Nonparametric regression penalizing deviations from additivity

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Mathematics

presented by
MICHAEL MARKUS STUDER
Dipl. Math. ETH
born September 1, 1972
citizen of Maschwanden ZH

accepted on the recommendation of
Prof. Dr. H. R. Künsch, examiner
Prof. Dr. P. Bühlmann, co-examiner
Prof. Dr. T. Gasser, co-examiner
PD. Dr. B. Seifert, co-examiner

2002
Contents

1 Introduction .................................................. 6
   1.1 Nonparametric regression estimation ......................... 6
   1.2 Contents of the thesis ..................................... 9

2 General .................................................. 11
   2.1 Local linear regression ................................... 11
   2.2 Additive modeling ....................................... 13
   2.3 MLN estimator ........................................... 14
      2.3.1 On notation ........................................ 14
      2.3.2 A versatile function space .......................... 14
      2.3.3 Estimation in the additive model ..................... 15
      2.3.4 Backfitting algorithm ............................... 17
      2.3.5 Asymptotics for local polynomials ................... 17

3 Non–additivity penalized estimation ........................... 19
   3.1 Model and notation ...................................... 19
   3.2 Nadaraya–Watson with penalty ............................. 19
      3.2.1 Nadaraya–Watson ................................... 19
      3.2.2 Dimension reduction ................................ 21
      3.2.3 Interpretation ...................................... 22
      3.2.4 Additional remarks .................................. 22
   3.3 Generalizing to local polynomials .......................... 23
      3.3.1 Local polynomial regression .......................... 23
      3.3.2 Estimation of the regression function itself .......... 24
   3.4 Penalized local linear estimation ........................... 26
      3.4.1 Regularizing the nuisance parameters ................ 26
      3.4.2 Comments on convex combination ..................... 28
      3.4.3 On different penalties on intercept and slopes .......... 29
   3.5 Pairwise interaction terms ................................ 30
   3.6 Semiparametric modeling .................................. 30

4 Properties of the penalized estimator ....................... 32
   4.1 Comparison with MLN ..................................... 32
      4.1.1 Function spaces and norms .......................... 32
      4.1.2 Some operators ..................................... 33
      4.1.3 Projection to \(F_{\text{add}}\) .......................... 34
   4.2 Finite grid properties .................................... 35
      4.2.1 Specification of \(Z\) ................................ 35
      4.2.2 Specification of \(\tilde{Z}\) ............................. 37
      4.2.3 Illustration of grid dependence ....................... 37
   4.3 About parameters ........................................ 40
      4.3.1 Influence of the parameters on variance and bias .... 40
      4.3.2 Local alternatives to the additive model ............... 42
      4.3.3 On different penalties for intercept and slopes ........ 42

5 Numerical evaluation ....................................... 44
   5.1 Regression function ...................................... 44
   5.2 Design .................................................. 46
   5.3 Implementation .......................................... 48
      5.3.1 Output grid ......................................... 48
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.2</td>
<td>Bandwidths</td>
<td>48</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Extreme penalties</td>
<td>49</td>
</tr>
<tr>
<td>5.4</td>
<td>MISE</td>
<td>49</td>
</tr>
<tr>
<td>5.4.1</td>
<td>MISE for superposed peaks regression function and 200 design points</td>
<td>49</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Odd behavior of the bias for small bandwidths</td>
<td>52</td>
</tr>
<tr>
<td>5.5</td>
<td>AIC and ISE</td>
<td>53</td>
</tr>
<tr>
<td>5.5.1</td>
<td>ISE</td>
<td>53</td>
</tr>
<tr>
<td>5.5.2</td>
<td>AIC</td>
<td>53</td>
</tr>
<tr>
<td>5.5.3</td>
<td>AIC vs ISE for 200 observations</td>
<td>54</td>
</tr>
<tr>
<td>5.5.4</td>
<td>AIC vs ISE for 1600 observations</td>
<td>58</td>
</tr>
<tr>
<td>5.5.5</td>
<td>AIC vs ISE for 100 equidistant observations</td>
<td>58</td>
</tr>
<tr>
<td>5.5.6</td>
<td>AIC vs ISE for 104 rotated equidistant observations</td>
<td>59</td>
</tr>
<tr>
<td>5.5.7</td>
<td>Others</td>
<td>59</td>
</tr>
<tr>
<td>5.6</td>
<td>Conclusion</td>
<td>60</td>
</tr>
</tbody>
</table>

### A Remarks on MLN
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>Notation</td>
<td>61</td>
</tr>
<tr>
<td>A.2</td>
<td>Details about normal equations</td>
<td>61</td>
</tr>
<tr>
<td>A.3</td>
<td>Details of theorem MLN:4'</td>
<td>62</td>
</tr>
<tr>
<td>A.4</td>
<td>Assumptions in MLN</td>
<td>64</td>
</tr>
</tbody>
</table>

### B Proofs for section 3
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>Section 3.2.2</td>
<td>66</td>
</tr>
<tr>
<td>B.2</td>
<td>Section 3.2.3</td>
<td>67</td>
</tr>
<tr>
<td>B.3</td>
<td>Section 3.2.4</td>
<td>67</td>
</tr>
<tr>
<td>B.4</td>
<td>Section 3.3</td>
<td>68</td>
</tr>
</tbody>
</table>

### C Proofs for section 4.1
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1</td>
<td>Proof of lemma 4.1</td>
<td>69</td>
</tr>
<tr>
<td>C.1.1</td>
<td>Overview</td>
<td>69</td>
</tr>
<tr>
<td>C.1.2</td>
<td>Lemma MLN:1'</td>
<td>71</td>
</tr>
<tr>
<td>C.1.3</td>
<td>Lemma MLN:2'</td>
<td>72</td>
</tr>
<tr>
<td>C.2</td>
<td>Some remarks</td>
<td>76</td>
</tr>
<tr>
<td>C.2.1</td>
<td>Backfitting</td>
<td>76</td>
</tr>
<tr>
<td>C.2.2</td>
<td>Convex combination</td>
<td>77</td>
</tr>
<tr>
<td>C.2.3</td>
<td>Remark on lower bound and continuous inverse</td>
<td>78</td>
</tr>
<tr>
<td>C.3</td>
<td>Proof of lemma 4.2</td>
<td>78</td>
</tr>
<tr>
<td>C.4</td>
<td>Proof of theorem 4.3</td>
<td>79</td>
</tr>
</tbody>
</table>

### D Remarks and proofs for section 4.2
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1</td>
<td>Redundancy in $\mathbf{Z}$</td>
<td>80</td>
</tr>
<tr>
<td>D.2</td>
<td>Fixed design in a neighborhood of $[0, 1]^d$</td>
<td>80</td>
</tr>
<tr>
<td>D.3</td>
<td>Lower bound for $K^{*2}$</td>
<td>80</td>
</tr>
<tr>
<td>D.4</td>
<td>Diagonal elements</td>
<td>81</td>
</tr>
</tbody>
</table>

### E Proofs for section 4.3
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.1</td>
<td>Section 4.3.1</td>
<td>82</td>
</tr>
<tr>
<td>E.2</td>
<td>Section 4.3.3</td>
<td>82</td>
</tr>
</tbody>
</table>
### Contents

**F** S–plus and R programs
- F.1 Get maximal dimension ................................................................. 84
- F.2 Output Grid ........................................................................ 84
- F.3 Local bandwidths ................................................................. 84
- F.4 Normalizations ...................................................................... 86
- F.5 Local linear estimator with non–additivity penalty ................. 88
- F.6 GridISE .............................................................................. 92
- F.7 GridAIC ............................................................................. 97
  - F.7.1 GridAIC.ISE .................................................................. 98
- F.8 GridMISE .......................................................................... 99

**G** C program code
- G.1 local.h ............................................................................. 101
- G.2 Storage order of the observations ........................................ 101
- G.3 LlPoint ............................................................................ 101
  - G.3.1 Kernel weight function .................................................... 102
  - G.3.2 Reparameterization .......................................................... 103
- G.4 Matrix S ........................................................................... 103
  - G.4.1 Memory layout ................................................................. 103
  - G.4.2 Cholesky decomposition .................................................. 104
  - G.4.3 Solving equation systems by Cholesky decomposition ... 104
  - G.4.4 Multiplication ................................................................. 105
- G.5 LlGrid ............................................................................. 105
  - G.5.1 Outgrid ......................................................................... 105
  - G.5.2 Bandwidths ..................................................................... 105
  - G.5.3 Normalization ................................................................. 106
  - G.5.4 LlGrid ........................................................................... 106
- G.6 Multiplication by $\tilde{Z}$ ......................................................... 107
  - G.6.1 Additive components ....................................................... 107
  - G.6.2 Related stuff ................................................................... 108
  - G.6.3 Initialize matrix ............................................................... 110
- G.7 Penalized estimator .............................................................. 111
- G.8 MISE ............................................................................... 113
- G.9 AIC .................................................................................. 115
  - G.9.1 GridAIC ....................................................................... 116
- G.10 matalg ........................................................................... 117

**H** About figures
- H.1 Regression functions ............................................................ 118
- H.2 Figures ............................................................................. 120
Abstract

Nonparametric regression in higher dimensions suffers from the ‘curse of dimensionality’. The assumption of an arbitrary multivariate smooth regression function is not sufficiently specific for practice, as there exists no acceptable compromise between variability and oversmoothing. This is the reason for using simpler models, even if they do not hold. Additive models have more restrictive assumptions and are hence easier to estimate. As a drawback, there is an additional model–bias as compared to the full model. We are investigating a continuous compromise between these two models, containing the above mentioned extremes as special cases.

This is achieved via a local linear estimator with penalty on the deviation from the additive model. We evaluate the estimator on a grid of output points. The subset of all possible parameter values, corresponding to the additive model, is denoted as additive subspace. The simultaneous local linear estimator for all output points is the solution of a quadratic minimization problem with many variables but simple structure. A multiple (parameter $R$) of the squared Euclidean distance to the additive subspace is added as a penalty term to the minimization problem. The ordinary local linear estimator corresponds to the case $R = 0$. Formally, $R = \infty$ corresponds to a local linear estimator restricted to the additive model, and the resulting estimator is asymptotically optimal if the grid is replaced by a continuum (Mammen, Linton, and Nielsen, 1999). The estimator with penalty on the non–additive component is continuous in $R$ — for $R = 0$ this applies only to the output points where the local linear estimator is unique. Accordingly, we have a parametric class of estimators, whose extremes are competitive for the corresponding classes of regression functions.

This indicates that the new method has the potential to improve the original estimators. The estimator was applied to simulated data for illustration. A simple parameter selection criterion is the improved Akaike information criterion (AIC) proposed by Hurvich, Simonoff, and Tsai (1998). The optimal parameters were in general neither 0 nor $\infty$. Extremes were detected by AIC, hence nothing was lost in these cases.

The special structure of the minimization problem allows reducing matrix size. This is necessary for computation. Furthermore, the conversion improves our understanding.

An attractive property of the estimator with penalty on the non–additive components is that the estimation is a data adaptive continuous compromise between multivariate local linear and additive estimation.

On the one hand, the local linear estimator has good asymptotic properties and is easily transcribed to higher dimensions. On the other hand, regularization is required in practice. A byproduct of penalizing is that the estimator is regularized even for small penalties.

Keywords: nonparametric estimation, additive models, shrinkage, model choice, curse of dimensionality, regularization, optimal bandwidth.
Zusammenfassung


Die spezielle Struktur des Minimierungsproblems erlaubt eine Reduktion der Größen der Matrizen, was angesichts der grossen Anzahl Parameter für die rechenotechnische Umsetzung notwendig ist. Diese Umformung führt zu einem besseren Verständnis der theoretischen Eigenschaften.


Der lokal lineare Schätzer hat einerseits gute asymptotische Eigenschaften und ist einfach auf höhere Dimensionen übertragbar. Andererseits ist in der Praxis eine Regularisierung notwendig. Ein Nebenprodukt ist die Regularisierung des Schätzers schon für kleine Bestrafungsterme.
1 Introduction

1.1 Nonparametric regression estimation

One dimensional regression

To model the dependence of an outcome variable $Y$ on the explanatory variable $X$ the following regression model is often postulated:

$$Y_i = r(X_i, \phi) + \varepsilon_i \quad (i = 1, \ldots, n)$$

where $r$ is the regression function and $\varepsilon_i$ are residuals with $E(\varepsilon_i) = 0$ and $\text{var}(\varepsilon_i) = \sigma^2$. In classical statistics knowledge of a parametric regression function with parameter vector $\phi$ is assumed, and only $\phi$ has to be estimated. Often such a parametric model cannot be postulated and for this case, non–parametric methods have been introduced in the last 30 years (see e.g. Fan and Gijbels, 1996). The only assumption about the regression function is that it is smooth. Popular methods include smoothing splines, kernel estimators and local polynomials.

The essential elements of these estimators are the weight function $K$ and the smoothing parameter $h$ (“bandwidth”), and the latter is crucial. For a small $h$ we obtain an undersmoothed estimate with a large variance, for a large $h$ an oversmoothed function with a large bias. Squared bias is asymptotically proportional to $h^4$ and variance to $(nh)^{-1}$. The optimal smoothing parameter is a compromise between squared bias and variance. It leads to an asymptotic rate of the mean squared error of $n^{-4/5}$, as compared with the better parametric rate $n^{-1}$.

Fan (1993) showed that the local linear method has attractive asymptotic optimality properties. It has also great flexibility in its use. Hidden in his arguments are some problems with the variance of the estimator. Since the variance can become unbounded in sparse regions, Seifert and Gasser (1996) proposed regularizations of the estimator.

Multidimensional regression

In principle, the non–parametric estimators can be generalized to more than one explanatory variable:

$$Y_i = r(X_{i,1}, \ldots, X_{i,d}) + \varepsilon_i$$

This generalization is particularly easy for local polynomials (see e.g. Fan, Gasser, Gijbels, Brockmann and Engel, 1997). Note that basically the bias depends on the diameter of the smoothing window and the variance depends on the number of observations in the smoothing window, i.e. its volume multiplied by $n$.

This is the point where problems arise: if the diameter of a ball on $d$ dimensions is divided by 2 then the volume is divided by $2^d$. Consequently, the bias might be reduced only at the cost of high increase in variability. Hence, the method works, but the benefit of larger sample size is reduced for increasing number of dimensions $d$: rate of convergence is $n^{-4/(4+d)}$.

Another point is that if the output points are uniformly spread over the support of the design, then large proportions of them are in a neighborhood of the boundary. These issues are summarized under the heading “curse of dimensionality”.

Example

The following example (figure 1) uses simulated data from Seifert and Gasser (2000), figure 6. The data were originally used to illustrate the usefulness of local ridging. The sample size is 200 and the range $([0.2, 1.1])$ of the values of the true regression function is about nine times as large as the residual standard deviation ($\sigma = 0.1$). Figure 2 displays two local linear estimations and illustrates the trade–off between variance and bias. Kernel weights are product Epanechnikov.
1 INTRODUCTION

1.1 Nonparametric regression estimation

![Figure 1: True regression function and design.](image1)

![Figure 2: Left plot uses ISE-optimal bandwidth $h = 0.174$. (Bandwidth is enlarged at the boundaries.) The right plot is wiggly due to the relatively small bandwidth $h = 0.117$. For bandwidths below 0.112, the estimator is not unique everywhere.](image2)

The bandwidth was enlarged at the boundary (see section 5.3.2) for the following reason: the number of observations within a certain distance from the output point is smaller at the boundary than in the interior. Without the boundary adjustments of the bandwidth, we would need larger bandwidths — $h > 0.21$ is required for uniqueness — but the left plot in figure 2 is already oversmoothed for $h = 0.174$.

ISE (= integrated squared error) is obtained by calculating the squared difference between the estimated and the true values and averaging them over the output grid. For convenience the result is multiplied by 2500.

**Additive models**

In order to avoid the curse of dimensionality, Stone (1985) proposed additive models:

$$ r(X_{i,1}, \ldots, X_{i,d}) = \sum_{k=1}^{d} r_i(X_{i,k}) . $$

This leads to a great improvement in the rate of convergence, i.e. to the one–dimensional rate $n^{-4/5}$. On the other hand a price has to be paid in flexibility when going down from the general full model defined in (1) to the more restrictive additive model.

Less restrictive models have been proposed by allowing certain interaction terms of the form $r_{kl}(x_k, x_l)$ into the model. In this thesis, a new approach is pursued by allowing a continuous compromise between the full and the additive model.
Non–additivity penalized estimation

A local linear estimator evaluated on a grid is regularized with a penalty term proportional to the squared deviation from additivity. This allows a smooth tuning between a full model and an additive model. The respective penalty factor is denoted by $R$, $R = 0$ corresponding to the full model, and $R = \infty$ corresponding to the additive model.

\[ \text{llR; } h = 0.117, \ R = 0.163, \ \text{ISE}=6.0 \]

\[ \text{llR; } h = 0.117, \ R = \infty, \ \text{ISE}=8.8 \]

Figure 3: On the left, the penalized estimator with parameters $(h = 0.117, R = 0.163)$ is displayed. On the right, an additive estimator with the same bandwidth is plotted.

As the true regression function (figure 1) is not additive, a small penalty $R = 0.163$ together with the bandwidth $h = 0.117$ is used. The resulting estimator (figure 3) has a much better ISE than the best ordinary local linear estimator (figure 2, left plot). The penalty $R$ has the following effect: variability is reduced while the peaks are preserved. Because of the smaller bandwidth, the peaks are not oversmoothed.

A general property of the penalized estimator with a small $R$ is: In regions, where the local linear estimator is not defined, an additive estimator is used instead. In well defined regions, the local linear estimator remains nearly unchanged. Figure 4 indicates at which output points the ordinary local linear estimator is stable. In this example, the estimator works because the features (peaks) are substantiated by data and the additive approximation (figure 3, right plot) is tolerable at instable output points. Note that penalizing also leads to an improvement if we rotate the design by ±90 degrees.

Figure 4: Left plot displays stability (gray scale, $h = 0.117$, bright=unstable) and true regression function (contour). Right plot shows design and smoothing windows for $h = 0.117$ (solid line) and $h = 0.174$ (dotted line) are displayed for output points (0,0) and (0.55,0.55). The bandwidth is enlarged at the boundary.
Other approaches for regularization

The problem with unstable local linear estimators is discussed in Seifert and Gasser (1996) for univariate functions. They propose two sorts of local regularizations — besides detecting numerical instability: First over-smoothing (i.e. locally increasing the bandwidth) results in a more favorable variance–bias compromise. One example is a $k$–nearest neighbor rule (Devroye, 1978), where the smoothing window contains $k$ observations. Second, ridge regression is used to shrink the local linear estimator towards the more stable Nadaraya–Watson estimator. This method is refined in Seifert and Gasser (2000) and works also for multivariate regression functions. Asymptotically, no regularizations of the local linear estimator are necessary and hence, regularization may be chosen to be asymptotically negligible.

The above mentioned methods are local regularizations as the estimation is based only on a neighborhood of the output point. Other approaches combine the estimates for different output–points. This leads to higher memory usage. For example, if the estimation for one output point depends on its neighbors, which themselves depend on their neighbors, then we have to calculate all output points simultaneously. Examples for global regularizations are below.

Typically, we have some idea about the shape of a regression function. This information is used as a constraint on the estimate, e.g. monotonicity for univariate functions — see Mammen, Marron, Turlach and Wand (2001). Another well–known example is additive modeling — see Mammen, Linton and Nielsen (1999) for a representation as multivariate local linear estimator with constraints.

Alternatively, we probably want some smooth estimate: For univariate regression functions, penalties on roughness are already known from smoothing splines, where a least squares fit is regularized with a penalty on integrated squared derivatives. For regression splines, Eilers and Marx (1996) proposed penalties on differences on coefficients of adjacent B–Splines and called it P–Splines. Local polynomials with penalties on roughness are also feasible.

1.2 Contents of the thesis

The thesis is organized as follows: An overview of related topics is given in section 2; i.e. local polynomial regression and additive modeling. In particular, a recent publication of Mammen et al. (1999) on estimation in additive models is sketched in section 2.3, because some of their results are used later.

In section 3, the idea of a local polynomial estimator with penalty on non–additivity is introduced. For simplicity, the idea is first applied to the Nadaraya–Watson estimator. Algebraic transformations are introduced to reduce memory consumption.

The parameters of local linear approximation may be divided into parameters of interest (e.g. intercept) and nuisance parameters (e.g. slopes). The question arises whether only the parameter of interest or also the nuisance parameters should be regularized: Local linear estimators with penalty on the parameter of interest and on all parameters are considered in sections 3.3 and 3.4, respectively. It will be seen that only with regularization of the nuisance parameters, the estimator gets rid of the curse of dimensionality for additive regression. In section 3.6, the penalized estimator is applied to semiparametric models.

Statistical properties are investigated in section 4. The estimator depends on the choice of the output grid. In section 4.1 we get rid of this grid–dependence by transferring the estimator to a function space, using techniques described in Mammen et al. (1999). The local linear estimator with a small non–additivity penalty may be used to stabilize local linear estimators of multivariate regression functions. For data originating from an additive regression function and a penalty $R$ that converges to infinity, the limit of this penalized estimator is asymptotically optimal in the additive model. Section 4.2 illustrates, that the number of output–points should grow as the bandwidth decreases — unless the penalty is chosen to dwindle asymptotically.

Section 5 contains finite sample evaluations. First, we investigate whether and by how
much improvements over the additive and the ordinary local linear estimator are possible using our estimator. As to be expected this is the case and the gains can be substantial. Even more important is whether finding the optimal parameters is feasible. As a simple parameter selection criterion, the corrected Akaike information criterion (Hurvich, Simonoff and Tsai, 1998) is evaluated (figure 5). In the examined cases, AIC correctly chooses $R = \infty$ or 0 if the regression function is additive or orthogonal to additive, respectively.

![Figure 5](image)

Figure 5: The left plot displays the ISE (a goodness of fit criterion, page 7) as a function of bandwidth and penalty parameter. A contour line bounds the region of parameters outperforming the ordinary local linear estimator. As this region is relatively large, data driven bandwidth selection leads to an improvement. The ISE–optimal parameters were used in the left plots of figures 2 and 3. The right plot displays AIC. In this example AIC works perfectly. Bandwidth is plotted in log–scale and penalty parameter is in $\frac{R}{1+R}$ scale.

2 Local polynomial and additive estimation

2.1 Local linear regression

Non-parametric regression

Local linear regression is one approach of non-parametric regression. Other popular methods are smoothing splines, Gasser–Müller estimator, orthogonal series, etc. The name non-parametric is used because the complexity of the fitted model may increase with the number of data. Essentially, all methods are weighted local averages of the data. The particular weight function or kernel is not crucial for the quality of the estimator. Furthermore, analytical solutions with good statistical properties are available. The smoothing parameter(s) or bandwidth(s) are important since they determine the extent of local averaging and, as a consequence, the complexity of the fitted model. Given a criterion for *goodness of fit*, we have to find the optimal smoothing parameter(s). For the *mean integrated squared error* (MISE) criterion, finding the optimum is a *variance-bias compromise* for all common methods.

A classical approach to bandwidth estimation is cross-validation. A more recent one is the *plug-in* method. The latter is based on estimating the asymptotically optimal bandwidths, which leads to better performance in terms of variance.

Stone (1980) investigated the *upper bound to the rate of convergence*, if we assume the true regression function to be in some class Θ. The optimal rate is \( n^{-(p-m)/(2p+d)} \), where \( n \) is the number of observations, \( d \) is the dimension, \( p \) the order of smoothness, and \( m \) the maximal order of derivative to be estimated. More exactly, let \( k \) be some integer smaller than \( p \). The class Θ consists of those functions \( g \) where the difference of \( g(x) \) and its Taylor approximation in the output point \( t \) up to order \( k \) is of order \( \|x - t\|^p \), where \( \|\cdot\| \) denotes the Euclidean norm.

Local polynomials

Local polynomials are weighted least-squares fits of a Taylor approximation to the regression function. The term *local* indicates that the weights are either zero or small outside a neighborhood of the output point. Stone (1980) proves that local polynomials achieve the optimal rate given above. This point-wise result is supplemented in Stone (1982) for \( L_p \)-norms.

A classical local polynomial estimator is the *Nadaraya–Watson* estimator, which uses a local constant approximation. Asymptotically it is outperformed by the local linear estimator.

Optimality of the one-dimensional local linear estimator

As mentioned above, the local linear estimator achieves the optimal rate of convergence. A nice property is, that the local linear estimator adjusts automatically at the boundary. In a breakthrough paper, Fan (1993) proved the following result: The local linear estimator is *asymptotically minimax optimal* with respect to MISE by achieving optimal constants among all *linear* estimators, under usual regularity conditions. The Epanechnikov kernel \( K(x) = 0.75 \max(1 - x^2, 0) \) is asymptotically optimal.

For finite sample sizes, problems arise for variance in sparse regions — see Seifert and Gasser (1996). This problem is also reflected by the fact that Fan (1993) had to introduce an asymptotically vanishing term \( n^{-2} \) into the estimator to ensure stability. For some output point \( t \) and a bandwidth \( h \), let \( K_i = K((X_i - t)/h) \) be the kernel weight for the observation \((Y_i, X_i)\). The estimator of Fan (1993) is:

\[
\sum_{i=1}^{n} \frac{w_i}{\sum_{k=1}^{n} w_k + n^{-2}} Y_i
\]

with weights \( w_i \) defined as

\[
w_i = \left( \sum_{k=1}^{n} K_k (X_k - t)^2 - (X_i - t) \sum_{k=1}^{n} K_k (X_k - t) \right) K_i.
\]
2.1 Local linear regression

Local linear estimator for dimensions larger than one

Let \( t = (t_1, \ldots, t_d)^\top \) denote some output point and \( K_i \) the kernel weight for the observation \( (Y_i, \textbf{X}_i) \). The components of \( \textbf{X}_i \) are \( X_{i,1}, \ldots, X_{i,d} \), for \( i = 1, \ldots, n \). The local linear estimator is defined as

\[
\hat{r}_{ll}(t) = \arg \min_{\beta_0} \min_{\beta_1, \ldots, \beta_d} \sum_{i=1}^n K_i \left( Y_i - \beta_0 - \sum_{k=1}^d \beta_k(X_{i,k} - t_k) \right)^2.
\]

More precisely, \( K_i \) is a rescaled kernel function applied to \( \textbf{X}_i - t \).

According to Fan et al. (1997), the mean squared error optimal kernel (in the interior) is the spherical Epanechnikov kernel, whose support is the unit circle:

\[
K(\mathbf{x}) = \frac{d(d+2)}{4\pi^{d/2} \Gamma(d/2)} \max(1 - \|\mathbf{x}\|^2, 0).
\]

When choosing bandwidths, we may consider three different classes: a scalar, a diagonal matrix, or a general matrix. The smoothing windows have the shape of a circle, an ellipsis with fixed axis, or a general ellipsis, respectively. Wand and Jones (1993) investigate the accuracy of these three types of bandwidths in the context of density estimation. They conclude that a diagonal bandwidth is in general adequate. Hence, we choose a diagonal bandwidth matrix

\[
H = \text{diag}(h_1, \ldots, h_d) = \begin{bmatrix}
    h_1 & 0 & \cdots & 0 \\
    0 & h_2 & \cdots & 0 \\
    \vdots & \ddots & \ddots & \vdots \\
    0 & \cdots & 0 & h_d
\end{bmatrix}
\]

and define \( K_i = (\prod_{k=1}^d h_k)^{-1} K(H^{-1}(\textbf{X}_i - t)) \).

Asymptotic bias and variance of the local linear estimator are given in Ruppert and Wand (1994): Assume that, for increasing \( n \), the bandwidths \( h_1, \ldots, h_d \) tend to zero and \( n \prod_{k=1}^d h_k \) tends to infinity. Furthermore \( h_1, \ldots, h_d \) have the same rate, i.e. \( h_k/h_1 \) is bounded away from zero and infinity. Assume that the output point \( t \) is in the interior of the support of the design density \( f \). At \( \mathbf{x} = t \), \( f(\mathbf{x}) \) is strictly greater than zero and continuously differentiable, the variance \( \sigma^2(\mathbf{x}) = \text{var}(Y|\mathbf{X} = \mathbf{x}) \) is strictly greater than zero and continuous, and the true regression function \( r(\mathbf{x}) = \mathbb{E}(Y|\mathbf{X} = \mathbf{x}) \) is twice continuously differentiable.

The second moment of the kernel \( \mu_2(K) \) is defined as

\[
\int \mathbf{x} \mathbf{x}^\top K(\mathbf{x}) d\mathbf{x} = \mu_2(K) I.
\]

The integrated squared kernel function \( R(K) \) is

\[
R(K) = \int \left( K(\mathbf{x}) \right)^2 d\mathbf{x}.
\]

Then the variance and bias of the local linear estimator are

\[
\text{bias(}r_{ll}(t)) = \frac{1}{2} \mu_2(K) \sum_{k=1}^d h_k^2 \left( \frac{\partial}{\partial t_k} \right)^2 r(t) + O_P(h_k^2)
\]

\[
\text{var(}r_{ll}(t)) = \frac{\sigma^2(t) R(K)}{n \prod_{k=1}^d h_k} \{ 1 + O_P(1) \}.
\]

Hence, the mean squared error is of order \( h_k^4 + (nh_k^4)^{-1} \), and the asymptotically optimal bandwidth is of order \( n^{-1/(4+d)} \). For general bandwidth matrices, the exact value is given in Fan et al. (1997) on page 87.

The multivariate local linear regression estimator possesses the same minimax properties as the univariate one, as shown in Fan et al. (1997) on page 92.
2.2 Additive modeling

Besides the problems with unbounded variance for random design, the local linear estimator (as well as other non-parametric estimators) suffers in high dimensions from the curse of dimensionality as indicated by the optimal rate of convergence, see section 2.1. To reduce model complexity, additive models are introduced: The unknown multivariate regression function is assumed to be a sum of unknown univariate regression functions:

\[ r(x) = \sum_{k=1}^{d} r_k(x_k). \]

The additive model may also be generalized, using a known link function, see Hastie and Tibshirani (1990).

**Optimal rate**

Assume that the design is rectangular, then the **optimal rate of convergence** in the additive model is the same as for univariate non-parametric regression. This is shown in Stone (1985, 1986), by demonstrating that a spline-based estimator achieves this rate. When using a backfitting estimator, i.e. an iterative algorithm using univariate smoothers, this is intuitively plausible, provided the algorithm converges.

If all except one additive components were known, estimating the remaining component would be a univariate non-parametric regression problem: optimal rate and even constants are known. An estimator using this additional information is named **oracle estimator** and may be used as a benchmark.

**Backfitting algorithm**

Buja, Hastie and Tibshirani (1989) investigated the convergence of backfitting algorithms, i.e. a Gauss-Seidel scheme for solving a set of normal equations. They considered linear smoothers. If the smoother matrices for the univariate estimators are symmetric with eigenvalues in \([0, 1]\), the backfitting algorithm converges to some solution of the normal equations. This condition holds for cubic smoothing splines and for regression splines. For local linear estimators, the singular values of the smoother matrix may be larger than one.

Opsomer and Ruppert (1997) investigated a bivariate backfitting algorithm. However, their proof required strong conditions about the design. While they achieved the asymptotically oracle variance, their bias was design dependent and therefore not oracle optimal.

The **marginal integration** estimator of Linton and Nielsen (1995) is generally not efficient. However it may be used as a pilot estimator — undersmoothing to reduce bias — to which a backfitting step is applied. According to Linton (1997), this resulting estimator is asymptotically oracle optimal.

**Projections**

The breakthrough was due to Mammen et al. (1999). The key idea was to use a **projection interpretation** for multivariate local linear estimators and to **introduce additivity constraints for the parameters**. The normal equations lead to a backfitting procedure that keeps track of intercept and slope during the iteration. For details see section 2.3.

In contrast to Buja et al. (1989), this backfitting algorithm is not an iteration of scatter-plot smoothers. Under general conditions, the backfitting steps may be considered as approximations of iterated projections, and one backfitting cycle corresponds to a shrinkage operator. Hence, convergence is guaranteed. Furthermore, the estimator **achieves asymptotically oracle optimality**.
2.3 Existence and asymptotic properties of a backfitting projection algorithm under weak conditions

The work of Mammen et al. (1999) is sketched here, because of its relevance for additive modeling and because the main results apply to the penalized estimator derived in this thesis. While they considered both local linear and local constant (Nadaraya–Watson) modeling, I outline only the local linear case.

2.3.1 On notation

Theorems, assumptions, and equations with a label starting with ‘MLN’ refer to corresponding items in Mammen et al. (1999). Sometimes, I use different variable names than MLN. In appendix A.1 a table with differences in notation is given.

2.3.2 A versatile function space

First consider the local linear estimator \( \hat{\beta}_{ll} \) given a bandwidth \( h \), data \((Y_i, X_{i,1}, \ldots, X_{i,d})_{i=1,\ldots,n}\), and a product kernel \( K_h\left(X_i, \cdot \right) = \prod_{k=1}^{d} K_{h}(X_{i,k}, x_k) \). As in MLN, we assume a scalar bandwidth \( h \). This assumption may be relaxed to diagonal bandwidths.

\[
\hat{\beta}_{ll}(x) = \arg \min_{\beta_0, \ldots, \beta_d} \frac{1}{n} \sum_{i=1}^{n} \left( Y_i - \beta_0 - \sum_{k=1}^{d} \beta_k \left( \frac{X_{i,k} - x_k}{h} \right) \right)^2 K_h(X_i, x) \quad \text{(MLN:1)}
\]

Now, function spaces \( \mathcal{F} \) and \( \mathcal{F}_{\text{full}} \) are introduced, such that

- \( Y_1, \ldots, Y_n \) are represented by \( r_Y \in \mathcal{F} \)
- All possible parameterizations \( \beta(\mathbf{x}) = (\beta_0(\mathbf{x}), \ldots, \beta_d(\mathbf{x})) \) form a subspace \( \mathcal{F}_{\text{full}} \) of \( \mathcal{F} \).
- (MLN:1) may be written as \( \hat{\beta}_{ll} = \arg \min_{r \in \mathcal{F}_{\text{full}}} \| r - r_Y \|_s^2 \) for some (semi-)norm \( \| \cdot \|_s \).

**Function spaces**

Define

\[
\mathcal{F} = \left\{ (r^{i,\ell}| i = 1, \ldots, n; \ell = 0, \ldots, d) \mid r^{i,\ell} \text{ is a function } \mathbb{R}^d \to \mathbb{R} \right\}.
\]

Define \( r_Y \in \mathcal{F} \) by

\[
r_Y^{i,\ell}(\mathbf{x}) = \begin{cases} Y_i & \ell = 0 \text{ (independent of } \mathbf{x} ) \\ 0 & \text{else} \end{cases}
\]

Because \( \beta \) in (MLN:1) does not depend on \( i \), we define

\[
\mathcal{F}_{\text{full}} = \left\{ (r^{0,\ldots,d}) \mid r^{\ell} \text{ is a function } \mathbb{R}^d \to \mathbb{R} \right\}.
\]

We embed \( r \in \mathcal{F}_{\text{full}} \) into \( \mathcal{F} \) by

\[
r^{i,\ell}(\mathbf{x}) = r^{\ell}(\mathbf{x}) \quad \forall i = 1, \ldots, n; \ell = 0, \ldots, d; \text{ and } (r^{i,\ell}) \in \mathcal{F}.
\]

For additive modeling, we need some subspace \( \mathcal{F}_{\text{add}} \subset \mathcal{F}_{\text{full}} \)

\[
\mathcal{F}_{\text{add}} = \left\{ r \in \mathcal{F}_{\text{full}} \mid r^{0}(\mathbf{x}) = g_1(x_1) + \ldots + g_d(x_d) \text{ and for } k = 1, \ldots, d \\
\quad r^{k}(\mathbf{x}) = g^k(x_k) \text{ for some functions } g_k, g^k : \mathbb{R} \to \mathbb{R} \right\}.
\]

This representation is not suitable for calculations, because of non–identifiability of \( g_k(x_k) \). See (3) on page 16 below for a definition including identifiability conditions.

Even though \( r^{0}(\mathbf{x}) + \sum_{k=1}^{d} r^{k}(\mathbf{x})(X_{i,k} - x_k)/h \) represents a Taylor approximation of \( r^{0}(X_i) \) and therefore \( r^{k} = h \frac{\partial}{\partial x_k} r^{0} \), we do not assume any relationship between \( r^{0} \) and \( r^{k} \).
2 GENERAL

2.3 MLN estimator

Semi–norm
Define a semi–norm \( \| \cdot \|_* \) on \( F \) by

\[
\| r \|_*^2 = \int \frac{1}{n} \sum_{i=1}^{n} \left\{ r^{i,0}(x) + \sum_{k=1}^{d} r^{i,k}(x) \left( \frac{X_{i,k} - x_k}{h} \right) \right\}^2 K_h(X_i, x) d\mathbf{x}.
\]

For \( r \in F_{\text{full}} \), \( \| r_Y - r \|_*^2 \) is written as

\[
\int \frac{1}{n} \sum_{i=1}^{n} \left\{ Y_i - r^{0}(x) - \sum_{k=1}^{d} r^{k}(x) \left( \frac{X_{i,k} - x_k}{h} \right) \right\}^2 K_h(X_i, x) d\mathbf{x}.
\]

Hence, (MLN:1) may be written as the orthogonal projection, with respect to \( \| \cdot \|_* \), of \( r_Y \) onto \( F_{\text{full}} \):

\[
\hat{r}_l = \arg \min_{r \in F_{\text{full}}} \| r_Y - r \|_*^2.
\]

2.3.3 Estimation in the additive model

The additive estimator \( \hat{r}_{\text{add}} \) of \( r_{\text{true}} \) is defined as the projection of \( r_Y \) into \( F_{\text{add}} \), instead of \( F_{\text{full}} \) as for \( \hat{r}_l \):

\[
\hat{r}_{\text{add}} = \arg \min_{r \in F_{\text{add}}} \| r_Y - r \|_*^2. \tag{2}
\]

Notation
For the normal equations of (2), we need some notation. Define for \( j, k = 1, \ldots, d \):

\[
\hat{V}_{0,0}(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(X_i, x) \\
\hat{V}_{0,j}(x) = \hat{V}_{j,0}(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(X_i, x) \frac{X_{i,j} - x_j}{h} \\
\hat{V}_{j,k}(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(X_i, x) \frac{X_{i,j} - x_j}{h} \frac{X_{i,k} - x_k}{h}
\]

and for \( k = 1, \ldots, d \):

\[
\hat{L}_0(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(X_i, x) Y_i \\
\hat{L}_k(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(X_i, x) \frac{X_{i,k} - x_k}{h} Y_i.
\]

Denote by \( \hat{V}(x) \) the \((d+1) \times (d+1)\) matrix with elements \( \hat{V}_{l,j}(x) \) and \( \hat{L}(x) = \text{col}_{l=0}^{d}(\hat{L}_l(x)) \).

The normal equations for the local linear estimator (MLN:1) are

\[
\hat{V}(x) \hat{r}_l = \hat{L}(x). \tag{MLN:31}
\]

Note that for all \( r \in F_{\text{full}} \)

\[
\| r \|_*^2 = \int \sum_{\ell,l=0}^{d} r^{\ell}(x) r^{l}(x) \hat{V}_{\ell,l}(x) d\mathbf{x}.
\]
Define the following abbreviations used in integrals

\[ dx_{-j} = \prod_{m=1, \ldots, d \atop m \neq j} dx_m \quad \text{and} \quad dx_{-j,k} = \prod_{m=1, \ldots, d \atop m \neq j,k} dx_m. \]

The following quantities are of interest \((j,k)\):

\[ \hat{V}_{i,l}^j(x_j) = \int \hat{V}_{i,l}(x) dx_{-j} \quad \text{(MLN:34)} \]

\[ \hat{V}_{i,l}^{jk}(x_j, x_k) = \int \hat{V}_{i,l}(x) dx_{-j,k}. \quad \text{(MLN:35)} \]

Now, we have all the notation needed to provide \(\mathcal{F}_{\text{add}}\) with identifiability conditions:

\[ \mathcal{F}_{\text{add}} = \left\{(r_0^0(x), r_1^0(x), \ldots, r_d^0(x)) \in \mathcal{F}_{\text{full}} \left| \right. \begin{array}{l} \exists g_0 \in \mathbb{R} \text{ and functions } g_j, g_j^d : \mathbb{R} \to \mathbb{R} \text{ with} \\ r_0^0(x) = g_0 + g_1(x_1) + \ldots + g_d(x_d), \\ r_1^0(x) = g_1(x_1), \ldots, r_d^0(x) = g_d(x_d), \\ \text{and } \int g_j(x_j) \hat{V}_{0,0}^j(x) dx_{-j} = \int g_j(x_j) \hat{V}_{0,0}^j(x) dx_j = 0 \end{array} \right\}. \quad \text{(3)} \]

For arbitrary \(r \in \mathcal{F}_{\text{add}}\), we use sub- and superscripts to refer to its decomposition given in (3). For example,

\[ \hat{r}_{\text{add}}^0(x) = \hat{r}_{\text{add},0} + \hat{r}_{\text{add},1}(x_1) + \ldots + \hat{r}_{\text{add},d}(x_d) \]

is an identifiable decomposition of the intercept \(\hat{r}_{\text{add}}^0\) of \(\hat{r}_{\text{add}}\). The slopes of \(\hat{r}_{\text{add}}\) are denoted by \(\hat{r}_{\text{add}}^1, \ldots, \hat{r}_{\text{add}}^d\).

For ease of notation, define the matrices \(\hat{M}_j\) and \(\hat{S}_{k,j}\) for \(j,k = 1, \ldots, d\):

\[ \hat{M}_j(x_j) = \begin{bmatrix} \hat{V}_{0,0}^j(x_j) & \hat{V}_{1,0}^j(x_j) \\ \hat{V}_{j,0}^j(x_j) & \hat{V}_{j,j}^j(x_j) \end{bmatrix} \quad \text{(MLN:46)} \]

\[ \hat{S}_{k,j}(x_k, x_j) = \begin{bmatrix} \hat{V}_{0,0}^{k,j}(x_k, x_j) & \hat{V}_{1,0}^{k,j}(x_k, x_j) \\ \hat{V}_{j,0}^{k,j}(x_k, x_j) & \hat{V}_{j,j}^{k,j}(x_k, x_j) \end{bmatrix}. \quad \text{(MLN:47)} \]

**Normal equations for \(\hat{r}_{\text{add}}\)**

The normal equations for arg \(\min_{\hat{r}_{\text{add}} \in \mathcal{F}_{\text{add}}} \|\hat{r}_{\text{add}} - r_Y\|_2^2\) are (see appendix A.2 for details)

\[ \hat{M}_j(x_j) \left( \frac{\hat{r}_{\text{add},j}(x_j) + \hat{r}_{\text{add},0}}{\hat{r}_{\text{add},j}(x_j)} \right) = \frac{1}{n} \sum_{i=1}^{n} \int K_h(X_i, x) dx_{-j} \left( X_{i,j} \right) Y_i - \sum_{k \neq j} \int \hat{S}_{k,j}(x_k, x_j) \left( \frac{\hat{r}_{\text{add},k}(x_k)}{\hat{r}_{\text{add},j}(x_j)} \right) dx_k. \quad \text{(MLN:32/33)} \]

Recall the normalizing conditions

\[ \int \hat{r}_{\text{add},j}(x_j) \hat{V}_{0,0}^j(x_j) dx_j = 0. \quad \text{(MLN:36)} \]

Formulae (MLN:32/33) and (MLN:36) define \(\hat{r}_{\text{add},0}, \hat{r}_{\text{add},j}(x_j)\), and \(\hat{r}_{\text{add}}^j(x_j)\) for given \(Y_i\) and \([\hat{r}_{\text{add},k}(x_k), \hat{r}_{\text{add}}^k(x_k); k \neq j]\).
The normal equations for a naive one–dimensional local linear estimator of the additive components based on \((Y_i, X_{i,j})_{i=1,...,n}\) are

\[
\widetilde{M}_j(x_j) \left( \hat{m}_j(x_j) \right) = \frac{1}{n} \sum_{i=1}^{n} \int K_h(X_i, \overline{x}) d\overline{x}_{-j} \left( \frac{1}{h} \right) Y_i. \tag{MLN:39/40}
\]

For product kernels, \(\int K_h(X_i, \overline{x}) d\overline{x}_{-j} = K_h(X_{i,j}, x_j)\). Now, (MLN:32/33) may be rewritten as

\[
\widetilde{M}_j(x_j) \left( \hat{r}_{\text{add},j}(x_j) + \hat{r}_{\text{add},0,j} - \hat{m}_j(x_j) \right) = -\sum_{k \neq j} \int \tilde{S}_{k,j}(x_k, x_j) \left( \hat{r}_{k,j}(x_k) \right) dx_k \tag{MLN:44}
\]

\[
\int \hat{r}_{\text{add},j}(x_j) \hat{V}_{0,0}(x_j) dx_j = 0. \tag{MLN:45}
\]

### 2.3.4 Backfitting algorithm

Define \(\hat{r}_{\text{add},j+}(x_j) = \hat{r}_{\text{add},j}(x_j) + \hat{r}_{\text{add},0}\). For each iteration step \(a (a = 0, 1, \ldots)\), the estimator \(\hat{r}_{\text{add}}^{[a+1]}\) is defined by

\[
\begin{aligned}
\left( \hat{r}_{\text{add},j+}^{[a+1]}(x_j) \right) &= \left( \hat{m}_j(x_j) \right) \\
- \widetilde{M}_j^{-1}(x_j) \sum_{k \neq j} \int \tilde{S}_{k,j}(x_k, x_j) \left( \hat{r}_{k,j}^{[a]}(x_k) \right) dx_k &- \phi_{\text{add},j}^{[a+1]}(x_j) = \phi_{\text{add},j}^{[a+1]}(x_j) - \int \hat{r}_{\text{add},j+}^{[a+1]}(x_j) \hat{V}_{0,0}(x_j) dx_j.
\end{aligned}
\tag{MLN:49}
\]

\[
\phi_{\text{add},j}^{[a+1]}(x_j) = \phi_{\text{add},j}^{[a]}(x_j) - \int \hat{r}_{\text{add},j+}^{[a]}(x_j) \hat{V}_{0,0}(x_j) dx_j. \tag{MLN:50}
\]

See also appendix C.2.1. MLN mention on page 1460, remark 5 that in practice all quantities have to be calculated on a grid and that the integrals have to be replaced by averages.

The starting values may be chosen as zero or as \(\hat{r}_{\text{add},j+}^{[0]}(x_j) = \hat{m}_j(x_j)\) and \(\hat{r}_{\text{add},j}^{[0]}(x_j) = \hat{m}_j(x_j)\), one iteration–step later.

### 2.3.5 Asymptotics for local polynomials

MLN formulate their theory quite generally. We consider the case, where the output grid is restricted to \([0, 1]^d\) and the observations \((Y_i, X_i)\) are independent identically distributed. Theorem MLN:4’ applies to this case and we are going to specify the sufficient assumptions:

**MLN:B2’**: The \(d\)-dimensional vector \(X\) has compact support \([0, 1]^d\) and its density \(f\) is bounded away from zero and infinity on \([0, 1]^d\).

**MLN:B1**: The kernel \(K\) is bounded, has compact support \((-C_1, C_1]\), is symmetric about zero, and is Lipschitz continuous. The product kernel \(K_h\) with bandwidth \(h\) is constructed from the univariate kernel \(K\) by

\[
K_h(X, \overline{x}) = \prod_{k=1}^{d} K((X - \overline{x})_k/h)/h.
\]

Furthermore, the kernel is rescaled at the boundary such that for all \(X_i \in [0, 1]^d\) holds

\[
\int_{[0,1]^d} K_h(X_i, \overline{x}) d\overline{x} = 1.
\]

This modification does not affect the local linear estimator, but it changes its projection to the additive model. Hence, the estimation of the marginal design density is equal to an integrated full–dimensional density estimation.
(MLN:B4'): The true regression function \( r_{\text{true},0}(x) = \mathbb{E}[Y|X = x] \) is twice continuously differentiable and \( f \) is once continuously differentiable.

(MLN:B3'): For some \( \theta > 5/2 \), \( \mathbb{E}[|Y|^{\theta}] < \infty \). Let
\[
\sigma_j^2(x_j) = \text{var}(Y - r_{\text{true},0}(x)|X_j = x_j).
\]

(C1): Suppose, the following model holds for \( x \in [0,1]^d \)
\[
r_{\text{true},0}(x) = r_{\text{true},0} + r_{\text{add},1}(x_1) + \ldots + r_{\text{add},d}(x_d)
\]
with the identifiability condition \( \mathbb{E}[r_{\text{add},j}(X_{i,j})|X_i \in [0,1]^d] = 0 \).

(C2): Assume, there exists a constant \( c_h \) with \( n^{1/5}h \to c_h \).

MLN use rather technical conditions (MLN:A1–A9). They are given in appendix A.4.

According to Theorem MLN:4’, (MLN:B1,B2’–B4’) and (C2) imply (MLN:A1–A9) for appropriate choices of \( \hat{m} \), \( \hat{M}_j \), \( \hat{S}_{k,j} \) — see appendix A.3. In this case, Theorem MLN:1’ asserts uniqueness and exponential convergence of the backfitting algorithm (MLN:49) on page 17. The same holds for the decomposition into expectation and stochastic part.

Assuming the regression function to be additive (C1), Theorem MLN:4’ provides the asymptotic distribution up to order \( h^2 \) of \( \hat{r}_{\text{add}} \). Comparing this with univariate local linear estimators, MLN conclude that (in the interior) \( \hat{r}_{\text{add}} \) achieves the same asymptotics as the oracle estimator \( \hat{r}_{\text{ora},j}(x_j) \) — see below.

Oracle optimality
When \( r_{\text{add},k}(x_k) \) for \( k \neq j \) are known, the problem of estimating \( \hat{r}_{\text{add},j}(x_j) \) given \( (Y_i, X_i)_{i=1,\ldots,n} \), reduces to the problem of estimating \( \hat{r}_{\text{add},j}(x_j) \) given \( (Y_i, X_i, X_j)_{i=1,\ldots,n} \), with
\[
Y_{i,-j} = Y_i - \sum_{k \neq j} r_{\text{true},k}(X_{i,k}).
\]
The oracle estimator is a one–dimensional local linear estimator based on \( (Y_i, X_{i,j})_{i=1,\ldots,n} \)
\[
\left( \hat{r}_{\text{ora},j}(x_j) + \hat{r}_{\text{ora},0} \right) = \hat{M}_j^{-1}(x_j) \frac{1}{n} \sum_{i=1,\ldots,n} K_h(X_{i,j}, x_j) \left( \frac{1}{h} \right) Y_{i,-j}.
\]
Because of the identifiability condition \( \int r_{\text{true},x_j}(x) f_j(x_j) = 0 \), we assume that \( \hat{r}_{\text{ora},j}(x_j) \) has mean zero. As in (MLN:45), the mean is taken with respect to the marginal–density estimation \( \hat{V}_{0,0}(x_j) \). MLN call this a centered oracle estimator.

MLN show that \( \hat{r}_{\text{ora},j}(x_j) \) and \( \hat{r}_{\text{add},j}(x_j) \) have the same asymptotic variance and bias. Hence, \( \hat{r}_{\text{add}} \) inherits the optimal properties of \( \hat{r}_{\text{ora}} \).

Above, we mentioned the properties of full and additive models. In the next section, we introduce a local linear estimator that allows a smooth choice between the full model and the additive model. In order to bound the variance, we may choose among a bias due to oversmoothing and a bias due to misspecification of the model.
3 NON–ADDITIVITY PENALIZED ESTIMATION

3 An estimator with a penalty on deviations from additivity

First, a modification of the Nadaraya–Watson estimator is introduced, and some properties are derived. This is later generalized to the local linear estimator.

3.1 Model and notation

Let \((X_1, Y_1), \ldots, (X_n, Y_n)\) be random variables fulfilling

\[
E(Y|X) = r(X)
\]

and \(\text{var}(Y|X) = \sigma^2(X)\). We assume that the observations \(Y_i\) are independently distributed conditionally on \(X_1, \ldots, X_n\). The \(X_i = [X_{i,1}, \ldots, X_{i,d}]^\top\) have design density \(f(x)\). We distinguish two cases: For random design, the \(X_i\) are independently identically distributed random vectors. For fixed design, the \(X_i\) are “regularly spaced” such that kernel density estimates converge uniformly to \(f(x)\) with the rate of regular integral approximations.

In this thesis, we use standard matrix notation, i.e. bold characters denote matrices \((A, P_{add})\) or, if underlined, column vectors \((\underline{x}, \underline{\beta})\). \(A^{-}, A^\top, \text{tr}(A), \text{Im}(A)\) denote generalized inverse, transposed, trace, and the vector space (image) generated by the column vectors of \(A\), respectively. Let \(B_k\) be some matrices, vectors or scalars. Then, \(\text{diag}_k(B_k)\) and \(\text{col}_k(B_k)\) denote a block–diagonal matrix with diagonal blocks \(B_k\) and the matrix, obtained by vertically stacking the matrices \(B_k\), respectively. Throughout this thesis, \(\|\cdot\|\) denotes the Euclidean norm.

3.2 A Nadaraya–Watson estimator penalized for non–additivity

A Nadaraya–Watson estimator will be defined, which diminishes the deviation from the sub–model of additive functions. As a by–product, this stabilizes the estimator. Algebraic transformations make the computations feasible and lead to better understanding.

The idea is carried over to local linear estimation due to their asymptotic superiority. If the additive model holds, better convergence rates are obtained. Hence, the question arises, whether these rates may be obtained for an appropriate choice of the penalty (model choice). This is considered in section 4.1.

3.2.1 Nadaraya–Watson estimation

Let \(K_H(X_i)\) be non–negative weights (denoted by \(K_i\) on page 12). Given a candidate value \(\tilde{r}\), the goodness of fit criterion may be written as

\[
\frac{1}{n} \sum_{i=1}^{n} K_H(X_i)(\tilde{r} - Y_i)^2 = a(\tilde{r} - \hat{r}_{NW})^2 + \text{const}
\]  

with \(a = \frac{1}{n} \sum_{i=1}^{n} K_H(X_i)\) and \(\hat{r}_{NW} = \frac{1}{n} \sum_{i=1}^{n} K_H(X_i)Y_i/a\), the minimizer. The product \(an(2h)^d\) is the effective number of observations in the smoothing window. It indicates how stable the Nadaraya–Watson estimator is. For a small \(a\), a modification of \(\tilde{r}\) results in a small increase of the weighted sum of squared residuals. If no data is in the smoothing window, \(a = 0\) and the minimum is ambiguous.

Output grid

Now, the estimator is evaluated simultaneously at output points \(t_1, \ldots, t_m\). A superscript \((j)\) indicates that a \(t\)–dependent quantity is evaluated at \(t = t_j\). In order to refer to the quantities \(a, \tilde{r}_{NW}, \text{and } \tilde{r}\) for \(t = t_j\), we write \(a^{(j)}, \tilde{r}_{NW}^{(j)}, \text{and } \tilde{r}^{(j)}\) (for convenience also denoted by \(a_j, \tilde{r}_j, \text{and } \tilde{r}_j\)). Note that local polynomial estimators are defined only at output points — unlike e.g. splines, where an element of a function space is estimated. Because some interdependence
between the estimates for the output points will later be introduced, the estimator will depend on the choice of the output points. In fact, the grid may be replaced by a function space, see section 4.1 for the local linear case.

**Subspace of additive estimates**

For additive models, an estimator is defined on a product space. Consequently, the output points must form a grid, to use the notion of additivity. We assume that \( t_j = [t_{j,1}, \ldots, t_{j,d}]^\top \), \( j = 1, \ldots, m \) are an enumeration of \( \{t_{1,1}, \ldots, t_{1,m_1}\} \times \cdots \times \{t_{d,1}, \ldots, t_{d,m_d}\} \subset [0,1]^d \)

and \( m = \prod_{\ell=1}^d m_\ell \). See equation (29) in section 4.2.1 for details about the enumeration. The estimates simultaneously at \( t_1, \ldots, t_m \) are minima of the high–dimensional paraboloid in \( \tilde{r}_1, \ldots, \tilde{r}_m \)

\[
\sum_{j=1}^m a_j (\tilde{r}_j - \hat{r}_j)^2.
\] (5)

Define the subspace of \( \mathbb{R}^m \), containing vectors \( \tilde{r} := [\tilde{r}_1, \ldots, \tilde{r}_m]^\top \) which may be obtained by evaluating an additive function at the output points \( t_1, \ldots, t_m \):

\[
\tilde{r}_j = r_1(t_{j,1}) + \ldots + r_d(t_{j,d})
\]

for arbitrary functions \( r_1, \ldots, r_d \). This subspace is called *additive* and \( P_{\text{add}} : \mathbb{R}^m \to \mathbb{R}^m \) denotes the orthogonal projection of \( \tilde{r} \) onto the additive subspace (with respect to the Euclidean scalar product).

**Penalizing deviations from additivity**

\( (I - P_{\text{add}}) \tilde{r} \) is the non–additive part of \( \tilde{r} \). We add a regularization term proportional to

\[
\|(I - P_{\text{add}}) \tilde{r}\|^2 = \tilde{r}^\top (I - P_{\text{add}}) \tilde{r}
\]

in order to penalize deviations from additivity. The penalty \( R > 0 \) is a tuning parameter.

Choosing \( A := \text{diag}(a_j) \), criterion (6) may be rewritten as a sum of two non–negative quadratic forms

\[
(\tilde{r} - \hat{r}_{\text{NW}})^\top A(\tilde{r} - \hat{r}_{\text{NW}}) + R \tilde{r}^\top (I - P_{\text{add}}) \tilde{r},
\]

leading to the normal equations

\[
(A + R(I - P_{\text{add}})) \hat{r} = A \hat{r}_{\text{NW}}.
\]

(7)

The minimum is unique if and only if \( A + R(I - P_{\text{add}}) \) is positive definite. In this case, the penalized estimator has the explicit form

\[
\hat{r}_R := \{A + R(I - P_{\text{add}})\}^{-1} A \hat{r}_{\text{NW}}.
\]

(8)

If the minimum is unique for one \( R \), it is unique for every \( R > 0 \), and \( \hat{r}_R \) is continuous in \( R \). Uniqueness depends only on the number and location of those \( j \) with \( a_j = 0 \). We have non–uniqueness, if the null space of \( A \) contains a vector in the additive subspace.
3.2.2 Algebraic transformations to avoid high–dimensional matrix inversions

In contrast to the usual estimator, where computation is done for each output point separately, the minimum of the penalized estimator needs to be evaluated simultaneously on a grid. Naively solving the normal equations would result in huge memory consumption and hence the estimator would be intractable: A three dimensional grid with 30 points in each direction contains 27000 points, and we have to invert a matrix with 729 million entries. This illustrates the need for dimension reduction. More importantly, this calculation leads to better understanding of the statistical properties.

The minimum of (6) is achieved at

\[ \hat{\mathbf{L}}_R = \{ (\mathbf{A} + \mathbf{R} \mathbf{I}) - \mathbf{R} \mathbf{P}_{add} \}^{-1} \mathbf{A} \hat{\mathbf{L}}_{NW}, \]

where \( \{ \cdot \}^{-1} \) is a generalized inverse. The matrix to be inverted is highly structured. It consists of a diagonal part \( \mathbf{A} + \mathbf{R} \mathbf{I} \) and a part \( \mathbf{R} \mathbf{P}_{add} \) of rank \( \sum \ell m_{\ell} + 1 - d \) which is generally much lower than \( m = \prod \ell m_{\ell} \).

We use a formula from Rao and Kleffe, 1988 page 5 to take advantage of this special structure: Choose some matrix \( \mathbf{Z} : \mathbb{R}^m \to \mathbb{R}^{m^*} \) with \( \mathbf{P}_{add} = \mathbf{Z}^\top \mathbf{Z} \). An explicit choice of \( \mathbf{Z} \) with \( m^* = \sum \ell m_{\ell} \) will be given in section 4.2.1.

**Proposition 3.1**

Given matrices \( \mathbf{B}, \mathbf{C}, \mathbf{D} \) with \( \mathbf{B} = \mathbf{B}^\top, \mathbf{D} = \mathbf{D}^\top \), and \( \text{Im}(\mathbf{B}) \supseteq \text{Im}(\mathbf{C}) \)

\[ \mathbf{B}^\top - \mathbf{B}^\top \mathbf{C} \mathbf{D} (\mathbf{I} + \mathbf{C}^\top \mathbf{B}^\top \mathbf{C} \mathbf{D})^{-1} \mathbf{C}^\top \mathbf{B}^\top \]

is a generalized inverse of \( \mathbf{B} + \mathbf{C} \mathbf{D} \mathbf{C}^\top \).

Apply Proposition 3.1 with \( (\mathbf{A} + \mathbf{R} \mathbf{I}, \mathbf{Z}^\top, -\mathbf{R} \mathbf{I}_{m^*}) \) instead of \( (\mathbf{B}, \mathbf{C}, \mathbf{D}) \). Since \( \mathbf{A} + \mathbf{R} \mathbf{I} \) has full rank, the conditions are obviously fulfilled. Define

\[ \mathbf{A}^{(R)} := R (\mathbf{A} + \mathbf{R} \mathbf{I})^{-1}. \]

Note that

\[ \mathbf{A}^{(R)} = \text{diag} \left( \frac{R}{a_j + R} \right) \quad \text{and} \quad \mathbf{I} - \mathbf{A}^{(R)} = (\mathbf{A} + \mathbf{R} \mathbf{I})^{-1} \mathbf{A}. \]

Using Proposition 3.1, we find that

\[ \hat{\mathbf{L}}_R := (\mathbf{I} + \mathbf{A}^{(R)} \mathbf{Z}^\top (\mathbf{I} - \mathbf{Z} \mathbf{A}^{(R)} \mathbf{Z}^\top)^{-1} \mathbf{I} - \mathbf{A}^{(R)}) \hat{\mathbf{L}}_{NW} \]

is a solution of (7).

**Computational details**

\( (\mathbf{I} - \mathbf{A}^{(R)}) \hat{\mathbf{L}}_{NW} \) is the minimizer of

\[ \sum_{j=1}^m \left( \frac{1}{n} \sum_{i=1}^n k^{(j)}_H (x_i - \bar{x}_j)^2 + R \hat{r}_j^2 \right). \]

This is a Nadaraya–Watson estimator shrunk to zero and therefore unique. The amount of shrinking depends on the ratio \( a_j/(a_j + R) \). Hence, \( a_j \) indicates how stable the Nadaraya–Watson estimator is when using ridge regression.

Multiplication by \( \mathbf{Z} \) corresponds to computing averages, and multiplication by \( \mathbf{Z}^\top \) is just a sum (see section 4.2.1). \( \mathbf{I} - \mathbf{Z} \mathbf{A}^{(R)} \mathbf{Z}^\top \) can be calculated efficiently and — because its dimension \( m^* \) is generally much smaller than \( m \) — solving the equation is feasible. Hence, formula (9) is a computational simplification compared to formula (8).

The location of the minimum in (6) is unique, if and only if \( (\mathbf{I} - \mathbf{Z} \mathbf{A}^{(R)} \mathbf{Z}^\top) \) is a non–singular matrix. See appendix B.1.
3.2.3 Interpretation of the penalized Nadaraya–Watson estimator

Formula (9) allows two nice ways of interpretation of the penalized estimator. Define the oblique projection \( P_A^{(R)} \) into the subspace of additive vectors

\[
P_A^{(R)} := Z^\top (I - ZA^{(R)} Z^\top)^{-1} Z (I - A^{(R)}).
\]

(10)

See also appendix B.2. Then, formula (9) may be rewritten as

\[
\hat{\tau}_R = (I - A^{(R)})\hat{\tau}_{NW} + A^{(R)}P_A^{(R)}\hat{\tau}_{NW}.
\]

(11)

Consequently, \( [\hat{\tau}_R]_j \) is a convex combination between the full dimensional estimator \( \hat{\tau}_j \) and an additive estimator \( [P_A^{(R)}\hat{\tau}_{NW}]_j \) with weights \( a_j / (a_j + R) \) and \( R / (a_j + R) \), respectively. Remember that \( a_j \) indicates how stable the full dimensional estimator is.

An equivalent form of (11) is:

\[
\hat{\tau}_R = (I - A^{(R)})(I - P_A^{(R)})\hat{\tau}_{NW} + P_A^{(R)}\hat{\tau}_{NW}.
\]

(12)

Formula (12) may be interpreted, as preserving the additive components of \( \hat{\tau}_{NW} \), while the “non–additive” components are shrunk.

3.2.4 Additional remarks

The additive components of \( \hat{\tau}_R \) depend on \( R \).

The additive components of \( \hat{\tau}_R \) are

\[
P_{\text{add}}\hat{\tau}_R = P_A^{(R)}\hat{\tau}_{NW},
\]

which generally depend on \( R \). Remark that formula (12) is a decomposition of \( \hat{\tau}_R \) into non–additive and additive components, because

\[
P_{\text{add}}P_A^{(R)} = P_A^{(R)}
\]

\[
P_{\text{add}}(I - A^{(R)})(I - P_A^{(R)}) = 0.
\]

The convex combination property (11) and the decomposition into additive and orthogonal components in (12) are desirable. Unfortunately, the projection \( P_A^{(R)} \) depends on \( R \), which is a conclusion of the two other properties: If we would use a modified version of the decomposition (12) as a definition for \( \hat{\tau}_R \) with \( P_{\text{add}}\hat{\tau}_R \) independent of \( R \), then the convex combination property (11) would not hold anymore. See appendix B.3.

Behaviour for singular \( (I - ZA^{(R)} Z^\top) \)

The following proposition states that components of \( \hat{\tau}_R \) at those output points, where the full estimator is unique, remain unique. This holds even if the minimization problem (6) is not unique.

**Proposition 3.2** For every \( j \) with \( a_j > 0 \), \( [\hat{\tau}_R]_j \) does not depend on the choice of the generalized inverse of \( (I - ZA^{(R)} Z^\top) \).

Uniqueness holds due to formula (9) and the remark about non–ambiguity of \( (I - A^{(R)})\hat{\tau}_{NW} \).

Grid dependence

The output grid needs to grow for increasing \( n \), because otherwise the smoothing windows do not cover the whole \([0, 1]^d\) cube.

Consider the decomposition of \( \hat{\tau}_R \) in (11): using a finer grid reduces the variability of the additive term \( P_A^{(R)}\hat{\tau}_{NW} \) (see section 4.2.3 for details). On the other hand, the local term \( \hat{\tau}_{NW} \) is not affected by the choice of the other output points.
3 NON–ADDITIVITY PENALIZED ESTIMATION 3.3 Generalizing to local polynomials

3.3 Generalizing from local constant to local polynomials

Comparing the bias of Nadaraya–Watson and local linear estimators, the Nadaraya–Watson estimator additionally has a design–dependent term. Hence, local linear estimators are preferred. Furthermore, the additive projection estimator in Mammen et al. (1999) is oracle optimal in the local linear case, but not in the Nadaraya–Watson case. Hence, in this section we generalize our estimator to local polynomials. However, a naive approach works only in low dimensions and has to be modified in an appropriate way later.

We introduce first the notation for general local polynomial estimation. The goal is to bring it into a form, which may be handled the same way as for the Nadaraya–Watson estimator. We want to keep the notation general, such that estimation of several local characteristics of \( r(\cdot) \) at some output point \( t \) is possible. In principle, application to higher order polynomials or estimation of derivatives are feasible, but from section 3.3.2 on we concentrate on local linear estimation of the regression function itself.

3.3.1 Local polynomial regression

Choose a local polynomial approximation \( y(\tilde{\beta}, \mathbf{x}) \) of \( r(\mathbf{x}) \) in a neighborhood of \( t \). For given \( \tilde{\beta} \) and \( \mathbf{x} \) define the fitted polynomial as

\[
y(\tilde{\beta}, \mathbf{x}) := \sum_{k=0}^{p} \tilde{\beta}_k p_k(\mathbf{x} - t),
\]

where \( \{p_k(\mathbf{x} - t)\}_{k=0, \ldots, p} \) denote a polynomial base and \( \tilde{\beta} = \text{col}_{k=0, \ldots, p}(\tilde{\beta}_k) \) the coefficients. The parameter of interest is a linear combination \( r_u \) of the coefficients \( \tilde{\beta}_k \):

\[
r_u(\tilde{\beta}) := \sum_{k=0}^{p} u_k \tilde{\beta}_k = \mathbf{u}^\top \tilde{\beta}.
\]

Typically \( u_k = p_k(\mathbf{0}) \). We want to minimize the weighted mean squared residuals

\[
\text{SSR}(\tilde{\beta}) := \frac{1}{n} \sum_{i=1}^{n} K_H(\mathbf{X}_i) \{ Y_i - y(\tilde{\beta}, \mathbf{X}_i) \}^2.
\]

Standard algebra yields that \( \text{SSR}(\tilde{\beta}) \) is a paraboloid in \( \tilde{\beta} \)

\[
\text{SSR}(\tilde{\beta}) = (\tilde{\beta} - \tilde{\beta}_{lp})^\top S(\tilde{\beta} - \tilde{\beta}_{lp}) + \text{const},
\]

where \( S \) is a non–negative definite matrix (\( |S|_{kl} = \frac{1}{n} \sum_{i=1}^{n} K_H(\mathbf{X}_i)p_k(\mathbf{X}_i - t)p_l(\mathbf{X}_i - t) \) for \( k, l = 0, \ldots, p \)), and every minimizer \( \tilde{\beta}_{lp} \) of \( \text{SSR}(\tilde{\beta}) \) is a local polynomial estimator of \( \tilde{\beta} \). The local polynomial estimator of \( r_u(\tilde{\beta}) \) is then \( \hat{\tau}_{lp} = r_u(\tilde{\beta}_{lp}) \). \( \hat{\tau}_{lp} \) is unique if and only if \( \mathbf{u} \in \text{Im}(S) \). We are not only interested in the location of the minimum, but also in the value of the minimization criterion for any candidate value \( \tilde{r} \) of \( r_u(\tilde{\beta}) \). Using the ‘unified theory of linear estimation’ (Rao, 1973, pages 294ff), we will perform a dimension reduction of the paraboloid \( \text{SSR}(\tilde{\beta}) \) to a parabola \( \text{SSR}(\tilde{r}) \) in \( \tilde{r} \). This is of the same form as equation (4) and allows us to handle this general case in the same way as the Nadaraya–Watson case. Define

\[
\text{SSR}(\tilde{r}) := \min_{\tilde{\beta} : r_u(\tilde{\beta}) = \tilde{r}} \text{SSR}(\tilde{\beta}),
\]

which attains its minimum at \( \hat{\tau}_{lp} \). Using formulas for ‘extrema of quadratic forms under linear restrictions’ (Rao, 1973, page 61), there exists some \( a \) with

\[
\frac{1}{n} \sum_{i=1}^{n} K_H(\mathbf{X}_i)p_k(\mathbf{X}_i - t)p_l(\mathbf{X}_i - t) a(\tilde{r} - \hat{\tau}_{lp})^2.
\]
3.3 Generalizing to local polynomials

Details are explained in appendix B.4. Consequently, the conditional minimum SSR(\(\tilde{r}\)) of SSR(\(\hat{\beta}\)) is a parabola (or a constant):

\[
\text{SSR}(\tilde{r}) = a(\tilde{r} - \hat{r}_p)^2 + \text{const}.
\]

**Proposition 3.3** Assume \(u \neq 0\). If \(u \in \text{Im}(S)\), then \(u^\top S^- u\) is independent of the choice of \(S^-\) and positive. We have

\[
a = (u^\top S^- u)^{-1} > 0.
\]

If \(u \notin \text{Im}(S)\), we get

\[
a = 0.
\]

### 3.3.2 Estimation of the regression function itself

The Nadaraya–Watson estimator corresponds to \(p = 0\) and \(p_0(x - t) = 1\) (local constant approximation). Formula (4) on page 19 states that SSR\(_{NW}\)(\(\hat{\beta}\)) = \(a_{NW}(\hat{\beta} - \hat{r}_{NW})^2 + \text{const}\) with \(a_{NW} = \frac{1}{n} \sum_{i=1}^{n} K_H(X_i)\).

Now, we consider the **local linear estimator**. The following proposition allows us to compare all (i.e. \(a\) as described in proposition 3.3 for the local linear estimator) with \(a_{NW}\) (i.e. \(a\) for the Nadaraya–Watson estimator).

**Proposition 3.4**

Use the following decomposition of \(S\) in (13)

\[
S = \begin{pmatrix} s_0 & \underline{s}_1 \\bar{s}_1^\top \\ \underline{s}_1 & S_2 \end{pmatrix}
\]

where \(s_0\) is a scalar, and \(S_2\) has dimension \(p \times p\).

If \(u = e_1 \equiv [1, 0, \ldots, 0]^\top\), then we have

\[
a = s_0 - \underline{s}_1^\top S^- \underline{s}_1
\]

for any generalized inverse of \(S_2\).

Note that the formula for \(a\) is also valid for \(u \notin \text{Im}(S)\), where \(a = 0\).

Using the notation of proposition 3.4, the local linear estimator corresponds to the case where \(p = d\) with \(p_0(x) = 1\) and \(\text{col}_{t=1,\ldots,d}(p_0(x)) = x - t\). We have \(s_0 = a_{NW}\),

\[
\underline{s}_1 = \frac{1}{n} \sum_{i=1}^{n} K_H(X_i)(X_i - t),
\]

and

\[
S_2 = \frac{1}{n} \sum_{i=1}^{n} K_H(X_i)(X_i - t)(X_i - t)^\top.
\]

If \(\underline{s}_1 \neq 0\), we have \(a_{ll} < a_{NW}\): \((S_2 \geq s_0^{-1} \underline{s}_1 \underline{s}_1^\top\) implies \(\underline{s}_1 \in \text{Im}(S_2)\) and therefore \(\underline{s}_1^\top S^- \underline{s}_1 > 0\)

\[
a_{ll} = a_{NW} - \underline{s}_1^\top S^- \underline{s}_1.
\]

Now apply the same procedure as for the Nadaraya–Watson estimator. Recall that a superscript \((j)\) indicates that the corresponding quantity is evaluated at \(t = t_j\). Hence, \(a_j\) denotes \(a_{ll}^{(j)}\), and \(\hat{r}_j\) denotes \(\hat{r}_{ll}^{(j)}\). Define

\[
\hat{r}_R = \arg\min_{\tilde{r}} \sum_{j=1}^{m} a_j(\hat{r}_j - \hat{r}_j)^2 + R \| (I - P_{\text{add}})\tilde{r} \|^2.
\]

(6’)

24
This is equivalent to finding estimates \( \hat{\beta}^{(1)}, \ldots, \hat{\beta}^{(m)} \), which minimize
\[
\sum_{j=1}^{m} \frac{1}{n} \sum_{i=1}^{n} K_{H}^{(j)}(X_i) \{ Y_i - y^{(j)}(\hat{\beta}^{(j)}, X_i) \}^2 + R \left\| (I - P_{\text{add}}) \text{col}_j \{ r^{(j)}(\hat{\beta}^{(j)}) \} \right\|^2,
\]
and then set \( \hat{\tau}_R = \text{col}_j \{ r^{(j)}(\hat{\beta}^{(j)}) \} \).

Define \( A = \text{diag}_j \{ a^{(j)}_u \} \) and \( \hat{\tau}_U = \text{col}_j (\hat{r}^{(j)}) \).
The condition for uniqueness of minimization problem (6') is that the matrix
\[
A + R(I - P_{\text{add}}) > 0 \quad \text{(positive definite)}.
\]

We proceed by generalizing the penalized Nadaraya–Watson estimator to the local linear case. The penalized estimator (6') is of the form
\[
\hat{\tau}_R = \{ A + R(I - P_{\text{add}}) \}^{-1} A \hat{\tau}_U.
\]
Again we write
\[
A^{(R)} = R(A + RI)^{-1} = \text{diag}_j \left( \frac{R}{a^{(j)}_u + R} \right).
\]
The oblique projection \( P_{A}^{(R)} \) is defined in (10).

The estimator is a pointwise convex combination of the local linear and some additive estimator:
\[
\hat{\tau}_R = (I - A^{(R)})\hat{\tau}_U + A^{(R)}P_{A}^{(R)}\hat{\tau}_U.
\]
The interpretation of shrinking non–additive components to zero remains valid:
\[
\hat{\tau}_R = (I - A^{(R)})(I - P_{A}^{(R)})\hat{\tau}_U + P_{A}^{(R)}\hat{\tau}_U.
\]

### Drawbacks

While asymptotic properties show that local linear estimation is preferable to Nadaraya–Watson estimation, the variability of the local linear estimator is considerably larger for small sample sizes. Since \( P_{A}^{(R)} \) essentially calculates averages, we expect that the variability is considerably reduced for increasing \( R \). However, in some instances, this expectation is not correct: As an example, consider high dimensional \((d > 5)\) estimation of an additive regression function with the asymptotically optimal bandwidth \((h \approx n^{-1/3})\). In this case, \( nh^d \) tends to zero for large \( n \). Hence, we have typically no or one observation in the smoothing window. To fit the local linear estimator, we need at least \( d + 1 \) observations, otherwise it is not well determined \((\mu^{(j)}_U = 0)\).

The penalized Nadaraya–Watson estimator avoids this problem when evaluated on a dense grid (i.e. smoothing windows which cover \([0, 1]^d\)). The ordinary Nadaraya–Watson estimator is well defined even if there is only one observation in the smoothing region, and hence the additive components are generally well defined.

If we modify the local linear estimator such that it is well defined with one observation in the smoothing window, this estimator will presumably be essentially a Nadaraya–Watson estimator. Therefore adding a (local) stabilization term to the minimization criterion (13) on page 23 does not help us achieving the advantages of the local linear estimator. The solution is to penalize all parameters, intercept and slope, towards additivity, not only the intercept. This approach is pursued in the next section.

For \( R \) tending to infinity and a very dense grid, we expect the estimator introduced below to be analogous to the additive estimator of Mammen et al. (1999), and the latter is known to achieve asymptotic oracle optimality.
3.4 Local linear estimation with full penalty

In the previous sections, we penalized the estimated value only, and thus the regularizing term did not depend on the actual parameterization. Accordingly, the resulting estimator did not depend on the parameterization of the local linear estimator. In this section, we will penalize all parameters. Hence, we must take care of the parameterization.

3.4.1 Regularizing the nuisance parameters

For some output point \( t_j \), minimize

\[
\hat{\beta}^{(j)} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} K_H^{(j)}(x_i) \left( Y_i - \beta_0 - \sum_{k=1}^{d} \beta_k X_{i,k} - t_{j,k} \right)^2
\]

where \( \beta = (\beta_0, \beta_1, \ldots, \beta_d) \).

The estimate \( \hat{\beta}^{(j)}_t \) is defined as \( \hat{\beta}^{(j)} \) with \( \hat{\beta}^{(j)}_t = [1, 0, \ldots, 0] \). The factor \( h_k^{-1} \) in the slope term will be justified later (sections 3.4.2 and 3.4.3). The matrix \( S^{(j)} \) is defined in (17) on page 27 below.

The parameters for the local linear estimator on the output–grid \( t_j, j = 1, \ldots, m \) are

\[
\hat{\beta} = \text{col}_{j=1,\ldots,m} \left( \hat{\beta}^{(j)} \right) = \text{col}_{j=1,\ldots,m} \left( \text{col}_{\ell=0,\ldots,d} \left( \hat{\beta}^{(j)}_{\ell} \right) \right) \in \mathbb{R}^{(d+1)m}.
\]

Additive subspace

Define the subspace of additive parameters by

\[
\left\{ \text{col}_{j=1,\ldots,m} \left( \hat{\beta}^{(j)} \right) \right\} \exists \text{ functions } r_k, r^k : [0, 1] \to \mathbb{R} \text{ such that}
\]

\[
\forall j = 1, \ldots, m : \hat{\beta}^{(j)}_0 = \sum_{k=1}^{d} r_k(t_{j,k})
\]

and for \( k = 1, \ldots, d : \hat{\beta}^{(j)}_k = r^k(t_{j,k}) \subset \mathbb{R}^{(d+1)m}.

Minimization criterion

Denote by \( \bar{P}_\text{add} \) the orthogonal projection into the above subspace. See section 4.2.2 for details. As in (6), we add a penalty for deviations from the additive subspace: \( \bar{\beta}_R \) is defined as the \( \hat{\beta} = \text{col}_j \text{col}_k(\hat{\beta}_k^{(j)}) \) minimizing

\[
\sum_{j=1}^{m} \frac{1}{n} \sum_{i=1}^{n} K_H^{(j)}(x_i) \left( Y_i - \hat{\beta}^{(j)}_0 - \sum_{k=1}^{d} \hat{\beta}^{(j)}_k X_{i,k} - t_{j,k} \right)^2
+ R \left\| (I - \bar{P}_\text{add}) \bar{\beta} \right\|^2
\]

One question arises: Why is the same \( R \) used to regularize both intercept and slope? Principally, this is done for ease of notation. In fact, it is possible to replace \( R \left\| (I - \bar{P}_\text{add}) \bar{\beta} \right\|^2 \) by \( R_1 \left\| \text{intercept} \right\|^2 + R_2 \left\| \text{slope} \right\|^2 \), where intercept and slope refer to the corresponding parts of \( (I - \bar{P}_\text{add}) \bar{\beta} \). Algorithmically, this is equivalent to a reparameterization. (In 16) we have \( R_2 = R_1 / h^2 \) if \( h_1 = \ldots = h_d = h \).
Normal equations
Minimizing (16) leads to the following normal equations
\[(\tilde{A} + R(I - \tilde{P}_{\text{add}}))\hat{\beta}_R = \mathbf{T},\]
with
\[\mathbf{T} = \text{col}_j(\mathbf{T}^{(j)}), \quad \mathbf{T}^{(j)} = \frac{1}{n} \sum_{i=1}^{n} K_H^{(j)}(X_i) \begin{bmatrix} \frac{1}{X_{i,1} - t_{i,1}} \\ \vdots \\ \frac{1}{X_{i,d} - t_{i,d}} \end{bmatrix} Y_i\]
and \(\tilde{A} = \text{diag}_{j=1,\ldots,m}(S^{(j)})\), where
\[S^{(j)} = \frac{1}{n} \sum_{i=1}^{n} K_H^{(j)}(X_i) \begin{bmatrix} \frac{1}{X_{i,1} - t_{i,1}} \\ \vdots \\ \frac{1}{X_{i,d} - t_{i,d}} \end{bmatrix} \begin{bmatrix} 1 & \frac{X_{i,1} - t_{i,1}}{h_1} & \cdots & \frac{X_{i,d} - t_{i,d}}{h_d} \end{bmatrix}.\]

The resulting estimate is \(\hat{\mathbf{r}}_R = U\hat{\mathbf{b}}_R \equiv \text{col}_j(\beta_0^{(j)})\) with \(U = \text{diag}_{j=1,\ldots,m}([1, 0, \ldots, 0])\).

Dimension reduction
We use a decomposition \(\tilde{P}_{\text{add}} = \tilde{Z}^\top \tilde{Z}\). One suitable choice is given in section 4.2.2. Let us introduce the notation
\[\tilde{A}^{(R)} = \tilde{A} + RI_{(d+1)m} \quad \text{and} \quad \tilde{A}^{(R)} = R(\tilde{A} + RI_{(d+1)m})^{-1}.\]

Now, we apply Proposition 3.1 with \((\tilde{A}^{(R)}, \tilde{Z}^\top, -RI_{2m^*})\) instead of \((B, C, D)\):
\[\left(\tilde{A}^{(R)} - R\tilde{P}_{\text{add}}\right)^{-1} = \left(\tilde{A}^{(R)}\right)^{-1} + R\left(\tilde{A}^{(R)}\right)^{-1} \tilde{Z}^\top \left\{I - \tilde{Z}R\left(\tilde{A}^{(R)}\right)^{-1} \tilde{Z}^\top\right\}^{-1} \tilde{Z} \left(\tilde{A}^{(R)}\right)^{-1}\]
\[= \left(I + \tilde{A}^{(R)} \tilde{Z}^\top \left\{I - \tilde{Z} \tilde{A}^{(R)} \tilde{Z}^\top\right\}^{-1} \tilde{Z}\right) \left(\tilde{A}^{(R)}\right)^{-1} \mathbf{T}.\]

Accordingly, we have the formula
\[\hat{\mathbf{r}}_R = U \left(I + \tilde{A}^{(R)} \tilde{Z}^\top \left\{I - \tilde{Z} \tilde{A}^{(R)} \tilde{Z}^\top\right\}^{-1} \tilde{Z}\right) \left(\tilde{A}^{(R)}\right)^{-1} \mathbf{T}.\]
The matrices \(\tilde{A}, \tilde{A}^{(R)}\), and \(\tilde{A}^{(R)}\) are block diagonal with \(m\) blocks of dimension \((d+1) \times (d+1)\).
The matrix \(\left\{I - \tilde{Z} \tilde{A}^{(R)} \tilde{Z}^\top\right\}\) has dimension \(2m^* \times 2m^*\). (Recall that \(m^* = \sum_{\ell=1}^{d} m_\ell\) is much smaller than \(m = \prod_{\ell=1}^{d} m_\ell\). Therefore, this implies a great reduction in both memory consumption and number of numeric operations.

Interpretation using oblique projections
Note that
\[
\left(\tilde{A}^{(R)}\right)^{-1} \mathbf{T} = \left(I - \tilde{A}^{(R)}\right)\tilde{A}^\top \mathbf{T} = \left(I - \tilde{A}^{(R)}\right)\hat{\mathbf{b}}^T
\]
is unique, even if \(\hat{\mathbf{b}}^T\) is not, and we may write
\[
\hat{\mathbf{r}}_R = U \left(I + \tilde{A}^{(R)} \tilde{Z}^\top \left\{I - \tilde{Z} \tilde{A}^{(R)} \tilde{Z}^\top\right\}^{-1} \tilde{Z}\right) \left(I - \tilde{A}^{(R)}\right)\hat{\mathbf{b}}^T.\]
Next, we mention that 
\[
\tilde{P}_A^{(R)} = \tilde{Z}^\top \left\{ I - \tilde{Z} \tilde{A}^{(R)} \tilde{Z}^\top \right\} \tilde{Z} \left( I - \tilde{A}^{(R)} \right) \tag{19}
\]
is an oblique projection into the image of \( \tilde{P}_{\text{add}} \). If \( \hat{\beta}_R \) is unique, we have \( \text{Im}(\tilde{P}_{\text{add}}) = \text{Im}(\tilde{P}_A^{(R)}) \).

Rewrite (18) as
\[
\hat{\epsilon}_R = U \left( (I - \tilde{A}^{(R)}) + \tilde{A}^{(R)} \tilde{P}_A^{(R)} \right) \hat{\beta}_{ll} \tag{20}
\]
and recall that \( \tilde{A}^{(R)} \) is block–diagonal and \( 0 \leq \tilde{A}^{(R)} \leq I \). Hence, \( \hat{\beta}_R \) is a “convex combination” of the parameters \( \hat{\beta}_{ll} \) and \( \tilde{P}_A^{(R)} \hat{\beta}_{ll} \), which is additive.

Again, non–additive components are shrunk:
\[
\tilde{\epsilon}_R = U \left( (I - \tilde{A}^{(R)}) (I - \tilde{P}_A^{(R)}) + \tilde{P}_A^{(R)} \right) \hat{\beta}_{ll} \, . \tag{21}
\]

### 3.4.2 Comments on convex combination

The penalized Nadaraya–Watson estimator has a nice representation as a convex combination of the ordinary Nadaraya–Watson estimator and an additive estimator, see equation (11). Formula (20) is also reminiscent to this interpretation, but there are some problems: Convex combinations of scalars in \([0, 1]\) are obvious. However, convex combinations of matrices (symmetric, with eigenvalues in \([0, 1]\)) may lead to some problems. Changing the base brings the matrix into diagonal form, but this also mixes up different components (intercept and slope). We want to see whether the slope terms affect the asymptotic behavior.

**Notation**

For a given output point \( t_j \), we need some notation for the terms in (20). The \( j^{th} \) component of \( \tilde{P}_A^{(R)} \hat{\beta}_{ll} \) is denoted by \( \hat{\beta}_{add,R}^{(j)} \). Say,
\[
\text{col}_j(\hat{\beta}_{add,R}) = \tilde{P}_A^{(R)} \hat{\beta}_{ll} \, .
\]

Then, the \( j^{th} \) component \( \tilde{\epsilon}_R^{(j)} \) of \( \tilde{\epsilon}_R \) may be written as
\[
\tilde{\epsilon}_R^{(j)} = (u - u_R^{(j)})^\top \hat{\beta}_{ll}^{(j)} + u_R^{(j)} \hat{\beta}_{add,R}^{(j)} \, . \tag{22}
\]

with \( u = [1, 0, \ldots, 0]^\top \) and \( u_R^{(j)} = S^{(j)}(S^{(j)} + RI)^{-1}u \). Note that
\[
U \tilde{A}^{(R)} = \text{diag}_{j=1,...,m}(u_R^{(j)})^\top \, .
\]

**Properties of \( u_R^{(j)} \)**

Because \( 0 \leq \tilde{A}^{(R)} \leq I \), we have
\[
\| u_R^{(j)} \| \leq 1 \quad \text{and} \quad \| u - u_R^{(j)} \| \leq 1 \, .
\]

Furthermore, \( u_R^{(j)} \) converges to \( 0 \) as \( R \) tends to infinity. If \( S^{(j)} \) is non–singular, \( u - u_R^{(j)} \) converges to \( 0 \) as \( R \) tends to zero.
Separation of intercept and slope

Rewrite formula (22) as

\[
(1 - [u_R^{(j)}]_0) [\hat{\beta}_n^{(j)}]_0 + [u_R^{(j)}]_0 [\tilde{\beta}_{add,R}^{(j)}]_0 + \sum_{k=1}^d [u_R^{(j)}]_k [\tilde{\beta}_{add,R}^{(j)} - \tilde{\beta}_n^{(j)}]_k \right) + \left\{ \sum_{k=1}^d [u_R^{(j)}]_k [\tilde{\beta}_{add,R}^{(j)} - \tilde{\beta}_n^{(j)}]_k \right\}. \tag{23}
\]

The first term \{\ldots\} in (23) is a convex combination of the local linear and some additive estimator.

The factor \( h^{-1} \) in the parameterization (15) ensures that the variability of \([\tilde{\beta}_n^{(j)}]_0\) and \([\tilde{\beta}_n^{(j)}]_k\) for \( k = 1, \ldots, d \) are of the same order \( O((nh^d)^{-1}) \).

### 3.4.3 On different penalties on intercept and slopes

Here, we consider choosing different penalties for intercept and slopes in the definition of \( \hat{\beta}_R \), formula (16) on page 26. Let \( P_t \) be the corresponding projection to the admissible parameterizations in the additive model such that, after reordering rows and columns, \( P_{add} \) corresponds to \( \text{diag}_{t=0, \ldots, d}(P_t) \): Projections to the subspace of vectors obtained by evaluating additive \( (P_0 := P_{add}) \) and univariate \( (P_k := Z_{0k} Z_{0k}, k = 1, \ldots, d \), see section 4.2.1) functions on a grid. We introduce new parameters \( \alpha_0 := 1 \) and \( \alpha_k \in (0, \infty) \) for \( k = 1, \ldots, d \). We redefine \( \tilde{\beta}_R \) as the minimizer of

\[
\sum_{j=1}^m \frac{1}{n} \sum_{i=1}^n K_{\beta}^{(j)}(X_i) \left( Y_i - \tilde{\beta}_0^{(j)} - \sum_{k=1}^d \tilde{\beta}_k^{(j)} X_{ik} - \frac{t_{jk}}{h_k} \right)^2 + R \sum_{t=0}^d \alpha_t^2 \left\| (I - P_t) \text{col}_j(\tilde{\beta}_t^{(j)}) \right\|^2. \]

This corresponds to a reparameterization \( \tilde{\beta}_t^{(j)} = \alpha_t \beta_t^{(j)} \) and write \( \tilde{\beta}^{(j)} := \text{col}_j(\text{col}_t(\beta_t^{(j)})) \), which minimizes

\[
\sum_{j=1}^m \frac{1}{n} \sum_{i=1}^n K_{\beta}^{(j)}(X_i) \left( Y_i - \beta_0^{(j)} - \sum_{k=1}^d \beta_k^{(j)} X_{ik} - \frac{t_{jk}}{h_k} \right)^2 + R \left\| (I - P_{add}) \tilde{\beta}^{(j)} \right\|^2. \]

Note that constant reparameterizations correspond to variable transformations.

For ease of notation, assume \( h_1 = \ldots = h_d \) and \( \alpha_1 = \ldots = \alpha_d \), and write \( h \) and \( \alpha \), respectively. The minimization problem is then

\[
\sum_{j=1}^m \frac{1}{n} \sum_{i=1}^n K_{\beta}^{(j)}(X_i) \left( Y_i - \beta_0^{(j)} - \sum_{k=1}^d \beta_k^{(j)} X_{ik} - \frac{t_{jk}}{h} \right)^2 + R \left\| (I - P_0) \text{col}_j(\tilde{\beta}_0^{(j)}) \right\|^2 + R \alpha^2 \sum_{k=1}^d \left\| (I - P_k) \text{col}_j(\tilde{\beta}_k^{(j)}) \right\|^2. \]

As the reparameterization is irrelevant for the cases \( R = 0 \) and \( R = \infty \), we consider the case \( R \) is constant in more detail. As the contribution of the full dimensional estimator in (20) is not vanishing, we need the assumption, that \( nh^d \) tends to infinity. See section 4.3.2 for details. Using asymptotic approximations for the full-dimensional terms, we compare the contributions of the intercept and the slope terms (section 4.3.3). With respect to the bias, \( \alpha \) should be chosen small: A large \( \alpha^2 R \) restricts the slopes to a (possibly inadequate) submodel — this is analogous to the Nadaraya–Watson estimator, where the slope of a local linear estimator is set to zero and the bias contains undesired design dependent terms. However, the variance indicates that \( \alpha^2 R \) corresponds to the ridge–parameter used to stabilize local linear regression in Seifert and Gasser (2000). We choose \( \alpha = 1 \) because the corresponding parameterization is suitable for asymptotic calculations and do not try to optimize this parameter.
3.5 Modeling with pairwise interaction terms

A smooth compromise between a full and an additive model is feasible in dimensions larger than two. However, does this always make sense? It would be a better idea to consider also additive models with bivariate interaction terms. This model is denoted by bivariate–additive model, while univariate–additive denotes the additive model used in the previous sections. The bivariate–additive model is out of scope of this thesis and therefore only sketched here.

**Smooth choice between full and bivariate–additive model**

The results in the previous section do not require \( P_{\text{add}} \) to be a projection to the additive subspace. A choice for \( Z \) for the univariate–additive case is given in section 4.2.1. For the bivariate–additive model

\[
r(t_j) = \sum_{l<\ell} r_{l\ell}(t_{j,l}, t_{j,\ell})
\]

we may define \( P_{\text{add}} = Z^T Z \) and

\[
Z = \text{col}_{1\leq l<\ell\leq d}(Z_{l\ell}) .
\]

With (\( \otimes \) denoting the Kronecker product)

\[
Z_{l\ell:k} = Z_{l\ell:d} \otimes \ldots \otimes Z_{l\ell:1}
\]

and

\[
Z_{l\ell:k} = \begin{cases} 
\frac{1}{m_{l\ell}} \mathbb{1}_{m_{k}}^T & k \notin \{l, \ell\} \\
I_{m_k} & \{1, k\} = \{l, \ell\} \text{ or } k = 1, \{1, 2\} = \{l, \ell\} .
\end{cases}
\]

Redefinition of \( \tilde{P}_{\text{add}} \) is analogous.

**Smooth choice between univariate–additive and bivariate–additive model**

In this case, the bivariate–additive model plays the role of the full model. This is not anymore local: While \( \tilde{A} \) is block–diagonal, \( \tilde{Z} \tilde{A} \tilde{Z}^T \) is not. (However, there may be an efficient way for inverting this matrix.) Hence, Proposition 3.1 does not lead to a dimension reduction. Despite of this algorithmic drawback, this is an interesting question from a statistical point of view.

For dimensions larger than four, the number of output points is of larger order than the squared number of parameters in the bivariate–additive model. Hence limits due to memory consumption will not be imposed by the size of \( \tilde{Z} \tilde{A} \tilde{Z}^T \). In this case, Proposition 3.1 allows a faster calculation in the case where different values of the penalty parameter \( R \) are used.

3.6 Semiparametric modeling

In this section, we sketch applications to semiparametric modeling. Estimating the parametric components using a small penalty \( R \) helps us to avoid problems with singularity. This is especially of interest, as undersmoothing the non–parametric part is indicated to avoid a bias in the parametric components. Statistical properties are not addressed any further, we concentrate on the algorithmic part. It will be seen that dimension reduction still works.

In semiparametric modeling, we assume that the regression function is of the form

\[
E[Y_i|X_i, \chi_i] = r(X_i) + \gamma^T \chi_i,
\]

where \( r \) is smooth as before and \( \chi_i = (\chi_{i,1}, \ldots, \chi_{i,q})^T \) are parametric terms.

The normal equations are of the form

\[
\begin{bmatrix} T \\ \tau \end{bmatrix} = \begin{bmatrix} B & C \\ C^T & D \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} .
\]

30
With $B = \tilde{A} + R(I - \tilde{P}_{\text{add}})$ (dimension $(md) \times (md)$). Let $F$ denote the linear mapping with $T = FY$. The $k^{th}$ column of $C$ may be obtained by $F\text{col}_i(\chi_{i,k})$ for $k = 1, \ldots, q$. Finally, $D$ has small dimension $q \times q$ and may be obtained by

$$D = \frac{1}{n} \sum_{i=1}^{n} K_{H}^{(j)}(X_i)[X_iX_i^\top]$$

We solve the normal equations using Rao and Kleffe (1988) formula (xiv) on page 6. Using the abbreviation $E = D - C^\top B^- C$ (dimension $q \times q$), one solution of (25) is

$$\gamma = E^{-1} \tau - E^{-1} C^\top B^- T$$

and

$$\beta_R = B^{-1}(T - C\gamma).$$

Multiplication by $B^-$ is done as in section 3.4.1. Hence neither memory consumption nor computation time prohibit estimation in semiparametric models.
4 Properties of the penalized estimator

For simplicity, the penalized estimator (16) was introduced on a grid. However, the grid \((t_j)\) has to grow as the bandwidth \(h\) shrinks, because otherwise not all observations are considered and the additive components do not get rid of the curse of dimensionality. Hence, we redo the calculations in a function space. The previous minimization on the grid is an approximation of the integral to be minimized. The main result in section 4.1 is that for \(R \to \infty\) and fixed \(n\), the estimator converges to the additive estimator of Mammen, Linton, and Nielsen. Furthermore, the convex combination property (20) holds. The dependence on the choice of the grid is investigated in section 4.2.

4.1 Comparisons with the estimator of Mammen, Linton, and Nielsen

Here, we apply the results of Mammen et al. (1999) to derive properties for the case \(R \to \infty\). The reader should have a look at section 2.3 before reading this section.

4.1.1 Function spaces and norms

We modify \(\mathcal{F}\) slightly (compared to the definition in section 2.3.2). The \(X_i\) are assumed to be in \([0, 1]^d\) which conforms to the assumptions in Theorem MLN:4’. Further we add some \(r^{0,k}\) \((k = 0, \ldots, d)\) to the tuple of functions for later extension. In the definition below the index \(i\) starts thus at zero instead of one.

**Normed vector space**

Define the space of tuples of \((n+1)(d+1)\) functions

\[
\mathcal{F} = \left\{ (r^{i,\ell}| i = 0, \ldots, n; \ell = 0, \ldots, d) \mid r^{i,\ell} : [0, 1]^d \to \mathbb{R} \right\}.
\]

Define \(r_Y \in \mathcal{F}\) with

\[
r_Y^{i,\ell}(x) = \begin{cases} Y_i & \text{for } i > 0 \text{ and } \ell = 0 \\ 0 & \text{else} \end{cases}
\]

Define the semi–norm \(\| \cdot \|_*\) by

\[
\|r\|_*^2 = \int \frac{1}{n} \sum_{i=1}^{n} \left[ r^{i,0}(\underline{x}) + \sum_{k=1}^{d} r^{i,k}(\underline{x}) X_{i,k} - x_k \right]^2 K_h(X_i, \underline{x}) d\underline{x}.
\]

Note that the sum starts at \(i = 1\). Therefore, \(\|r\|_*\) does not depend on \(r^{0,\ell}\).

**Subspace for full and additive estimation**

Define the subspace

\[
\mathcal{F}_{\text{full}} = \left\{ (r^0, \ldots, r^d) \mid r^\ell : [0, 1]^d \to \mathbb{R}, \ell = 0, \ldots, d \right\}
\]

and embed \(r \in \mathcal{F}_{\text{full}}\) into \(\mathcal{F}\) by

\[
r^{i,\ell}(\underline{x}) = r^\ell(\underline{x}) \quad \forall i = 0, \ldots, n; \ell = 0, \ldots, d; \text{ and } (r^{i,\ell}) \in \mathcal{F}.
\]

The subspace \(\mathcal{F}_{\text{add}} \subset \mathcal{F}_{\text{full}}\) of additive functions is defined by

\[
\mathcal{F}_{\text{add}} = \{ r \in \mathcal{F}_{\text{full}} \mid r^0(\underline{x}) \text{ is additive; for } k = 1, \ldots, d, r^k(\underline{x}) \text{ depends only on } x_k \}.
\]
Penalty for non–additivity
Define the $L^2$–norm on $\mathcal{F}$ by
\[
\|r\|_{L^2}^2 = \frac{1}{n+1} \sum_{i=0}^{n} \sum_{\ell=0}^{d} \int \left| r^{i,\ell}(x) \right|^2 dx .
\]

The ordinary local linear estimator is obtained by finding the minimizer $r$ of $\|r_Y - r\|^2_2$. By construction, $r^{i,\ell}_Y$ is zero for $i = 0$, and this term may be used to impose an additional regularization term on $r$. Denote by $P_0$ the projection from $\mathcal{F}$ to $\mathcal{F}_{\text{full}}$ which preserves the components $r^{0,\ell}(x)$ for $\ell = 0, \ldots, d$. Thus, $P_0$ restricted to $\mathcal{F}_{\text{full}}$ is the identity. Denote by $P_{\text{add}}$ the orthogonal projection from $\mathcal{F}_{\text{full}}$ to $\mathcal{F}_{\text{add}}$, orthogonal with respect to the $L^2$–norm. $P_{\text{add}}$ is extended from $\mathcal{F}_{\text{full}}$ to $\mathcal{F}$ as $P_{\text{add}} P_0$.

Define the semi–norm $\|\cdot\|_R$ on $\mathcal{F}$ by
\[
\|r\|^2_R = \|r\|^2_\ast + R \| (I - P_{\text{add}}) r \|_{L^2}^2 .
\]

The penalized estimator $\hat{r}_R(x)$ is defined as the minimizer for $r \in \mathcal{F}_{\text{full}}$ of
\[
\|r_Y - r\|^2_R = \frac{1}{n+1} \sum_{i=0}^{n} \sum_{\ell=0}^{d} \int \left[ Y_i - r^{0,\ell}(x) - \sum_{k=1}^{d} r^{k}(x) \frac{X_{i,k} - x_k}{h} \right]^2 K_h(X_i, x) dx + R \| (I - P_{\text{add}}) r \|_{L^2}^2 .
\]

This is the continuous equivalent of the penalized estimator $\hat{r}_R$ defined in (16).

4.1.2 Some operators
Define $\hat{V}_{\ell,l}$ with $\ell, l = 0, \ldots, d$ as in (MLN:31) on page 15 in section 2.3.

Define the linear operator $S_* : \mathcal{F}_{\text{full}} \to \mathcal{F}_{\text{full}}$
\[
\begin{pmatrix}
  r^{0}(x) \\
  \vdots \\
  r^{d}(x)
\end{pmatrix} \mapsto \begin{pmatrix}
  \hat{V}_{0,0}(x) r^{0}(x) + \hat{V}_{0,1}(x) r^{1}(x) + \ldots + \hat{V}_{0,d}(x) r^{d}(x) \\
  \vdots \\
  \hat{V}_{d,0}(x) r^{0}(x) + \hat{V}_{d,1}(x) r^{1}(x) + \ldots + \hat{V}_{d,d}(x) r^{d}(x)
\end{pmatrix} .
\]

$S_*$ has the property
\[
\langle r, S_* r \rangle_{L^2} = \|r\|^2_\ast .
\]

The normal equations for the local linear estimator are as in (MLN:31)
\[
S_* \hat{r}_{\text{full}} = \hat{L} .
\]

Define a symmetric, linear operator $S_R$ with $\|r\|^2_R = \langle r, S_R r \rangle_{L^2}, \forall r \in \mathcal{F}_{\text{full}}$
\[
S_R : \mathcal{F}_{\text{full}} \to \mathcal{F}_{\text{full}}, r \mapsto S_* r + R (r - P_{\text{add}} r) .
\]

The normal equations for the penalized local linear estimation as defined in (26) are
\[
S_R \hat{r}_R = \hat{L} .
\]

Define the linear operator $S_{\text{add}}$ as $P_{\text{add}} S_* P_{\text{add}}$ restricted to $\mathcal{F}_{\text{add}}$. $S_{\text{add}}$ is a linear operator $\mathcal{F}_{\text{add}} \to \mathcal{F}_{\text{add}}$. The normal equations for the additive estimator $\hat{r}_{\text{add}}$ defined in (2) at page 15 are
\[
S_{\text{add}} \hat{r}_{\text{add}} = P_{\text{add}} \hat{L} .
\]
4.1.3 Projection to $\mathcal{F}_{\text{add}}$

In the following, we assume that the semi-norm $\| \cdot \|_s$ is a norm on $\mathcal{F}_{\text{add}}$. This assumption is justified by Theorem MLN:1', which says that the normal equations (MLN:44)–(MLN:47) have a unique solution with probability tending to one.

Define the projection $\Pi_s : \mathcal{F}_{\text{full}} \to \mathcal{F}_{\text{add}}$ by

$$\Pi_s \hat{r} = \arg \min_{\hat{r}_{\text{add}} \in \mathcal{F}_{\text{add}}} \| \hat{r} - \hat{r}_{\text{add}} \|_s^2, \quad \forall \hat{r} \in \mathcal{F}_{\text{full}}.$$ 

The normal equations for this minimization problem may be written as

$$\tilde{M}_j(x_j) \left( \hat{r}_{\text{add},j}(x_j) + \hat{r}_{\text{add},0} - \hat{m}_j(x_j) \right) = -\sum_{k \neq j} \int \tilde{S}_{k,j}(x_k, x_j) \left( \hat{r}_{\text{add},k}(x_k) - \hat{m}_j(x_j) \right) dx_k. \quad (28)$$

For the definition of $\hat{m}_j$ and $\hat{m}_j$ see appendix C.1.1. The identifiability condition is the same as in (MLN:45). The only difference between (28) and (MLN:44) is that $\hat{m}$ replaces $\hat{m}$. The one-dimensional local linear estimators $\hat{m}_j$ and $\hat{m}_j$ are defined in (MLN:39/40). Recall that (MLN:44) and (MLN:45) define normal equations for

$$\arg \min_{\hat{r}_{\text{add}} \in \mathcal{F}_{\text{add}}} \| r_Y - \hat{r}_{\text{add}} \|_s^2.$$ 

See appendix C.1 for the details and the proof of the following Lemma.

**Lemma 4.1** Under the assumptions (MLN:B1), (MLN:B2')–(MLN:B4'), and (C2) given in section 2.3.5, $\Pi_s$ and $\mathcal{S}_{\text{add}}^{-1}$ are, with probability tending to one, continuous linear operators.

Lemma 4.1 implies that the assumptions in Lemma 4.2 and Theorem 4.3 hold with probability tending to one.

**Lemma 4.2**

1. $\hat{r}_{\text{add}}$ is unique, if and only if $\hat{r}_R$ is unique for $R > 0$.

2. If there exist positive constants $C_1$, $C_2$, and $C_3$ such that

$$\| r \|_s \geq C_1 \| r \|_{L^2} \quad \forall r \in \mathcal{F}_{\text{add}},$$

$$\| r \|_s \leq C_2 \| r \|_{L^2} \quad \forall r \in \mathcal{F}_{\text{full}},$$

$$\| \Pi_s r \|_{L^2} \leq C_3 \| r \|_{L^2} \quad \forall r \in \mathcal{F}_{\text{full}},$$

then for every $R > 0$ there exists a positive constant $C_4$

$$\| r \|_{R} \geq C_4 \| r \|_{L^2} \quad \forall r \in \mathcal{F}_{\text{full}}.$$ 

The existence of $C_1$ to $C_3$ is equivalent to the assumption that $\mathcal{S}_{\text{add}}$ has a continuous inverse, $\mathcal{S}$ is continuous, and $\Pi_s$ is continuous. The existence of $C_4 > 0$ in the above Lemma implies that $\mathcal{S}_R$ may be inverted, see appendix C.2.3. This is useful for solving the normal equations (27) for $\hat{r}_R$.

**Convergence of $\hat{r}_R$ to $\hat{r}_{\text{add}}$ for $R \to \infty$ and $n$ fixed.**

Let us now show, that for $R$ tending to infinity, the penalized estimator $\hat{r}_R$ converges to the additive estimator $\hat{r}_{\text{add}}$ of MLN introduced in section 2.3. That estimator is known to be asymptotically oracle optimal (section 2.3.5).

**Theorem 4.3** Under the conditions in Lemma 4.2, we have

$$\| \hat{r}_R - \hat{r}_{\text{add}} \|_{L^2} \leq \frac{(1 + C_3)^2}{R} 2C_2 \| (r_Y - \hat{r}_{\text{add}}) \|_s \quad (n \text{ fixed})$$

Thus, the rate of convergence of $\hat{r}_R$ to $\hat{r}_{\text{add}}$ for fixed $n$ and $R \to \infty$ is $R^{-1}$. Note that $\| r_Y - \hat{r}_{\text{add}} \|_s^2$ is an integrated sum of weighted squared residuals.
However, the assumption of uniform (with probability tending to one) bounds for $C_2$ and $C_3$ is generally not fulfilled. Therefore, finding a minimal rate for $R$, i.e. a sequence $R_n$ such that $\hat{r}_{R_n}$ has the optimal rate for $n$ tending to infinity needs further investigation. This fact is illustrated as follows: Because we assume in (MLN:B1) bounded kernel weights, $S_r$ is continuous. However, if we use bandwidths $h$ of order $n^{-1/5}$ in high dimensions, then $K_h(x,x)/n$ is of order $n^{(d/5)-1}$. Hence, for $d > 5$, we do not expect $C_2$ to be uniformly bounded for large $n$.

4.2 Finite grid properties

In the following, we give possible choices $Z$ and $\tilde{Z}$ for the decomposition of $P_{add}$ and $\tilde{P}_{add}$, respectively. In section 4.2.3, the relevance of the output grid is illustrated for a special case.

4.2.1 Specification of $Z$

As announced in section 3.2.2 (dimension reduction of the Nadaraya–Watson estimator with penalty), we introduce now a suitable choice for $Z$. Matrix multiplication by $Z$ and $Z^\top$ corresponds to calculation of averages. Because $A^{(R)}$ is diagonal, $ZA^{(R)}Z^\top$ may be calculated directly, using terms known from ANOVA. The sums are approximations of integrals.

Numeration of the output grid
The special form of the matrix $Z$ depends on the numeration of the grid points $t_j$. Therefore, we define the relationship between $j \in \{1, \ldots, m\}$ and $(j_1, \ldots, j_d) \in \{1, \ldots, m_1\} \times \ldots \times \{1, \ldots, m_d\}$ to be

$$j = j(j_1, \ldots, j_d) := j_1 + \sum_{k=2}^d (j_k - 1) \prod_{1 \leq \ell < k} m_\ell. \quad (29)$$

Consequently,

$$t_{j(j_1, \ldots, j_d)} = [t_{j_1}^1, \ldots, t_{j_d}^d]^\top.$$

This kind of inverse lexicographical ordering is the usual way to store multi-dimensional arrays in FORTRAN. $Z$ may be separated into $d$ blocks

$$Z = \begin{bmatrix}
Z_{01} \\
Z_2 \\
\vdots \\
Z_d
\end{bmatrix}$$

corresponding to the $d$ additive components.

Specification of $Z_\ell$ using Kronecker products
Denote by $A \otimes B$ the Kronecker product of $A$ and $B$, i.e. the block matrix with entries $((a_{ij}B))$. As an example, let $e_\ell$ be the $\ell$th unit vector and $B_\ell$, $\ell = 1, \ldots, N$ arbitrary matrices of the same dimension. Then,

$$\text{diag}_\ell(B_\ell) = \sum_\ell (e_\ell e_\ell^\top) \otimes B_\ell \quad (30)$$

$$\text{col}_\ell(B_\ell) = \sum_\ell e_\ell \otimes B_\ell.$$

For $k = 1, \ldots, d$, define

$$Z_{0k} := \frac{1}{\sqrt{m_d}} I_{m_d}^\top \otimes \ldots \otimes \frac{1}{\sqrt{m_{k+1}}} I_{m_{k+1}}^\top \otimes I_{m_k} \otimes \frac{1}{\sqrt{m_{k-1}}} I_{m_{k-1}}^\top \otimes \ldots \otimes \frac{1}{\sqrt{m_1}} I_{m_1}^\top,$$
4.2 Finite grid properties

Then for every $\mathbf{Z} \in \{1, \ldots, m\}$, we have the following properties: For ease of reading we fix the indices $j \in \{1, \ldots, m\}$ and $j_k \in \{1, \ldots, m\}$ for some $k = 1, \ldots, d$. Define the subset $\mathcal{J}_{j_k}$ of indices $j$ corresponding to $t_j$ with $k$th coordinate equal to $t_{jk}$. ($\mathcal{J}_{j_k}$ contains $\frac{m}{m_k}$ elements.)

$$\mathcal{J}_{j_k} = \{ j \mid t_{jk} = t_k \}$$

Then for every $m$ dimensional vector $\mathbf{v}$ we have

$$\mathbf{Z}_{jk} = \mathbf{v}$$

$$\mathbf{Z}_{jk} = \mathbf{v}$$

$$\mathbf{Z}_{jk} = \mathbf{v}$$

$$\mathbf{Z}_{jk} = \mathbf{v}$$

Integral approximation

Choose some smooth function $v(t)$ and define $\mathbf{v} = [v(t_1), \ldots, v(t_m)]^T$. We will use the following integral approximation for $m_1, \ldots, m_d \to \infty$ (assuming equidistant grid)

$$\sqrt{\frac{m_k}{m}} \mathbf{Z}_{jk} \to \int \int_{[0,1]^{d-1}} v(x_1, \ldots, x_d) \prod_{1 \leq \ell \leq d, \ell \neq k} dx_\ell \quad \text{with} \quad x_k := t_{jk}.$$
4.2.2 Specification of \( \tilde{Z} \)

Recall the definition of the additive subspace in section 3.4 (local linear estimator with penalty on intercept and slopes). There is no interdependence between terms representing intercept \((\tilde{\beta}_0^j ; j = 1, \ldots, m)\) and partial derivatives \((\tilde{\beta}_k^j ; j = 1, \ldots, m; k = 1, \ldots, d)\). Hence, we apply \( Z \) (as defined in section 4.2.1 for the Nadaraya–Watson case) to the parameters representing intercept terms. Moreover, the different partial derivatives do not depend on each other, hence we choose \( Z \) as

\[
\tilde{Z} = \begin{bmatrix} Z_{\text{col} j=1, \ldots, m} (\tilde{\beta}_0^j) \\
Z_{01} \text{col} j=1, \ldots, m (\tilde{\beta}_1^j) \\
\vdots \\
Z_{0d} \text{col} j=1, \ldots, m (\tilde{\beta}_d^j) \end{bmatrix}.
\]

Note that the \( \tilde{\beta}_j^j \) are stored in \( \tilde{Z} \in \mathbb{R}^{(d+1)m} \) as follows:

\[
\tilde{\beta} = \text{col} j=1, \ldots, m \left( \begin{bmatrix} \tilde{\beta}_0^j \\
\tilde{\beta}_1^j \\
\vdots \\
\tilde{\beta}_d^j \end{bmatrix} \right) = \text{col} j=1, \ldots, m \left( \text{col} \ell=0, \ldots, d \left( \begin{bmatrix} \tilde{\beta}_0^j \\
\tilde{\beta}_1^j \\
\vdots \\
\tilde{\beta}_d^j \end{bmatrix} \right) \right).
\]

Hence, the above choice of \( \tilde{Z} \) may be written as

\[
\tilde{Z} = \begin{bmatrix} Z \otimes [1, 0, 0, \ldots, 0] \\
Z_{01} \otimes [0, 1, 0, \ldots, 0] \\
\vdots \\
Z_{0d} \otimes [0, 0, \ldots, 0, 1] \end{bmatrix}.
\]

4.2.3 Illustration of grid dependence

Using infinite grids (i.e., function spaces) and an additive regression function, \( \hat{r}_R \) has a mean integrated squared error of \( O(h^4 + (nh)^{-1}) \) if \( R \) grows fast enough (section 4.1, but minimal rate of \( R \) is not known). The question is, whether this rate is also achievable for a finite output grid.

To avoid oblique projections, we assume that the design density \( f \) is constant. Additionally, we want the \( S^{(j)} \), as defined in formula (17) on page 27, to be independent of \( j \). To achieve this, both design and output points are assumed to form an equidistant grid and the spacings of the output grid are a multiple of the spacings of the design. In fact, the output points form a sub–grid of the design points. In this case, the local linear estimator does not depend on the values of the slope parameters. This implies that Nadaraya–Watson and local linear estimators are equal. Because the number of output points will be required to grow with \( n \), the above scenario applies only to low dimensions.

To avoid boundary effects, we need data in a neighborhood of \([0, 1]^d\). For a formal definition, see appendix D.2. In this special case, \( P^{(R)}_A \) is equal to \( P_{\text{add}} \) because \( a_j = a \) for all \( j \) — if \( h < \tilde{h} \). Recall that \( a \) is an approximation of the design density \( f \). The parameter \( a_j \) is defined in section 3.2.1 equation (5) for the Nadaraya–Watson case and section 3.3.2 for the local linear case.

For ease of notation, we assume that \( m_1 = \ldots = m_d \) and that the bandwidths \( h_1, \ldots, h_d \) are equal, and write \( h \) (as in section 2.3.2).

Bias of the penalized estimator

Let \( B_{\text{nl}} \) be the first order bias = \( O(h^2) \) term of \( \hat{r}_H \). Then the bias of \( \hat{r}_R \) is

\[
\frac{1}{m} \| \text{bias}(\hat{r}_R) \|^2 = \frac{1}{m} \left\| P_{\text{add}} B_{\text{nl}} + (I - P_{\text{add}}) \left( \frac{a}{a + R} B_{\text{nl}} + \frac{R}{a + R} \right) \right\|^2.
\]

In general, the term \( P_{\text{add}} B_{\text{nl}} \) is still of order \( h^2 \). If the true regression function is additive, the squared bias is \( O(h^4) \), otherwise it is \( O(\max(h^4, (\frac{R}{a + R})^2)) \).

37
It turns out that the choice of the output grid is not crucial for the rate of convergence of the bias.

**Covariance of the local linear estimator**

The following proposition generalizes Theorem 2.1 of Ruppert and Wand (1994) which gives the variance of the local linear estimator.

**Proposition 4.4** Assume:

- The bandwidth $h$ tends to zero and $nh^d$ tends to infinity for increasing $n$. The bandwidth–matrix $H$ is defined as $hI$.
- Let $K$ be a univariate kernel function. Assume that $K$ is bounded, symmetric around zero, and has support $[-1,1]$. Define $\mu_2(K) = \int u^2K(u)du$, which is assumed to be nonzero.
- The weights $K_H^{(j)}(x)$ are defined as $\prod_{\ell=1}^{d} h^{-1}K((x_\ell - t_{j,\ell})/h)$.
- Assume that the conditional variance $\sigma^2$ of $Y$ given $X$ is constant. The regression function $r(\cdot)$ is twice continuously differentiable in a neighborhood of $[0,1]^d$. The design density $f(x) = f$ is constant.

Define $K^{*2}$ as the convolution of the kernel $K$:

$$K^{*2}(u) = K*K(u) \equiv \int K(v)K(v+u)dv.$$  

Denote the covariance of two local linear estimations at $t_j$ and $t_k$ by

$$[\Sigma_{ll}]_{jk} = \text{Cov}(\hat{r}_{ll}(t_j), \hat{r}_{ll}(t_k)\mid X_1, \ldots, X_n).$$

Then, the covariance is asymptotically

$$[\Sigma_{ll}]_{jk} = \sigma^2 fnh^d \prod_{\ell=1}^{d} K^{*2}\left(t_{j,\ell} - t_{k,\ell}\right) \{1 + o(1)\}.$$  

**Integrated variance of $\hat{R}_R$**

Let $\Sigma_R$ be the covariance matrix of $\hat{R}_R$:

$$[\Sigma_R]_{jk} = \text{Cov}(\hat{r}_R(t_j), \hat{r}_R(t_k)\mid X_1, \ldots, X_n).$$

Its trace divided by the number of output points

$$\frac{1}{m}\text{tr}(\Sigma_R)$$

is the integrated variance. The choice of the output grid is not crucial for non–additive regression functions, as $R \to 0$ will be chosen. Let us evaluate the rate of convergence for additive regression functions, where $R \to \infty$.

Using formula (11) on page 22, we get

$$\frac{1}{m}\text{tr}(\Sigma_R) = \left(\frac{a}{a+R}\right)^2 \frac{1}{m}\text{tr}(\Sigma_{ll}) + \left(1 - \left(\frac{a}{a+R}\right)^2\right) \frac{1}{m}\text{tr}(Z\Sigma_{ll}Z^\top).$$  

We want to establish that formula (31) achieves the rate $(nh)^{-1}$, which is optimal for additive and univariate local linear estimators. The term with $\text{tr}(\Sigma_{ll})$ is asymptotically negligible if
\[(\frac{a}{a + R})^2 = o(h^{d - 1})\]: \(R\) should grow faster than \(h^{-(d - 1)/2}\). The goal is to establish that the term with \(\text{tr}(Z\Sigma_{ll}Z')\) achieves the rate \((nh)^{-1}\). This term is decomposed as follows:

\[
\frac{1}{m} \text{tr}(Z\Sigma_{ll}Z') = \sum_{\ell=1}^{d} \frac{1}{m} \text{tr}(Z_{\ell\ell}\Sigma_{ll}Z'_{\ell\ell}) - \frac{d - 1}{m^2} (\frac{1}{m} \Sigma_{ll}1_m).
\] (32)

The last term in (32) is of smaller order and, since it is negative, we do not care about it. For the calculation of the order of formula (32), we distinguish between non–overlapping \((m_1 h \to c \in (0, \frac{1}{2}))\) and strongly overlapping \((m_1 h \to \infty)\) smoothing windows.

**Non–overlapping smoothing windows**

For non–overlapping smoothing windows, the estimates for different output points are independent. In this case, \(\Sigma_{ll}\) is a multiple of the identity:

\[
\Sigma_{ll} = \frac{\sigma^2(K^{*2}(0))}{fnh^d} \{1 + o(1)\} I.
\]

Because of the special choice of the design, the terms of smaller order in (32) are equal for all output points. Hence,

\[
\frac{1}{m} \text{tr}(Z\Sigma_{ll}Z') = \frac{\sigma^2K^{*2}(0)d}{fnh^d} \frac{1}{m} \text{tr}(P_{adl}) \{1 + o(1)\}
\]

\[
= \frac{\sigma^2K^{*2}(0)}{fnh} \left( \frac{K^{*2}(0)}{m_1h} \right)^{d-1} \left( d - \frac{d-1}{m_1} \right) \{1 + o(1)\}.
\]

Since we assume the spacings between the output points \(\frac{1}{m_1h}\) to be at least as large as the smoothing windows \((2h)\), we get \(m_1 h \leq \frac{1}{2}\). If \(m_1 h \to c \in (0, \frac{1}{2})\), then the variance has the desired rate.

As shown in appendix D.3, the integrated squared kernel \(K^{*2}(0)\) is bounded from below by \(\frac{1}{2}\) and achieves its minimum for the uniform kernel \(\frac{1}{2}[\frac{1}{2}, \frac{1}{2}]\). Using a grid with \(m_1 h = \frac{1}{2}\) and the uniform kernel, we achieve the optimal constant

\[
\frac{1}{m} \text{tr}(Z\Sigma_{ll}Z') \approx \frac{d\sigma^2}{2fnh}.
\]

Hence for the uniform kernel, for \(R \to \infty\) the integrated variance is asymptotically equal to the variance of the sum of \(d\) independent univariate local linear oracle estimators. For other kernel functions or larger gaps of the output grid, variability increases.

**Overlapping smoothing windows**

The penalized estimator \(\hat{r}_R\) is a design–dependent linear mapping of \(\hat{r}_{ll}\). Therefore, its covariance matrix is known. Above we assumed that the number of output points does not grow too fast. Here, we consider the case when the number of output points in the smoothing windows \((m_1 h)\) tends to infinity. This allows us to approximate sums by integrals.

\[
\frac{1}{m} \text{tr}(Z_{\ell\ell}\Sigma_{ll}Z'_{\ell\ell}) \to \frac{\sigma^2}{fnh^d} \frac{m_\ell}{m} K^{*2}(0) \prod_{k=1,d \neq \ell} m_k \int_{-1}^{1} (1 - |v|)K^{*2}(\frac{v}{h}) dv
\]

\[
\leq \frac{\sigma^2K^{*2}(0)}{fnh}.
\]

See appendix D.4 for details.
For comparison with the case of non–overlapping smoothing windows, consider
\[
\frac{1}{m} \text{tr} \left( Z_0 \Sigma h Z_0^\top \right) = \frac{\sigma^2 K^{2(0)}}{f n h} \{1 + o(1)\} \left\{ \begin{array}{ll}
1 & \text{if } m_1 h \rightarrow \infty \\
\left( \frac{K^{2(0)}}{m_1 h} \right)^{d-1} & \text{if } m_1 h \leq \frac{1}{2} \end{array} \right.
\]
Note that, for the non–overlapping case \( m_1 h \leq \frac{1}{2} \), we have
\[
\frac{K^{2(0)}}{m_1 h} \geq 1.
\]
Thus, we pay a price in terms of the constant for non–overlapping smoothing windows.

**Conclusion**
In the case of equidistant design, the variance achieves the optimal rate for the additive model, if
- the bandwidth \( h \) has the optimal rate \( n^{-1/5} \).
- the output grid grows fast enough: \( \frac{1}{m_1} \leq 2h \).
- the penalty \( R \) grows at a rate of at least \( h^{-(d-1)/2} \).

However, if the true regression function \( r \) is not additive, the bias is not vanishing — unless \( R \) is tending to zero. In this case, choosing \( R = O(h^2) \) gives us the desired rate \( h^4 \) for the mean squared bias, and the choice of the output grid does not affect convergence rates.

### 4.3 About parameters

For the penalized local linear estimator with penalty parameter \( R \), the cases \( R = 0 \) and \( R = \infty \) are understood. What about \( R \approx 1 \)? Furthermore, the effect of reparameterization of the slope (section 3.4.3) cannot be investigated in the cases \( R = 0 \) and \( R = \infty \). In section 4.3.2 we will investigate local (\( n \)-dependent) alternatives to the additive model: For every \( n \), the deviation of the true regression function from the additive model is chosen such that \( R = 1 \) is optimal. (The required asymptotic expansions are in section 4.3.1.) Consequently, in situations where \( R = 1 \) is expedient, the bandwidth should be of order \( n^{-1/(4+d)} \). This facilitates asymptotic expansions used to compare different parameterizations of intercept and slopes.

#### 4.3.1 Influence of the parameters on variance and bias

The effect of the variability of \( Y_i \) and the curvature of the regression function (i.e. the \( h^2 \) term with the Hessian matrix in the bias) is known. In a first step we have a closer look at the dependence of \( \hat{\beta}_R \) on \( \text{col}_j(\mathbf{T}^{(j)}) \). Later, this is combined with a Taylor expansion of the regression function.

We assume that the design density is bounded away from 0 and \( \infty \). Furthermore, the output grid is dense enough. Alternatively, the calculations can also be done as in section 4.1 (see also appendix C.2.2).

For the definition of the additive components, we distinguish between small and large \( R \). For small \( R \), we assume that \( n h^d \rightarrow \infty \) and therefore \( S^{(j)} \) is non–singular. Then
\[
\hat{\beta}_{\text{add}, R} = \mathbf{Z}^\top \left( (I - \mathbf{Z} \mathbf{Z}^\top) + \mathbf{Z} \text{diag}_j(S^{(j)}(S^{(j)} + RI)^{-1}) \mathbf{Z}^\top \right)^{-1} \text{col}_j \left( (S^{(j)} + RI)^{-1} \mathbf{T}^{(j)} \right).
\]

Because \( \hat{\beta}_{\text{add}, R} \) is a projection of \( \hat{\beta}_{ll} \), we expect that the mapping \( \text{col}_j(\mathbf{T}^{(j)}) \mapsto \hat{\beta}_{\text{add}, R} \) is \( O(I) \).
For large $R$, we replace $(S^{(j)} + RI)^{-1}$ by $R^{-1}(R^{-1}S^{(j)} + I)^{-1}$ and obtain

$$
\hat{\beta}_{\text{add}, R} = \tilde{Z}^\top \left( (I - \tilde{Z}\tilde{Z}^\top) + \tilde{Z}\text{diag}_j(S^{(j)})(R^{-1}S^{(j)} + I)^{-1}\tilde{Z}^\top \right)^{-1} \tilde{Z}\text{col}_j \left( (R^{-1}S^{(j)} + I)^{-1}\tilde{Z}^{(j)} \right).
$$

We have the bound $(R^{-1}S^{(j)} + I)^{-1} \preceq I$. For the other matrix inversion we use

$$
\tilde{Z}\text{diag}_j \left( S^{(j)}(R^{-1}S^{(j)} + I)^{-1} \right) \tilde{Z} \succeq \frac{1}{R^{-1}\|A\|_{\sup} + 1} \tilde{Z}A\tilde{Z}^\top,
$$

and $\tilde{Z}A\tilde{Z}^\top$ are basically two-dimensional marginal density estimates. Hence, $((I - \tilde{Z}\tilde{Z}^\top) + \ldots)^{-1}$ is $O(R^{-1}\|A\|_{\sup} + 1)$ for $h \geq h_{\text{add}} \propto n^{-1/5}$ and we write

$$
\hat{\beta}_{R} = (S^{(j)} + RI)^{-1}T^{(j)} + R(S^{(j)} + RI)^{-1}\hat{\beta}_{\text{add}, R}^{(j)}.
$$

### Taylor approximation

Consider the following decomposition of $Y_i$:

$$
Y_i = r(X_i) + \varepsilon_i = r(t_j) + (X_i - t_j)^\top \nabla_r(t_j) + O(\|X_i - t_j\|^2) + \varepsilon_i,
$$

where $\nabla_r(x)$ denotes the gradient $\left( \frac{\partial}{\partial x_1} r(x), \ldots, \frac{\partial}{\partial x_d} r(x) \right)^\top$. For the deterministic part, we get

$$
E(T^{(j)}|X_1, \ldots, X_n) = \frac{1}{n} \sum_{i=1}^{n} K_H^{(j)}(X_i) r(X_i) \begin{bmatrix} 1 \\ (X_{i1} - t_{j1})/h_1 \\ \vdots \\ (X_{id} - t_{jd})/h_d \end{bmatrix} = S^{(j)} \begin{bmatrix} r(t_j) \\ \nabla_r(t_j) \end{bmatrix} + O(h^2 \max(1, \frac{1}{nh^2}) I).
$$

The term $\max(1, \frac{1}{nh^2})$ reflects the fact, that on the one hand $\frac{1}{n} K_H^{(j)}(X_i)$ is $O(\frac{1}{nh^2})$. On the other hand, if $nh^d \to \infty$, the sum over $i$ of $\frac{1}{n} K_H^{(j)}(X_i)$ is an estimate of the design density.

For the additive part, $\text{col}_j(r(t_j), \nabla_r(t_j))$ is replaced by some additive expression denoted by $\tilde{P}_A^{(R)} \left[ \frac{r}{\nabla_r} \right](t_j)$ with

$$
\text{col}_j \left( \tilde{P}_A^{(R)} \left[ \frac{r}{\nabla_r} \right](t_j) \right) = \tilde{P}_A^{(R)} \text{col}_j \left( \left[ \frac{r(t_j)}{\nabla_r(t_j)} \right] \right).
$$

For the $O(\ldots)$ terms see appendix E.1.

### Conditional variance

The conditional variance is

$$
\text{var}(T^{(j)}|X_1, \ldots, X_n) = \begin{cases} O\left(\frac{1}{nh^2} \right) & nh^d \to \infty \\ O\left(\frac{1}{n^2h^2} \right) & nh^d \to 0. \end{cases}
$$

For the additive part, we have

$$
\text{var} \left( \hat{\beta}_{\text{add}, R}^{(j)} \right) = \begin{cases} O\left(\frac{1}{nh^{d/2}} \right) & nh^d \to \infty \\ O(\max(1, \frac{1}{nh^{d/2}}) \frac{1}{nh}) & R \text{ bounded away from zero}. \end{cases}
$$
Conditions on \( R \) and \( h \)
When \( h \) has the appropriate rate of the full-dimensional model \( n^{-1/(4+d)} \), the penalty \( R \) may be chosen arbitrarily \( (0, \infty) \). If the rate of \( h \) is smaller than \( n^{-1/5} \) (the rate of the additive model), \( R \) has to be large enough, such that the \( O(\ldots) \) terms are sufficiently small.

If \( nh^d \to \infty \), we use the approximation

\[
\widehat{\beta}_R = (S^{(j)} + RI)^{-1}\left\{ S^{(j)} \left[ \frac{r}{\nabla r} \right] + \hat{P}_A^{(R)} \left[ \frac{r}{\nabla r} \right] \right\} + O(h^2) + O_P \left( \frac{1}{(R+1)\sqrt{nh^d}} + \frac{1}{\sqrt{nh^d}} \right)
\]

otherwise, we use

\[
\widehat{\beta}_R = (S^{(j)} + RI)^{-1}\left\{ S^{(j)} \left[ \frac{r}{\nabla r} \right] + \hat{P}_A^{(R)} \left[ \frac{r}{\nabla r} \right] \right\} + O \left( k^2 + \frac{h^2}{Rnh^d} \right) + O_P \left( \frac{1}{Rnh^d} + \frac{1}{\sqrt{nh^d}} \right).
\]

If the true regression function is additive, the bias is \( O(h^2) \) or \( O(h^2 + h^2/(Rnh^d)) \). Otherwise, the \( \hat{P}_A^{(R)} \) term also contributes to the bias.

### 4.3.2 Local alternatives to the additive model

We consider a regression function in the neighborhood of the additive model:

\[
r_n(\mathbf{x}) = (1 - \lambda_n)r_{\text{add}}(\mathbf{x}) + \lambda_nr_{-\text{add}}(\mathbf{x}),
\]

where \( r_{\text{add}} \) is additive and \( r_{-\text{add}} \) is not. We choose \( \lambda_n \) such that \( R \approx 1 \) is optimal. For \( R \approx 1 \), the stochastic term \( O_P\left( \frac{1}{(R+1)\sqrt{nh^d}} \right) \) implies that the leading term of the variance is \((nh^d)^{-1}\) and we will choose \( h \propto n^{-1/(4+d)} \). Hence, \( S^{(j)} \) is \( O(1) \). The bias is of the form \( O(h^2 + \frac{R}{\pi\lambda_n}) \) and the stochastic term is \( O_P\left( \frac{1}{\sqrt{nh^d}} + \frac{1}{R\sqrt{nh^d}} \right) \). Since \( R \approx 1 \) is the minimum, we choose \( \lambda_n \propto h^2 \). This minimum is within the range, where the above approximations are accurate.

This is the reason for assuming \( nh^d \to \infty \) if \( R \neq 1 \).

### 4.3.3 On different penalties for intercept and slopes

In section 3.4.3, the question was whether different penalties for intercept and slopes should be used. This is equivalent to a reparameterization, where the \( k^{th} \) slope is multiplied by \( \alpha_k \). We assume \( \alpha_1 = \ldots = \alpha_d \) and write for simplicity \( \alpha \). Define \( \hat{P}_A^{(R)} \left[ \frac{r}{\alpha \nabla r} \right] (t_j) \) by

\[
\text{col}_j \left( \hat{P}_A^{(R)} \left[ \frac{r}{\alpha \nabla r} \right] (t_j) \right) = \hat{P}_A^{(R)} \text{col}_j \left( \left[ \frac{r(t_j)}{\alpha \nabla r(t_j)} \right] \right),
\]

where \( \hat{P}_A^{(R)} \) is defined as \( \hat{P}_A^{(R)} \) with \( S^{(j)\alpha} \) instead of \( S^{(j)} \), see appendix E.2.

Assume \( nh^d \to \infty \). Then

\[
\widehat{\beta}_R = \hat{P}_A^{(R)} \left[ \frac{r}{\alpha \nabla r} \right] (t_j) + (S^{(j)} + RI)^{-1}S^{(j)} \left( \left[ \frac{r(t_j)}{\alpha \nabla r(t_j)} \right] - \hat{P}_A^{(R)} \left[ \frac{r}{\alpha \nabla r} \right] (t_j) \right) + O(h^2 \mathbf{1}) + O_P(\frac{1}{\sqrt{nh^d}}).
\]

Recall the abbreviation

\[
S^{(j)} = \begin{bmatrix}
s_0 & s_1 \\
s_1 & S_2
\end{bmatrix}
\]

42
Now consider
\[(1, 0, \ldots, 0) \left( S_{(j)}^{(\alpha)} + RI \right)^{-1} S_{(j)}^{(\alpha)} \begin{bmatrix} r(t_j) \\ \alpha \nabla r(t_j) \end{bmatrix} = \frac{s_0 - s_1^T(S_2 + \alpha^2 RI)^{-1}s_1}{R + s_0 - s_1^T(S_2 + \alpha^2 RI)^{-1}s_1} r(t_j) + \frac{\alpha^2 R s_1^T(S_2 + \alpha^2 RI)^{-1}\nabla r(t_j)}{R + s_0 - s_1^T(S_2 + \alpha^2 RI)^{-1}s_1}.
\]

We do not want \( \alpha \to \infty \), because then we might choose \( R \to 0 \) and \( \alpha^2 R \to \infty \) and then the asymptotic bias would contain an undesired term as the Nadaraya–Watson estimator. (Stabilizing the local linear estimator towards the Nadaraya–Watson estimator instead of the additive model is out of scope of this thesis.)

Choosing \( \alpha \to 0 \) reduces the bias but increases the variance. To find reasons for not doing this, we need to look at the \( O_p(\ldots) \) terms. (The influence of \( \alpha \) on the projection \( \tilde{P}_{A_R}^{(R)} \) is not investigated.)

Furthermore \( \alpha = 1 \) is reasonable as \([\beta_{(j)}^R]_0 \) and \([\beta_{(j)}^R]_k, k = 1, \ldots, d \) use the “same model”.
5 Numerical evaluation

In this section, finite sample properties of the penalized estimator are evaluated. For 4 two-dimensional regression functions (see section 5.1), 5 designs (see section 5.2), and 3 residual variances $\sigma^2$, the mean integrated squared error (MISE) depending on $(h, R)$ is evaluated, see section 5.4. We conclude that penalizing for non-additivity has the potential to improve the ordinary local linear estimator.

Data-adaptive choice of the parameters $(h, R)$ is beyond the scope of this thesis. In section 5.5, we investigate if the corrected Akaike information criterion (AIC) may lead to a parameter-selection rule that is acceptable in terms of integrated squared error (ISE). In the situations analyzed, AIC leads to a good choice of the parameters with respect to ISE.

5.1 Regression function

The following regression functions have been used:

**Superposed peaks**

As a non-additive regression function, we use (figure 6)

$$r^N(x_1, x_2) = 0.3 \exp \left( -4 \left\| \frac{x_1 - 0.25}{x_2 - 0.25} \right\|^2 \right) + 0.7 \exp \left( -8 \left\| \frac{x_1 - 0.75}{x_2 - 0.75} \right\|^2 \right) + 0.5 \exp \left( - \left\| \frac{x_1 - 0.5}{x_2 - 0.5} \right\|^2 \right).$$

This function was used in Seifert and Gasser (2000) figure 6.

![Figure 6: Regression function superposed peaks.](image)

Evaluated on a 200 $\times$ 200 grid, the “variance” of this function is 0.015 and 60% of its variability is explained by the additive components.

**Approximately additive peaks**

Rotating the above regression function by 45 degrees is still not additive but may be approximated by an additive function reasonably well (figure 7).

$$r^R(x_1, x_2) = r^N \left( x_1 + x_2 - 0.5 \right)$$

$$- r^N \left( -x_1 + x_2 - 0.5 \right).$$
Figure 7: Regression function approximately additive peaks.

Evaluated on a 200 × 200 grid, the “variance” of this function is 0.02 and 86% of its variability is explained by the additive components.

Additive peaks
If the regression function is additive, a penalty $R = \infty$ is optimal. The following regression function is used to see if AIC detects additive functions (figure 8).

$$r^A(x_1, x_2) = \frac{1}{2} \sum_{k=1}^{2} \left( 0.3 \exp(-4(x_k - 0.25)^2) + 0.7 \exp(-8(x_k - 0.75)^2) + 0.5 \exp(-(x_k - 0.5)^2) \right)$$

Figure 8: Regression function additive peaks.

Evaluated on a 200 × 200 grid, the “variance” of this function is 0.02 and 100% of its variability is explained by the additive components.
Fully non–additive function
The “diagonal sine” function defined on \([0,1]^2\) is orthogonal to the additive model (figure 9).

\[ r_S(x_1, x_2) = \sin(2\pi(x_1 + x_2)) \]

Figure 9: Regression function diagonal sine.

The “variance” of this function is 0.5 and 0% of its variability is explained by the additive components. Penalizing the full model towards additivity is not indicated. Ideally, parameter selection by AIC would choose \( R = 0 \). This regression function is not evaluated for all designs.

5.2 Design
The penalized estimator was evaluated for the following designs.

R200
This realization of 200 random uniform observations was used in Seifert and Gasser (2000) and is displayed in figure 10.

Figure 10: Design R200: 200 random uniformly distributed points.

R300
This design denotes a set of 300 random uniformly distributed design points (not shown). It is only used in conjunction with the “diagonal–sine” regression function.
5 NUMERICAL EVALUATION

5.2 Design

R1600
This is a realization of 1600 random uniform observations, see figure 11.

![Figure 11: Design R1600: 1600 random uniformly distributed points.](image)

F100
Consider an equidistant grid of $10 \times 10$ observations (figure 12). A bandwidth of at least $\frac{1}{9}$ is necessary for uniqueness of the ordinary local linear estimator.

![Figure 12: Design F100: $10 \times 10$ equidistant points.](image)

This design is ideal for estimation of additive functions — but only if input and output grid are the same: In this case, no non-parametric smoothing is required (unless $\sigma$ is large).

For a large output grid, $h = \frac{1}{9}$, and $R = \infty$ the penalized estimator interpolates the parametric estimates at the input points.

F104
For curiosity, an equidistant grid is rotated and shrunk to ensure that the margins are not clustered, see the tick marks (figure 13). In order to obtain uniform margins, a low discrepancy grid may be used alternatively. The results are qualitatively the same (and hence not shown).
5.3 Implementation

5.3.1 Output grid

The output grid was chosen to be $50 \times 50 \times 0, \frac{1}{49}, \ldots, 1$ taking into account

- Computation: a dense grid consumes more memory and requires more computation time.
- Statistical properties: smoothing windows should strongly overlap, see section 4.2.3.

Integration is always replaced by sums over the output grid.

5.3.2 Bandwidths

The evaluation uses a product Epanechnikov kernel: smoothing windows are rectangular. The bandwidths are determined by a one-dimensional parameter: $h_1 = h_2$ except for some boundary correction. This restriction facilitates visualization and is justified by the symmetry (i.e. exchange $x_1$ and $x_2$) of the regression functions $r^N$, $r^A$, and $r^S$.

The bandwidth parameters are chosen *equidistant* in log-scale.

**Adjustment of the bandwidth at the boundary**

Figure 13: Design F104: equidistant grid without clusters in the empirical marginal distribution.

Figure 14: Illustration of the adjustment at the boundary for the bandwidth. Originally, the bandwidth was $h = 0.20$. The lines at the sides represent smoothing windows. On each of these lines, a point marks the corresponding output point. The two rectangles illustrate how the smoothing windows are constructed.
Near the boundary, bandwidths are enlarged such that the size of the smoothing window within $[0, 1]^2$ remains constant (figure 14).

Note that the subroutines are more general: $h_1 \neq h_2$ is allowed and even local bandwidths — bandwidth is a univariate function of the corresponding component of the output point — are accepted. More general bandwidths are not implemented as this would complicate computation.

Reparameterization

For simplicity, the estimator was originally defined using a global bandwidth. Both the rescaling of the kernel and the parameterization of the slopes depend on the bandwidth (definition (16) of $\hat{\beta}_R$ on page 26). For kernel weights, use of the above boundary adjustment is indicated. But for the slopes we still use the global bandwidth. We use $\alpha_k = 1$, see section 3.4.3.

5.3.3 Extreme penalties

The penalty parameters are chosen equidistant in scale $\frac{R}{1+R}$ between 0 and 1. In order to compare goodness of fit averaged over the output grid, we need to avoid estimators with undefined regions. Accordingly, we use $R \geq R_{\text{min}}$ with $\frac{R_{\text{min}}}{1+R_{\text{min}}} = 10^{-4}$. Now, conditions for non–singularity are the same as for the additive model. On the other end, $R = \infty$ does not need regularization, but for numerical reasons a different implementation is needed (see section G.7 for details). Therefore a maximal $R$ of 9999 was used: $\frac{R_{\text{max}}}{1+R_{\text{max}}} = 1 - 10^{-4}$. This is justified by continuity in $R$ of the penalized estimator at $R = \infty$.

5.4 MISE

We investigate the dependence of the mean integrated squared error (MISE) on the bandwidth $h$ and the penalty $R$.

As the values of (M)ISE are small and leading zeroes in tables are not desired, (M)ISE is multiplied by the number of output points (2500). For comparison of MISE with residual variance $\sigma^2 \in \{0.05^2, 0.1^2, 0.2^2\}$, we define $\sigma^2_I = 2500\sigma^2 \in \{6.25, 25, 100\}$.

Calculation of the MISE

Let $G_{R,H}$ denote the linear mapping with $\tilde{\beta}_R = G_{R,H} Y$. Furthermore, $e_i$ is the $i^{\text{th}}$ unit vector (of length $n$). Assume that the stochastic part of $Y_i, \varepsilon_i (i = 1, \ldots, n)$ are independent $\mathcal{N}(0, \sigma^2)$ distributed. The ‘integrated’ variance $\frac{1}{m} \sum_{j=1}^{m} \text{var}(\tilde{\beta}_{R,(j)})$ is equal to

$$\frac{1}{m} \mathbb{E}(\|G_{R,H} e\|^2) = \frac{1}{m} \sum_{i=1}^{n} \mathbb{E}(\varepsilon_i^2) \|G_{R,H} e_i\|^2 \frac{1}{\sigma^2}.$$

Exact variance calculation is done via repeated evaluation of the penalized estimator on a unit vector. Fortunately, this has only to be done once for a design. Bias is obtained by calculating the difference between the true values and the estimates based on noise–free data.

5.4.1 MISE for superposed peaks regression function and 200 design points

For the regression function “superposed peaks” $r^N$, the design “R200”, and $\sigma^2_I = 25$ we get the integrated variance and integrated squared bias as a function of $h$ and $R$ (figure 15). Let us have a closer look at the MISE (figure 16): As the true regression function is clearly not additive, the usage of large penalties $R$ is not recommended. However, using a small penalty $R = 0.16$ allows us to use a smaller bandwidth compared to the ordinary local linear estimator and the MISE is reduced by 25% (table 1).
5.4 MISE

Variance

\[
\begin{array}{cccc}
0.2 & 0.4 & 0.6 & 0.8 \\
\end{array}
\]

\[
\begin{array}{cccc}
R/(1+R) & -1.2 & -1 & -0.8 & -0.6 \\
\end{array}
\]

\[
\begin{array}{cccc}
\log_{10}(h) & 0 & 5 & 10 \\
\end{array}
\]

Figure 15: Variance (left, $\sigma_I^2 = 25$) and squared bias (right) for “superposed peaks” regression function and design “R200”. Bandwidths $h$ in [0.05, 0.316] are plotted in log–scale and penalties $R$ in [$10^{-4}, 10^4$] are plotted in $\frac{R}{1+R}$ scale. The front corner corresponds to $(h, R) = (0.05, 10^{-4})$. Variability may be reduced by either increasing $h$ or $R$.

MISE for superposed peaks regression function, $\sigma_I^2 = 25$, and 200 design points.

Figure 16: On top, the MISE as a function of the bandwidth $h$ and the penalty $R$ is displayed as a perspective (left) and a gray scale (right) plot. The optimal penalty for a given bandwidth, $R_{\text{opt}}(h)$ (solid line), and the optimal bandwidth for a given penalty, $h_{\text{opt}}(R)$ (dashed line), are added to the grey scale plot. The MISE along these curves is displayed at the bottom: MISE($h, R_{\text{opt}}(h)$) vs. log($h$) (left) and MISE($h_{\text{opt}}(R), R$) vs. $\frac{R}{1+R}$ (right).
The following table 1 summarizes MISE for 3 regression functions, 3 residual standard deviations, and the design R200. Obviously, if the true regression function is additive \((r^A)\), maximal penalizing is optimal. If the true regression function is not additive, penalizing the deviation from additivity has the potential to improve the estimation even if the additive estimator is much worse than the ordinary local linear estimator.

Similar results were obtained for the design R1600 (table 2). The gain is however smaller. As expected, for the design R300 and the “diagonal sine” regression function \(r^S\), MISE is minimized for \(R \approx 0\) (not shown).

### Superposed peaks

<table>
<thead>
<tr>
<th>(\sigma_I^2)</th>
<th>MISE((h_{opt}, R = 10^{-4}))</th>
<th>MISE((h_{opt}, R_{opt}))</th>
<th>MISE((h_{opt}, R = 10^4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>3.98=135% (0.117, 10^{-4})</td>
<td>2.93=100% (0.104, 0.053)</td>
<td>16.2=550% (0.107, 10^4)</td>
</tr>
<tr>
<td>25</td>
<td>8.14=133% (0.166, 10^{-4})</td>
<td>6.12=100% (0.117, 0.149)</td>
<td>17.3=283% (0.117, 10^4)</td>
</tr>
<tr>
<td>100</td>
<td>15.2=117% (0.216, 10^{-4})</td>
<td>12.9=100% (0.18, 0.25)</td>
<td>20.4=158% (0.2, 10^4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\sigma_I^2)</th>
<th>MISE((h_{opt}, R = 10^{-4}))</th>
<th>MISE((h_{opt}, R_{opt}))</th>
<th>MISE((h_{opt}, R = 10^4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>3.78=126% (0.124, 10^{-4})</td>
<td>3=100% (0.097, 0.111)</td>
<td>8.53=284% (0.1, 10^4)</td>
</tr>
<tr>
<td>25</td>
<td>7.14=119% (0.17, 10^{-4})</td>
<td>5.96=100% (0.117, 0.282)</td>
<td>9.7=162% (0.124, 10^4)</td>
</tr>
<tr>
<td>100</td>
<td>13.9=115% (0.226, 10^{-4})</td>
<td>11.8=100% (0.178, 0.493)</td>
<td>13=110% (0.186, 10^4)</td>
</tr>
</tbody>
</table>

### Approximately additive peaks

<table>
<thead>
<tr>
<th>(\sigma_I^2)</th>
<th>MISE((h_{opt}, R = 10^{-4}))</th>
<th>MISE((h_{opt}, R_{opt}))</th>
<th>MISE((h_{opt}, R = 10^4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>7.67=667% (0.105, 10^{-4})</td>
<td>1.15=100% (0.054, 9999)</td>
<td>1.15=100% (0.054, 10^4)</td>
</tr>
<tr>
<td>25</td>
<td>14.9=446% (0.129, 10^{-4})</td>
<td>3.33=100% (0.072, 9999)</td>
<td>3.33=100% (0.072, 10^4)</td>
</tr>
<tr>
<td>100</td>
<td>24=247% (0.184, 10^{-4})</td>
<td>9.71=100% (0.094, 9999)</td>
<td>9.71=100% (0.094, 10^4)</td>
</tr>
</tbody>
</table>

### Additive peaks

<table>
<thead>
<tr>
<th>(\sigma_I^2)</th>
<th>MISE((h_{opt}, R = 10^{-4}))</th>
<th>MISE((h_{opt}, R_{opt}))</th>
<th>MISE((h_{opt}, R = 10^4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>0.79=101% (0.066, 10^{-4})</td>
<td>0.77=100% (0.063, 0.02)</td>
<td>15=1940% (0.072, 10^4)</td>
</tr>
<tr>
<td>25</td>
<td>1.87=105% (0.083, 10^{-4})</td>
<td>1.77=100% (0.079, 0.042)</td>
<td>15.2=859% (0.083, 10^4)</td>
</tr>
<tr>
<td>100</td>
<td>4.53=112% (0.11, 10^{-4})</td>
<td>4.02=100% (0.1, 0.111)</td>
<td>16=396% (0.1, 10^4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\sigma_I^2)</th>
<th>MISE((h_{opt}, R = 10^{-4}))</th>
<th>MISE((h_{opt}, R_{opt}))</th>
<th>MISE((h_{opt}, R = 10^4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>0.82=100% (0.066, 10^{-4})</td>
<td>0.82=100% (0.066, 0.02)</td>
<td>7.65=936% (0.066, 10^4)</td>
</tr>
<tr>
<td>25</td>
<td>1.96=105% (0.083, 10^{-4})</td>
<td>1.85=100% (0.076, 0.064)</td>
<td>7.88=427% (0.076, 10^4)</td>
</tr>
<tr>
<td>100</td>
<td>4.63=113% (0.11, 10^{-4})</td>
<td>4.09=100% (0.1, 0.136)</td>
<td>8.62=210% (0.1, 10^4)</td>
</tr>
</tbody>
</table>

Table 1: MISE for 3 regression functions, 3 residual standard deviations, and 200 design points (R200).

<table>
<thead>
<tr>
<th>(\sigma_I^2)</th>
<th>MISE((h_{opt}, R = 10^{-4}))</th>
<th>MISE((h_{opt}, R_{opt}))</th>
<th>MISE((h_{opt}, R = 10^4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>1.01=504% (0.06, 10^{-4})</td>
<td>0.2=100% (0.035, 9999)</td>
<td>0.2=100% (0.035, 10^4)</td>
</tr>
<tr>
<td>25</td>
<td>2.41=398% (0.076, 10^{-4})</td>
<td>0.61=100% (0.044, 9999)</td>
<td>0.61=100% (0.044, 10^4)</td>
</tr>
<tr>
<td>100</td>
<td>5.84=321% (0.095, 10^{-4})</td>
<td>1.82=100% (0.06, 9999)</td>
<td>1.82=100% (0.06, 10^4)</td>
</tr>
</tbody>
</table>

Table 2: MISE for 3 regression functions, 3 residual standard deviations, and 1600 design points (R1600).
5.4.2 Odd behavior of the bias for small bandwidths

Typically, the bias increases with the amount of smoothing and a larger bandwidth \( h \) or a penalty \( R \) leads to smoother estimates. Having a closer look at figure 16, we observe that the bias decreases with \( h \) around \( h = 10^{-1.2} \) and \( R = 10^{-4} \). To highlight this phenomenon, the range of bandwidths is extended to include extremely small values: from 0.03 to 0.25 (figure 17).

![Figure 17: The integrated squared bias is expected to increase with \( h \) or \( R \). Why is the minimal bias not attained at the minimal values of \( h \) and \( R \)? The reason is the regularization for extremely small bandwidths.](image)

For the bandwidth 0.03, the expected number of observations is smaller than one (0.7) and the tiny penalty \( R = 10^{-6} \) leads to a remarkable modification towards additivity (figure 18).

![Figure 18: The penalized estimator for the design “R200” and the regression function “superposed peaks” without noise (\( \sigma = 0 \)). The bandwidths are 0.03 (top/left), 0.05 (top/right), 0.1 (bottom/left), and 0.2 (bottom/right). the penalty parameter is \( R = 10^{-6} \).](image)
5.5 AIC and ISE

The optimal $h$ and $R$ are defined to be the minimum of integrated squared errors (ISE). Since the true regression function is involved in ISE, ISE cannot be used in applications. The Akaike information criterion (AIC) tries to find a compromise between a good fit (small estimated residuals) and a simple model (low number of degrees of freedom) and does not involve the unknown true regression function. For simulated data, we will investigate, whether the parameters $h_{AIC}$ and $R_{AIC}$ obtained by AIC are useful in terms of ISE.

5.5.1 ISE

The integrated squared error (ISE) is defined as the integral (mean over the output grid) of the squared difference between the estimate and the true regression function. As with MISE, values are multiplied by 2500.

5.5.2 AIC

Hurvich et al. (1998) proposed an improved Akaike information criterion for smoothing parameter selection in non–parametric regression. AIC estimates the Kullback–Leibler discrepancy and is applicable to linear estimators. An estimator is called linear if the fitted values $\hat{Y}$ depend linearly on the observations $Y$:

$$\hat{Y} = MY.$$  

The matrix $M$ is called hat matrix.

Consider the residual variance estimation

$$\hat{\sigma}^2 = \frac{1}{n} \| Y - MY \|^2.$$  

The goal is to find a trade–off between good fit (small $\hat{\sigma}^2$) and small complexity of the model:

$$\text{AIC}_C = \log(\hat{\sigma}^2) + \frac{1 + \text{tr}(M)/n}{1 - (\text{tr}(M) + 2)/n},$$

where $\text{tr}(M)$ denotes the trace of $M$, which is interpreted as degrees of freedom.

Application of GCV, FPE, T, and other bandwidth selectors depending only on $\hat{\sigma}^2$ and $\text{tr}(M)$ (see e.g. Härde and Marron, 1985) is easy.

Note that AIC uses residuals at the design points and is therefore related to average squared errors (ASE). As the penalized estimator returns values on a output grid, ISE is the natural criterion for goodness of fit. If the design is non–uniformly distributed, this distinction may become an issue.

Obtaining the hat matrix

As the local linear estimator with non–additivity penalty depends on the choice of the output grid, which is in general different from the design, we need to specify $M$. Choosing a larger output grid is not desirable, as this modifies the estimator and increases computational burden. However, we may use the convex combination interpretation (20) on page 28:

$$\hat{y}_i = [1, 0, \ldots, 0] (S(X_i) + RI)^{-1} (S(X_i) \hat{\beta}_d(X_i) + R \{ \tilde{P}_A^{(R)} \hat{\beta}_d \} (X_i))$$

where $S(X_i)$ is defined as $S^{(j)}$ in formula (17) with $t_j$ replaced by $X_i$, and analogously for $T(X_i) = S(X_i) \hat{\beta}_d(X_i)$. The additive part $\{ \tilde{P}_A^{(R)} \hat{\beta}_d \} (X_i)$ is defined as a linear interpolation of its values on the grid. As these values are additive, the interpolation is univariate.

If the estimator is defined on a function space instead of a grid, the convex combination property still holds. In this case the computation on a grid may be considered as an integral approximation, see also appendix C.2.2.
Problems
In Hurvich et al. (1998), one link in the chain of derivation of AIC<sub>C</sub> is the assumption that the smoother is unbiased. Nevertheless, AIC<sub>C</sub> is used for local linear estimators and is competitive to plug–in bandwidth choice in Monte–Carlo simulations. We might worry about the applicability as the estimations are shrunk towards additivity which imposes an additional bias. Hence, the worst case occurs if R<sub>opt</sub> is medium sized, but in this case the best additive and ordinary local linear estimator are also far from this optimum and we might hope that even a not so good choice of R leads to an improvement.

5.5.3 AIC vs ISE for 200 observations
The behavior of ISE at the minimizer (h<sub>ISE</sub>, R<sub>ISE</sub>) obtained by AIC<sub>C</sub> is compared with the optimal ISE(h<sub>ISE</sub>, R<sub>ISE</sub>) (table 3). For the “superposed peaks” regression function r<sup>N</sup>, AIC performs well. For the additive regression function, AIC detects the biggest penalty available (R = 10<sup>4</sup>). For the “approximately additive peaks” regression function r<sup>R</sup>, AIC uses a larger bandwidth (oversmoothing) instead of a regularization towards additivity. The case r<sup>R</sup> and σ<sup>2</sup> = 25 seems to be a positive exception, but figure 20 shows a second local minimum.

### Superposed peaks

<table>
<thead>
<tr>
<th>σ&lt;sup&gt;2&lt;/sup&gt;</th>
<th>ISE(h&lt;sub&gt;ISE&lt;/sub&gt;, R&lt;sub&gt;ISE&lt;/sub&gt;)</th>
<th>ISE(h&lt;sub&gt;AIC&lt;/sub&gt;, R&lt;sub&gt;AIC&lt;/sub&gt;)</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>(h=0.104,R=0.053)</td>
<td>3.1 (h=0.105,R=0.111)</td>
<td>103%</td>
</tr>
<tr>
<td>25</td>
<td>(h=0.117,R=0.163)</td>
<td>6.2 (h=0.126,R=0.176)</td>
<td>103%</td>
</tr>
<tr>
<td>100</td>
<td>(h=0.202,R=0.266)</td>
<td>11.8 (h=0.193,R=0.087)</td>
<td>109%</td>
</tr>
</tbody>
</table>

### Approximately additive peaks

<table>
<thead>
<tr>
<th>σ&lt;sup&gt;2&lt;/sup&gt;</th>
<th>ISE(h&lt;sub&gt;ISE&lt;/sub&gt;, R&lt;sub&gt;ISE&lt;/sub&gt;)</th>
<th>ISE(h&lt;sub&gt;AIC&lt;/sub&gt;, R&lt;sub&gt;AIC&lt;/sub&gt;)</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>(h=0.094,R=0.111)</td>
<td>3.6 (h=0.124,R=0.099)</td>
<td>124%</td>
</tr>
<tr>
<td>25</td>
<td>(h=0.098,R=0.389)</td>
<td>6.2 (h=0.133,R=0.266)</td>
<td>107%</td>
</tr>
<tr>
<td>100</td>
<td>(h=0.188,R=0.613)</td>
<td>13.1 (h=0.251,R=0.031)</td>
<td>122%</td>
</tr>
</tbody>
</table>

### Additive peaks

<table>
<thead>
<tr>
<th>σ&lt;sup&gt;2&lt;/sup&gt;</th>
<th>ISE(h&lt;sub&gt;ISE&lt;/sub&gt;, R&lt;sub&gt;ISE&lt;/sub&gt;)</th>
<th>ISE(h&lt;sub&gt;AIC&lt;/sub&gt;, R&lt;sub&gt;AIC&lt;/sub&gt;)</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>(h=0.052,R=49)</td>
<td>1 (h=0.057,R=9999)</td>
<td>102%</td>
</tr>
<tr>
<td>25</td>
<td>(h=0.073,R=24)</td>
<td>2.8 (h=0.071,R=9999)</td>
<td>100%</td>
</tr>
<tr>
<td>100</td>
<td>(h=0.093,R=49)</td>
<td>7.4 (h=0.087,R=9999)</td>
<td>101%</td>
</tr>
</tbody>
</table>

Table 3: Optimal ISE and ISE at minimizer of AIC are compared for 3 different regression functions and 3 residual variances. The design is 200 random uniformly distributed points (R200).

In figure 19, AIC detects the minimum and the behavior of AIC and ISE is similar even though the scales are inherently different. In figure 20, AIC fails to detect the ISE optimal parameters. Furthermore it has a second local minimum. We cannot conclude that this is due to the regression function or the design, as a second realization of y did not show this problems. Additive functions are detected by AIC, see figure 21.
**AIC vs. ISE** for superposed peaks regression function, $\sigma_f^2 = 25$, and 200 design points.

![Comparison of AIC (left column) and ISE (right column). In the top row, AIC and ISE are displayed as gray scale plots. For fixed $h$, AIC and ISE are minimized with respect to $R$: $R_{\text{opt}}(h)$, and the solid line connects $(h, R_{\text{opt}}(h))$. Conversely, the dotted line is $(h_{\text{opt}}(R), R)$, where $h_{\text{opt}}(R)$ is the minimizer with respect to $h$ of AIC and ISE for fixed $R$. In the middle row, AIC and ISE($h, R_{\text{opt}}(h)$) are plotted. This corresponds to the values along the solid line in the top row. In the bottom row, AIC and ISE($h_{\text{opt}}(R), R$) are plotted. A point marks the global minimum. Recall that ISE measures how good the estimator fits the true regression function, while AIC does not use the true regression function.](image-url)
**AIC vs. ISE** for approximately additive peaks regression function, $\sigma^2 = 25$, and 200 design points.

Figure 20: Comparison of AIC (left column) and ISE (right column). In the top row, AIC and ISE are displayed as gray scale plots. For fixed $h$, AIC and ISE are minimized with respect to $R$: $R_{\text{opt}}(h)$, and the solid line connects $(h, R_{\text{opt}}(h))$. Conversely, the dotted line is $(h_{\text{opt}}(R), R)$, where $h_{\text{opt}}(R)$ is the minimizer with respect to $h$ of AIC and ISE for fixed $R$. In the middle row, AIC and ISE$(h, R_{\text{opt}}(h))$ are plotted. This corresponds to the values along the solid line in the top row. In the bottom row, AIC and ISE$(h_{\text{opt}}(R), R)$ are plotted. A point marks the global minimum.
AIC vs. ISE for additive peaks regression function, $\sigma^2 = 25$, and 200 design points.

Figure 21: Comparison of AIC (left column) and ISE (right column). In the top row, AIC and ISE are displayed as gray scale plots. For fixed $h$, AIC and ISE are minimized with respect to $R$: $R_{\text{opt}}(h)$, and the solid line connects $(h, R_{\text{opt}}(h))$. Conversely, the dotted line is $(h_{\text{opt}}(R), R)$, where $h_{\text{opt}}(R)$ is the minimizer with respect to $h$ of AIC and ISE for fixed $R$. In the middle row, AIC and ISE($h, R_{\text{opt}}(h)$) are plotted. This corresponds to the values along the solid line in the top row. In the bottom row, AIC and ISE($h_{\text{opt}}(R), R$) are plotted. A point marks the global minimum.
5.5.4 AIC vs ISE for 1600 observations

With 1600 observations, AIC performs well for both additive and non–additive regression functions. At this sample size “approximately additive peaks” is recognized as non–additive.

**Superposed peaks**

<table>
<thead>
<tr>
<th>$\sigma^2$</th>
<th>ISE($h_{\text{ISE}}, R_{\text{ISE}}$)</th>
<th>ISE($h_{\text{AIC}}, R_{\text{AIC}}$)</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>0.9 ($h=0.063, R=0.02$)</td>
<td>0.9 ($h=0.069, R=10^{-4}$)</td>
<td>101%</td>
</tr>
<tr>
<td>25</td>
<td>2 ($h=0.079, R=0.042$)</td>
<td>2 ($h=0.076, R=0.042$)</td>
<td>100%</td>
</tr>
<tr>
<td>100</td>
<td>4.8 ($h=0.1, R=0.087$)</td>
<td>5 ($h=0.115, R=0.111$)</td>
<td>103%</td>
</tr>
</tbody>
</table>

**Approximately additive peaks**

<table>
<thead>
<tr>
<th>$\sigma^2$</th>
<th>ISE($h_{\text{ISE}}, R_{\text{ISE}}$)</th>
<th>ISE($h_{\text{AIC}}, R_{\text{AIC}}$)</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>0.8 ($h=0.066, R=0$)</td>
<td>0.8 ($h=0.069, R=0$)</td>
<td>100%</td>
</tr>
<tr>
<td>25</td>
<td>1.8 ($h=0.076, R=0.064$)</td>
<td>1.8 ($h=0.076, R=0.042$)</td>
<td>100%</td>
</tr>
<tr>
<td>100</td>
<td>4.1 ($h=0.091, R=0.22$)</td>
<td>4.1 ($h=0.091, R=0.19$)</td>
<td>100%</td>
</tr>
</tbody>
</table>

**Additive peaks**

<table>
<thead>
<tr>
<th>$\sigma^2$</th>
<th>ISE($h_{\text{ISE}}, R_{\text{ISE}}$)</th>
<th>ISE($h_{\text{AIC}}, R_{\text{AIC}}$)</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>0.2 ($h=0.033, R=9999$)</td>
<td>0.2 ($h=0.032, R=49$)</td>
<td>100%</td>
</tr>
<tr>
<td>25</td>
<td>0.6 ($h=0.042, R=9999$)</td>
<td>0.6 ($h=0.035, R=9999$)</td>
<td>104%</td>
</tr>
<tr>
<td>100</td>
<td>1.6 ($h=0.055, R=9999$)</td>
<td>1.8 ($h=0.066, R=9999$)</td>
<td>114%</td>
</tr>
</tbody>
</table>

Table 4: Optimal ISE and ISE at minimizer of AIC are compared for 3 different regression functions and 3 residual variances. The design is 1600 random uniformly distributed points (R1600).

5.5.5 AIC vs ISE for 100 equidistant observations

As 100 observation is quite small and the results depend on the realization, the equidistant design F100 is used. For the “superposed peaks” ($r^N$) and the “approximately additive peaks” ($r^R$) regression function, AIC leads to oversmoothing (table 5). For the “additive peaks” regression function ($r^A$), AIC recognizes that the data is from an additive function ($R = 9999$) but it oversmoothes for $\sigma^2 = 100$.

This design is not representative for the case “small sample size without sparse regions” for the following reasons:

Due to the special structure of the design, the additive model may be estimated also parametrically (at the design points). The ISE-optimal bandwidth 0.107 leads to an estimator whose additive components are piecewise linear. It looks like an interpolated parametric estimator.

Second, there is a bend in $\hat{\sigma}^2(h, R)$ resulting in the same estimated bandwidth $h_{\text{AIC}} = 0.214$ in different cases. (Precisely: 0.214 is just below twice the spacings of the grid.)

Furthermore, numerical problems arise for small bandwidths: for $h = 0.102 < \frac{1}{5}$ and $R = 0.0417$ the estimator broke down resulting in a negative trace of the smoother matrix. (The error is in the additive component.) As there are sharp peaks in the AIC plot, this kind of failure is easily detected.

To avoid these problems, the design F104 is used below, but there is no substantial improvement.
5 NUMERICAL EVALUATION

5.5 AIC and ISE

Superposed peaks

<table>
<thead>
<tr>
<th>$\sigma^2_I$</th>
<th>ISE($h$, $R$)</th>
<th>ISE($h_{AIC}$, $R_{AIC}$)</th>
<th>ratio</th>
<th>ISE($\bar{y}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>3.7 (h=0.112, R=0.087)</td>
<td>8 (h=0.214, R=0.042)</td>
<td>216%</td>
<td>40.0</td>
</tr>
<tr>
<td>25</td>
<td>7.4 (h=0.135, R=0.22)</td>
<td>9.1 (h=0.214, R=0.111)</td>
<td>121%</td>
<td>39.8</td>
</tr>
<tr>
<td>100</td>
<td>13 (h=0.204, R=0.19)</td>
<td>13 (h=0.214, R=0.316)</td>
<td>102%</td>
<td>39.6</td>
</tr>
</tbody>
</table>

Approximately additive peaks

<table>
<thead>
<tr>
<th>$\sigma^2_I$</th>
<th>ISE($h$, $R$)</th>
<th>ISE($h_{AIC}$, $R_{AIC}$)</th>
<th>ratio</th>
<th>ISE($\bar{y}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>5.3 (h=0.129, R=0.064)</td>
<td>8 (h=0.214, R=0.042)</td>
<td>150%</td>
<td>54.8</td>
</tr>
<tr>
<td>25</td>
<td>8.7 (h=0.135, R=0.613)</td>
<td>10 (h=0.214, R=0.22)</td>
<td>115%</td>
<td>54.6</td>
</tr>
<tr>
<td>100</td>
<td>14 (h=0.178, R=2.571)</td>
<td>17 (h=0.282, R=9999)</td>
<td>118%</td>
<td>54.3</td>
</tr>
</tbody>
</table>

Additive peaks

<table>
<thead>
<tr>
<th>$\sigma^2_I$</th>
<th>ISE($h$, $R$)</th>
<th>ISE($h_{AIC}$, $R_{AIC}$)</th>
<th>ratio</th>
<th>ISE($\bar{y}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>3.6 (h=0.107, R=9999)</td>
<td>3.6 (h=0.107, R=9999)</td>
<td>100%</td>
<td>54.2</td>
</tr>
<tr>
<td>25</td>
<td>5 (h=0.107, R=9999)</td>
<td>652 (h=0.102, R=0.042)</td>
<td>13164%</td>
<td>54.0</td>
</tr>
<tr>
<td>100</td>
<td>11 (h=0.107, R=9999)</td>
<td>16 (h=0.195, R=9999)</td>
<td>151%</td>
<td>53.7</td>
</tr>
</tbody>
</table>

Table 5: Optimal ISE and ISE at minimizer of AIC are compared for 3 different regression functions and 3 residual variances. ISE($\bar{y}$) denotes the ISE for a constant approximation. For additive peaks and $\sigma^2_I = 25$, AIC fails due to numerical instability. Skipping one ($h$, $R$)–pair solves the problem. The design is 100 fixed design points (F100).

5.5.6 AIC vs ISE for 104 rotated equidistant observations

Using the design F104, the additive components may be estimated well because the margins of the data are quite uniform. In table 6, we see that parameter selection by AIC works well for the “additive peaks” regression function ($r^A$). For the “superposed peaks” regression function ($r^N$), AIC oversmoothes. AIC also oversmoothes in the case of the “approximately additive peaks” regression function ($r^R$) except for $\sigma^2_I = 100$, but in this case both AIC and ISE fail to reconstruct the structure of the regression function.

5.5.7 Others

The “diagonal sine” ($r^S$) regression function was evaluated on the “R300” design. AIC selects the minimal penalty $R = 10^{-4}$ — this is correct, as the additive components of $r^S$ are zero. The estimations look well, there is however a tendency to oversmooth. Loss in ISE starts at 0% ($\sigma^2_I = 6.25$) to 12% ($\sigma^2_I = 625$) and is 38% for $\sigma^2_I = 2500$. 
### 5.6 Conclusion

The local linear estimator with penalty on deviation from additivity has the potential to improve the estimation: as the penalty reduces variability, we may select a smaller bandwidth.

Parameter selection by AIC works in general, but it may fail in difficult situations.

---

#### Table 6: Optimal ISE and ISE at minimizer of AIC are compared for 3 different regression functions and 3 residual variances. ISE(\(\bar{y}\)) denotes the ISE for a constant approximation. The design is 104 fixed design points (F104).

<table>
<thead>
<tr>
<th></th>
<th>ISE((h_{ISE}, R_{ISE}))</th>
<th>ISE((h_{AIC}, R_{AIC}))</th>
<th>ratio</th>
<th>ISE((\bar{y}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma^2 = 6.25)</td>
<td>3.3 ((h=0.107, R=0.111))</td>
<td>5.4 ((h=0.17, R=0.099))</td>
<td>164%</td>
<td>38.7</td>
</tr>
<tr>
<td>(\sigma^2 = 25)</td>
<td>7.7 ((h=0.148, R=0.176))</td>
<td>8.3 ((h=0.182, R=0.163))</td>
<td>108%</td>
<td>39.1</td>
</tr>
<tr>
<td>(\sigma^2 = 100)</td>
<td>19 ((h=0.195, R=0.266))</td>
<td>22 ((h=0.214, R=2.571))</td>
<td>121%</td>
<td>40.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ISE((h_{ISE}, R_{ISE}))</th>
<th>ISE((h_{AIC}, R_{AIC}))</th>
<th>ratio</th>
<th>ISE((\bar{y}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma^2 = 6.25)</td>
<td>5.1 ((h=0.112, R=0.136))</td>
<td>6.6 ((h=0.17, R=0.205))</td>
<td>129%</td>
<td>53.2</td>
</tr>
<tr>
<td>(\sigma^2 = 25)</td>
<td>10 ((h=0.141, R=0.351))</td>
<td>12 ((h=0.219, R=9999))</td>
<td>122%</td>
<td>53.5</td>
</tr>
<tr>
<td>(\sigma^2 = 100)</td>
<td>21 ((h=0.275, R=1.564))</td>
<td>21 ((h=0.251, R=9999))</td>
<td>101%</td>
<td>54.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ISE((h_{ISE}, R_{ISE}))</th>
<th>ISE((h_{AIC}, R_{AIC}))</th>
<th>ratio</th>
<th>ISE((\bar{y}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma^2 = 6.25)</td>
<td>1.9 ((h=0.058, R=9999))</td>
<td>2 ((h=0.071, R=9999))</td>
<td>106%</td>
<td>52.9</td>
</tr>
<tr>
<td>(\sigma^2 = 25)</td>
<td>6.2 ((h=0.083, R=9999))</td>
<td>6.3 ((h=0.093, R=9999))</td>
<td>101%</td>
<td>53.3</td>
</tr>
<tr>
<td>(\sigma^2 = 100)</td>
<td>19 ((h=0.145, R=49))</td>
<td>20 ((h=0.132, R=9999))</td>
<td>101%</td>
<td>54.7</td>
</tr>
</tbody>
</table>
A Remarks on the estimator of Mammen, Linton, and Nielsen

Calculations, details and assumptions omitted in section 2.3 are provided below.

A.1 Notation

For direct comparison with MLN, the table below compares the notation. Equation numbers (e.g. (54)) refer to equations in MLN.

<table>
<thead>
<tr>
<th>MLN</th>
<th>thesis</th>
<th>notion</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>( f )</td>
<td>design density ( \mathcal{L}(X) ), assumption (B2')</td>
</tr>
<tr>
<td>( q_0 )</td>
<td>( f_X ) (see appendix A.3)</td>
<td>design density ( \mathcal{L}(X) ), assumption (B2')</td>
</tr>
<tr>
<td>( f )</td>
<td></td>
<td>multivariate regression function</td>
</tr>
<tr>
<td>( \hat{m} )</td>
<td>( \hat{r}_{\text{add}} )</td>
<td>additive estimate</td>
</tr>
<tr>
<td>( \hat{m}_j(x_j), \hat{m}^j(x_j) )</td>
<td>( \hat{r}^{a}<em>{\text{add}}, \hat{r}</em>{\text{add},j}(x_j) )</td>
<td>components of ( \hat{r}_{\text{add}} ) (32/33/44)</td>
</tr>
<tr>
<td>( \hat{m}_{j,[a]}(x_j), \hat{m}^{[a],j}(x_j) )</td>
<td>( \hat{r}^{[a]}<em>{\text{add},j}(x_j), \hat{r}^{[a],j}</em>{\text{add}}(x_j) )</td>
<td>backfitting steps (49/50)</td>
</tr>
<tr>
<td>( m_0 + \sum_{j=1}^{d} m_j(x_j) )</td>
<td>( m_j(x_j), \hat{m}^j(x_j) )</td>
<td>univariate estimator (39/40)</td>
</tr>
<tr>
<td></td>
<td>( r_{\text{true},j}(x) )</td>
<td>E(( Y \mid X = x )) decomposition of ( r_{\text{true},0}(x) ) if additive model holds (54)</td>
</tr>
<tr>
<td></td>
<td>( r_{\text{true},0} + \sum_{j=1}^{d} r_{\text{true},j}(x_j) )</td>
<td></td>
</tr>
</tbody>
</table>

A.2 Details about normal equations

\( \hat{r}_{\text{add}} \) is the minimizer over \( r \in \mathcal{F}_{\text{add}} \) if \( ||Y - r||^2 \) if and only if

\[
\langle r_Y - \hat{r}_{\text{add}}, r \rangle_s = 0 \quad \forall r \in \mathcal{F}_{\text{add}} \quad (\spadesuit)
\]

**Intercept terms**

Consider \( r \in \mathcal{F}_{\text{add}} \) with \( r^0(x) = g_0 + g_j(x_j) \) and \( r^k(x) = 0 \) for \( k = 1, \ldots, d \). Condition (\( \spadesuit \)) implies

\[
\int \left( \frac{1}{n} \sum_{i=1}^{n} \left( Y_i - \hat{r}_{\text{add},0} - \sum_{k=1}^{d} \left( \hat{r}_{\text{add},k}(x_k) + \hat{r}^k_{\text{add}}(x_k) \frac{X_{i,k} - x_k}{h} \right) \right) \right) K_h(\mathbf{X}_i, \mathbf{z}) d\mathbf{z}_{-j} x_j = 0.
\]

Hence the integrand must be zero:

\[
\frac{1}{n} \sum_{i=1}^{n} Y_i \int K_h(\mathbf{X}_i, \mathbf{z}) d\mathbf{z}_{-j} - (\hat{r}_{\text{add},0} + \hat{r}_{\text{add},j}(x_j)) \hat{V}_{j,0}(x_j) \int \hat{r}_{\text{add},k}(x_k) \hat{V}_{j,k,0}(x_j, x_k) d\mathbf{z}_{-j} - \sum_{k \neq j} \int \hat{r}^{k}_{\text{add}}(x_k) \hat{V}^{j,k}_{j,0}(x_j, x_k) d\mathbf{z}_{-j} = 0.
\]

The details are given below:

\[
\int \hat{r}_{\text{add},j}(x_j) \frac{1}{n} \sum_{i=1}^{n} K_h(\mathbf{X}_i, \mathbf{z}) d\mathbf{z}_{-j} = \hat{r}_{\text{add},j}(x_j) \hat{V}_{j,0}(x_j)
\]

\[
\int \hat{r}^j_{\text{add}}(x_j) \frac{1}{n} \sum_{i=1}^{n} \frac{X_{i,j} - x_j}{h} K_h(\mathbf{X}_i, \mathbf{z}) d\mathbf{z}_{-j} = \hat{r}^j_{\text{add}}(x_j) \hat{V}^{j}_{j,0}(x_j)
\]
\[
\int \hat{r}_{\text{add},k}(x_k) \frac{1}{n} \sum_{i=1}^{n} K_h(\mathbf{X}_i, X_d) d\mathbf{x}_{-j} = \int \hat{r}_{\text{add},k}(x_k) \frac{1}{n} \sum_{i=1}^{n} K_h(\mathbf{X}_i, X_d) d\mathbf{x}_{-j,k} dx_k = \int \hat{r}_{\text{add},k}(x_k) \tilde{V}_{j,k,0}(x_j, x_k) dx_k 
\]

\[
\int \hat{r}_{\text{add},k}(x_k) \frac{1}{n} \sum_{i=1}^{n} \frac{X_{i,j} - x_j}{h} K_h(\mathbf{X}_i, X_d) d\mathbf{x}_{-j} = \int \hat{r}_{\text{add},k}(x_k) \tilde{V}_{j,k,0}(x_j, x_k) dx_k.
\]

### Slope terms

Consider \( r \in \mathcal{F}_{\text{add}} \) with \( r^0(\mathbf{x}) = 0 \), \( r^j(\mathbf{x}) = g^j(x_j) \), and \( r^k(\mathbf{x}) = 0 \) for \( k \neq j \). For the slope terms, condition (♠) implies that

\[
\frac{1}{n} \sum_{i=1}^{n} Y_i \int K_h(\mathbf{X}_i, X_d) d\mathbf{x}_{-j} \frac{X_{i,j} - x_j}{h} - (\hat{r}_{\text{add},0} + \hat{r}_{\text{add},j}(x_j)) \tilde{V}_{j,0}(x_j) 
- \sum_{k \neq j} \hat{r}_{\text{add},k}(x_k) \tilde{V}_{j,k,0}(x_j, x_k) d\mathbf{x}_k - \sum_{k \neq j} \hat{r}_{\text{add},k}(x_k) \tilde{V}_{j,k,0}(x_j, x_k) dx_k = 0.
\]

### A.3 Details of theorem MLN:4’

#### Details about the choice of the kernel \( K_h \)

\( K_h \) is a product kernel, and is defined by rescaling some univariate kernel function \( K \), which is bounded, symmetric around zero, Lipschitz-continuous, and has compact support \([-C_1, C_1]\).

Because we assume that there are no observations outside \([0,1]^d\), we modify \( K_h \) such that its integral is one:

Define \( K_h(u) = K(u/h)/h \) and

\[
K_h(u,v) = \begin{cases} 
\frac{K_h(u-v)}{\int_0^u K_h(w-v)dw} & \text{for } u,v \in [0,1] \\
0 & \text{else}
\end{cases}.
\]

(MLN:58)

Finally, define the product kernel \( K_h(\mathbf{X}_i, \mathbf{X}) = \prod_{j=1}^{d} K_h(\mathbf{X}_{i,j}, x_j) \).

#### Comments on observations

MLN allow observations outside \([0,1]^d\), those are omitted, however, this implies that averages over all observations are replaced by averages over all observations in \([0,1]^d\). Hence we need to distinguish between the density \( f_X \) of \( \mathbf{X} \) and the conditional density \( f(\mathbf{x}) \) of \( \{ \mathbf{X} = \mathbf{x} | \mathbf{X} \in [0,1]^d \} \).

Denote by \( N \) the number of observations \( \mathbf{X} \) with \( \mathbf{X}_i \in [0,1]^d \). The expected number of observations in \([0,1]^d\) is \( n_0 = nP(\mathbf{X} \in [0,1]^d) \equiv E(N) \).

Hence, we have to replace \( n \) with \( n_0 \) in assumption (C2): “Assume, there exists a constant \( c_h \) with \( n_0^{1/3} h \to c_h \).”

Define the \( \tilde{m}_{j}(x_j), \tilde{M}_{j}(x_j) \) by

\[
\tilde{M}_{j}(x_j) \left( \tilde{m}_{j}(x_j) \right) = \frac{1}{N} \sum_{i=1,...,n}^{} K_h(\mathbf{X}_{i,j}, x_j) \left( \frac{1}{h} \frac{X_{i,j} - x_j}{h} \right) Y_i
\]

(MLN:62)
Furthermore, define

\[ \tilde{M}_j(x_j) = \begin{bmatrix} \hat{V}_{j0}^j(x_j) \\ \hat{V}_{j0}^j(x_j) \end{bmatrix} \]

\[ = \frac{1}{N} \sum_{i=1}^{n} K_h(X_{i,j}, x_j) \begin{bmatrix} 1 \\ \frac{X_{i,j}-x_j}{h} \end{bmatrix} \]

(MLN:63)

Furthermore, define

\[ \tilde{S}_{k,j}(x_k, x_j) = \begin{bmatrix} \hat{V}_{k0}^j(x_k, x_j) \\ \hat{V}_{k0}^j(x_k, x_j) \end{bmatrix} \]

\[ = \frac{1}{N} \sum_{i=1}^{n} K_h(X_{i,j}, x_j) K_h(X_{i,k}, x_k) \times \begin{bmatrix} 1 \\ \frac{X_{i,k}-x_k}{h} \frac{X_{i,j}-x_j}{h} \end{bmatrix} \]

(MLN:64)

\( \hat{r}_{\text{add},0}, \hat{r}_{\text{add},j}, \) and \( \hat{r}_j^j \) are defined by the normal equations (MLN:44) and (MLN:45).

Because \( \int \hat{V}_{k0}^j(x_k, x_j) dx_k = \hat{V}_{j0}^j(x_j) \) and \( \int \hat{V}_{j0}^j(x_j) dx_j = 1 \), \( \hat{r}_{\text{add},0} \) does not depend on \( j \). In this case, the normalization (MLN:50) may be omitted.

**Choice of the constants in assumptions (MLN:A1')–(MLN:A9')**

For the assumptions, see appendix A.4 below. Define

\[ \Delta_n = h^2 \]

\[ W = \begin{bmatrix} 1 \\ 0^* \end{bmatrix} \int u^2 K(u) du I \]

\[ \left( \begin{array}{c} \hat{m}^A_j(x_j) \\ \hat{m}^A_{A,j}(x_j) \end{array} \right) = \tilde{M}_j^{-1}(x_j) \frac{1}{N} \sum_{i=1}^{n} K_h(X_{i,j}, x_j) \left( \frac{1}{h} \right) (Y_i - r_{\text{true},0}(X_i)) \]

\[ \alpha_{n,j}(x_j) = r_{\text{add},j}^n(x_j) + h^2 r_{\text{add},j}^n(x_j) \int u^2 K(u) du \]

\[ \alpha_j^n(x_j) = hr_{\text{add},j}^n(x_j) \]

\[ \gamma_{n,j} = \nu_{n,j} + h^2 \int u^2 K(u) du \int r_{\text{add},j}^n(x_j) f_j(x_j) dx_j \]

\[ \nu_{n,j} = \int r_{\text{add},j}^n(x_j) K_h(x_j, u) f_j(u) du \int dx_j \]

Choose \( S_1, \ldots, S_d \) closed subsets of \( (0, 1) \). If we assume (MLN:B1,B2'–B4') and (C2) (say: \( h \) has order \( n^{-1/5} \)), then the conditions (MLN:A1')–(MLN:A6'),(MLN:A8'), and (MLN:A9') are satisfied.

Using Theorem MLN:3', assumption (MLN:A7') is also satisfied with

\[ \mu_{n,j}(x_j) = \alpha_{n,j}(x_j) - \gamma_{n,j}. \]

Theorems MLN:2' and MLN:3' mention asymptotic behavior in general cases and are therefore omitted here.
A.4 Assumptions in MLN

The following assumptions are used in the proofs. Theorem MLN:4’ assures that they are fulfilled.

Assumptions

Suppose, there exists a density function $f$ on $\mathbb{R}^d$ with marginals

$$f_j(x_j) = \int f(x) dx_{-j}$$

and

$$f_{j,k}(x_j, x_k) = \int f(x) dx_{-j,k} \quad (j \neq k)$$

and a deterministic, symmetric, positive definite Matrix $W$ of dimension $d + 1$ with elements $W_{\ell,l}$, $\ell, l = 0, \ldots, d$ and $W_{0,0} = 1$.

$$M_j(x_j) = \begin{bmatrix} W_{0,0} & W_{j,0} \\ W_{j,0} & W_{j,j} \end{bmatrix} f_j(x_j); S_{k,j}(x_k, x_j) = \begin{bmatrix} W_{0,0} & W_{k,0} \\ W_{j,0} & W_{k,j} \end{bmatrix} f_{k,j}(x_k, x_j)$$

$\hat{M}_j(x_j)$ and $\hat{S}_{k,j}(x_k, x_j)$ are defined as in (MLN:46) and (MLN:47), respectively.

(MLN:A1’): For all $j \neq k$, it holds that

$$\int \frac{f_{j,k}^2(x_j, x_k)}{f_j(x_j)f_k(x_k)} dx_j dx_k < \infty$$

(MLN:A2’): For all $j \neq k$, it holds that

$$\int \left( \frac{\hat{V}_{j,0}^j(x_j) - f_j(x_j)}{f_j(x_j)} \right)^2 f_j(x_j) dx_j = o_P(1),$$

$$\int \left( \frac{\hat{V}_{j,k}^{j,k}(x_j, x_k) - f_{j,k}(x_j, x_k)}{f_j(x_j)f_k(x_k)} \right)^2 f_k(x_k)f_j(x_j) dx_j dx_k = o_P(1),$$

$$\int \left[ \hat{M}^{-1}_j(x_j) \hat{S}_{k,j}(x_k, x_j) - M^{-1}_j(x_j) S_{k,j}(x_k, x_j) \right]^2 f_k(x_k)^{-1} f_j(x_j) dx_j dx_k = \begin{bmatrix} o_P(1) & o_P(1) \\ o_P(1) & o_P(1) \end{bmatrix}$$

Furthermore, $\hat{M}_j$ and $\hat{S}_{j,k}$ vanish outside the support of $f_j$ and $f_{j,k}$, respectively.

(MLN:A3’): There exists a constant $C$ such that with probability tending to one for all $j$

$$\int \hat{m}_j(x_j)^2 f_j(x_j) dx_j \leq C$$

and

$$\int \hat{m}_j(x_j)^2 f_j(x_j) dx_j \leq C.$$
(MLN:A4'): For some finite intervals $S_j \subset \mathbb{R}$ that are contained in the support of $f_j$, $j = 1, \ldots, d$, we suppose that there exists a finite constant $C$ such that with probability tending to one for all $j \neq k$,

$$\sup_{x_j \in S_j} \int \text{trace}(\hat{S}_{j,k}(x_j, x_k) \hat{M}_j^{-2}(x_j) \hat{S}_{k,j}(x_k, x_j)) f_k(x_k)^{-1} dx_k \leq C.$$  

Decompose the smoothers $\hat{\mu}_j$ and $\hat{\mu}_j^A$ into a stochastic part $A$ and an expectation part $B$: $\hat{\mu}_j = \hat{\mu}_j^A + \hat{\mu}_j^B$ and $\hat{\mu}_j = \hat{\mu}_j^A + \hat{\mu}_j^B$. For $s \in \{A, B\}$ define $\hat{r}_{\text{add}, j}^s$, $\hat{r}_{\text{add}}^s A$, and $\hat{r}_{\text{add}, 0j}$ as the solution of the following equations

$$\hat{M}_j(x_j) \left( \hat{r}_{\text{add}, j}^s(x_j) + \hat{r}_{\text{add}, 0j}^s - \hat{\mu}_j^s(x_j) \right) = -\sum_{k \neq j} \int \hat{S}_{k,j}(x_k, x_j) \left( \hat{r}_{\text{add}, j}^s(x_j) \hat{r}_{\text{add}}^A(x_j) \right) dx_k \quad (MLN:51)$$

$$\int \hat{r}_{\text{add}, j}^s(x_j) \hat{V}_{0,0}^j(x_j) dx_j = 0 \quad (MLN:52)$$

Compare this with (MLN:44) and (MLN:45). Obviously, $\hat{r}_{\text{add}} = \hat{r}_{\text{add}}^A + \hat{r}_{\text{add}}^B$.

(MLN:A5'): There exists a constant $C$ such that with probability tending to one for all $j$

$$\int \hat{\mu}_j^s(x_j)^2 f_j(x_j) dx_j \leq C, \quad s = A, B$$

and

$$\int \hat{\mu}_j^s(x_j)^2 f_j(x_j) dx_j \leq C, \quad s = A, B.$$  

(MLN:A6'): Define

$$\left( \begin{array}{c} \hat{\mu}_k^A[j](x_k) \\ \hat{\mu}_k^B[j,k](x_k) \end{array} \right) = \int \hat{M}_k^{-1}(x_k) \hat{S}_{j,k}(x_j, x_k) \left( \begin{array}{c} \hat{\mu}_j^A(x_j) \\ \hat{\mu}_j^B(x_j) \end{array} \right) dx_j.$$  

We suppose there exists a sequence $\Delta_n$ with

$$\sup_{x_k \in S_k} \left( \hat{\mu}_k^A[j](x_k)^2 + \hat{\mu}_k^A[j,k](x_k)^2 \right) = o_P(\Delta_n)$$

$$\int \left( \begin{array}{c} \hat{\mu}_k^A[j](x_k) \\ \hat{\mu}_k^B[j,k](x_k) \end{array} \right)^\top M_k(x_k) \left( \begin{array}{c} \hat{\mu}_k^A[j](x_k) \\ \hat{\mu}_k^B[j,k](x_k) \end{array} \right) dx_k = o_P(\Delta_n)$$

where the $S_k$ have been introduced in (MLN:A4').

(MLN:A7'): There exist deterministic functions $\mu_{n,j}(x_j)$ with

$$\sup_{x_k \in S_k} \left| \hat{\mu}_{\text{add}}^B[j](x_j) - \mu_{n,j}(x_j) \right| = o_P(\Delta_n)$$

with $\Delta_n$ as in (MLN:A6') and $S_k$ as in (MLN:A4').

Assumption (MLN:A7') introduces the asymptotic expectation $\mu_{n,j}(x_j)$. To calculate this, use the assumptions (MLN:A8') and (MLN:A9') in conjunction with (MLN:A1')–(MLN:A6'). See also Theorem MLN:3'.

(MLN:A8'): Suppose for all $j \neq k$ and $s, t = 1, 2$

$$\sup_{x_k \in S_k} \left| \left[ \hat{M}_k^{-1}(x_k) \hat{S}_{j,k}(x_j, x_k) - M_k^{-1}(x_k) S_{j,k}(x_j, x_k) \right]_{s,t} \right| f_j(x_j) dx_j = o_P(\Delta_n)$$
Because the second term adds a penalty on non–additive $\tilde{r}$ of ($\ast$). According to Rao (1973), page 27, this holds if and only if

$$I - \hat{\alpha} \hat{x} = \mu_n, \ldots, \gamma_n$$

This condition may be rewritten as

$$I - \hat{\alpha} \hat{x} = \mu_n, \ldots, \gamma_n$$

This is non–singular if and only if

$$I - \hat{\alpha} \hat{x} = \mu_n, \ldots, \gamma_n$$

Decompose ($I - Z A^{(R)} Z^\top)$ into two orthogonal, non–negative definite parts:

$$I - Z A^{(R)} Z^\top = \{I - Z Z^\top\} + \{Z(I - A^{(R)}) Z^\top\}.$$ This is non–singular if and only if

$$\text{Im}(Z Z^\top) = \text{Im}(Z(I - A^{(R)}) Z^\top).$$

This condition may be rewritten as

$$\text{Im}(Z) = \text{Im}(Z(I - A^{(R)})).$$

According to Rao (1973), page 27, this holds if and only if

$$\text{Im}(I - A^{(R)}) \supseteq \text{Im}(Z^\top) \equiv \text{Im}(P_{\text{add}}) \quad (\ast)$$

Now, prove that ($\ast$) is equivalent to the uniqueness of the minimum in (6). The minimum of (6) is unique if and only if

$$A + (I - P_{\text{add}}) > 0 \quad (\text{positive definite}).$$

Because the second term adds a penalty on non–additive $\tilde{r}$, the null space of (**) must be additive. Accordingly, (**) is non–singular if and only if

$$\text{Im}(P_{\text{add}} A P_{\text{add}}) = \text{Im}(P_{\text{add}}).$$

This is equivalent to ($\ast$) using analogous arguments.

Therefore, the minimum of (6) is unique if and only if $I - Z A^{(R)} Z^\top$ is non–singular.
B.2 Section 3.2.3

$P_A^{(R)}$ is a projection:

Define

$$B := I - ZA^{(R)}Z^\top \equiv (I - ZZ^\top) + Z(I - A^{(R)})Z^\top. \quad (*)$$

$BB^-$ restricted to $\text{Im}(B)$ is the identity. This is an alternative formalization of $BB^-B = B$.

Because $\text{Im}(Z(I - A^{(R)})) = \text{Im}(Z(I - A^{(R)})Z^\top)$ is a subset of $\text{Im}(B)$, we get

$$BB^-Z(I - A^{(R)}) = Z(I - A^{(R)}). \quad (*)$$

Furthermore, using

$$(I - ZZ^\top)Z = 0 \quad (*)$$

and the definition of $B$, we get

$$(I - ZZ^\top)B = (I - ZZ^\top). \quad (*)$$

Applying $(*)$–$(*)$ results in

$$(I - ZZ^\top)B^-Z(I - A^{(R)}) = 0 \quad (*)$$

$P_A^{(R)} = Z^\top B^-Z(I - A^{(R)})$ is a projection because

$$(P_A^{(R)})^2 = Z^\top B^-Z(I - A^{(R)}Z^\top B^-Z(I - A^{(R)})$$

$$(=) Z^\top B^-B - B(I - ZZ^\top)B^-Z(I - A^{(R)})$$

$$(=) Z^\top B^-BB^-Z(I - A^{(R)})$$

$$(=) Z^\top B^-Z(I - A^{(R)}) = P_A^{(R)}$$

In general, $\text{Im}(P_A^{(R)}) \subseteq \text{Im}(Z^\top) \equiv \text{Im}(P_{\text{add}})$.

If $B$ is non–singular, we get equality: We claim that $P_A^{(R)}$ restricted to $\text{Im}(P_{\text{add}}) = \text{Im}(Z^\top)$ is the identity. This is proved by

$$(I - P_A^{(R)})Z^\top Z = Z^\top (I - B^{-1}Z(I - A^{(R)})Z^\top Z$$

$$(=) Z^\top (I - B^{-1}(B - (I - ZZ^\top)))Z$$

$$(=) Z^\top (I - B^{-1}B)Z = 0. \quad (*)$$

B.3 Section 3.2.4

Additive components of $\tilde{t}_{rt}$:

Because $P_{\text{add}}Z^\top = Z^\top$ we immediately get $P_{\text{add}}P_A^{(R)} = P_A^{(R)}$. Because

$$(I - A^{(R)})P_A^{(R)} = (I - A^{(R)})Z^\top B^-Z(I - A^{(R)}) = P_A^{(R)}(I - A^{(R)})$$

we obtain (assume: $B$ is non–singular)

$$P_{\text{add}}(I - A^{(R)})(I - P_A^{(R)}) = Z^\top Z(I - P_A^{(R)})^\top (I - A^{(R)}) \quad (*)$$
Without the assumption that $B$ is non–singular, we obtain
\[
Z(I - A^{(R)})P_{A}^{(R)} = (Z(I - A^{(R)})Z^\top)B^{-}Z(I - A^{(R)})
\]
\[
= (B - (I - ZZ^\top))B^{-}Z(I - A^{(R)})
\]
\[
= BB^{-}Z(I - A^{(R)})
\]
\[
= Z(I - A^{(R)}).
\]

Therefore
\[
Z^\top Z(I - A^{(R)})(I - P_{A}^{(R)}) = 0.
\]

Consider the following modification of (12) as a definition for $\tilde{\ell}_R$
\[
\tilde{\ell}_R := (I - A^{(R)})(I - P_{A}^{(R)})\tilde{\ell}_{NW} + \tilde{\ell}_{NW}^{add}
\]
where $\tilde{\ell}_{NW}^{add}$ is an arbitrary additive estimator. In order to preserve the decomposition into additive and orthogonal components, we did not modify the projection in the first term. Hence,
\[
\tilde{\ell}_R = (I - A^{(R)})\tilde{\ell}_{NW} + A^{(R)}P_{A}^{(R)}\tilde{\ell}_{NW} + \left(\tilde{\ell}_{NW}^{add} - P_{A}^{(R)}\tilde{\ell}_{NW}\right)
\]
and the estimator is not any more a pointwise (design adaptive) convex combination of a $\tilde{\ell}_{NW}$ and some additive estimator.

**Proof of proposition 3.2:**
Let $R_0 \subseteq \mathbb{R}^{m^*}$ be the null space of $(I - ZA^{(R)}Z^\top)$. If $R_0 \neq \{0\}$, then $P_{A}^{(R)}\tilde{\ell}_{NW}$ is ambiguous. The difference between two solutions is an element of $Z^\top R_0$. Now, we annihilate all components $j'$ with $a_{j'} = 0$. This is done by multiplying with $I - A^{(R)} = diag_j(a_{j})^{(1)}$.

$R_0$ (the null space of $(I - ZZ^\top) + Z(I - A^{(R)}Z^\top)$ is a subset of the null space of $Z(I - A^{(R)}Z^\top)$, which is equal to the null space of $(I - A^{(R)})Z^\top$:
\[
(I - A^{(R)})Z^\top R_0 = \{0\}
\]
We conclude that for every vector $\mathbf{v} \equiv \text{col}_j(v_j) \in Z^\top R_0$, $v_j = 0$ for every $j$ with $a_j > 0$.

**B.4 Section 3.3**

**Proof of equation (14) and proposition 3.3:**
The following proposition is adapted from Rao (1973). We will apply formula 1f.1.11 page 61:

**Proposition B.1** **Extrema of quadratic forms under linear restrictions.** Assume $\mathbf{u} \neq 0$. $S$ non–negative definite and some generalized inverse of the form
\[
\begin{pmatrix}
S & \mathbf{u} \\
\mathbf{u}^\top & 0
\end{pmatrix}^{-1}
= 
\begin{pmatrix}
C_1 & C_2 \\
C_3 & -a
\end{pmatrix}.
\]

Then
\[
\min_{\|\beta\|_2 = \tau} \beta^\top S\beta = \frac{1}{a} \tau^2.
\]

Formula (4i.1.7) in Rao (1973) establishes that $a(\bar{r} - \hat{r}_p)^2$ is invariant for any choice of the generalized inverse in (1f.1.10).

If $\mathbf{u} \in \text{Im}(S)$, then $\mathbf{u}^\top S^\top \mathbf{u} > 0$ and we may apply formula 4i.1(iv)(a) to obtain
\[
a = \frac{1}{\mathbf{u}^\top S^\top \mathbf{u}} > 0.
\]

If $\mathbf{u} \not\in \text{Im}S$, there exists a $\mathbf{w}$ with $\mathbf{w}^\top \mathbf{u} = 1$ and $S\mathbf{w} = 0$. Assume $\hat{\beta}_p$ is a minimizer of SSR($\hat{\beta}$). Then SSR($\hat{\beta}_p + \lambda \mathbf{w}$) = SSR($\hat{\beta}_p$) and $\mathbf{u}^\top (\hat{\beta}_p + \lambda \mathbf{w}) = \hat{r}_p + \lambda$. Hence, SSR($\bar{r}$) must be constant and $a = 0$.
Proof of proposition 3.4:
For $S = S^r \geq 0$ and $u = [1, 0, \ldots, 0]^T$ use the notation:

$$S = \begin{pmatrix} s_0 & s_1^T \\ s_1 & S_2 \end{pmatrix}$$

$S$ has dimensions $p \times p$.

Because $S \geq 0$, we may apply Rao and Kleffe (1988), paragraph 1.2(xiv):

$$(s_0 - s_1^T S_2 s_1)^{-1} - (s_0 - s_1^T S_2 s_1)^{-1} s_1 s_1^T (s_0 - s_1^T S_2 s_1)^{-1} s_1^T$$

is a generalized inverse of $S$.

Applying Rao and Kleffe (1988) 1.8(v), page 23, we obtain $s_0 - s_1^T S_2 s_1 \geq 0$ and $s_1 \in \text{Im}(S_2)$. The latter implies $S_2 S_2 s_1 = s_1$. The following calculation employs the above choice of $S^{-1}$.

$$SS^{-1} u = (s_0 - s_1^T S_2 s_1) \left[ \begin{array}{c} s_0 - s_1^T S_2 s_1 \\ s_1 - S_2 s_1 \\ S_2 - S_2 s_1 \end{array} \right] = \begin{cases} u, & \text{if } s_0 - s_1^T S_2 s_1 \neq 0 \\ 0, & \text{if } s_0 - s_1^T S_2 s_1 = 0. \end{cases}$$

Apply the property that $SS^{-1} u = u$ if and only if $u \in \text{Im}(S)$. Hence, $u \in \text{Im}(S)$ is equivalent to $(s_0 - s_1^T S_2 s_1) > 0$.

We conclude that

$$s_0 - s_1^T S_2 s_1 = \begin{cases} \frac{1}{S^{-1} u}, & \text{if } u \in \text{Im}(S) \\ 0, & \text{if } u \notin \text{Im}(S) \end{cases}$$

which is equal to $a$ by proposition 3.3.

C Proofs for section 4.1

C.1 Proof of lemma 4.1

C.1.1 Overview

Here, we calculate the projection $\Pi_* : \mathcal{F}_{\text{full}} \rightarrow \mathcal{F}_{\text{add}}$

$$\Pi_* \hat{r} = \arg \min_{\hat{r}_{\text{add}} \in \mathcal{F}_{\text{add}}} \| \hat{r} - \hat{r}_{\text{add}} \|_2$$

We proceed as MLN, who solve

$$\arg \min_{\hat{r}_{\text{add}} \in \mathcal{F}_{\text{add}}} \| r_Y - \hat{r}_{\text{add}} \|_2.$$  

The normal equations for $\Pi_*$ look like (MLN:32/33) but with

$$\frac{1}{n} \sum_{i=1}^n K_k(X_{i,j}, x_j) Y_i \left( \frac{1}{n} \sum_{i,j} - x_j \right)$$

replaced by

$$\sum_{k=0}^d \int \left( \hat{V}_{k,0}(x) \hat{p}^k(x) \right) d\hat{\pi}_{-j}.$$

Here we define $\hat{m}_j(x_j)$ and $\hat{m}^j(x_j)$ as

$$\hat{M}_j(x_j) \left( \hat{m}_j(x_j) \hat{m}^j(x_j) \right) = \sum_{k=0}^d \int \left( \hat{V}_{k,0}(x) \hat{p}^k(x) \right) d\hat{\pi}_{-j}.$$  

This corresponds to the definition of $\hat{m}_j(x_j)$ and $\hat{m}^j(x_j)$ in (MLN:39/40) with the above replacement.

Lemma 4.1 states that $\Pi_*$ and $S_{\text{add}}^{-1}$ are continuous with probability tending to one.
Projection $\Pi_s$ is continuous

Looking into the details of the proofs in MLN, we find that the calculation of $\hat{r}_{\text{add}}$ from $(\hat{m}_j, \hat{m}^j)_{j=1,...,d}$, corresponds to a continuous linear mapping. MLN proved this for the Nadaraya–Watson case and gave a sketch for the proof of the local linear case. The relevant Lemmas MLN:1 and MLN:2 are transcribed as Lemma C.1 and C.2 below. Adaptations of formulas for the Nadaraya–Watson case to the local linear case are indicated by a prime, see e.g. (MLN:70') and (MLN:70) below. Using (MLN:84') on page 74 proves the claim about the representation as a continuous linear mapping.

Hence it remains to write $(\hat{m}_j, \hat{m}^j)$ as a continuous (with probability tending to one) linear mapping of $\hat{r}$: Define the linear operator $C_j : \mathcal{F}_{\text{full}} \rightarrow L^2([0, 1]) \times L^2([0, 1])$ by

$$C_j r = \int \left( \frac{\hat{r}_j(x)}{\hat{m}_j(x)} \right) d\mathcal{F}_{-j}.$$ 

Then (✱) may be rewritten as

$$\hat{M}_j \left( \frac{\hat{m}_j}{\hat{m}^j} \right) = C_j \hat{P}_{\text{add}} S_s \hat{r}.$$  

MLN mention in their sketch of proof of Theorem MLN:4’ that “First one shows uniform convergence of $\hat{M}_j(x_j)$ to $M_j(x_j)$” (page 1488). Hence, we may assume (with probability tending to one) that $\hat{M}_j^{-1}(x_j)$ is “good natured”.

Now, we put the pieces together:

- calculating $\hat{m}_j, \hat{m}^j$ from $\hat{r}$ is continuous (from (✱)).
- The calculation of $\hat{r}$ (defined in (36) on page 74) is continuous. For the Nadaraya–Watson case, $\hat{r}$ is defined in Lemma MLN:3. For the local linear case, see also pages 1481 and 1482.
- $\hat{r}_{\text{add}}$ is of the form $\hat{r}_{\text{add}} = \sum_{k=0}^{\infty} \hat{T}^k \hat{r}$, and $\sum_{k=0}^{\infty} \hat{T}^k$ is continuous. The operator $\hat{T}$ is used in the backfitting algorithm (section C.2.1). If $\hat{T}$ is a contraction, the backfitting algorithm converges. In section C.1.3 the operator $\hat{T}$ converges to some operator $T$, which is a contraction (section C.1.2).

In each step, continuity holds with probability tending to one. Note that MLN use a different (design dependent) norm, but that norm is equivalent to the $L^2$-norm, because of assumption (MLN:B2').

We conclude that under the assumptions (MLN:B1), (MLN:B2')–(MLN:B4'), (C2) and with probability tending to one as $n$ tends to infinity, $\Pi_s$ is continuous and has bounded operator–norm.

Operator $S_{\text{add}}$ has continuous inverse

The normal equations for $\Pi_s \hat{r} = \arg \min_{\hat{r}_{\text{add}} \in \mathcal{F}_{\text{add}}} \| \hat{r} - \hat{r}_{\text{add}} \|_{s, s}^2$, $\hat{r} \in \mathcal{F}_{\text{full}}$ are

$$S_{\text{add}} \hat{r}_{\text{add}} = \hat{P}_{\text{add}} S_s \hat{r}.$$ 

In the proof of the continuity of $\Pi_s$, we calculated $\Pi_s$ as follows

$$\hat{r} \xrightarrow{\hat{P}_{\text{add}} S_s} \hat{M}_j^{-1} C_j \hat{m}_j, \hat{m}^j \xrightarrow{(36)} \hat{r}_{\text{add}}.$$ 

Because $\hat{P}_{\text{add}} S_s \hat{r}_{\text{add}} = S_{\text{add}} \hat{r}_{\text{add}}$ and $\Pi_s \hat{r}_{\text{add}} = \hat{r}_{\text{add}}$ (for $\hat{r}_{\text{add}} \in \mathcal{F}_{\text{add}}$) we have $\hat{r}_{\text{add}} = S_u \hat{r}_{\text{add}}$. Hence, with probability tending to one, $S_u$ and $S_{\text{add}}$ are continuous and there exist lower $(C_4)$ and upper $(C_5)$ bounds

$$C_5 \| \hat{r}_{\text{add}} \|_{L^2} \geq \| \hat{r}_{\text{add}} \|_{s} \geq C_4 \| \hat{r}_{\text{add}} \|_{L^2} \quad \forall \hat{r}_{\text{add}} \in \mathcal{F}_{\text{add}}.$$ 

70
By the same argument as in section C.2.3 we conclude that $S_{\text{add}}$ is bijective and $S_{\text{u}} = S_{\text{add}}^{-1}$ is continuous.

C.1.2 Lemma MLN:1’

**Notation**

Using the notation $\text{var}(K) = \int u^2 K(u) du$, $W$ has the special form

$$W = \text{diag}(1, \text{var}(K), \ldots, \text{var}(K))$$

and

$$M_j(x_j) = \begin{bmatrix} 1 & 0 & \text{var}(K) \\ 0 & 1 & 0 \end{bmatrix} f_j(x_j), \quad S_{\ell,j}(x_\ell, x_j) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} f_{\ell,j}(x_\ell, x_j)$$

Define the spaces

$$H = \left\{ r = (r^0, \ldots, r^d) \mid r^k \in L^2(f) \ \forall k = 0, \ldots, d \text{ and there exist functions } r_j \in L^2(f_j) \text{ and } r^0(x) = r_1(x_1) + \ldots + r_d(x_d) \text{ and } r^j(x) \text{ depends only on } x_j \right\} \subset L^2(W_f)$$

$$H^0 = \left\{ r \in H \mid \int r^0(x) f(x) dx = 0 \right\}$$

$$H_j = \left\{ r \in H^0 \mid \text{ with } r^\ell(x_\ell) = 0 \text{ and } r_\ell(x_\ell) = 0 \ \forall \ell \neq j \right\}$$

Define the norm on $H$ by

$$\|r\|_H^2 = \int r^0(x)^2 f(x) dx + \text{var}(K) \sum_{\ell=1}^d \int r^\ell(x_\ell)^2 f_\ell(x_\ell) dx_\ell$$

and introduce a notation for its restriction to $H_j$

$$\left\| \begin{bmatrix} r_j \\ r^j \end{bmatrix} \right\|_{H_j}^2 = \int r_j(x_j)^2 f_j(x_j) dx_j + \text{var}(K) \int r^j(x_j)^2 f_j(x_j) dx_j.$$

This notation will also be used, when $\int r_j(u) f_j(u) du \neq 0$.

Define $\Psi_j : H \to H$ with

$$\Psi_j r(x) = (\Psi^0_j r(x), \Psi^1_j r(x), \ldots, \Psi^d_j r(x))$$

$$\Psi^0_j r(x) = \int r_j(u) f_j(u) du + \sum_{k \neq j} r_k(x_k)$$

$$- \sum_{k \neq j} \int \frac{f_{j,k}(x_j, v)}{f_j(x_j)} \left( r_k(v) - \int r_k(u) f_k(u) du \right) dv$$

$$\Psi^j_j r(x) = 0$$

$$\Psi_j r^k(x) = r^k(x_k), \quad k \neq j, 0$$

$I - \Psi_j$ is the orthogonal (with respect to $\|\cdot\|_H$) projection from $H$ to $H_j$. 

71
C.1 Proof of lemma 4.1 C.1 PROOFS FOR SECTION 4.1

For linear operators \( S : \mathcal{H} \rightarrow \mathcal{H} \) define

\[
\|S\|_{\sup(\mathcal{H})} = \sup \left\{ \|Sr\|_{\mathcal{H}} \mid r \in \mathcal{H} \text{ with } \|r\|_{\mathcal{H}} \leq 1 \right\}
\]

\[
\|S\|_{\sup(\mathcal{H}^0)} = \sup \left\{ \|Sr\|_{\mathcal{H}} \mid r \in \mathcal{H}^0 \text{ with } \|r\|_{\mathcal{H}} \leq 1 \right\}
\]

\[
\|S\|_{\sup(\mathcal{H}^{0,n})} = \sup \left\{ \|Sr\|_{\mathcal{H}} \mid r \in \mathcal{H}^{0,n} \text{ with } \|r\|_{\mathcal{H}} \leq 1 \right\}
\]

\( \mathcal{H}^{0,n} \) is a subspace of \( \mathcal{H} \) with a condition depending on the empirical density, and therefore defined in the next paragraph below.

Define \( T = \Psi_d \cdots \Psi_1 \).

Transcription of Lemma MLN:1. In the following, we present an adaptation to the local linear case of the Lemma 1 in the appendix of MLN.

**Lemma C.1** Suppose that condition (MLN:A1’) holds. Then \( T : \mathcal{H} \rightarrow \mathcal{H} \) is a positive, self-adjoint operator with \( \|T\|_{\sup(\mathcal{H}^0)} < 1 \). Hence, for \( r = 1, 2, \ldots \)

\[
\|Tr\|_{\sup(\mathcal{H}^0)} \leq \|T\|_{\sup(\mathcal{H}^0)} (\text{MLN:69'})
\]

Furthermore, for every \( r \in \mathcal{H}^0 \) there exist functions \( r_j \) and \( r_j^i \) in \( L^2(\mathcal{H}_j) \) with identifiability condition \( \int r_j(x_j)f_j(x_j)dx_j = 0 \) and \( r(\mathbf{x}) = (r_1(x_1) + \ldots + r_d(x_d), r_1(x_1), \ldots, r_d(x_d)) \) (f a.s.) and there exists a constant \( c > 0 \) with

\[
\|r\|_{\mathcal{H}} \geq c \max_{j=1,\ldots,d} \left\| \left( r_j \right) \right\|_{\mathcal{H}_j} \quad \text{(MLN:70')}
\]

MLN proved the above Lemma for the Nadaraya–Watson case. For the local linear case, they gave a sketch of proof. Denote by \( T^{NW} \) the corresponding operator in the Nadaraya–Watson case.

(MLN:69’) directly follows from (MLN:69) because

\[
Tr = (T^{NW}r^0, 0, \ldots, 0)
\]

Choose \( r \in \mathcal{H}^0 \) and apply

\[
\|r^0\|_f \geq c \max_{j=1,\ldots,d} \|r_j\|_f \quad \text{(MLN:70)}
\]

to obtain

\[
\|r\|^2_{\mathcal{H}} = \|r^0\|^2_f + \text{var}(K) \sum_{\ell=1}^d \|r^\ell\|^2_f \\
\quad \geq c^2 \max_{j=1,\ldots,d} \|r_j\|^2_f + d \text{var}(K) \max_{\ell=1,\ldots,d} \|r^\ell\|^2_f \\
\quad \geq \min(c^2, d) \max \left( \frac{\|m_j\|_{\mathcal{H}_j}}{m_j} \right).
\]

This finishes the proof of Lemma C.1.

C.1.3 Lemma MLN:2’

The operator \( \Psi_j \) utilizes the (unknown) density \( f \). Approximate \( \Psi_j \) by an operator \( \hat{\Psi}_j \), which uses estimations for the two-dimensional marginal distributions. The goal is to demonstrate that the sup–norm of the approximation of \( T \) is still strictly smaller than one (with probability tending to one).
Define the spaces
\[ \mathcal{H}^{0,n} = \left\{ r \in \mathcal{H} : \int r_j(x_j) \hat{V}^j_{0,0}(x_j) dx_j = 0 \right\} \]
\[ \mathcal{H}^n_j = \left\{ r \in \mathcal{H}^{n,0} : \text{with } r^\ell(x_\ell) = 0 \text{ and } r_\ell(x_\ell) = 0 \forall \ell \neq j \right\} \]

Define \( \hat{\Psi}_j : \mathcal{H}^{0,n} \to \mathcal{H}^{0,n} \) by
\[
\hat{\Psi}_j(r(x)) = \left( \hat{\Psi}^0_j(r(x)), \hat{\Psi}^1_j(r(x_1), \ldots, \hat{\Psi}^d_j(r(x_d)) \right)
\]
\[
\hat{\Psi}^0_j(r(x)) = \hat{\Psi}_{j,1}r(x_1) + \ldots + \hat{\Psi}_{j,d}r(x_d)
\]
\[
\hat{\Psi}_{j,k}r(x_k) = \begin{cases} \hat{g}_j(x_j) - \int \hat{g}_j(u) \hat{V}^j_{0,0}(u) du & k = j \\ r_k(x_k) & k \neq j \end{cases}
\]
\[
\hat{\Psi}_{j,k}r(x_k) = \begin{cases} \hat{g}^j(x_j) & k = j \\ r^k(x_k) & k \neq j \end{cases}
\]
\[
\left( \begin{array}{c} \hat{g}_j(x_j) \\ \hat{g}^j(x_j) \end{array} \right) = -\sum_{k \neq j} \int \hat{M}^{-1}_j(x_j) \hat{S}_{k,j}(x_k, x_j) \left( \begin{array}{c} r_k(x_k) \\ r^k(x_k) \end{array} \right) dx_k.
\]

Furthermore, define
\[ \hat{T} = \hat{\Psi}_d \cdots \hat{\Psi}_1 \]

In the following, we show how these operators may be used to solve the normal equations. \( \hat{T} \) converges to \( T \), when restricted to \( \mathcal{H}^{0,n} \). Hence, \( \hat{T} \) is a contraction (with probability tending to one) and the backfitting algorithm converges.

**Transcription of Lemma MLN:2.** In the following, we present an adaptation to the local linear case of the Lemma 2 in the appendix of MLN. The lemma states that the operator \( \hat{T} \) converges to the operator \( T \) (which depends on the unknown design density).

**Lemma C.2** Suppose that conditions (MLN:A1') and (MLN:A2') hold, then
\[ \| \hat{\Psi}_j - \Psi_j \|_{\sup(\mathcal{H}^{0,n})} = o_P(1) \]
\[ \| \hat{T} - T \|_{\sup(\mathcal{H}^{0,n})} = o_P(1) \]

Choose \( \gamma < 1 \) with \( \| T \|_{\sup(\mathcal{H}^0)} < \gamma \). Then, with probability tending to one,
\[ \| \hat{T} \|_{\sup(\mathcal{H}^{0,n})} \leq \gamma. \]

Furthermore, for every \( r \in \mathcal{H}^{0,n} \) there exist functions \( r_j \) and \( r^j \) in \( L^2(f_j) \) with identifiability condition \( \int r_j(x_j) \hat{V}^j_{0,0}(x_j) dx_j = 0 \) and \( r(x) = (r_1(x_1) + \ldots + r_d(x_d), r^1(x_1), \ldots, r^d(x_d)) \) (f. a. s.) and there exists a constant \( c > 0 \) such that with probability tending to one it holds
\[ \| r \|_{\mathcal{H}} \geq c \max_{j=1,\ldots,d} \left\| \begin{array}{c} r_j \\ r^j \end{array} \right\|_{\mathcal{H}_j}. \]

**Some motivation.** In the following, we write the normal equations using these operators — see equation (MLN:84') below. Because of (MLN:76') in the previous lemma, this equation has a unique solution in \( \mathcal{H}^{0,n} \) with probability tending to one. Furthermore the backfitting algorithm mentioned in section C.2.1 below converges.
C.1 Proof of lemma 4.1

Recall that the normal equations (MLN:44) state
\[
\left(\hat{\tau}_{\text{add},j}(x_j)\right) = \left(\hat{m}_j(x_j) - \text{const}\right) - \sum_{k \neq j} \int M_j^{-1}(x_j) S_k(x_k, x_j) \left(\hat{\tau}_{\text{add},k}(x_k)\right) dx_k
\]

Because we do the calculations in \(H^{0,n}\), we have to subtract some constants: Define
\[
\hat{\tau}_{\text{add}}(x) = (\hat{\tau}_{\text{add},1}(x_1) + \ldots + \hat{\tau}_{\text{add},d}(x_d), \hat{\tau}_{\text{add},1}(x_1), \ldots, \hat{\tau}_{\text{add},d}(x_d))
\]

and
\[
\hat{m}_j(x_j) = \hat{m}_j(x_j) - \int \hat{m}_j(u) \hat{V}_{0,0}(u) du.
\]

Then the normal equations are
\[
\hat{\tau}_{\text{add}}(x) = \hat{\Psi} \hat{\tau}_{\text{add}}(x) + (\hat{\tau}_{\text{add},1}(x_1), 0, \ldots, 0, \hat{\tau}_{\text{add},1}(x_1), 0, \ldots, 0)
\]

Define
\[
\hat{\tau} = \hat{\Psi}_d \cdots \hat{\Psi}_2 \hat{m}_1(x_1) \quad \hat{\tau}_2 \hat{m}_2(x_2) \quad \hat{\tau}_3 \hat{m}_3(x_3) \quad \ldots + \hat{\tau}_{\text{add}}(x) - \hat{\tau}(x).
\]

and we get
\[
\hat{\tau}_{\text{add}}(x) = \hat{T} \hat{\tau}_{\text{add}}(x) + \hat{\tau}(x).
\]

Proof of (MLN:77'). The definitions of the spaces \(H^0\) and \(H^{0,n}\) differ only in the norming conditions. Hence, given an element in one space, we may choose some in the other, such that they differ only by a constant added to the first component. This difference is asymptotically negligible, as shown below.

We choose some pair \((r, r^c) \in H^0 \times H^{0,n}\), which differs only by a constant: Let \(r = (r_1 + \ldots + r_d, r^1, \ldots, r^d) \in H^0\) (say: \(\int r_j(u) f_j(u) du = 0\)) and define \(r^\Delta = \int r_j(u) \hat{V}_{0,0}(u) du\), \(r_j(x_j) = r_j(x_j) - r_j^\Delta\)

\[
r^c(x) = (r^c_1(x_1) + \ldots + r^c_d(x_d), r^1(x_1), \ldots, r^d(x_d)) \in H^{0,n}
\]

Alternatively, \(r^c\) may be chosen and \(r\) depends on \(r^c\).

Because of the first item in assumption (MLN:A2'),
\[
|r^\Delta_j| = o_p(1) \|r_j\|_{f_j} = o_p(1) \|r_j^\Delta\|_{f_j}.
\]

Combining this with (MLN:70'), the claim (MLN:77') is easy to prove.

Proof of (MLN:76'). We already know from Lemma MLN:1' that for some \(\gamma < 1\), \(\|Tr\|_H < \gamma \|r\|_H\) for each \(r \in H^0 - \{0\}\). To find an upper bound for \(r^c \in H^{0,n}\), we use the above result that the difference is asymptotically negligible.

For the proof of claim (MLN:76'), consider
\[
\|Tr^c\|_H = \|Tr + \sum_{\ell=1}^d r^\Delta_\ell, 0, \ldots, 0\|_H
\]
\[
\leq \|Tr\|_H + \left|\sum_{\ell=1}^d r^\Delta_\ell\right|
\]
\[
\leq \|T\|_{H^0} \|r\|_H + o_p(1) \|r^c\|_H.
\]

74
Because \( \|r\|_\mathcal{H} \leq \|r^c\|_\mathcal{H} \), we get
\[
\|T\|_{\sup(\mathcal{H}^0, n)} \leq \|T\|_{\sup(\mathcal{H}^0)} + o_p(1).
\]

**Proof of (MLN:74').** We split \( \hat{\Psi}_j = \Psi \) into several pieces. For the first, we use the third item of assumption (MLN:A2') — see section A.4. The remaining terms are are approximations of sums of \( r_k^\Delta \), which was shown to be small enough.

Define
\[
g_j(x_j) = \sum_{k \neq j} \int \frac{f_{k,j}(x_k, x_j)}{f_j(x_j)} r_k^f(x_k)dx_k.
\]

We want to compare \( g_j \) with \( \hat{g}_j(x_j) \) in the definition of \( \hat{\Psi}_{j,r^c} \), because
\[
\begin{align*}
\left( \begin{array}{c} g_j(x_j) \\ 0 \end{array} \right) &= -\sum_{k \neq j} \int M_j^{-1}(x_j) S_{k,j}(x_k, x_j) \left( \frac{r_k^f(x_k)}{r_k^r(x_k)} \right) dx_k.
\end{align*}
\]

The relation between \( g_j \) and \( \Psi_j r^c \) is
\[
\Psi_j^0 r^c(x) = \left( g_j(x_j) + \sum_{\ell=1}^d r_{\ell}^\Delta \right) + \sum_{k \neq j} r_k^r(x_k).
\]

Now, decompose the left hand side of (MLN:74') into pieces, that may be handled by using assumption (MLN:A2'):
\[
\begin{align*}
\left\| (\hat{\Psi}_j - \Psi_j) r^c \right\|_{\mathcal{H}} &= \left\| \left( \frac{g_j(x_j) - \int \hat{g}_j(u) \hat{V}_{0,0}^i(u)du}{\hat{g}^j(x_j)} - 0 \right) \right\|_{\mathcal{H}_j} \\
&\leq \left\| \left( \frac{\hat{g}_j(x_j) - g_j(x_j)}{\hat{g}^j(x_j)} \right) \right\|_{\mathcal{H}_j} + \left| \int \hat{g}_j(u) \hat{V}_{0,0}^i(u)du \right| + \sum_{\ell=1}^d \left| r_{\ell}^\Delta \right|
\end{align*}
\]

The first term is bounded with the third item in assumption (MLN:A2')
\[
\begin{align*}
\left\| \frac{g_j(x_j) - \int \hat{g}_j(u) \hat{V}_{0,0}^i(u)du}{\hat{g}^j(x_j)} \right\|_{\mathcal{H}_j} &= \left\| \sum_{k \neq j} \int \left( M_j^{-1}(x_j) S_{k,j}(x_k, x_j) - \tilde{M}_j^{-1}(x_j) \tilde{S}_{k,j}(x_k, x_j) \right) \left( \frac{r_k^f(x_k)}{r_k^r(x_k)} \right) dx_k \right\|_{\mathcal{H}_j} \\
&= o_p(1) \left( r_k^c \right)_{\mathcal{H}_j} \quad \text{(MLN:77')}
\end{align*}
\]

This implies that \( \|\hat{g}_j - g_j\|_{f_j} = o_p(1) \|r^c\|_{\mathcal{H}} \) and therefore
\[
\begin{align*}
\left| \int \hat{g}_j(u) \left( \hat{V}_{0,0}^i(u) - f_j(u) \right)du \right| &= o_p(1) \|\hat{g}_j\|_{f_j} = o_p(1) \|r^c\|_{\mathcal{H}} \\
\left| \int \left( \hat{g}_j(u) - g_j(u) \right) f_j(u)du \right| &= o_p(1) \|r^c\|_{\mathcal{H}} \\
\left| \int g_j(u) f_j(u)du \right| &= \sum_{k \neq j} r_k^\Delta = o_p(1) \|r^c\|_{\mathcal{H}}.
\end{align*}
\]
Proof of (MLN:75').
Because $\Psi_j$ is an orthogonal projection
$$\|\Psi_j\|_{sup(\mathcal{H})} = 1.$$
In conjunction with (MLN:74'), this implies
$$\|\hat{\Psi}_j\|_{sup(\mathcal{H},n)} = \|\Psi_j\|_{sup(\mathcal{H},n)} + o_P(1)$$
Claim (MLN:75') is proven by induction.
Suppose that (MLN:75') holds for $k-1$, say:
$$\|\hat{\Psi}_{k-1} \cdots \hat{\Psi}_1 - \Psi_{k-1} \cdots \Psi_1\|_{sup(\mathcal{H},n)} = o_P(1). \quad (\circ)$$
For $k-1 = 1$, $(\circ)$ holds because of (MLN:74').
The induction step $k-1 \rightarrow k$ for $(\circ)$ is
$$\|\hat{\Psi}_k \cdots \hat{\Psi}_1 - \Psi_k \cdots \Psi_1\|_{sup(\mathcal{H},n)} \leq$$
$$\|\hat{\Psi}_k - \Psi_k\|_{sup(\mathcal{H},n)} \|\hat{\Psi}_{k-1} \cdots \hat{\Psi}_1\|_{sup(\mathcal{H},n)} + \|\Psi_k\|_{sup(\mathcal{H})} \|\hat{\Psi}_{k-1} \cdots \hat{\Psi}_1 - \Psi_{k-1} \cdots \Psi_1\|_{sup(\mathcal{H},n)} = o_P(1)$$

C.2 Some remarks

C.2.1 Backfitting
We may define a backfitting algorithm by
$$\hat{r}_{add}^{c,[a+1]} = \hat{T}_{add}^{c,[a]} + \hat{r}.$$ 
Because $\|\hat{T}\|_{sup(\mathcal{H},n)} \leq \gamma < 1$ (with probability tending to one), convergence is guaranteed (asymptotically) for any starting value $\hat{r}_{add}^{c,[0]}$ in $\mathcal{H}^{0,n}$.
The difference between the algorithms (MLN:49) and $(\ast)$ is that the former updates $\hat{r}_{add}^{c,j}$ after all components have been calculated (Jacobi iteration), while the latter updates $\hat{r}_{add,j}^{c,j}$ and $\hat{r}_{add,j+1}^{c,j+1}$ (Gauss–Seidel iteration). Formally, on the right hand side (MLN:49) has $[a]$ and $(\ast)$ has $[a + 1_{k<j}]$:
$$\left( \begin{array}{c} \hat{r}_{add}^{c,[a+1]}(x_j) \\ \hat{r}_{add}^{c,[a+1],j}(x_j) \end{array} \right) = - \sum_{k \neq j} M_j^{-1}(x_j) S_{k,j}(x_k, x_j) \left( \begin{array}{c} \hat{r}_{add,k}^{c,[a+1],j}(x_k) \\ \hat{r}_{add,k}^{c,[a+1],j}(x_k) \end{array} \right) dx_k$$
$$+ \left( \begin{array}{c} \hat{m}_j^c(x_j) \\ \hat{m}_j^c(x_j) \end{array} \right) - \left( \begin{array}{c} \text{average} \\ 0 \end{array} \right)$$
with $1_{k<j} = \begin{cases} 1, & k<j \\ 0, & \text{else} \end{cases}$.
The constant $\hat{r}_{add,0}$ may be easily calculated:
$$\hat{r}_{add,0} = \int (\hat{m}_j(x_j) + \hat{g}_j(x_j)) V_{0,0}(x_j) dx_j.$$
C.2.2 Convex combination

The convex combination property (20) holds also for the grid independent approach. Note that

\[ S_s + R I : \mathcal{F}_{\text{full}} \to \mathcal{F}_{\text{full}} \]

is symmetric, bijective, continuous with continuous inverse. Furthermore

\[ S_s(S_s + RI)^{-1} = (S_s + RI)^{-1}S_s = I - R(S_s + RI)^{-1} \]

is symmetric, and we have the following bound

\[ S_s(S_s + RI)^{-1} \geq \frac{1}{\|S_s\|_{\text{sup}(\mathcal{F}_{\text{full}})} + R} S_s \]

Define the operator \( \Lambda : \mathcal{F}_{\text{add}} \to \mathcal{F}_{\text{add}} \) by

\[ \Lambda = \tilde{P}_{\text{add}}(S_s + RI)^{-1}S_s \tilde{P}_{\text{add}} \]

If \( S_{\text{add}}^{-1} \) exists and is continuous, then \( \Lambda \) has a continuous inverse, because

\[ \Lambda \geq \frac{1}{\|S_s\|_{\text{sup}(\mathcal{F}_{\text{full}})} + R} S_{\text{add}}. \]

Recall \( S_{\text{add}} = \tilde{P}_{\text{add}}S_s \tilde{P}_{\text{add}} \), restricted to \( \mathcal{F}_{\text{add}} \). Obviously,

\[ P_s^{(R)} = \tilde{P}_{\text{add}}\Lambda^{-1} \tilde{P}_{\text{add}}(S_s + RI)^{-1}S_s \]

is a projection.

We need the following properties: \( \tilde{P}_{\text{add}}P_s^{(R)} = P_s^{(R)} \) and

\[ \tilde{P}_{\text{add}}(I - (S_s + RI)^{-1}R)P_s^{(R)} = \Lambda \Lambda^{-1} \tilde{P}_{\text{add}}(S_s + RI)^{-1}S_s \]

\[ = \tilde{P}_{\text{add}}(S_s + RI)^{-1}S_s. \] \((\Box)\)

Recall the normal equations (see equation (27) on page 33)

\[ S_s \hat{r}_{ll} = \hat{L} \quad \text{and} \quad S_R \hat{r}_R = \hat{L}. \]

We claim that \( \hat{r}_R \) may be written as

\[ \hat{r}_R = \left\{ (S_s + RI)^{-1}S_s + R(S_s + RI)^{-1}P_s^{(R)} \right\} \hat{r}_{ll} \]

Hence, \( \hat{r}_R \) is a convex combination of \( \hat{r}_{ll} \) and \( P_s^{(R)}\hat{r}_{ll} \).

This may be verified, using the normal equations:

\[ S_R \hat{r}_R = (\{S_s + RI\} - R\tilde{P}_{\text{add}})(S_s + RI)^{-1} \left\{ S_s + R P_s^{(R)} \right\} \hat{r}_{ll} \]

\[ = S_s \hat{r}_{ll} + R \tilde{P}_{\text{add}} \left\{ (I - (S_s + RI)^{-1}R)P_s^{(R)} - (S_s + RI)^{-1}S_s \right\} \hat{r}_{ll} \]

\[ = S_s \hat{r}_{ll} + R \left\{ \tilde{P}_{\text{add}}(S_s + RI)^{-1}S_s - \tilde{P}_{\text{add}}(S_s + RI)^{-1}S_s \right\} \hat{r}_{ll} = S_s \hat{r}_{ll}. \] \((\text{Def})\)

Note that \( \hat{r}_{ll} \) may be ambiguous, but \( S_s \hat{r}_{ll} \) is unique.
C.2.3 Remark on lower bound and continuous inverse

Now, we claim that
\[ C_5 \|r\|_{L^2} \geq \|r\|_R \geq C_4 \|r\|_{L^2} \quad \forall r \in \mathcal{F}_{\text{full}} \]  \hspace{1cm} (\bullet)\]
implies that \( \mathcal{S}_R \) is bijective and has a continuous inverse.

The property (\bullet) implies that \( \mathcal{S}_R \) is injective.

Note that \( \mathcal{F}_{\text{full}} \) equipped with the norm \( \| \cdot \|_{L^2} \) is complete. The graph \( \{(r, S_R r)| r \in \mathcal{F}_{\text{full}}\} \subseteq \mathcal{F}_{\text{full}}^2 \) is closed (with respect to the norm \( \| (r, S_R r) \| = \|r\|_{L^2} + \| S_R r \|_{L^2} \)), because \( \mathcal{S}_R \) is continuous.

Next, we are going to prove that the image of \( \mathcal{S}_R \) is closed. Choose \( r_s \in \mathcal{F}_{\text{full}} \) for \( s = 1, 2, \ldots \) such that \( (S_R r_s) \) is a Cauchy sequence. Hence, \( S_R r_s \) tends to \( r \in \mathcal{F}_{\text{full}} \). (\bullet) implies \( \| S_R r_s \|_{L^2} \geq C_4^2 \| r \|_{L^2} \). Therefore, also \( (r_s) \) is a Cauchy sequence and the limit is denoted by \( r \). Hence, \( (r_s, S_R r_s) \) is a Cauchy sequence in the graph of \( \mathcal{S}_R \) with limit \((r, w)\). Because the graph is closed, \((r, w)\) is an element of the graph, say: \( w = S_R r \). Accordingly, the image of \( \mathcal{S}_R \) is a closed, linear subspace of \( \mathcal{F}_{\text{full}} \).

If \( \mathcal{S}_R \) is not surjective, then we chose some element of the orthogonal complement of the image of \( \mathcal{S}_R \) and get some contradiction: In that case there exists some \( r^\perp \) with \( \| r^\perp \|_{L^2} = 1 \) and \( r^\perp \) is orthogonal (with respect to the norm \( \| \cdot \|_{L^2} \)) to the image of \( \mathcal{S}_R \). But this would imply \( \| r^\perp \|_{R}^2 = \langle r^\perp, S_R r^\perp \rangle_{L^2} = 0 \) and therefore \( r^\perp = 0 \). Hence, \( \mathcal{S}_R \) is surjective. The existence of \( r^\perp \) is obvious or may be established as some special case of an application of a theorem due to Hahn and Banach. (That theorem utilizes the axiom of choice.)

C.3 Proof of lemma 4.2

The difference between a semi–norm and a norm is that in the latter we have the property \( \|r\| = 0 \) implies \( r = 0 \), for every \( r \) in the appropriate vector space where the norm is defined on. Assume \( \| \cdot \|_* \) restricted to \( \mathcal{F}_{\text{add}} \) is a norm. (Say: \( \tilde{r}_{\text{add}} \) is unique.) Now, choose \( R > 0 \) and \( r \in \mathcal{F}_{\text{full}} \) with \( \|r\|_R = 0 \). \( r \) must be in \( \mathcal{F}_{\text{add}} \), because otherwise \( \| (I - \tilde{P}_{\text{add}}) r \|_{L^2} \) would be non–zero. In this case, we have \( \| r \|_R = \| r \|_* \) and therefore, by assumption, \( r = 0 \). Hence \( \| \cdot \|_R \) restricted to \( \mathcal{F}_{\text{full}} \) is a norm and \( \tilde{r}_R \) is unique.

Conversely, assume \( \| \cdot \|_R \) restricted to \( \mathcal{F}_{\text{full}} \) is a norm. Because the norms \( \| r \|_R \) and \( \| r \|_* \) are identical for \( r \in \mathcal{F}_{\text{add}} \subset \mathcal{F}_{\text{full}} \), \( \| \cdot \|_* \) restricted to \( \mathcal{F}_{\text{add}} \) is a norm.

Choose \( r \in \mathcal{F}_{\text{full}} \) with \( \| r \|_{R} \leq 1 \) and decompose it into \( r = r_{\text{add}} + r_{\perp} \) with \( r_{\text{add}} = \tilde{P}_{\text{add}} r \in \mathcal{F}_{\text{add}} \). By definition of the norm, \( \| r \|_{R} \leq 1 \) implies \( \| r_{\perp} \|_{L^2}^2 \leq R^{-1} \).

Note that \( r_{\perp} - \Pi_s r_{\perp} \) is orthogonal (with respect to \( \| \cdot \|_* \)) to \( \mathcal{F}_{\text{add}} \). And therefore
\[ \| r_{\text{add}} + r_{\perp} \|_*^2 = \| r_{\text{add}} + \Pi_s r_{\perp} \|_*^2 + \| r_{\perp} - \Pi_s r_{\perp} \|_*^2 \]
Now, we show, the existence of a lower bound for \( \| r \|_{R}^2 \)
\[ 1 \geq \| r \|_{R}^2 = \| r_{\text{add}} + \Pi_s r_{\perp} \|_*^2 + \| r_{\perp} - \Pi_s r_{\perp} \|_*^2 + R \| r_{\perp} \|_{L^2}^2 \]
\[ \geq \| r_{\text{add}} + \Pi_s r_{\perp} \|_*^2 + R \| r_{\perp} \|_{L^2}^2 \]
\[ \geq C_7^2 (\| r_{\text{add}} \|_{L^2} + \| \Pi_s r_{\perp} \|_{L^2})^2 + R \| r_{\perp} \|_{L^2}^2 \]
\[ \geq C_7^2 (\| r_{\text{add}} \|_{L^2}^2 - \Pi_s r_{\perp} \|_{L^2}^2 + R \| r_{\perp} \|_{L^2}^2 \]
\[ \geq \{C_1 \min(0, \| r_{\text{add}} \|_{L^2}^2 - C_7 R^{-1/2})\}^2 + R \| r_{\perp} \|_{L^2}^2 \]
Hence: \( \| r \|_{R} \leq 1 \) implies \( \| r_{\text{add}} \|_{L^2} \) and \( \| r_{\perp} \|_{L^2} \) are bounded. Hence there exists a \( C_4 > 0 \) with
\[ \| r \|_{R} \geq C_4 \| r \|_{L^2} \]

78
C.4 Proof of theorem 4.3

We assume — as in Lemma 4.2 — that there exist positive constants $C_1$, $C_2$, and $C_3$ such that

$$
\|r\|_\infty \geq C_1 \|r\|_{L^2} \quad \forall r \in F_{\text{add}}
$$

$$
\|r\|_\infty \leq C_2 \|r\|_{L^2} \quad \forall r \in F_{\text{full}}
$$

$$
\|\Pi_r r\| \leq C_3 \|r\|_{L^2} \quad \forall r \in F_{\text{full}}
$$

Consider

$$
\|\hat{r}_R - \hat{r}_{\text{add}}\|_{L^2} \leq \|\hat{r}_R - \hat{P}_{\text{add}} \hat{r}_R\|_{L^2} + \|\hat{P}_{\text{add}} \hat{r}_R - \hat{r}_{\text{add}}\|_{L^2}
$$

$$
= \| (I - \hat{P}_{\text{add}}) \hat{r}_R\|_{L^2} + \| (\hat{P}_{\text{add}} - \Pi_x) \hat{r}_R\|_{L^2}
$$

$$
= \| (I - \hat{P}_{\text{add}}) \hat{r}_R\|_{L^2} + \|\Pi_x (\hat{P}_{\text{add}} - I) \hat{r}_R\|_{L^2}
$$

$$
\leq (1 + C_3) \| (I - \hat{P}_{\text{add}}) \hat{r}_R\|_{L^2}. \tag{*1}
$$

Note that we have an $R$–independent bound for $\|r_y - \hat{r}_R\|_R$:

$$
\|r_y - \hat{r}_R\|_R \leq \|r_y - \hat{r}_{\text{add}}\|_R = \|r_y - \hat{r}_{\text{add}}\|_\infty. \tag{*}
$$

By definition $\|r_y - \hat{r}_R\|_R^2 = \|r_y - \hat{r}_{\text{add}}\|_R^2 + R \| (I - \hat{P}_{\text{add}}) \hat{r}_R\|_{L^2}$. Hence

$$
R \| (I - \hat{P}_{\text{add}}) \hat{r}_R\|_{L^2}^2 = \|r_y - \hat{r}_R\|_R^2 - \|r_y - \hat{r}_{\text{add}}\|_R^2
$$

$$
\leq \|r_y - \hat{r}_R\|_R^2 - \|r_y - \hat{r}_{\text{add}}\|_R^2
$$

$$
= \|\hat{r}_{\text{add}} - \hat{r}_R\|_R^2 - 2r_Y - (\hat{r}_{\text{add}} + \hat{r}_R)
$$

$$
\leq \|\hat{r}_{\text{add}} - \hat{r}_R\|_R \|2r_Y - (\hat{r}_{\text{add}} + \hat{r}_R)\|_R
$$

$$
\leq C_2 \|\hat{r}_{\text{add}} - \hat{r}_R\|_R \|2r_Y - (\hat{r}_{\text{add}} + \hat{r}_R)\|_R. \tag{*2}
$$

Next, we examine $\|2r_Y - (\hat{r}_{\text{add}} + \hat{r}_R)\|_R^2$. Recall that

$$
\|2r_Y - (\hat{r}_{\text{add}} + \hat{r}_R)\|_R^2 \leq \|2r_Y - (\hat{r}_{\text{add}} + \hat{r}_R)\|_R^2 \tag{*3}
$$

and we use the abbreviations $a = 2(r_Y - \hat{r}_R)$ and $b = \hat{r}_R - \hat{r}_{\text{add}}$. Hence, $a + b = 2r_Y - (\hat{r}_{\text{add}} + \hat{r}_R)$. Because $b \in F_{\text{full}}$, and $\hat{r}_R$ minimizes the $\|r_y - \hat{r}\|_R$ for all $\hat{r} \in F_{\text{full}}$, $a$ and $b$ are orthogonal with respect to the norm $\|\cdot\|_R$:

$$
\langle a, b \rangle_R = 2 \langle r_Y - \hat{r}_R, \hat{r}_R - \hat{r}_{\text{add}} \rangle_R = 0.
$$

We write $c = a + 2b = 2(r_Y - \hat{r}_{\text{add}})$, and we have

$$
\|a + b\|_R^2 = \|c - b\|_R^2 = \|c\|_R^2 - 2 \langle c, b \rangle_R + \|b\|_R^2
$$

$$
= \|c\|_R^2 - 2 \langle a + b, b \rangle_R + \|b\|_R^2
$$

$$
= \|c\|_R^2 - 3 \|b\|_R^2 \leq \|c\|_R^2.
$$

Hence (because of (*))

$$
\|2r_Y - (\hat{r}_{\text{add}} + \hat{r}_R)\|_R^2 \leq \|2(r_Y - \hat{r}_{\text{add}})\|_R^2 = \|2(r_Y - \hat{r}_{\text{add}})\|_R^2. \tag{*4}
$$

Using (*1)–(*4), we obtain

$$
\|\hat{r}_R - \hat{r}_{\text{add}}\|_{L^2} \leq \frac{(1 + C_3)^2}{R} C_2 \|2(r_Y - \hat{r}_{\text{add}})\|_R. \tag{79}
$$
D Remarks and proofs for section 4.2

D.1 Redundancy in Z

In section 4.2.1, we defined a matrix $Z_k$ which had not full rank. The question arises, if it is worth trying to find a smaller matrix $Z^*_k$ with full rank. As the matrix $Z_0k$ has only $m$ nonzero entries, multiplication by $Z_0k$ (and by $Z_k$) is very efficient. For faster multiplication by $Z^*_k$ we use the decomposition

$$Z^*_k = \begin{bmatrix} 0 & 1 & \cdot & \cdot & \cdot \\ \vdots \\ 0 & 0 & 1 \end{bmatrix} QZ_0k$$

where $Q$ is some orthogonal matrix with $Q(1, \ldots, 1)^T = \sqrt{m_k}(1, 0, \ldots, 0)^T$. I assume that multiplication of a vector by $[0 I]Q$ is $O(m^2_k)$. (Calculating $Z_kv$ from $Z_0kv$ is $O(m_kv)$.)

Now, we demonstrate that $Z^*_k$ leads to a slower implementation than $Z_k$. On the one hand, inverting $I -ZA(R)Z^\top$ is simpler if $Z_k$ were replaced by $Z^*_k$. The reduced burden due to smaller dimension is $O(\sum_k m_k^2 d) = O(m^2_k d^3)$ for the Cholesky decomposition, and $O(d \sum_k m_k)$ for solving (which is inferior to the difference $O(m^2_k)$ for multiplication by $Z_k$ compared to $Z^*_k$). On the other hand, obtaining the matrix $I -ZA(R)Z^\top$ is more complicated: Intuitively, it will at least compensate the above mentioned gain. Otherwise, the calculation of MISE (section 5.4) or AIC (section 5.5.2) uses $n + 1$ times a multiplication by $Z$ while the Cholesky decomposition is done only once for every combination of parameters.

Furthermore $Z_k$ is preferred over $Z^*_k$, because implementation is simpler.

D.2 Fixed design in a neighborhood of $[0, 1]^d$

Formally, we have a sequence $(n^*_s, m^*_s)_{s \geq 1}$ with $n^*_s$ tending to infinity for increasing $s$, and $n^*_s$ is a multiple of $m^*_s$. The output grid is chosen as

$$\left\{ 0, \frac{1}{m^*_1}, \ldots, \frac{m^*_s - 1}{m^*_1} \right\}^d.$$

For any upper bound $\tilde{h}$ of the bandwidths, define the design as

$$\left\{ -\left\lfloor \tilde{h}n^*_1 \right\rfloor, \ldots, 0, \frac{1}{n^*_1}, \ldots, \frac{n^*_s - 1}{n^*_1}, \ldots, \frac{n^*_s - 1 + \left\lfloor \tilde{h}n^*_1 \right\rfloor}{n^*_1} \right\}^d.$$

The design density is $f(x) = (1 + 2\tilde{h})^{-d}$, $n = n^*_1 + 2\left\lfloor \tilde{h}n^*_1 \right\rfloor$

For every $h < \tilde{h}$, $S(j)$ is independent of $j$.

D.3 Lower bound for $K^{*2}$

Because $\text{supp}(K) = [-1, 1]$ and

$$1 = \left( \int_{-1}^{1} K(u) du \right)^2 \leq \int_{-1}^{1} K(u)^2 du \cdot \int_{-1}^{1} du$$

we get

$$\int_{-1}^{1} K(u)^2 du \geq \frac{1}{2}.$$ 

For non–overlapping smoothing windows, we have $m_1h \leq \frac{1}{2}$ and therefore

$$\frac{K^{*2}(0)}{m_1h} \geq 1.$$
D.4 Diagonal elements of $Z_{0\ell}^T \Sigma_{ll} Z_{0\ell}$

The diagonal elements of $Z_{0\ell}^T \Sigma_{ll} Z_{0\ell}$ are

$$[Z_{0\ell}^T \Sigma_{ll} Z_{0\ell}]_{j\ell, j\ell} = \frac{\sigma^2}{f_n h^d} \frac{m_{\ell}}{m} \sum_{j, k \in J_{\ell}^*} \prod_{\ell=1}^d K^{*2} \left( \frac{t_{j\ell} - t_{k\ell}}{h} \right) \{1 + o(1)\}.$$ 

Rewrite the sum and approximate it by an integral

$$\frac{\sigma^2}{f_n h^d} \frac{m_{\ell}}{m} \sum_{j, k \in J_{\ell}^*} \prod_{\ell=1}^d K^{*2} \left( \frac{t_{j\ell} - t_{k\ell}}{h} \right)$$

$$= \frac{\sigma^2}{f_n h^d} K^{*2}(0) \prod_{k=1, \ldots, d, k \neq \ell} K^{*2} \left( \frac{u_k - u_j}{m_k h} \right)$$

$$= \frac{\sigma^2}{f_n h^d} K^{*2}(0) \prod_{k=1, \ldots, d, k \neq \ell} \sum_{j=1}^{m_k-1} \frac{m_k - |j|}{m_k} K^{*2} \left( \frac{u_j}{m_k h} \right)$$

$$\rightarrow \frac{\sigma^2}{f_n h^d} K^{*2}(0) \prod_{k=1, \ldots, d, k \neq \ell} m_k \int_{-1}^1 (1 - |v|) K^{*2} \left( \frac{v}{h} \right) dv_k$$

$$= \frac{\sigma^2}{f_n h^d} m \frac{m_{\ell}}{m} K^{*2}(0) \left( \frac{1}{h} \int_{-1}^1 (1 - |v|) K^{*2} \left( \frac{v}{h} \right) dv \right)^{d-1}.$$ 

For the integral, we get the following bound (because $K \geq 0$)

$$\frac{1}{h} \int_{-1}^1 (1 - |v|) K^{*2} \left( \frac{v}{h} \right) dv = \int_{-h}^{h-1} (1 - |u|) K^{*2}(u) du \leq \int_{-\infty}^{\infty} K^{*2}(u) du = 1.$$ 

This bound is exact for $h \to 0$.

The matrix $Z_{0\ell}$ has dimensions $m_{\ell} \times m$. This compensates the factor $m/m_{\ell}$.

$$\frac{1}{m} \text{tr}(Z_{0\ell}^T \Sigma_{ll} Z_{0\ell}) = \frac{1}{m} \sum_{j_{\ell}=1}^{m_{\ell}} [Z_{0\ell}^T \Sigma_{ll} Z_{0\ell}]_{j_{\ell}j_{\ell}}$$
E Proofs for section 4.3

E.1 Section 4.3.1

Case \( \text{nh}^d \) large and \( R \) arbitrary

In this case, we would expect that \( \tilde{P}^{(R)}_A \) is \( O(I) \). Hence, the bias terms are as desired: \( \mathbb{E}(\hat{\beta}^{(j)}_I) \) and \( \mathbb{E}(\hat{\beta}^{(j)}_{\text{add},R}) \) both have \( O(h^2) \) terms.

The variance of \( \hat{\beta}^{(j)}_{\text{add},R} \) is reduced to \( O(\frac{1}{\text{nh}}) \).

Case \( R \) large — bias

For \( R \) large, the structure of

\[
\tilde{Z}^T \tilde{Z}\mathrm{col}_j \left( (R^{-1} S^{(j)} + I)^{-1} T^{(j)} \right)
\]

reduces the bias term \( O(h^2 \max(1, \frac{1}{\text{nh}^2})) \) to \( O(h^2) \). This holds because \( (R^{-1} S^{(j)} + I)^{-1} \) is bounded by \( I \), and \( \tilde{Z}^T \tilde{Z}\mathrm{col}_j(T^{(j)}) \) is related to univariate estimation. Furthermore

\[
\left( \tilde{Z}\mathrm{diag}_j(S^{(j)}(R^{-1} S^{(j)} + I)^{-1} \tilde{Z}) \right)^{-1} = O \left( 1 + R^{-1} \max(1, \frac{1}{\text{nh}^2}) \right).
\]

Using formula (34), we get \( O(h^2 \max(1, \frac{1}{\text{Rnh}^2}) + h^2 \max(1, \frac{1}{\text{Rnh}^2}, 1)) = O(h^2 + \frac{h^2}{\text{Rnh}^2} + \frac{h^2}{\text{R}}) \) for \( \mathbb{E}(\hat{\beta}^{(j)}_R) \). Hence, the \( O(h^2 \max(1, \frac{1}{\text{nh}^2})) \) term in \( T^{(j)} \) is transformed into \( O(h^2 \max(1, \frac{1}{\text{Rnh}^2})) \) in \( \mathbb{E}(\hat{\beta}^{(j)}_{\text{add},R}) \).

Case \( R \) large — variance

For \( R \) large, (in fact: ‘bounded away from zero’ is sufficient)

\[
\text{var}(\hat{\beta}^{(j)}_{\text{add},R}) = \max(1, \frac{1}{\text{Rnh}^2}) \frac{1}{\text{nh}}
\]

for \( \ell = 0, \ldots, d \).

E.2 Section 4.3.3

Definition of \( S^{(j)}_A \) and \( T^{(j)}_A \)

We use the following abbreviations

\[
S^{(j)}_A = \begin{bmatrix} s_0 & S^\alpha_1' \\ S^\alpha_1 & S^\alpha_2 \end{bmatrix} \quad T^{(j)} = \begin{bmatrix} T_0 \\ T_0' \end{bmatrix},
\]

where

\[
s_0 = \frac{1}{n} \sum_{i=1}^n K^{(j)}_H(X_i) \quad T_0 = \frac{1}{n} \sum_{i=1}^n K^{(j)}_H(X_i)Y_i
\]

\[
S^\alpha_1 = \frac{1}{n} \sum_{i=1}^n K^{(j)}_H(X_i) \begin{bmatrix} X_{i,1} - t_{j,1} \cr \vdots \cr X_{i,d} - t_{j,d} \end{bmatrix} \quad S^\alpha_2 = \frac{1}{n} \sum_{i=1}^n K^{(j)}_H(X_i) \begin{bmatrix} X_{i,1} - t_{j,1} & X_{i,1} - t_{j,1} & \cdots & X_{i,1} - t_{j,1} \\ \frac{1}{\alpha_1 h_1} & \frac{1}{\alpha_1 h_1} & \cdots & \frac{1}{\alpha_1 h_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\alpha_d h_d} & \frac{1}{\alpha_d h_d} & \cdots & \frac{1}{\alpha_d h_d} \end{bmatrix}
\]

82
\[ T_{a^0} = \frac{1}{n} \sum_{i=1}^{n} K_R^{(j)}(X_i) \begin{bmatrix} \frac{X_{i,1} - t_{i,1}}{\alpha_1 h_1} \\ \vdots \\ \frac{X_{i,d} - t_{i,d}}{\alpha_d h_d} \end{bmatrix} Y_i. \]

If \( \alpha_1 = \ldots = \alpha_d \), we have

\[
s_0 - s_1^{\alpha^T}(S_2^{\alpha^T} + RI)^{-1}s_1 = s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1,
\]

and

\[
s_1^{\alpha^T}(S_2^{\alpha^T} + RI)^{-1}T_{a^0}^{(j)} = s_1(S_2 + \alpha^2 RI)^{-1}T_{a^0}^{(j)}. \tag{38} \]

**Influence of \( \alpha \) on the bias**

The goal is to calculate \((1, 0, \ldots, 0)(S^{(j)}\alpha + RI)^{-1}T^{(j)}\alpha\). Consider first

\[
(1, 0, \ldots, 0)(S^{(j)}\alpha + RI)^{-1} = \{R + s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1\}^{-1}[1, -\alpha s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}]. \tag{39}
\]

The leading term of \( T^{(j)}\alpha \) is \( S^{(j)}\alpha \) \( \text{col}(r(t_j), \alpha \nabla_r(t_j)) \) and we calculate

\[
(1, 0, \ldots, 0)(S^{(j)}\alpha + RI)^{-1} = \frac{(s_0 r + s_1^{\alpha^T} \nabla_r) - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}(s_1 r - S_2 \nabla_r)}{R + s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1}
\]

\[
= \frac{(s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1)r}{R + s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1} + \frac{s_1^{\alpha^T}(1 - (S_2 + \alpha^2 RI)^{-1}s_1) \nabla_r}{R + s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1}
\]

\[
= \frac{(s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1)r}{R + s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1} + \frac{s_1^{\alpha^T}(\alpha^2 R(S_2 + \alpha^2 RI)^{-1}) \nabla_r}{R + s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1}
\]

Hence, choosing \( R \to 0 \) and \( \alpha^2 R \to \infty \), the slope terms \( \nabla_r \) do not vanish. Using formula (35) on page 42, we conclude that the bias of the first component of \( \hat{\beta}_R \) contains terms depending on the difference of \( \nabla_r \) and the additive part of \( \nabla_r \). Hence the bias has the same disadvantages as the Nadaraya–Watson estimator. If \( r \) is additive, these terms cancel each other. Furthermore, \( r \) additive implies \( R \to \infty \) and hence, these terms vanish.

**Influence of \( \alpha \) on the variance**

Finding the optimal ridge parameter is a variance–bias compromise and the bias is generally larger for the additive part. Accordingly, the variance of \( \hat{\beta}_R^{(j)} \) is typically dominated by the local part \( T^{(j)} \) in the decomposition (34) on page 41. Hence, we calculate the variance of

\[
(1, 0, \ldots, 0)(S^{(j)}\alpha + RI)^{-1} \begin{bmatrix} T_{a^0}^{(j)} \\ \alpha^{-1}T^{(j)}_{a^0} \end{bmatrix}.
\]

The conditional variances of \( T_{a^0}^{(j)} \) and \( T^{(j)}_{a^0} \) are of the same order. Using formula (39), we want to ensure that the variance of

\[
s_1^{\alpha^T}(S_2 + \alpha^2 R)^{-1}T_{a^0}^{(j)}
\]

is not larger than the variance of \( T_{a^0}^{(j)} \). (The term \((R + s_0 - s_1^{\alpha^T}(S_2 + \alpha^2 RI)^{-1}s_1)^{-1}\) is common.)

Seifert and Gasser (2000) investigated a local linear estimator with a penalty on the slope. As a rule of thumb, they propose to use a penalty which is proportional to \( \|s_1\| \). Thus, \( \alpha^2 R \) should not be smaller than this bound.

Recall that \( s_1 \) is \( O_p(f'h_1) \) and \( S_2 \) is \( O_p(I) \).
F S–PLUS AND R PROGRAMS

F S–plus and R programs

In this section we describe the routines for evaluating the local linear estimator with penalty on non–additivity. The main function is in section F.5. Before we specify the construction of local bandwidths and the normalizations. Sections F.6 and later contain functions for calculating ISE, AIC, and MISE.

F.1 Get maximal dimension

When using high dimensional data, some C functions need to be compiled again after the constant MAX_DIM was increased. The function call NonAddPenaltyGetDim(5) returns the current built–in limit and prints some error message if 5 exceeds this value.

```r
NonAddPenaltyGetDim <- function(dimension=0)
{
  if (!is.R()){
    return(.C("NonAddPenaltyGetDim",MAXDIM=as.integer(dimension),
          COPY=T,CLASSES="integer")$MAXDIM)
  } else {
    return(.C("NonAddPenaltyGetDim",MAXDIM=as.integer(dimension),
              NAOK=FALSE,DUP=FALSE,PACKAGE="NonAddPenalty")$MAXDIM)
  }
}
```

If you are using S–PLUS instead of R, define

```r
is.R <- function(){ return(F) }
```

F.2 Output Grid

The variable outgrid is a list describing the output grid:

```r
outgrid <- list(c(t^1_1, . . . , t^1_{m^1}) , . . . , c(t^d_1, . . . , t^d_{m^d}) )
```

F.3 Local bandwidths

In order to reduce the complexity of bandwidth choice, we assume that the bandwidths depend only on d parameters (h_1, . . . , h_d). However, we better not use the same bandwidth in the interior and at the boundary. See also figure 14 in section 5.3.2.

In the univariate case, the following function calculates local bandwidths h_k(t^k_1), . . . , h_k(t^k_{m_k}) given bandwidth=h_k, outgrid=c(t^k_1, . . . , t^k_{m_k}) and boundary=NULL or c(0,1) for the interval [0,1]. For Gaussian kernels (kernel=2) we assume the smoothing window is [−2,2].

```r
BandCalc1 <- function(bandwidth,outgrid,boundary=NULL,kernel=0)
{
  if (length(bandwidth)!==1){
    stop("BandCalc1 : bandwidth has wrong length \n")
  }
  if (mode(outgrid)=="list"){
    stop("BandCalc1 : outgrid should be a vector \n")
  }
  if (is.null(boundary)) { # No boundary adjustment
    return(rep(bandwidth,times=length(outgrid)))
  }
}
```
if (length(boundary) != 2)
  stop("BandCalc1: boundary should have length 2")
}
if (any(outgrid < boundary[1]) || any(outgrid > boundary[2])){
  print(range(output))
  print(boundary)
  stop("BandCalc1: output grid points outside boundary")
}
win.len <- -1 # smoothing window is [-win.len, win.len]
if (kernel == 2) { # Gauss kernel
  win.len <- 2
}
# Increase the bandwidth by the amount the smoothing windows overlap
# the boundary. No check if diff(boundary) > 2*win.len*bandwidth.
# This check is omitted, because otherwise the local bandwidths
# would not depend smoothly on the unmodified bandwidth.
leftoverlap <- (boundary[1] + win.len * bandwidth) - outgrid
leftoverlap[leftoverlap < 0] <- 0
rightoverlap[rightoverlap < 0] <- 0
thebandwidth <- rep(bandwidth, times = length(outgrid))
return(thebandwidth + (leftoverlap + rightoverlap) / win.len)

Example:

> BandCalc1(bandwidth = 0.2, outgrid = seq(0, 1, by = 0.1), boundary = c(0, 1))
[1] 0.4 0.3 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.3 0.4

Univariate boundary adjustment is done by BandCalc1. To obtain a list of local bandwidths
we use the function BandCalc which simply calls d times BandCalc1:

BandCalc <- function(bandwidth, outgrid, boundary = NULL, kernel = 0)
{
  if (mode(outgrid) != "list"){
    stop("BandCalc: outgrid has to be a list")
  }
  dimensions <- length(outgrid)
  if (mode(bandwidth) == "list"){
    stop("BandCalc: bandwidth should be a vector")
  }
  if (length(bandwidth) == 1){
    bandwidth <- rep(bandwidth, times = dimensions)
  } else if (dimensions != length(bandwidth)){
    stop("BandCalc: sizes of outgrid and bandwidth", "do not match")
  }
  if (is.null(boundary)){
    warning("BandCalc: No boundary correction performed")
  }
  dimbound <- 1

85
Define indexes \( j \) and \( j_1, \ldots, j_d \) with \( t_j = (t_{j1}, \ldots, t_{jd})^\top \). We use a product Epanechnikov kernel and store the normalizations in a list \( \text{normalization} \) which has the same size and structure as \( \text{outgrid} \) or \( \text{bandwidths} \). For the local linear estimator, weights are defined by

\[
K(X_i - t_j; h) = \prod_{k=1}^{d} \text{normalization}[[k]][j_k] \max \left( 0, 1 - \left( \frac{X_{i,k} - t_{j,k}}{h_k(t_{j,k})} \right)^2 \right).
\]

This corresponds to a normalization of type 0 in section G.5.3 below.

Without boundary adjustments (boundary=NULL) we define

\[
\text{normalization}[[k]][j_k] = \frac{3}{4h_k(t_{j,k})}.
\]

If the output interval is \([a,b]\) (boundary=c(a,b)), we define

\[
\text{normalization}[[k]][j_k] = \left( h_k(t_{j,k}) \int_{lo}^{hi} 1 - x^2 dx \right)^{-1}
\]

\[
\begin{align*}
lo &= \min \left( 1, \frac{b - t_{j,k}}{h_k(t_{j,k})} \right) \\
hi &= \min \left( 1, \frac{t_{j,k} - a}{h_k(t_{j,k})} \right)
\end{align*}
\]

\[
\text{NormCalcPE}\leftarrow \text{function}(\text{bandwidths} . \text{list} , \text{outgrid} , \text{boundary} = \text{NULL}){
\text{if} (\text{mode}(\text{bandwidths} . \text{list}) != "\text{list}"
\text{stop}("\text{NormCalcPE: \text{bandwidths} . \text{list} \ has \ to \ be \ a \ \text{list} \ \text{\n"})
\text{if} (\text{mode}(\text{outgrid}) != "\text{list}"
\text{stop}("\text{NormCalcPE: \text{outgrid} \ has \ to \ be \ a \ \text{list} \ \text{\n"})
\text{if} (\text{length}(\text{bandwidths} . \text{list}) != \text{length}(\text{outgrid})
\text{stop}("\text{NormCalcPE: \text{bandwidths} . \text{list} \ and \ \text{outgrid} \ should \ have \),
"the same length."

normalizations<−list(NULL)
if ( is.null(boundary)){
  warning("NormCalcPE: No boundary correction performed\n")
  for ( i in 1:length(bandwidths.list)){
    normalizations<−c(normalizations ,
                   list(0.75/bandwidths.list[[i]]))
  }
  return(normalizations[−1])
}
dimbound<−F
if ( is.null(dim(boundary))){
  thisboundary<−boundary
} else {
  dimbound<−T
}

normalizations<−list(NULL)
for ( i in 1:length(bandwidths.list)){
  if ( dimbound){
    thisboundary<−boundary[d ,]
  }
  hi<−bandwidths.list[[i]]
  some<−hi>thisboundary[2]−outgrid[[i]]
  hi[some]<−(thisboundary[2]−outgrid[[i]])[some]
  hi<−hi/bandwidths.list[[i]]
  lo<−bandwidths.list[[i]]
  some<−lo>outgrid[[i]]−thisboundary[1]
  lo[some]<−(outgrid[[i]]−thisboundary[1])[some]
  lo<−lo/bandwidths.list[[i]]
  norms<−1/(lo−lo ˆ3/3+hi−hi ˆ3/3)∗bandwidths.list[[i]]
  normalizations<−c(normalizations , list(norms))
}
return(normalizations[−1])

Example:

> outgrid<−list(seq(0,1,by=0.1))
> ( bw<−BandCalc(0.2,outgrid,c(0,1)) )[[1]]
[1] 0.4 0.3 0.2 0.2 0.2 0.2 0.2 0.2 0.3 0.4
> 3/4/NormCalcPE(bw, outgrid ,NULL) )[[1]]
[1] 0.4 0.3 0.2 0.2 0.2 0.2 0.2 0.2 0.3 0.4

Warning messages:
NormCalcPE: No boundary correction performed
in: NormCalcPE(bw, outgrid , NULL)
> 3/4/NormCalcPE(bw, outgrid ,c(0,1)) )[[1]], 3
[1] 0.2 0.222 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.222 0.2

For Gaussian kernel weights, use the function NormCalcG.
F.5 Local linear estimator with non–additivity penalty

The function NonAddPenaltyLlR calculates the penalized estimator as defined in formula (16) on page \(26\). See also section \(G.7\) for the documentation of the C code. The syntax is:

\[
\text{NonAddPenaltyLlR}(x, y, \text{outgrid}, \text{bandwidths}, \text{boundary}, \text{kernel}, R, \text{invR}=NULL)
\]

**Arguments**

- **x**
  - Matrix with dimensions \((n,d)\)
  - Observations. \(x[i,k]=X_{i,k}\).

- **y**
  - Vector of length \(n\)
  - \(y[i]=Y_i\).

- **outgrid**
  - List of \(d\) vectors
  - \(\text{outgrid} \leftarrow \text{list}(c(t_1^1, \ldots, t_{m_1}^1), \ldots, c(t_d^1, \ldots, t_{m_d}^d))\)

- **bandwidths**
  - Vector of length \(d\)
  - To be processed by BandCalc (section \(F.3\)). Reparameterization via division by the (global) bandwidth, independently of the output points.

- **boundary**
  - See section \(F.3\).

- **kernel**
  - Specify the kernel weight function
    - 0 or "PE": product Epanechnikov (default)
    - 1 or "SE": spherical symmetric Epanechnikov. *Not fully supported.*
    - 2 or "G": Gauss

- **R**
  - Scalar, default=NULL
  - Penalty parameter \(R\). Conflicts with parameter \(\text{invR}\).

- **invR**
  - Scalar, default=NULL
  - Inverse penalty parameter \(R^{-1}\). (Suitable for large values of \(R\).)

**Values**

The function returns a list containing the arguments of the C function NonAddPenaltyEstimator defined in section \(G.7\).

\[
dims = (n, d, m_1, \ldots, m_d)
\]

- **x**
- **y**
- **matS**
  - Dimension= \((d+1, d+2, m_1, \ldots, m_d)\)
- **matT**
  - Dimension= \((d+1, m_1, \ldots, m_d)\)
- **outgrid.concat**
  - Length= \(m_1 + \ldots + m_d\)
- **bandwidths.concat**
  - Length= \(m_1 + \ldots + m_d\)
- **addtmp**
  - Dimension= \((m_1 + \ldots + m_d, 2)\)
- **Options**
  - Length=3
  - Kernel type, type of normalization, and type of reparameterization.
- **normalizations.concat**
  - Size depends on Options.
- **Param**
- **mat**
  - **R xor invR**
    - Scalar
  - Penalty parameter
- **est**
  - dimension= \((m_1, \ldots, m_d)\)
  - Estimated values.
Local linear estimator with non-additivity penalty

Source code

```r
NonAddPenaltyLlR <- function(x, y, outgrid, bandwidths, boundary=NULL, 
R=NULL, invR=NULL, kernel=0)
{
  if (!xor(is.null(R), is.null(invR)))
    if (is.null(R))
      stop("NonAddPenaltyLlR: Argument R or invR required.
    
  } else {
    stop("NonAddPenaltyLlR: do not use the options R, "
    "and invR together.

  }
}

if (mode(outgrid)!="list")
  stop("NonAddPenaltyLlR: ",deparse(substitute(outgrid)),
    "has to be a list."
}

if (NonAddPenaltyGetDim(length(outgrid))<=length(outgrid))
  stop("NonAddPenaltyLlR: Dimension is too large.

if (mode(kernel)=="character")
  if (kernel=="PE")
    kernel<-0
  else if (kernel=="SE")
    kernel<-1
  else if (kernel=="G")
    kernel<-2
  else {
    stop("NonAddPenaltyLlR: kernel should be",
      " 0, 1, 2, PE", 
      " SE", " or G."
    
  }
}

if (mode(bandwidths)!="list")
  #bandwidths are global bandwidths.
  # calculate local bandwidth
  bandwidths.list <- BandCalc(bandwidths, outgrid, boundary, kernel)
  paramtype<-2 
  # reparameterization by global bandwidths
  Param<-1/bandwidths
  if (length(Param)==1)
    Param<-rep(Param, times=length(outgrid))
  }
}

else {
  # bandwidths are local bandwidths
  if (length(bandwidths)!=length(outgrid))
    stop("NonAddPenaltyLlR: lengths of bandwidth and",
      " outgrid do not match."
}

bandwidths.list <- bandwidths
  paramtype<-1 
  # reparameterization by local bandwidths
  Param<-1:length(bandwidths) 

if (kernel==0)
  normtype<-0
  normaliziation.list <- NormCalcPE(bandwidths.list,
```
outgrid, boundary)
}
else if (kernel == 1) {
    warning("NonAddPenaltyLlR: Normalization for kernel="SE", not yet implemented")
    normtype <- 0
    normalization.list <- NormCalcPE(bandwidths.list, outgrid, boundary)
}

## NormCalcSE() is not yet implemented.
## As soon as available uncomment the following two lines.
# normtype <- 1
# normalization.array <- NormCalcSE(bandwidths.list, outgrid, boundary)
}
else if (kernel == 2) {
    normtype <- 0
    normalization.list <- NormCalcG(bandwidths.list, outgrid, boundary)
}
else {
    stop("NonAddPenaltyLlR: kernel should be 0, 1 or 2."
}

# check if data has correct dimension
if (dim(x)[1] != length(y)) {
    stop("NonAddPenaltyLlR: dim(x)[1] \ne length(y)."
}
if (dim(x)[2] != length(outgrid)) {
    stop("NonAddPenaltyLlR: dim(x)[2] \ne length(outgrid)."
}

# concatenate list to vector
outgrid.concat <- NULL
bandwidths.concat <- NULL
if (normtype == 0) {
    normalizations.concat <- NULL
}
dims <- dim(x)
for (d in 1:length(outgrid)) {
    outgrid.concat <- c(outgrid.concat, outgrid[[d]])
    bandwidths.concat <- c(bandwidths.concat, bandwidths.list[[d]])
    if (normtype == 0) {
        normalizations.concat <- c(normalizations.concat, normalization.list[[d]])
    }
dims <- c(dims, length(bandwidths.list[[d]])]
}

# storage for normal equations
matS <- numeric(prod(dimS))
dim(matS) <- dimS
dimT <- dimS[2]
vecT <- numeric(prod(dimT))
dim(vecT) <- dimT

# storage for result
est <- numeric(prod(dimT[2]))
dim(est) <- dimT[2]
# temporary storage
addtmp <- numeric(2*sum(dimT[-1]))
mat <- numeric(length(addtmp)*2)
dim(mat) <- rep(length(addtmp), times=2)
#
if (normtype==1)
  normalizations.concat <- normalization.array
}
Options <- c(kernel, normtype, paramtype)
#
storage.mode(dims) <- "integer"
storage.mode(Options) <- "integer"
storage.mode(x) <- "double"
storage.mode(y) <- "double"
storage.mode(outgrid.concat) <- "double"
storage.mode(bandwidths.concat) <- "double"
storage.mode(normalizations.concat) <- "double"
storage.mode(Param) <- "double"
if (!is.null(R)) {
  storage.mode(R) <- "double"
  if (!is.R()) {
    # Splus
    res <- .C("NonAddPenaltyEstimator",
      dims=dims, x=x, y=y, matS=matS, vecT=vecT,
      outgrid.concat=outgrid.concat,
      bandwidths.concat=bandwidths.concat,
      addtmp=addtmp, Options=Options,
      normalizations.concat=normalizations.concat,
      Param=Param, mat=mat.R-R, est=est,
      COPY=rep(F, times=14),
      CLASSES=c("integer", rep("double", times=7),"integer",
        rep("double",times=5))
  }
  else {
    # R
    res <- .C("NonAddPenaltyEstimator",
      dims=dims, x=x, y=y, matS=matS, vecT=vecT,
      outgrid.concat=outgrid.concat,
      bandwidths.concat=bandwidths.concat,
      addtmp=addtmp, Options=Options,
      normalizations.concat=normalizations.concat,
      Param=Param, mat=mat.R-R, est=est,
      NAOK=FALSE, DUP=FALSE, PACKAGE="NonAddPenalty"
  }
} else {
  # optimized code for large R=1/invR
  storage.mode(invR) <- "double"
  if (!is.R()) {
    # Splus
    res <- .C("NonAddPenaltyEstimator_inv",
      dims=dims, x=x, y=y, matS=matS, vecT=vecT,
      outgrid.concat=outgrid.concat,
      bandwidths.concat=bandwidths.concat,
      addtmp=addtmp, Options=Options,
    }
  }
}
normalizations.concat=normalizations.concat,
Param=Param,  mat=mat, invR=invR, est=est,
COPY=rep(F, times=14),
CLASSES=c("integer", rep("double", times=7),"integer",
 rep("double", times=5))}
} else {  # R
res$C("NonAddPenaltyEstimator_inv",
dims=dims, x=x, y=y, matS=matS, vecT=vecT,
outgrid.concat=outgrid.concat,
bandwidths.concat=bandwidths.concat,
addtmp=addtmp, Options=Options,
normalizations.concat=normalizations.concat,
Param=Param,  mat=mat, invR=invR, est=est,
NAOK=FALSE, DUP=FALSE, PACKAGE="NonAddPenalty"
}
}

dim(res$x)<-dim(x)
dim(res$matS)<-dim(matS)
dim(res$vecT)<-dim(vecT)
dim(res$mat)<-dim(mat)
dim(res$est)<-dim(est)
dim(res$addtmp)<-dim(addtmp)
return(res)

F.6 GridISE

GridISE calculates the integrated squared error (ISE) simultaneously for different penalties and (global) bandwidths. It calls the C function NonAddPenaltyGridISE(), which is described in appendix G.8. The syntax is as follows:

GridISE(x, y, true, outgrid, testR, testB, boundary=NULL, kernel=0,
desc="", printstep=F, testRtrsf=testR/(1+testR),
testBtrsf=log10(testB), timestamp=F)

Arguments

x  Matrix with dimensions \((n,d)\)
Observations. \(x_{i,k}=X_{i,k}\).

y  Vector of length \(n\)
Observations. \(y_i=Y_i\).

testB  Vector of (global) bandwidths
The local bandwidths are calculated by BandCalc(testB[k],outgrid,boundary)
boundary  Parameter to be passed to BandCalc

kernel Specify the kernel weight function
  • 0 or "PE": product Epanechnikov (default)
  • 1 or "SE": spherical symmetric Epanechnikov. Not fully supported.
  • 2 or "G": Gauss

desc Character string, default=""

printstep Boolean, default=T
Displays progress.

testRtrsf Vector of the same length as testR, default: testRtrsf=testR/(1+testR)
This variable is not used here but the plots will depend on it.

testBtrsf Vector of the same length as testB, default: testBtrsf[k]=log10(testB[k])
This variable is not used here but the plots will depend on it.

timestamp Boolean, default=F
Stores the output of date() before and after executing the C function.

Details
The calculation uses the following formula:

\[
\text{for (} j \text{ in 1:length(testB))}{ \\
  \text{for (} k \text{ in 1:length(testR))}{ \\
    \text{est} \leftarrow \text{NonAddPenaltyLIR}(x, y, \text{outgrid}, \text{bandwidths}=\text{testB}[j], \text{boundary}, \text{R}=\text{testR}[k], \text{kernel}) \\
    \text{ISE}[k, j] \leftarrow \text{sum}((\text{est} - \text{true})^2) \\
  } \\
}\]

ISE uses the same global bandwidth for every coordinate. This restriction makes sense for visualization. For other applications, the S-Plus/R interface should be modified. Note that in the above algorithm the outer for loop (variable j) is done in S-Plus, while the inner loop (variable k) is done in C. Hence more general bandwidths may be used.

Actually we calculate a sum of squared errors. However, the discrete analog of an integral would be an average.

To avoid confounding, the result is a list containing ISE and the arguments:

Values a list with

ISE matrix of dimension (\text{length}(\text{testR}), \text{length}(\text{testB}))
  Sum of squared errors.

testB

testR

outgrid

x

y

kernel

ture

testRtrsf

testBtrsf
desc

boundary
date1 if timestamp==T: date() before .C() call.
date2 if timestamp==T: date() after .C() call.
Source code

GridISE <- function(x, y, true, outgrid, testR, testB, boundary=NULL,
                     kernel=0, desc="", printstep=F,
                     testRtransf=testR/(1+testR),
                     testBtransf=log10(testB), timestamp=F)
{
  if (mode(outgrid)!="list")
    stop("GridISE: outgrid has to be a list."
  }
  if (NonAddPenaltyGetDim(length(outgrid))<=length(outgrid))
    stop("GridISE: Dimension is too large."
  }
  if (mode(kernel)=="character")
    if (kernel=="PE")
      kernel<0
    else if (kernel=="SE")
      kernel<-1
    else if (kernel=="G")
      kernel<-2
    else
      stop("GridISE: kernel should be ",
            "0, 1, 2, PE", SE" or "G"."
  }
  # check if data has correct dimension
  if (dim(x)[1]!=length(y))
    stop("GridISE: dim(x)[1] # length(y)."
  }
  if (dim(x)[2]!=length(outgrid))
    stop("GridISE: dim(x)[2] # length(outgrid)."
  }
  # concatenate list to vector
  outgrid.concat<-NULL
  dims<-dim(x)
  for (d in 1:length(outgrid))
    dims<-c(dims, length(outgrid[[d]]))
  outgrid.concat<-c(outgrid.concat, outgrid[[d]])
  # storage for normal equations
  dimS<-c(dims[2]+1, dims[2]+2, dims[-c(1, 2)])
  matS<-numeric(prod(dimS))
  dim(matS)<-dimS
  dimT<-dimS[-2]
  vecT<-numeric(prod(dimT))
  dim(vecT)<-dimT
  # temporary storage
  addvec<-numeric(2*sum(dimT[-1]))
  mat<-numeric(length(addvec)**2)
  dim(mat)<-rep(length(addvec), times=2)
  # storage for result
  est<-numeric(prod(dimT[-1]))
}

94
dim(est) <- dimT[-1]
if (length(est) != length(true)) {
  stop("GridISE: dimension of \texttt{true} does not match the size of the output grid.\n")
}
dim(true) <- dim(est)
dims <- c(dims, length(testR))
storage.mode(dims) <- "integer"
storage.mode(x) <- "double"
storage.mode(true) <- "double"
storage.mode(outgrid.concat) <- "double"
storage.mode(testR) <- "double"
# storage for ISE

for (index in 1: testB.len) {
  # loop over all bandwidths
  if (printstep) {
    if (index %% 10 == 0) {
      cat(":")
    } else {
      cat(".")
    }
  }
  # calculate local bandwidths
  bandwidths.list <- BandCalc(testB[index], outgrid, boundary, kernel)

  Param <- rep(1.0 / testB[index], times=dims[2])
  paramtype <- 2  # reparameterization by global bandwidths
  storage.mode(Param) <- "double"
  if (kernel == 0) {
    normtype <- 0
    normalization.list <- NormCalcPE(bandwidths.list, outgrid, boundary)
  } else if (kernel == 1) {
    warning("GridISE: Normalization for \texttt{SE} not yet implemented")
    normtype <- 0
    normalization.list <- NormCalcPE(bandwidths.list, outgrid, boundary)
  }
}

# NormCalcSE() is not yet implemented.
# As soon as available uncomment the following two lines.
```
# normtype<-1
# normalization.array<-NormCalcSE(bandwidths.list,outgrid,boundary)
} else if (kernel==2) {
  normtype<-0
  normalization.list<-NormCalcG(bandwidths.list,
      outgrid,boundary)
} else {
  stop("GridISE: kernel should be 0, 1 or 2.\n")
}
# concatenate list to vector
bandwidths.concat<-NULL
if (normtype==0){
  normalizations.concat<-NULL
}
for (d in 1:length(outgrid)){
  bandwidths.concat<-c(bandwidths.concat,
      bandwidths.list[[d]])
  if (normtype==0){
    normalizations.concat<-c(normalizations.concat,
        normalizations.list[[d]])
  }
}
if (normtype==1){
  normalizations.concat<-normalization.array
}
# Options and storage.mode
Options<-c(kernel,normtype,paramtype)
storage.mode(Options)<-"integer"
storage.mode(bandwidths.concat)<-"double"
storage.mode(normalizations.concat)<-"double"
if (!is.R()){res<-.C("NonAddPenaltyGridISE",
    dims,x,y,true,outgrid.concat,Options,
    bandwidths.concat,normalizations.concat,
   Param,matS,vecT,mat,addvec.est,
testR,RISE=RISE,
specialsok=T,
    CLASSES=c("integer","rep("double",times=4),"integer",
        rep("double",times=10)),
    COPY=rep(T,times=16),
    ## F may cause some problems with missing values.
)
} else {
  res<-.C("NonAddPenaltyGridISE",
    dims,x,y,true,outgrid.concat,Options,
    bandwidths.concat,normalizations.concat,
    Param,matS,vecT,mat,addvec.est,
testR,RISE=RISE,
    NAOK=TRUE,DUP=TRUE,PACKAGE="NonAddPenalty"
)
}
```
F.7 GridAIC

AIC is explained in section 5.5.2. The corresponding C function is specified in appendix G.9.1.

GridAIC(x, y, outgrid, testR, testB, boundary=NULL, kernel=0, desc="", printstep=F, testRtrsf=testR/(1+testR), testBtrsf=log10(testB), timestamp=F)

Arguments

x Matrix with dimensions \((n,d)\)
Observations. \(x[i,k]=X_{i,k}\).

y Vector of length \(n\)
Observations. \(y[i]=Y_i\).

outgrid List of \(d\) vectors
outgrid <- list(c(t1_1, …, t1_m1), …, c(td_1, …, td_md))

testR Vector of penalties
testB Vector of (global) bandwidths
The local bandwidths are calculated by BandCalc(testB[k],outgrid,boundary)
boundary Parameter to be passed to BandCalc
kernel Specify the kernel weight function
• 0 or ”PE”: product Epanechnikov (default)
• 2 or ”G”: Gauss
desc Character string, default=""
printstep Boolean, default=T
Displays progress.
testRtrsf Vector of the same length as testR, default: testRtrsf=testR/(1+testR)
This variable is not used here but the plots will depend on it.
testBtrsf Vector of the same length as testB, default: testBtrsf[k]=log10(testB[k])
This variable is not used here but the plots will depend on it.
timestamp Boolean, default=F
Stores the output of date() before and after executing the C function.

Values a list with

\(\sigma\) matrix of dimension \((\text{length}(\text{testR}), \text{length}(\text{testB}))\)
Residual variance estimation.
tracemat: matrix of dimension \(\text{length}(\text{testR}), \text{length}(\text{testB})\)

Trace of the hat matrix divided by the number of observations.

testB, testR, outgrid, x, y, kernel, testRtrsf, testBtrsf, desc, boundary, date1, date2

if timestamp==T: date() before \(\text{.C()}\) call.

if timestamp==T: date() after \(\text{.C()}\) call.

**Details**

The AIC criterion is

\[
\log(\text{sigmahat}) + \frac{(1 + \text{tracemat})}{(1 - \text{tracemat} - 2/\text{length}(y))}
\]

The source code is qualitatively similar to GridISE.

**F.7.1 GridAIC.ISE**

In simulations, we want to compare AIC and ISE. The function GridAIC.ISE returns a list containing both the values of GridAIC and GridISE. The function outgrid2design is used for the calculation of the parameter true from the list outgrid and the function \(r\).

**Source code**

\[
\text{GridAIC.ISE} \leftarrow \text{function} (x, y, r, \text{outgrid}, \text{testR}, \text{testB}, \text{boundary}=\text{NULL}, \text{kernel}=0, \text{desc}='', \text{printstep}=\text{T}, \text{testRtrsf}=\text{testR}/(1+\text{testR}), \text{testBtrsf}=\log10(\text{testB}), \text{sigma} = \text{sqrt}(\text{var}(y-f(x))), \text{timestamp}=\text{F})
\]

\{
if (timestamp){
date1 <- date()
}
d <- \text{dim}(x)[2]
dims <- NULL
for (dc in 1:d){
dims <- c(dims, \text{length}(\text{outgrid}[[dc]]))
}
designout <- \text{outgrid2design}(\text{outgrid})
true <- r(designout)
dim(true) <- dims
res1 <- \text{GridAIC}(x, y, \text{outgrid}, \text{testR}, \text{testB}, \text{boundary}, \text{kernel}, \text{desc}, 
\text{printstep}, \text{testRtrsf}, \text{testBtrsf}, \text{timestamp}=\text{F})
res <- \text{GridISE}(x, y, true, \text{outgrid}, \text{testR}, \text{testB}, \text{boundary}, \text{kernel}, \text{desc}, 
\text{printstep}, \text{testRtrsf}, \text{testBtrsf}, \text{timestamp}=\text{F})
res <- c(res, \text{list}(\text{sigma}=sigma, r=r, sigmahat=res1$\text{sigmahat}, 
tracemat=res1$\text{tracemat}))
GridMISE calculates integrates squared bias and variance for different penalties and (global) bandwidths. The C function is described in appendix G.8. The syntax is as follows:

GridMISE(x, rx, true, outgrid, testR, testB, boundary=NULL, kernel=0, desc="", printstep=F, testRtrsft=testR/(1+testR), testBtrsft=log10(testB), timestamp=F)

**Arguments**

- **x**: Matrix with dimensions \((n,d)\)
  Observations. \(x[i,k]=X_{i,k}\).
- **rx**: Vector of length \(n\)
  Observations. \(rx[i]=r(X_i) = E(Y_i)\).
- **true**: Array with dimensions \((m_1,\ldots,m_d)\)
  True regression function evaluated on the output grid.
outgrid List of $d$ vectors

\[
\text{outgrid} \leftarrow \text{list}\left(c(t_{11}^1, \ldots, t_{m_1}^1), \ldots, c(t_{1d}^d, \ldots, t_{md}^d)\right)
\]

testR Vector of penalties

testB Vector of (global) bandwidths

The local bandwidths are calculated by BandCalc(testB[k], outgrid, boundary)

boundary Parameter to be passed to BandCalc

kernel Specify the kernel weight function

* 0 or "PE": product Epanechnikov (default)
* 1 or "SE": spherical symmetric Epanechnikov. Not fully supported.
* 2 or "G": Gauss

desc Character string, default=""

printstep Boolean, default=T

Displays progress.

testRtrsf Vector of the same length as testR, default: testRtrsf=testR/(1+testR)
This variable is not used here but the plots will depend on it.

testBtrsf Vector of the same length as testB, default: testBtrsf[k]=\log_{10}(testB[k])
This variable is not used here but the plots will depend on it.

timestamp Boolean, default=F
Stores the output of date() before and after executing the C function.

**Values** a list with

**var** matrix of dimension (length(testR), length(testB))

Sum of variance (assuming i.i.d standard normal errors).

bias2 matrix of dimension (length(testR), length(testB))

Sum of squared bias.

testB testR outgrid x rx kernel true testRtrsf testBtrsf desc boundary
date1 if timestamp==T: date() before .C() call.
date2 if timestamp==T: date() after .C() call.

The source code is qualitatively similar to GridISE and therefore omitted here.
G  C program code

See section G.7 to see how the pieces are put together. Output variables are underlined.

G.1 local.h

The software is designed as a library that is dynamically linked to some statistical software. As S-Plus and R use different types of integer variables, we define a type INT which is either long int or int.

The size of some temporary variables depends on the dimension $d$. Occasionally, increase the constant MAX_INT and recompile.

A minimal version of local.h is displayed below.

```c
#ifndef LOCAL_H
#define LOCAL_H

typedef int INT; /* integer definition for R */
// typedef long int INT; /* integer definition for S+ */
#define MAX_DIM 10 /* upper limit for dimension */
#define CHECK_INDEX /* activate some debugging code */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#endif
```

Often two functions have a large part in common and differ only in details. Therefore I make use of #define and include files repeatedly. For better understanding preprocess the source files: gcc -E -C -P -nostdinc matrixZ.c.

G.2 Storage order of the observations

Multidimensional arrays are stored in FORTRAN storage order. As an example, the observations $X_1, \ldots, X_n$ are (logically) stored in a matrix (right table). The memory layout is as below:

\[
\begin{array}{cccccc}
\vdots & X_{1,1} \ldots X_{1,n} & X_{1,2} \ldots X_{1,n} & X_{1,3} \ldots X_{1,n} & \ldots & Y_1 \ldots Y_n & \ldots \\
\uparrow & x & X_{2,1} \ldots X_{2,n} & X_{2,2} \ldots X_{2,n} & \vdots & \vdots & \vdots \\
& y & \vdots & \vdots & X_{n,1} \ldots X_{n,n} & X_{n,2} \ldots X_{n,n} & \vdots \\
\end{array}
\]

$x$ and $y$ are pointers to the address of the first elements.

G.3 LlPoint

The normal equations for the local linear estimator at output point $t_j$ are

\[
S^{(j)} \hat{\beta}^{(j)}_{U} = T^{(j)}.
\]

The function NonAddPenaltyLlPoint calculates the matrix $S^{(j)}$ and the vector $T^{(j)}$.

```c
void NonAddPenaltyLlPoint (dim, x, y, S, T, outpoint, bandwidths, Options, normalization)
```

const INT *dim;
const double *x;
const double *y;
double *S;
double *T;
const double *outpoint;
const double *bandwidths;
const INT *Options;
const double *normalization;

Arguments

dim (input) INT vector, length = 2
      dim = (n, d).
x (input) double matrix, dimensions = (n, d)
      Observations \(X_i\), \(i = 1, \ldots, n\).
y (input) double vector, length = n
      Observations \(Y_i\), \(i = 1, \ldots, n\).
S (output) double matrix, dimensions = \((d + 1, d + 2)\)
      Values of \(S^{(j)}\). Only half of the storage is currently used — see section G.4.1 below.
T (output) double vector, length = \(d + 1\)
      Values of \(T^{(j)}\).
outpoint (input) double vector, length = \(d\)
      Coordinates of the output point \(t_j\): outpoint = \((t_{j,1}, \ldots, t_{j,d})\).
bandwidths (input) double vector, length = \(d\)
      Bandwidths: bandwidths = \((h_1, \ldots, h_d)\).
Options (input) INT vector, length = 2
      Options[0] specifies the weight function (section G.3.1 below).
      Options[1] specifies the reparameterization (section G.3.2 below).

G.3.1 Kernel weight function

The product Epanechnikov kernel (Options[0]==0) is defined as

\[
K \left( X_i - t_j; h \right) = \prod_{k=1}^{d} \frac{3}{4h_k} \prod_{k=1}^{d} \max \left( 0, 1 - \left( \frac{X_{i,k} - t_{j,k}}{h_k} \right)^2 \right)
\]

The spherical symmetric Epanechnikov kernel (Options[0]==1) is defined as

\[
K \left( X_i - t_j; h \right) = \frac{d(d + 2)}{4\pi^{d/2}\Gamma(d/2)} \prod_{k=1}^{d} \frac{1}{h_k} \max \left( 0, 1 - \sum_{k=1}^{d} \left( \frac{X_{i,k} - t_{j,k}}{h_k} \right)^2 \right)
\]

The Gaussian kernel (Options[0]==2) is defined as

\[
K \left( X_i - t_j; h \right) = \prod_{k=1}^{d} \frac{1}{\sqrt{2\pi h_k}} \prod_{k=1}^{d} \exp \left( - \frac{(X_{i,k} - t_{j,k})^2}{2h_k^2} \right)
\]
The underbraced terms are not calculated by the C program but are passed to the function by normalization[0]. This allows some boundary adjustment.

### G.3.2 Reparameterization

Let \( k, \ell \in \{1, \ldots, d\} \). The elements of the matrix \( S^{(j)} \) are defined by

\[
\begin{align*}
    s_{0,0} &= \frac{1}{n} \sum_{i=1}^{n} K(X_i - t_j; h) \\
    s_{k,0} &= \frac{1}{n} \sum_{i=1}^{n} K(X_i - t_j; h) (X_{i,k} - t_{j,k}) \\
    s_{k,\ell} &= \frac{1}{n} \sum_{i=1}^{n} K(X_i - t_j; h) (X_{i,k} - t_{j,k})(X_{i,\ell} - t_{j,\ell}).
\end{align*}
\]

For generality, we allow some reparameterizations. The reason is that in section G.7 the same regularization parameter is used for intercept and slope. The simplest generalization to use different parameters is to allow reparameterizations of the slope terms.

- If Options[1] == 0, no reparameterization is performed.
- If Options[1] == 1, \((X_{i,k} - t_{j,k})\) is replaced by \((X_{i,k} - t_{j,k}) / h_k\).
- If Options[1] == 2, \((X_{i,k} - t_{j,k})\) is multiplied by normalization[k].
- If Options[1] == 3, \((X_{i,k} - t_{j,k})\) is replaced by \((X_{i,k} - t_{j,k}) \text{ normalization}[k] / h_k\).

Obviously, the analog is done with \( k \) replaced by \( \ell \).

Note that when using the function NonAddPenaltyLiGrid instead of NonAddPenaltyLiPoint, param[k−1] is used instead of normalization[k]. Also: Options[2] instead of Options[1].

### G.4 Matrix S

#### G.4.1 Memory layout

Later, we want to solve \((S^{(j)} + RI)^{-1}\) without overwriting the memory where \( S^{(j)} \) is stored. As \( S^{(j)} \) is symmetric, we calculate the Cholesky decomposition of \( S + RI \):

\[
S^{(j)} + RI = \begin{bmatrix}
1 & 0 & 0 & 0 & \quad \begin{bmatrix}
    s_{0,0}^* & 0 & 0 & 0 \\
    0 & s_{1,1}^* & 0 & 0 \\
    0 & 0 & s_{2,2}^* & 0 \\
    0 & 0 & 0 & s_{3,3}^*
\end{bmatrix} & \quad \begin{bmatrix}
1 & s_{0,1}^* & s_{0,2}^* & s_{0,3}^* \\
0 & 1 & s_{1,2}^* & s_{1,3}^* \\
0 & 0 & 1 & s_{2,3}^* \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{bmatrix}.
\]

The coefficients \( s_{k,\ell}^* \) are stored in the unused entries of the variable S of NonAddPenaltyLiPoint(). The Elements of \( S^{(j)} \) are denoted by \( s_{k,\ell} \). (Right table). As the memory would have to be allocated anyhow, accessing \( s_{k,\ell} \) or \( s_{k,\ell}^* \) is simpler than in the case of some other packed storage scheme.
G.4.2 Cholesky decomposition

The function NonAddPenaltyCholdecS calculates the Cholesky decomposition.

```c
void NonAddPenaltyCholdecS (const INT *dim, double *S, const double *R);
```

For $R \to \infty$ the formula for the penalized local linear estimator is modified and we need to solve $R(S^{(j)} + RI)^{-1} \equiv (R^{-1}S^{(j)} + I)^{-1}$. The corresponding Cholesky decomposition is calculated by NonAddPenaltyCholdecS_inv and invR=$R^{-1}$ is used as parameter.

```c
void NonAddPenaltyCholdecS_inv (const INT *dim, double *S, const double *invR);
```

Arguments

- **dim** (input) INT pointer to scalar
  
  $\text{dim}=d$.

- **S** (input/output) double matrix, dimensions = $(d + 1, d + 2)$
  
  The lower/left half of S contains the entries of $S^{(j)}$. The upper/right half is overwritten by the coefficients of the Cholesky decomposition.

- **R** (input) double pointer to scalar
  
  The matrix $(S^{(j)} + RI)$ is Cholesky decomposed.

- **invR** (input) double pointer to scalar
  
  The matrix $(invRS^{(j)} + I)$ is Cholesky decomposed.

G.4.3 Solving equation systems by Cholesky decomposition

If we assume $R > 0$, then the matrix $(S^{(j)} + RI)$ is positive definite. Hence, divisions by zero should not occur, and — if the flag THEORY_SAYS_ITS_OK is defined in matrixS.c — we omit the error handling routines.

Solving an equation system by Cholesky decomposition is done by NonAddPenaltyInvCholS:

```c
#define THEORY_SAYS_ITS_OK
void NonAddPenaltyInvCholS (const INT *dim, double *S, double *x);
```

Arguments

- **dim** (input) INT pointer to scalar
  
  $\text{dim}=d$.

- **S** (input) double matrix, dimensions = $(d + 1, d + 2)$
  
  The coefficients of a Cholesky decomposition are stored in the upper/right half of S.

- **x** (input/output) double vector, length = $d + 1$
  
  The (input) vector x is multiplied by the inverse of the Cholesky decomposed matrix and the result overwrites x.
G.4.4 Multiplication

Multiplication by $S^{(j)}$ is done by

```c
void NonAddPenaltyMultiplyS (dim, S, v, res)
    const INT *dim;
    const double *S; /* matrix with dimensions (dim+1, dim+2) */
    const double *v; /* vector with length (dim+1) */
    double *res; /* vector with length (dim+1) */
```

Arguments

- `dim` (input) `INT` pointer to scalar `dim = d`.
- `S` (input) `double` matrix, dimensions = $(d + 1, d + 2)$
  The lower/left half of $S$ is defines some symmetric matrix $S^{(j)}$, which is typically initialized by the function NonAddPenaltyLlPoint (section G.3).
- `v` (input) `double` vector, length = $d + 1$
  Some vector $v$.
- `res` (output) `double` vector, length = $d + 1$
  The result of $S^{(j)}v$ is stored in res.

G.5 LlGrid

G.5.1 Outgrid

Now, we choose some output grid

$$\{t_1^1, \ldots, t_{m_1}^1\} \times \cdots \times \{t_1^d, \ldots, t_{m_d}^d\} \subset [0,1]^d$$

The information about the output grid is stored in a vector `outgrid` and organized as follows:

<table>
<thead>
<tr>
<th>$t_1^1$</th>
<th>$t_1^2$</th>
<th>$t_1^3$</th>
<th>$t_1^4$</th>
<th>$t_1^5$</th>
<th>$t_1^6$</th>
<th>$t_1^7$</th>
<th>$t_1^8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{m_1}^1$</td>
<td>$t_{m_1}^2$</td>
<td>$t_{m_1}^3$</td>
<td>$t_{m_1}^4$</td>
<td>$t_{m_1}^5$</td>
<td>$t_{m_1}^6$</td>
<td>$t_{m_1}^7$</td>
<td>$t_{m_1}^8$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outgrid[0]</td>
<td>Outgrid[1]</td>
<td>Outgrid[d−1]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output points are accessed by (Outgrid[0][index[0]], Outgrid[d−1][index[d−1]]), where Outgrid[ ] is some vector of pointers whose length is bounded by the constant MAX_DIM.

G.5.2 Bandwidths

We want to evaluate NonAddPenaltyLlPoint (section G.3) at all points on the output grid.

As a restriction, we assume that the $k^{th}$ bandwidth depends only on the $k^{th}$ coordinate:

$$h_k(t_j) = h_k(t_{j,k})$$

This restriction simplifies computation. The bandwidths are stored in the same order as the output grid. The pointer to the different blocks are stored in `Band[ ]`.

<table>
<thead>
<tr>
<th>$h_1(t_1^1), \ldots, h_1(t_{m_1}^1)$</th>
<th>$h_2(t_1^1), \ldots, h_2(t_{m_2}^1)$</th>
<th>$\ldots$</th>
<th>$h_d(t_1^1), \ldots, h_d(t_{m_d}^1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Band[0]</td>
<td>Band[1]</td>
<td>Band[d−1]</td>
<td></td>
</tr>
</tbody>
</table>
G.5.3 Normalization

As the normalization depends on the bandwidth and probably also on the range of data (i.e., boundary adjustment), the normalization may be different for each output point. If we use product kernels, the structure of the normalization is simplified. The parameter Options[1] is used to select what the structure of the parameter normalization looks like:

0 normalizes has the same length as bandwidths and the pointers to the blocks are denoted by Norms[ ]. The equivalent normalization [0] in NonAddPenaltyLlPoint would be

\[ \prod_{k=1}^{d} \text{Norms}[k-1][ \text{indexes}[k-1] ] \]

1 every output point has its own normalization.

3 every output point has the same normalization.

The case Options[1]=2 calculates normalizations of type 0 from the bandwidths without worrying about boundary adjustment. The parameter normalization is used as temporary storage. Do not use this option.

Reparameterization
The reparameterization is the same as in section G.3.2 — except that the data is stored in param[0], . . . , param[d−1] instead of normalization[1], . . . , normalization[d].

G.5.4 LlGrid

```c
void NonAddPenaltyLlGrid (const INT *dim, x , y , S , T , outgrid , bandwidths , addtmp ,
                         Options , normalization , param )
```

Arguments

- `dim` (input) INT vector, length = d + 2
- `x` (input) double matrix, dimensions = (n, d)
- `y` (input) double vector, length = n
- `S` (output) double array, dimensions = (d + 1, d + 2, m1, . . . , md)

Values of S = diag_j(S_j). Simply a concatenation of the output S of the function NonAddPenaltyLlPoint.

Only half of the storage is currently used — see section G.4.1 on page 103.
T  (output) double array, dimensions = (d + 1, m₁, . . . , mₜ)
Values of \( T = \text{col}_j(\mathbf{T}^{(j)}) \).

outgrid (input) double vector, length = m₁ + . . . + mₜ
Output grid, see section G.5.1.

bandwidths (input) double vector, length = m₁ + . . . + mₜ
Bandwidths, see section G.5.2.

addtmp (output) double matrix, dimensions = (m₁ + . . . + mₜ, 2)
Temporary storage for distances between grid points and one observation.

Options (input) INT vector, length = 3
Options[0] specifies the weight function (section G.3.1 on page 102).
Options[1] specifies the usage of the parameter normalization.

normalization (input) double vector, length = m₁ + . . . + mₜ (if Options[1] = 0)
Normalization for product kernels.

normalization (input) double array, dimensions = (m₁, . . . , mₜ) (if Options[1] = 1)
Normalization for each output point may be chosen individually.

normalization (input) double pointer to scalar (if Options[1] = 2)
One size fits all?

normalization (output) double vector, length = m₁ + . . . + mₜ (if Options[1] = 2)
Not implemented for spherical Epanechnikov kernel (Options[0] = 1). Using this option is deprecated.

param (input) double vector, length = d (if Options[2] = 2 or 3)
Reparameterization, see section G.3.2 on page 103.

param (input) void — not used if Options[2] = 0 or 1.

G.6 Multiplication by \( \mathbf{Z} \)

NonAddPenaltyMultiplyZ(dim, vec, addvec)
const INT *dim;
const double *vec;
double *addvec;

Arguments

dim (input) INT vector, length = d + 1
\text{dim} = (d, m₁, . . . , mₜ).

vec (input) double array, dimensions = (d + 1, m₁, . . . , mₜ)
Parameters of the local linear estimator on a grid.

addvec (output) double matrix, dimensions = (m₁ + . . . + mₜ, 2)
Parameters of the additive components (see below) obtained by multiplication of vec with the matrix \( \mathbf{Z} \).

G.6.1 Additive components

If the parameters stored in vec were additive, i.e. \( \hat{\mathbf{\beta}}^{(j)} \) is of the form

\[
\hat{\mathbf{\beta}}^{(j)} = \left( \sum_{k=1}^{d} r_k(t_{j,k}), r_1(t_{j,1}), \ldots, r_d(t_{j,d}) \right) ^\top,
\]
then addvec consists of two columns, one for the intercept and one for the slope terms. Each column has length $m_1 + \ldots + m_d$. The following table shows the transposed entries of addvec.

\[
\begin{array}{c}
\sqrt{\frac{m_1}{m_1} r_1^1(t_1^{m_1})}, \ldots, \sqrt{\frac{m_1}{m_1} r_1^d(t_1^{m_1})} & \ldots & \sqrt{\frac{m_d}{m_d} r_d^1(t_1^{m_1})}, \ldots, \sqrt{\frac{m_d}{m_d} r_d^d(t_1^{m_1})} \\
\sqrt{\frac{m_1}{m_2} r_1'(t_1^{m_2})}, \ldots, \sqrt{\frac{m_1}{m_2} r_1'(t_1^{m_2})} & \ldots & \sqrt{\frac{m_d}{m_2} r_d'(t_1^{m_2})}, \ldots, \sqrt{\frac{m_d}{m_2} r_d'(t_1^{m_2})}
\end{array}
\]

As identifiability condition we assume $\sum_{j=1}^{m_k} r_k(t_{jk}^k) = 0$, for $k = 2, \ldots, d$.

### G.6.2 Related stuff

**MultiplyZSolveRS**

In order to avoid storing large temporary variables, multiplication by $\tilde{Z} \text{diag}(S^{(j)} + RI)^{-1}$ or $\tilde{Z} \text{diag}(R^{-1}S^{(j)} + I)^{-1}$, respectively is done in one step:

\[
\text{NonAddPenaltyMultiplyZSolveRS}(\text{dim}, S, \text{vec}, \text{addvec})
\]

**Arguments**

- **dim** (input) INT vector, length $= d + 1$
  - dim = $(d, m_1, \ldots, m_d)$
- **S** (input) double array, dimensions $= (d + 1, d + 2, m_1, \ldots, m_d)$
  - Cholesky decomposition of $(S + RI)^{-1}$ or $(R^{-1}S + I)^{-1}$, respectively.
  - S is not initialized here. For the lower/left half use NonAddPenaltyLlGrid and for the upper/right half use e.g. NonAddPenaltyInitializeMat.
- **vec** (input) double array, dimensions $= (d + 1, m_1, \ldots, m_d)$
  - Parameters of the local linear estimator on a grid.
- **addvec** (output) double matrix, dimensions $= (m_1 + \ldots + m_d, 2)$
  - Parameters of the additive components (see above).

**MultiplyZtrans**

Multiplication by $\tilde{Z}^\top$ is done by

\[
\text{NonAddPenaltyMultiplyZtrans}(\text{dim}, \text{addvec}, \text{vec})
\]

**Arguments**

- **dim** (input) INT vector, length $= d + 1$
  - dim = $(d, m_1, \ldots, m_d)$
- **vec** (input) double array, dimensions $= (d + 1, m_1, \ldots, m_d)$
  - Parameters of the local linear estimator on a grid.
- **addvec** (output) double matrix, dimensions $= (m_1 + \ldots + m_d, 2)$
  - Parameters of the additive components (see above).

**G.6.2 Related stuff**

**MultiplyZSolveRS**

In order to avoid storing large temporary variables, multiplication by $\tilde{Z} \text{diag}(S^{(j)} + RI)^{-1}$ or $\tilde{Z} \text{diag}(R^{-1}S^{(j)} + I)^{-1}$, respectively is done in one step:

\[
\text{NonAddPenaltyMultiplyZSolveRS}(\text{dim}, S, \text{vec}, \text{addvec})
\]

**Arguments**

- **dim** (input) INT vector, length $= d + 1$
  - dim = $(d, m_1, \ldots, m_d)$
- **S** (input) double array, dimensions $= (d + 1, d + 2, m_1, \ldots, m_d)$
  - Cholesky decomposition of $(S + RI)^{-1}$ or $(R^{-1}S + I)^{-1}$, respectively.
  - S is not initialized here. For the lower/left half use NonAddPenaltyLlGrid and for the upper/right half use e.g. NonAddPenaltyInitializeMat.
- **vec** (input) double array, dimensions $= (d + 1, m_1, \ldots, m_d)$
  - Parameters of the local linear estimator on a grid.
- **addvec** (output) double matrix, dimensions $= (m_1 + \ldots + m_d, 2)$
  - Parameters of the additive components (see above).

**MultiplyZtrans**

Multiplication by $\tilde{Z}^\top$ is done by

\[
\text{NonAddPenaltyMultiplyZtrans}(\text{dim}, \text{addvec}, \text{vec})
\]

**Arguments**

- **dim** (input) INT vector, length $= d + 1$
  - dim = $(d, m_1, \ldots, m_d)$
- **vec** (input) double array, dimensions $= (d + 1, m_1, \ldots, m_d)$
  - Parameters of the local linear estimator on a grid.
- **addvec** (output) double matrix, dimensions $= (m_1 + \ldots + m_d, 2)$
  - Parameters of the additive components (see above).
Calculate $(R^{-1}S + I)^{-1}(R^{-1}T + \tilde{Z}^\top \text{addvec})$ by

\begin{verbatim}
NonAddPenaltySolveAddZtrans_inv(dim, S, T, invR, addvec, vec)
  const INT *dim;
  const double *S;
  const double *T;
  const double *invR;
  const double *addvec;
  double *vec;
\end{verbatim}

Arguments
\begin{itemize}
  \item **dim** (input) INT vector, length = \(d + 1\)
  \item **S** (input) double array, dimensions = \((d + 1, d + 2, m_1, \ldots, m_d)\)
    Cholesky decomposition of \((S + RI)^{-1}\) or \((R^{-1}S + I)^{-1}\), respectively.
    \(S\) is not initialized here. For the lower/left half use NonAddPenaltyLlGrid
    and for the upper/right half use e.g. NonAddPenaltyInitializeMat.
  \item **T** (input) double array, dimensions = \((d + 1, m_1, \ldots, m_d)\)
    Values of \(T = \text{col}_j(T^{(j)})\).
  \item **R** (input) double pointer to scalar
    Penalty parameter \(R\).
  \item **invR** (input) double pointer to scalar
    Inverse penalty parameter \(R^{-1}\).
  \item **addvec** (input) double matrix, dimensions = \((m_1 + \ldots + m_d, 2)\)
    Parameters of the additive components (see section G.6.1).
  \item **vec** (output) double array, dimensions = \((d + 1, m_1, \ldots, m_d)\)
    Parameters of the local linear estimator on a grid.
\end{itemize}

To ensure the identifiability condition in section G.6.1 use

\begin{verbatim}
void NonAddPenaltyAddvecSubMean(dim, addvec)
  const INT *dim;
  double *addvec;
\end{verbatim}

This is not included in NonAddPenaltyMultiplyZtrans because the conditions are fulfilled in our application and separating this step allows us to define the parameter addvec as **const**.

Arguments
\begin{itemize}
  \item **dim** (input) INT vector, length = \(d + 1\)
    \item **dim** = \((d, m_1, \ldots, m_d)\).
  \item **addvec** (input/output) double matrix, dimensions = \((m_1 + \ldots + m_d, 2)\)
    Parameters of the additive components (see section G.6.1).
\end{itemize}

Do not care about the slopes in variable vec
If we are only interested in the intercepts, we save memory by using

\begin{verbatim}
NonAddPenaltyInterAddZtrans(dim, S, T, R, addvec, vec)
  const INT *dim;
  const double *S;
  const double *T;
  const double *R;
  const double *addvec;
  double *vec;
\end{verbatim}
NonAddPenaltyInterAddZtrans_inv (dim, S, T, invR, addvec, vec)
    const INT *dim;
    const double *S;
    const double *T;
    const double *invR;
    const double *addvec;
    double *vec;

Arguments
All parameters are the same as in NonAddPenaltySolveAddZtrans (with and without “_inv”) — except vec, which has smaller size:
vec (output) double array, dimensions = \((m_1, \ldots, m_d)\)
Values on the grid.

G.6.3 Initialize matrix

\[
\text{mat} = I - \tilde{Z} R (S + RI)^{-1} \tilde{Z}^T
\]

NonAddPenaltyInitializeMat (dim, S, mat, R)
NonAddPenaltyInitializeMatCD (dim, S, mat, R)
    const INT *dim;
    double *S;
    double *mat;
    const double *R;

\[
\text{mat} = I - \tilde{Z} (I - S (R^{-1} S + I)^{-1}) \tilde{Z}^T
\]

NonAddPenaltyInitializeMat_inv (dim, S, mat, invR)
NonAddPenaltyInitializeMatCD_inv (dim, S, mat, invR)
    const INT *dim;
    double *S;
    double *mat;
    const double *invR;

Arguments

\begin{verbatim}
dim (input) INT vector, length = d + 1
dim=(d, m_1, \ldots, m_d).
S (input/output) double array, dimensions = \((d + 1, d + 2, m_1, \ldots, m_d)\)
Assume the lower/left half of S was initialized by NonAddPenaltyLlGrid and contains the matrix S. This function calculates the Cholesky decomposition of \((S + RI)^{-1}\) or \((R^{-1} S + I)^{-1}\), respectively and stores the coefficients in the upper/right half of S.
mat (output) double matrix, dimensions = \((2(m_1 + \ldots + m_d), 2(m_1 + \ldots + m_d))\)
Matrix of the size of the number of parameters of the additive model. If the name of the function contains “CD”, \text{mat} is Cholesky decomposed by chol_dec defined in matalg.h.
R (input) double pointer to scalar
Penalty parameter R.
invR (input) double pointer to scalar
Inverse penalty parameter \(R^{-1}\).
\end{verbatim}
G.7 Penalized estimator

These functions calculate the estimator defined in formula (16) on page 26. For small penalties $R$, we use the following formula ($U = I \otimes (1, 0, \ldots, 0)$):

$$\text{est} = U(S + RI)^{-1}(T + \tilde{R}Z(I - \tilde{Z}R(S + RI)^{-1}\tilde{Z})_{\text{mat}}^{-1}\tilde{Z}^T(S + RI)^{-1}T).$$

This is done by

```c
void NonAddPenaltyEstimator (dim, x, y, S, T, outgrid, bandwidths, addtmp, Options, normalization, param, mat, R, est)
{
    const INT *dim;
    const double *x;
    const double *y;
    double *S;
    double *T;
    const double *outgrid;
    const double *bandwidths;
    double *addtmp;
    const INT *Options;
    /* const */ double *normalization;
    const double *param;
    double *mat;
    const double *R;
    double *est;

    NonAddPenaltyLlGrid (dim, x, y, S, T, outgrid, bandwidths, addtmp, Options, normalization, param);
    NonAddPenaltyInitializeMatCD (&dim[1], S, mat, R);
    NonAddPenaltyMultiplyZSolveRS (&dim[1], S, T, addtmp);
    inv_chol (mat, addtmp, &2(m1 + \ldots + md));
    // NonAddPenaltyAddvecSubMean (&dim[1], addtmp);
    NonAddPenaltyInterAddZtrans (&dim[1], S, T, R, addtmp, est);
}
```

The expression &dim[1] corresponds to a pointer to the vector obtained by removing the first element of dim[].

$$\ldots n \quad d \quad m_1 \quad \ldots \quad m_d \quad \ldots$$

\[\text{dim} \quad &\text{dim}[1]\]

For large penalties $R$, we use the following formula ($U = I \otimes (1, 0, \ldots, 0)$):

$$\text{est} = U(R^{-1}S + I)^{-1}(R^{-1}T + \tilde{Z}(I - \tilde{Z}Z) + \tilde{Z}^T S(R^{-1}S + I)^{-1}\tilde{Z})_{\text{mat}}^{-1}\tilde{Z}^T(R^{-1}S + I)^{-1}T).$$

void NonAddPenaltyEstimator_inv (dim, x, y, S, T, outgrid, bandwidths, addtmp, Options, normalization, param, mat, invR, est)
{
    const INT *dim;
    const double *x;
    const double *y;

    // Code...
```
double *S;
double *T;
const double *outgrid;
const double *bandwidths;
double *addtmp;
const INT *Options;

NonAddPenaltyLlGrid(dim, x, y, S, T, outgrid, bandwidths, addtmp, Options, normalization, param);
NonAddPenaltyInitializeMatCD_inv(&dim[1], S, mat, invR);
NonAddPenaltyMultiplyZSolveRS(&dim[1], S, T, addtmp);
inv_chol(mat, addtmp, &2(m1 + ... + md));
// NonAddPenaltyAddvecSubMean(&dim[1], addtmp);
NonAddPenaltyInterAddZtrans_inv(&dim[1], S, T, invR, addtmp, est);

Arguments

dim (input) INT vector, length = d + 2

dim = (n, d, m1, ..., md).
x (input) double matrix, dimensions = (n, d)
Observations \( \mathbf{X}_i, i = 1, \ldots, n. \)
y (input) double vector, length = n
Observations \( Y_i, i = 1, \ldots, n. \)
S (output) double array, dimensions = (d + 1, d + 2, m1, ..., md)
For the lower/left half see NonAddPenaltyLlGrid in section G.5.4. For the upper/right half see section G.4.2.
T (output) double array, dimensions = (d + 1, m1, ..., md)
Values of \( \mathbf{T} = \text{col}_j(\mathbf{T}_{(j)}). \)
outgrid (input) double vector, length = \( m1 + \ldots + md \)
Output grid, see section G.5.1.
bandwidths (input) double vector, length = \( m1 + \ldots + md \)
Bandwidths, see section G.5.2.
addtmp (output) double matrix, dimensions = \( (m1 + \ldots + md, 2) \)
Temporary storage for distances between grid points and one observation (in LlGrid) and for additive components. Returned values fulfill \( \tilde{Z} \text{ addtmp} = \tilde{P}_A^{(R)} \tilde{\beta}_l, \) which is addtmp = \( \tilde{Z} \tilde{\beta}_R \) (defined in formula (16)).
Options (input) INT vector, length = 3
Options[0] specifies the weight function (section G.3.1 on page 102).
Options[1] specifies the usage of the parameter normalization.
normalization (input) double vector, length = \( m1 + \ldots + md \) (if Options[1] = 0)
Normalization for product kernels.
normalization (input) double array, dimensions = \( (m1, \ldots, md) \) (if Options[1] = 1)
Normalization for each output point may be chosen individually.
G C PROGRAM CODE

G.8 MISE

normalization (input) double pointer to scalar (if Options[1] = 2)
One size fits all?

normalization (output) double vector, length = \(m_1 + \ldots + m_d\) (if Options[1] = 2)
Not implemented for spherical Epanechnikov kernel (Options[0] = 1). Using this option is deprecated.

param (input) double vector, length = \(d\) (if Options[2] = 2 or 3)
Reparameterization, see section G.3.2 on page 103.

param (input) void — not used if Options[2] = 0 or 1.

mat (output) double matrix, dimensions = \(2(m_1 + \ldots + m_d), 2(m_1 + \ldots + m_d)\)
see NonAddPenaltyInitializeMatCD or NonAddPenaltyInitializeMatCD_inv in section G.6.3.

R (input) double pointer to scalar (NonAddPenaltyEstimator only)
Penalty parameter \(R\).

invR (input) double pointer to scalar (NonAddPenaltyEstimator_inv only)
Inverse penalty parameter \(R^{-1}\).

est (output) double array, dimensions = \((m_1, \ldots, m_d)\)
The estimated values.

G.8 MISE

The integrated squared error (ISE) is the sum of squared difference between estimated and true values. When calculating ISE for different penalty parameters \(R\), the NonAddPenaltyLlGrid part in the function NonAddPenaltyEstimator is only processed once. The difference between ISE and integrated squared bias is that the estimator is applied to \(Y\) in the former and \(E[Y]\) in the latter case. The integrated variance is calculated by repeated evaluations of the estimator on a unit vector \((0, \ldots, 0, 1, 0, \ldots, 0)\). As \(S\) and \(mat\) were already initialized by the bias it remains to calculate \(T\) (stored in otherT). This is simplified by the fact that only one observation is non–zero.

void NonAddPenaltyGridMISE(dim, x, rx, rout, outgrid, Options, bandwidths, normalization, param, S, T, otherT, mat, addtmp, addvec, est, testR, var, bias2)

void NonAddPenaltyGridISE(dim, x, rx, rout, outgrid, Options, bandwidths, normalization, param, S, T, mat, addvec, est, testR, ise)
double *var;    /* MISE only */
double *bias2;   /* MISE only */
double *ise;     /* ISE only */

Arguments
Those parameters without description are the same as in NonAddPenaltyEstimator.
dim    (input) INT vector, length = \(d + 3\)
\(\text{dim} = (n, d, m_1, \ldots, m_d, \#R)\).
x     (input) double matrix, dimensions = \((n, d)\)
Observations \(\mathbf{X}_i, i = 1, \ldots, n\).
rx     (input) double vector, length = \(n\)
Observations \(\mathbb{E}[Y_i] \text{ or } Y_i, i = 1, \ldots, n\).
rout     (input) double array, dimensions = \((m_1, \ldots, m_d)\)
True regression function evaluated on the grid.
outgrid     (input) double vector, length = \(m_1 + \ldots + m_d\)
Output grid.
Options     (input) INT vector, length = 3
Options[0] specifies the weight function (section G.3.1 on page 102).
Options[1] specifies the usage of the parameter normalization.
bandwidths     (input) double vector, length = \(m_1 + \ldots + m_d\)
Bandwidths.
normalization     (input) double  — size depends on Options[1]
Normalization.
param     (input) double  — size depends on Options[2]
Reparameterization.
S     (output) double array, dimensions = \((d + 1, d + 2, m_1, \ldots, m_d)\)
T     (output) double array, dimensions = \((d + 1, m_1, \ldots, m_d)\)
otherT     (output) double array, dimensions = \((d + 1, m_1, \ldots, m_d)\) (MISE only)
Temporary storage. Same as \(T\) but with \(\mathbf{Y}\) replaced by a unit vector.
mat     (output) double matrix, dimensions = \((2(m_1 + \ldots + m_d), 2(m_1 + \ldots + m_d))\)
addtmp     (output) double matrix, dimensions = \((m_1 + \ldots + m_d, 2)\) (MISE only)
Temporary storage for distances between grid points and one observation
addvec     (output) double matrix, dimensions = \((m_1 + \ldots + m_d, 2)\)
Temporary storage for additive components.
est     (output) double array, dimensions = \((m_1, \ldots, m_d)\)
Temporary storage.
testR     (input) double vector, length = \(\#R\)
Different penalty parameters for which (M)ISE is calculated. Size is stored at
the end of \(\text{dim}\).
var     (output) double vector, length = \(\#R\) (MISE only)
Integrated variance. Assuming i.i.d. standard normal noise, the variance is
exactly evaluated.
bias2     (output) double vector, length = \(\#R\) (MISE only)
Integrated squared bias: Sum of squared difference between true values (rout)
and estimates based on noise–free data (rx).
ise     (output) double vector, length = \(\#R\) (ISE only)
The Akaike Information Criterion (AIC) compares observed and estimated values. As the non-additivity penalized estimator is defined only at the output points, and increasing the output grid such that every observation lies on a gridpoint is not a good idea, we interpolate the estimator. As the estimator is a pointwise compromise between the local linear and some additive estimator, we only interpolate the additive part. For the local linear part, we simply apply NonAddPenaltyLlPoint.

For the calculation of the trace of the hat matrix, we repeatedly (for every \( i \)) apply the estimator to \( \mathbf{Y} = \mathbf{e}_i \), the \( i \)th unit vector. In this case the calculation of \( T \) is simplified. Moreover the equivalent of \( T^{(j)} \) for some output point \( t_j = \mathbf{X}_j \) and \( \mathbf{Y} = \mathbf{e}_i \) is simply \( T^{(\mathbf{X}_j)}_{\mathbf{e}_i} = (\text{normalization}/n, 0, \ldots, 0) \).

**Interpolation**

The following function is used for (univariate) piecewise linear interpolation of

- additive components
- bandwidths
- inverse normalizations. (we do not check if normalization is larger than zero.)

In the case Options[1]=1, interpolation is different and not yet implemented.

```c
static void NonAddPenaltyGetWeight (x, outgrid, outlength, index, weight)
    const double x;
    const double *outgrid;
    const long int outlength;
    long int *index;
    double *weight;
```

**Arguments**

- **x** (input) double scalar (passed by value, not by reference) 
  \( x \in [\text{outgrid}[0], \text{outgrid}[\text{outlength}-1]] \).
  If \( x \) is outside, we replace \( x \) by the nearer one of the interval endpoints.

- **outgrid** (input) double vector, length = outlength 
  Increasingly sorted vector, e.g. Outgrid[k] (defined in section G.5.1).

- **outlength** (input) long int scalar (passed by value, not by reference) 
  Length of argument outgrid.

- **index** (output) long int pointer to scalar 
  \(*\text{index} \in \{0, \ldots, \text{outlength}-2\} \) with \( \text{outgrid}[\text{index}] \leq x \leq \text{outgrid}[\text{index}+1] \)

- **weight** (output) double pointer to scalar 
  \( x = (*\text{weight}) \times \text{outgrid}[\text{index}] + (1.0 - (*\text{weight})) \times \text{outgrid}[\text{index}-1] \)
G.9.1 GridAIC

```c
void NonAddPenaltyGridAIC(
    const INT *dim,
    const double *x,
    const double *y,
    const double *outgrid,
    const INT *Options,
    const double *bandwidths,
    const double *normalization,
    const double *param,
    double *S,
    double *T,
    double *mat,
    double *addtmp,
    double *addvec,
    double *addvec0,
    const double *testR,
    double *tracemat)
```

**Arguments**

- **dim** (input) INT vector, length = \( d + 3 \)
  
  \( \text{dim}=(n, d, m_1, \ldots, m_d, \#R) \).

- **x** (input) double matrix, dimensions = \( (n, d) \)
  
  Observations \( \mathbf{X}_i, i = 1, \ldots, n \).

- **y** (input) double vector, length = \( n \)
  
  Observations \( Y_i, i = 1, \ldots, n \).

- **outgrid** (input) double vector, length = \( m_1 + \ldots + m_d \)
  
  Output grid.

- **Options** (input) INT vector, length = \( 3 \)
  
  Options[0] specifies the weight function (section G.3.1 on page 102).
  Options[1] specifies the usage of the parameter normalization.
  Note that Options[1]=1 is not allowed.

- **bandwidths** (input) double vector, length = \( m_1 + \ldots + m_d \)
  
  Bandwidths.

- **normalization** (input) double — size depends on Options[1]
  
  Normalization.

- **param** (input) double — size depends on Options[2]
  
  Reparameterization.

- **S** (output) double array, dimensions = \( (d + 1, d + 2, m_1, \ldots, m_d) \)

- **T** (output) double array, dimensions = \( (d + 1, m_1, \ldots, m_d) \)

- **mat** (output) double matrix, dimensions = \( (2(m_1 + \ldots + m_d), 2(m_1 + \ldots + m_d)) \)

- **addtmp** (output) double matrix, dimensions = \( (m_1 + \ldots + m_d, 2) \)
  
  Temporary storage for distances between grid points and one observation.
addvec (output) double matrix, dimensions = \((m_1 + \ldots + m_d, 2)\)
Temporary storage for additive components (for original \(\mathbf{Y}\)).

addvec0 (output) double matrix, dimensions = \((m_1 + \ldots + m_d, 2)\)
Temporary storage for additive components (for \(\mathbf{Y}\) replaced by \(\mathbf{e}\)).

testR (input) double vector, length = \#R
Different penalty parameters for which AIC is calculated. Size is stored at the end of \(\mathbf{dim}\).

sigmahat (output) double vector, length = \#R
average (=sum divided by \(n\)) of squared residuals.

tracemat (output) double vector, length = \#R
Trace of hat matrix divided by \(n\)

\[
AIC_C(\text{testR}[k]) = \log(\text{sigmahat}[k]) + \frac{1 + \text{tracemat}[k]}{1 - \text{tracemat}[k] - 2/n}
\]

Parameter \(R = \infty\)
In order to calculate MISE or AIC for \(R = \infty\), the source code has to be modified. Look out for “invR” in AIC.c (6 occurrences), GridMISE.c (6) and GridMISE_include.c (1). The replacements are inside comments starting with “/\* or:”

G.10 matalg
In matalg.c we define functions for solving equations.

\textbf{void} chol_dec(\textbf{double} *\textbf{data}, \textbf{const} \textbf{INT} *\textbf{dim})

Arguments
\textbf{dim} (input) \textbf{INT} pointer to scalar
Dimension. Set matalg_debug=1; for debugging.

\textbf{data} (input/output) \textbf{double} matrix, dimensions = \((\text{dim}, \text{dim})\)
\textbf{data} contains the elements of a symmetric matrix. It is overwritten by the coefficients of a Cholesky decomposition.

\textbf{INT} ic_status;
\textbf{#define} IC_unique 0 /* unique solution */
\textbf{#define} IC_exist 1 /* not unique, but solution exists */
\textbf{#define} IC_fail 2 /* no solution exists */
\textbf{void} inv_chol(\textbf{const} \textbf{double} *\textbf{mat}, \textbf{double} *\textbf{x}, \textbf{const} \textbf{INT} *\textbf{dim})

Arguments
\textbf{dim} (input) \textbf{INT} pointer to scalar

\textbf{data} (input) \textbf{double} matrix, dimensions = \((\text{dim}, \text{dim})\)
Coefficients of a Cholesky decomposition. Obtained by chol_dec.

\textbf{x} (input/output) \textbf{double} vector, length = \(d + 1\)
The (input) vector \(\mathbf{x}\) is multiplied by the inverse of the Cholesky decomposed matrix and the result overwrites \(\mathbf{x}\). The variable ic_status displays the success.

The above functions are not very sophisticated. If desired, one may replace them by wrappers for BLAS... When replacing the functions in matrixS.c, be cautious, because somewhere in matrixZ_include.c we directly access \(S^{(j)}\) stored in S. Use \textbf{#define} CHECK_MATMUL for verifying correctness.

117
H About figures

Informations on ‘how to reproduce figures and tables are given in this section. The source code is stored in appendix.tar.gz available at http://stat.ethz.ch/~mstuder/thesis/ or http://www.unizh.ch/biostat/People/mstuder/thesis/. Most of the figures are obtained by selecting a type and passing the output of GridAIC/GridISE/GridMISE to a plot function plotAIC/plotISE/plotSimul. Hence, we may plot the design (type="D"), the true regression function ("F"), estimation at minimum of AIC/ISE (perspective: "e", gray scale: "E"), and AIC/ISE/MISE (perspective plot: "P"; gray scale: "G"/"g"; gray scale and contour: "c"; cross-section along $R_{\text{opt}}(h)$ or $h_{\text{opt}}(R)$: "H" or "R"/"r").

The calculation of the MISE is somewhat special, as the variance is calculated only once for each design: To calculate variance and bias for the function rfull (figure 15 on page 50), use

```r
mise.rfull.200 <- doSimul("Data200", c(50,50), rfull, testB, testR)
```

If we want the same for the function radd we copy the variance from above and calculate the bias by GridISE. The design, output grid, and parameters are extracted from mise.rfull.200:

```r
mise.radd.200 <- doLazySimul(mise.rfull.200, radd)
```

H.1 Regression functions

The source code for the regression functions in section 5.1 is given below.

```r
rfull <- function(x)
{
  if (is.null(x)){
    return("3 Normal [BS,TG(2000)|JCGS:9.388-360; Figure 6]" )
  }
  if (is.null(dim(x))){
    dim(x) <- c(1,length(x))
  }
  if (length(dim(x)) != 2){
    stop(deparse(substitute(x))," has wrong dimension. \n")
  }
  if (any(x<0)||any(x>1)){
    stop("Observation outside [0,1]. \n")
  }
  y <- dnorm((x[,1]-.25)*8)*dnorm((x[,2]-.25)*8)*0.3*2*pi +
       dnorm((x[,1]-.75)*16)*dnorm((x[,2]-.75)*16)*0.7*2*pi +
       dnorm((x[,1]-.5)*2)*dnorm((x[,2]-.5)*2)*0.5*2*pi
  return(y)
}

rrot <- function(x)
{
  if (is.null(x)){
    return("rot45 ; 3 Normal")
  }
  if (is.null(dim(x))){
    dim(x) <- c(1,length(x))
  }
  if (length(dim(x)) != 2){
    stop(deparse(substitute(x))," has wrong dimension. \n")
  }
  if (any(x>0)||any(x<1)){
    stop("Observation outside [0,1]. \n")
  }
}
```

118
if ( any(x<0) | any(x>1)) {
    stop("Observation outside \[0 , 1\].\n"")
}
rx1 <- x[ ,1] + x[ ,2] - 0.5
rx2 <- -x[ ,1] + x[ ,2] + 0.5
y <- dnorm((rx1 - .25) * 8) * dnorm((rx2 - .25) * 8) * 0.3 * 2 * pi +
    dnorm((rx1 - .75) * 16) * dnorm((rx2 - .75) * 16) * 0.7 * 2 * pi +
    dnorm((rx1 - .5) * 2) * dnorm((rx2 - .5) * 2) * 0.5 * 2 * pi
return(y)
}

radd <- function(x)
{
    if (is.null(x)){
        return("Some additive function")
    }
    if (is.null(dim(x))){
        dim(x) <- c(1, length(x))
    }
    if (length(dim(x)) != 2){
        stop(deparse(substitute(x)), "has wrong dimension.\n"")
    }
    if (any(x<0) | any(x>1)){
        stop("Observation outside \[0 , 1\].\n"")
    }
    y <- dnorm((x[ ,1] - .25) * 8) * 0.3 * sqrt(2 * pi) +
        dnorm((x[ ,1] - .75) * 16) * 0.7 * sqrt(2 * pi) +
        dnorm((x[ ,1] - .5) * 2) * 0.5 * sqrt(2 * pi) +
        dnorm((x[ ,2] - .25) * 8) * 0.3 * sqrt(2 * pi) +
        dnorm((x[ ,2] - .75) * 16) * 0.7 * sqrt(2 * pi) +
        dnorm((x[ ,2] - .5) * 2) * 0.5 * sqrt(2 * pi)
    return(y/2)
}

diagsin <- function(x)
{
    if (is.null(x)){
        return("sin(2Pi(x1+x2))")
    }
    if(is.null(dim(x))) {
        dim(x) <- c(1, length(x))
    }
    if(length(dim(x)) != 2) {
        stop(deparse(substitute(x)), "has wrong dimension.\n"")
    }
    if(any(x < 0) | any(x > 1)) {
        stop("Observation outside \[0 , 1\].\n"")
    }
    y <- sin(2 * pi * (x[ ,1] + x[ ,2]))
    return(y)
}
H.2 Figures

Note that the following code should reproduce the figures. Be warned, that calculating high resolution plots will take a long time. This holds especially for the example with 1600 observations.

```r
# Test the transformation

# By = 0.02

testRtrsf <- seq(0, 1, by = 0.01)  # by = 0.02

testRtrsf[1] <- 0.0001

testRtrsf[51] <- 1 - 0.0001

testR <- testRtrsf / (1 - testRtrsf)

testBtrsf <- seq(-1.3, -0.5, by = 0.005)  # by = 0.02

testB <- 10^testBtrsf

outgrid50x50 <- getFdesign(c(50, 50), F)

f <- rfull

x <- cbind(Data200$twodimdat$x1, Data200$twodimdat$x2)

epsilon <- Data200$twodimdat$y - Data200$twodimdat$Ey

y <- f(x) + epsilon

gaic.ise.rfull.200.1 <- GridAIC.ISE(x, y, f, outgrid50x50, testR = testR, testB = testB, boundary = c(0, 1), kernel = 0, desc = "3. Normal", printstep = T, testRtrsf = testRtrsf, testBtrsf = testBtrsf, sigma = Data200$sigma, timestamp = T)

gaic.ise.rfull.200.2 <- GridAIC.ISE(x, y, f, outgrid50x50, testR = testR, testB = testB, boundary = c(0, 1), kernel = 0, desc = "3. Normal", printstep = T, testRtrsf = testRtrsf, testBtrsf = testBtrsf, sigma = 2 * Data200$sigma, timestamp = T)

gaic.ise.rfull.200.05 <- GridAIC.ISE(x, y, f, outgrid50x50, testR = testR, testB = testB, boundary = c(0, 1), kernel = 0, desc = "3. Normal", printstep = T, testRtrsf = testRtrsf, testBtrsf = testBtrsf, sigma = 0.5 * Data200$sigma, timestamp = T)

Figure 1, left: plotISE(aic.ise.rfull.200.1, type="F", noTitle=T)
Figure 1, left: plotISE(aic.ise.rfull.200.1, type="D", noTitle=T)
Figure 3, left: plotISE(aic.ise.rfull.200.1, type="e", noTitle=T)
Figure 5, left: plotISE(aic.ise.rfull.200.1, type="e", noTitle=T)
Figure 5, right: plotAIC(aic.ise.rfull.200.1, type="e", noTitle=T)
Figure 19: par(mfcol=c(3,2))

plotAIC(aic.ise.rfull.200.1, type=c("G","H","r"),add=T)

plotISE(aic.ise.rfull.200.1, type=c("G","H","r"),add=T)

Figure 16: plotSimul(mise.rfull.200.01, type=c("P","G","H","r"),maximum=30,noTitle=T)

Figure 4, left displays a^{(j)}_{il} defined in section 3.3. The current version of the software does not calculate this. The figure was drawn using an older version.
References


121


Curriculum vitae

Name: Studer
First name: Michael Markus

Date of birth: September 1, 1972
Place of birth: Zurich, Switzerland

Citizenship: Swiss
Marital status: Single

Home address: Trottenstrasse 16
CH–8037 Zurich

Work address: Department of Biostatistics
ISPM
University Zurich
Sumatrastrasse 30
CH-8006 Zurich
++41–1–634 46 47

Email: mstuder@ifspm.unizh.ch

Education

1979–1985 Primary school in Dagmersellen, LU, Switzerland
1985–1992 High school in Sursee, LU, Switzerland
1992 Matura type C at Kantonsschule Sursee


1997–2002 Ph. D. studies. Advisors: Prof. Dr. T. Gasser and PD Dr. B. Seifert.

Acknowledgment

This work was supported by Swiss National Science Foundation, project number 2100–052567.97, entitled ‘Semiparametric Modeling’.

I would like to thank my advisors Burkhardt Seifert and Theo Gasser for their support, and sharing their experience. The colleagues from the Department of Biostatistics provided a stimulating environment for me. Furthermore, I thank Hans Rudolf Künsch and Peter Bühlmann for accepting to be reviewers of my thesis at the ETH.

More generally, there are various people (family and friends) I would like to thank for sharing their time with me.