrix with non-linear effects, X matrix with linear
      effects.
  # num.knotsf is the number of interior knots.
  # w are the adaptive weights.
  # power is the exponent for group size to balance for
      different group sizes.
  # lambda is an optional grid of tuning parameters.

  #Vector with group index for each column in the design
```

```
      matrix.
  groups <- as.vector(t(matrix(rep((1):(ncol(Z)), (num.
      knotsf + 2)),
          ncol(Z), (num.knotsf + 2))))
  groups <- c(groups, seq(ncol(Z) + 1, ncol(Z) + ncol(X),
      by = 1))
  Yc <- Y - mean(Y)
  Z_I <- monotone.splines(Z, num.knotsf)
  design.matrix <- cbind(Z_I, X)
  weight <- (tabulate(groups))^power
  coop.fit <- coop.lasso(design.matrix, Yc, groups,
  wk = w * weight, intercept = FALSE, lambda = lambda)
  return(coop.fit)
}

cv.ms.part.lin <- function(X, Z, Y, K, method, num.knots,
   w = NULL, lambda, power = 0.8) {
  # Z matrix with non-linear effects, X matrix with linear
       effects.
  # K is the number of folds for the cross-validation.
  # method is "part.lin" for the initial fit, or "adaptive
      .part.lin" for the adaptive fit.
  # num.knotsf is the number of interior knots.
  # w are the adaptive weights.
  # lambda is the grid of tuning parameters, and must be
      provided.
  # power is the exponent for group size to balance for
      different group sizes.

  error <- matrix(NA, nrow = K, ncol = 100)
  n = nrow(X)
  antf <- floor(n / K)

  if(n == (K * antf)) {
    set.seed(1)
    fold <- sample(rep(1:K, antf))
  }
  else {
    fold <- sample(c(rep(1:K, antf), 1:(n - (K * antf))))
  }

  for(k in 1:K){
    test.ind <- which(fold == k)
    trainX <- X[-test.ind, ]
    trainZ = Z[-test.ind, ]
    testX <- X[test.ind, ]
    testZ = Z[test.ind, ]
    trainY <- Y[-test.ind]
    testy <- Y[test.ind]
    if(method == "part.lin") {
      cv.fit <- part.lin.ms(trainX, trainZ, trainY, "
         gaussian", num.knots, lambda, power)
      Ztest <- monotone.splines(testZ, num.knots)
    }
    else if(method == "adaptive.part.lin") {
      cv.fit = ad.part.lin.ms(trainX, trainZ, trainY, "
         gaussian", num.knots, w, lambda, power)
      Ztest <- monotone.splines(testZ, num.knots)
    }
    error.k <- (testy - predict(cv.fit, newx = cbind(Ztest
       , testX)))^2
    error[k, ] <- colSums(error.k)
  }
  MSE <- colMeans(error)
  return(MSE)
}

library(scoop)

non.monotone <- NULL # Vector to store indicies of non-
   monotonic fits.
num.knots <- 6

# Partially linear fit with monotone splines lasso,
# x1 is the matrix with linear effects, x2 is the matrix
   with non-linear effects.
fit.part.lin <- part.lin.ms(x1, x2, y, "gaussian", num.
   knotsf = num.knots)  # Partially linear fit.
lambda.seq.part.lin <- fit.part.lin@lambda  # Extract
   lambda sequence used.
cv.part.lin <- cv.ms.part.lin(x1, x2, y, K = 10,
            method = "part.lin", num.knots = num.knots,
            lambda = lambda.seq.part.lin)  #
               Crossvalidation error for these lambdas.
coef.part.lin <- fit.part.lin@coefficients
# Extract coefficients with smallest prediction error
```

```
coef.part.lin <- coef.part.lin[cv.part.lin == min(cv.part.
   lin), ]

weights.adaptive <- rep(10000000, p) #Vector with adaptive
    weights

counter = 1
for (i in 1:d2){
  #Check if the coefficients for each non-linear p are
     zero.
    if(sum(coef.part.lin[counter:(counter + num.knots + 1)
       ]) != 0){
      if(abs(sum(coef.part.lin[counter:(counter+num.knots
          + 1)])) !=
      sum(abs(coef.part.lin[counter:(counter + num.knots +
          1)]))) {
        non.monotone <- c(non.monotone, c(k, i))
      }
      if(i == 1) {
        tp1[k] <- 1
      }
      else if(i == 2) {
        tp2[k] <- 1
      }
      else if(i == 3) {
        tp3[k] <- 1
      }
      else if(i == 4) {
        tp4[k] <-  1
      }
      else {
        fp[k] <- fp[k] + 1
      }
      weights.adaptive[i] <- 1 / sqrt(sum(coef.part.lin
      [counter:(counter + num.knots + 1)]^2))
    }
  counter <- counter + num.knots + 2 #Indices for next p
}

# Linear parameters.
beta1[k] <- coef.part.lin[counter]
beta2[k] <- coef.part.lin[counter + 1]
beta3[k] <- coef.part.lin[counter + 2]
```

```
beta4[k] <- coef.part.lin[counter + 3]

#Store indicators for the linear variables selected.
for (i in counter:(counter + d1 - 1)){
  if(coef.part.lin[i] != 0) {
    if(i == counter) {
      tp1.lin[k] <- 1
    }
    else if(i == (counter + 1)) {
      tp2.lin[k] <- 1
    }
    else if(i == (counter + 2)) {
      tp3.lin[k] <- 1
    }
    else if(i == (counter + 3)) {
      tp4.lin[k] <- 1
    }
    else {
      fp.lin[k] <- fp.lin[k] + 1
    }
    weights.adaptive[d2 + i - counter + 1] <- 1 / abs(coef
       .part.lin[i])
  }
}

fit.ad.part.lin <- ad.part.lin.ms(x1, x2, y,"gaussian",
   num.knotsf = num.knots,
                w = weights.adaptive)  # Adaptive
                   partially linear fit.
lambda.seq.ad.part.lin <- fit.ad.part.lin@lambda  #
   Extract lambda values.
cv.ad.part.lin <- cv.ms.part.lin(x1, x2, y, K = 10, method
    = "adaptive.part.lin",
               num.knots = num.knots, w = weights.adaptive
                  ,
               lambda = lambda.seq.ad.part.lin)  # Cross-
                  validation error for these lambdas.
coef.ad.part.lin <- fit.ad.part.lin@coefficients
# Extract coefficients with smallest prediction error.
coef.ad.part.lin <- coef.ad.part.lin[cv.ad.part.lin == min
   (cv.ad.part.lin), ]
```

H R-code for PLAMM-2 and APLAMM-2

This is an R-script with methods (functions) for PLAMM-2 and APLAMM-2. The methods in monotone splines lasso are called without definitions. This is part of the simulation experiments with four true non-linear covariates and four true linear covariates, see section 6.3.3.

```
library(scoop)
library(glmnet)

non.monotonic <- NULL  # Vector to store indices of non-
   monotonic fits.

num.knots = 6  # Number of interior knots.

# x1 is the matrix with the linear covariates.
# x2 is the matrix with non-linear covariates.

tol <- 10^(-4)  # Convergence criterion.
beta.old <- rep(0, d1)  # Beta from last step, first guess
   .
beta.new <- NULL  # Beta in current step.
coef.old <- NULL  # Spline coefficients from last step.
coef.new <- NULL  # Spline coefficients in current step.
y.now <- y - x1%*%beta.old  # What is to be explained by
   the splines in the first run.
cont1 <- TRUE  # Variable to indicate whether or not we
   have convergence in the betas. "cont" for continue.
cont2 <- TRUE  # Variable to indicate whether or not we
   have convergence in the spline coeffiencts.
stop <- FALSE  # Variable to tell us whether or not to
   stop.
iter <- 0  # Number of iterations.

while(!stop) {
  # This while loop continues until we have convergence in
      both parts.
  iter <- iter + 1
  print(c("iter", iter))
  # Monotone fit with currently best guess for beta
  fit.ms <- monotone.lasso(x2, y.now, "gaussian", num.
     knotsf = num.knots)
  lambda.seq.ms <- fit.ms@lambda
  cv.ms <- cvmslasso(x2, y.now, K = 10, method = "monotone
     .lasso",
  num.knots = num.knots, lambda = lambda.seq.ms)
  coef.ms <- fit.ms@coefficients
  coef.ms <- coef.ms[cv.ms == min(cv.ms), ]
  coef.new <- coef.ms
  if (!is.null(coef.old)) {
    if (sum((coef.new - coef.old)^2) < tol) {  # Check for
        convergence.
      cont2 <- FALSE  # Convergence in non-linear part.
      if (!cont1){  # If convergence in both parts, stop.
        stop <- TRUE
      }
    }
    else {
      cont2 <- TRUE
    }
}

  if (!stop) {
    if(sum(cv.ms == min(cv.ms)) == 1) {  # Unique
        estimated tuning parameter.
      ms.fitted <- fitted(fit.ms)[, cv.ms == min(cv.ms)]
         # Extract fitted y from spline part.
    }
    else {
      ms.fitted <- 0
    }
    y.now <- y - ms.fitted  # Observations to be explained
        by the linear terms.
    cv.lasso <- cv.glmnet(x1, y.now, intercept = FALSE,
               standardize = TRUE, alpha = 1)  # Fit the
                  linear part.
    lambda.min.lasso <- cv.lasso$lambda.min
    beta.new <- coef(cv.lasso, s = lambda.min.lasso)
    beta.new <- as.matrix(beta.new)
  }
  beta.new <- beta.new[2:(d1 + 1)]  # Remove intercept.
  if (sum((beta.new-beta.old)^2) < tol) {  # Check for
     convergence.
    cont1 <- FALSE  # Convergence in the linear parameters
        .
    if(!cont2) {  # If convergence in both parts, stop.
```

```
      stop <- TRUE
    }
  }
  else{
    cont1 <- TRUE
  }
  beta.old <- beta.new  # Update beta.
  coef.old <- coef.new  # Update spline coefficients.
  y.now <- y - x1%*%beta.old  # Update y to be explained
     by the splines.
}

weights.lin <- rep(100000, d1)  # Adaptive weights for the
    linear terms.
# Count true and false linear terms.
if(beta.new[1] != 0) {
  weights.lin[1] <- 1 / abs(beta.new[1])
  tp1.lin[k] <- 1
}
if(beta.new[2] != 0) {
  tp2.lin[k] <- 1
  weights.lin[2] <- 1 / abs(beta.new[2])
}
if(beta.new[3] != 0) {
  tp3.lin[k] <- 1
  weights.lin[3] <- 1 / abs(beta.new[3])
}
if(beta.new[4] != 0) {
  tp4.lin[k] <- 1
  weights.lin[4] <- 1 / abs(beta.new[4])
}
for (i in 5:d1) {
  if(beta.new[i] != 0) {
    fp.lin[k] <- fp.lin[k] + 1
    weights.lin[i] <- 1 / abs(beta.new[i])
  }
}

weights.nonlin <- rep(10000, d2)  # Adaptive non-linear
   weights.

# Count true and false covariates with non-linear effect.
counter <- 1
```

```
for (i in 1:d2){
  if(sum(coef.new[counter:(counter + 7)]) != 0) {
    if(abs(sum(coef.new[counter:(counter + 7)])) !=
    sum(abs(coef.new[counter:(counter + 7)]))) {
      #Non-monotone fit.
      non.monotonic = c(non.monotonic, c(k, i))
    }
    if(i == 1) {
      tp1.ms[k] <- 1
    }
    else if(i == 2) {
      tp2.ms[k] <- 1
    }
    else if(i == 3) {
      tp3.ms[k] <- 1
    }
    else if(i == 4) {
      tp4.ms[k] <- 1
    }
    else{
      fp.ms[k] <- fp.ms[k] + 1
    }
    weights.nonlin[i] <- 1 / sqrt(sum(coef.new[counter:(
       counter + 7)]^2))
  }
  counter <- counter + 8
}


tol <- 10^(-8)  # Convergence criterion.
coef.old.ad <- NULL  # Spline coefficients from last step.
coef.new.ad <- NULL  # Spline coefficients in current step
   .
beta.old.ad <- beta.new
beta.new.ad <- beta.new
y.now <- y - x1%*%beta.old.ad  # What is to be explained
   by the splines in the first run.
cont1 <- TRUE  # Variable to indicate whether or not we
   have convergence in the betas. "cont" for continue.
cont2 <- TRUE  # Variable to indicate whether or not we
   have convergence in the spline coeffiencts.
stop <- FALSE  # Variable to tell us whether or not to
   stop.
```

```
iter <- 0  # Number of iterations
while(!stop) {
  # This while loop continues until we have convergence in
      both parts.
  iter <- iter + 1
  print(c("iter", iter))
  # Monotone fit with currently best guess for beta.
  fit.ams <- adaptive.monotone(x2, y.now, "gaussian", num.
     knotsf = num.knots, w = weights.nonlin)
  lambda.seq.ad <- fit.ams@lambda
  cv.ams <- cvmslasso(x2, y.now, K = 10, method  = "
     adaptive.monotone", num.knotsf = num.knots,
  w = weights.nonlin, lambda = lambda.seq.ad)
  coef.ams <- fit.ams@coefficients
  coef.ams <- coef.ams[cv.ams == min(cv.ams), ]
  coef.new.ad <- coef.ams
  if (!is.null(coef.old.ad)) {
    if (sum((coef.new.ad - coef.old.ad)^2) < tol) {  #
       Check for convergence.
      cont2 <- FALSE  # Do not continue.
      if (!cont1) {  # If convergence in both parts, stop.
        stop <- TRUE
      }
    }
    else {
      cont2 <- TRUE
    }
  }

  if (!stop) {
    if(tp1.lin[k] == 0 & tp2.lin[k] == 0 & tp3.lin[k] == 0
    & tp4.lin[k] == 0 & fp.lin[k] == 0) {
      cont1 <- FALSE
    }
    else {
      if(sum(cv.ams == min(cv.ams)) == 1) {  # Unique
         penalty parameter.
        ams.fitted <- fitted(fit.ams)[, cv.ams == min(cv.
           ams)]  # Extract fitted y from spline part.
      }
      else {
        ams.fitted <- 0
      }
      y.now <- y - ams.fitted  # Observations to be
         explained by the linear terms.
      cv.alasso <- cv.glmnet(x1, y.now, intercept = FALSE,
          standardize = TRUE,
                alpha = 1, penalty.factor = weights.lin,
                exclude = which(beta.new == 0))  # Fit the
                    linear part.
      lambda.min.alasso <- cv.alasso$lambda.min
      beta.new.ad <- coef(cv.alasso, s = lambda.min.alasso
         )
      beta.new.ad <- as.matrix(beta.new.ad)
    }
  }
  beta.new.ad <- beta.new.ad[2:(d1 + 1)]  # Remove
     intercept.
  if(sum((beta.new.ad - beta.old.ad)^2) < tol){  # Check
     for convergence.
    cont1 <- FALSE  # Do not continue.
    if(!cont2) {  # If convergence in both parts, stop.
      stop <- TRUE
    }
  }
  else {
    cont1 <- TRUE
  }
  beta.old.ad <- beta.new.ad  # Update beta.
  coef.old.ad <- coef.new.ad  # Update spline coefficients
     .
  y.now <- y - x1%*%beta.old.ad  # Update y to be
     explained by the splines.
}
```