# Additive Modelling with Penalized Regression Splines and Genetic Algorithms

Rüdiger Krause[1] and Gerhard Tutz

Department of Statistics,
Ludwig-Maximilians University, Akademiestr.1, 80799 München, Germany

**Summary.** Additive models of the type $y = f_1(x_1) + \ldots + f_p(x_p) + \epsilon$ where $f_j, j = 1, \ldots, p$, have unspecified functional form, are flexible statistical regression models which can be used to characterize nonlinear regression effects. The basic tools used for fitting the additive model are the expansion in B-splines and penalization which prevents the problem of overfitting. This penalized B-spline (called P-spline) approach strongly depends on the choice of the amount of smoothing used for components $f_j$. In this paper we treat the problem of choosing the smoothing parameters by genetic algorithms. In several simulation studies our approach of automatically calculation of the smoothing parameters is compared to alternative methods given in literature. In particular functions with different spatial variability are considered and the effect of constant respectively local adaptive smoothing parameters is evaluated.

## Keywords

Additive model, Genetic algorithm, Penalized regression splines, B-splines, Improved AIC criterion.

## 1 Introduction

Traditionally, there have been two basic approaches to address the problem of choosing basis functions. One technique places knots (and their corresponding basis functions) adaptively, i.e. the function is estimated by only a small set of basis functions which are adaptively chosen by a selection procedure. Some known examples for this technique are MARS (Friedman (1991)), forward selection, backward elemination and stepwise regression (Rawlings, Pantula & Dickey (1998)). In a broad sense we also can add the Support Vector Machines (Chapelle & Vapnik (1999), Vapnik (1995), Vapnik (1998)) and its extension to Relevance Vector Machines (Tipping (2000), Tipping (2001)) to this group.

The alternative approach (which we apply in this paper) avoids the problem of knot selection problems by using a large number of basis functions in combination with penalization of the coefficients. The danger of overfitting resulting in wiggly estimated curves is avoided by introducing a penalty term. There exists a large number of proposals for specifying an accurate penalty term (see e.g. Eilers & Marx

[1] krause@stat.uni-muenchen.de

(1996), Hastie, Tibshirani & Friedman (2001)). All proposals have in common that each penalty term is characterized by a smoothing parameter $\lambda$. This smoothing parameter controls the influence of the penalty term and hence the smoothness of the estimation function. A large parameter value tends to result in smooth estimators (e.g. $\lambda \to \infty$ leads to a linear estimator). In contrast, a small parameter value yields wiggly estimated curves (the extreme case is an interpolation of data for $\lambda = 0$). To prevent over- respectively underfitting of data (Bishop (1995)) accurate choice of the smoothing parameter is essential. For simple problems a grid search is sufficient for choosing a suitable smoothing parameter (Eilers & Marx (1996)). However, for more complex problems this approach is not any longer efficient.

A solution of this problem we propose the application of genetic algorithms (Holland (1975), Goldberg (1989)). Based on randomly stochastical search, genetic algorithms also yield accurate results for complex problems in many dimensions.

In this paper we mainly apply the new approach to the choice of smoothing parameters in simulated data, which are modelled by additive models (Hastie & Tibshirani (1990)). The paper is structured as follows: in the next section we generally describe the class of additive models and the flexible representation of functions by expansions in B-spline basis functions. Section 3 presents the penalization concept of Eilers & Marx (1996) and adapts it to our problem. In section 4 we introduce the genetic algorithm for the choice of the smoothing parameters. Finally section 6 compares our approach with other methods proposed in literature (and shortly sketched in 5) by several simulation studies.

## 2 Additive Model and B-splines

A very popular and flexible approach which assumes some structure in the predictor space is the *additive model* discussed in detail by Hastie & Tibshirani (1990). Suppose that we have observations $(y_i, \mathbf{x}_i), i = 1, \ldots, n$, where each $\mathbf{x}_i$ is now a vector of $p$ components $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$. Then it is assumed that the response variable $y_i$ depends on $\mathbf{x}_i$ by

$$\begin{aligned} y_i &= \beta_0 + f_1(x_{i1}) + \ldots + f_p(x_{ip}) + \epsilon_i \\ &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \end{aligned} \tag{1}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. It is obvious that the additive model replaces the problem of estimating a function $f$ of a $p$-dimensional variable $\mathbf{x}_i$ by one of estimating $p$ separate one-dimensional functions $f_j(x_{ij})$. The advantage of (1) is its potential as a data analytic tool: since each variable is represented separately one can plot the $p$ coordinate functions separately and thus evaluate the roles of the single predictors.

An approach which allows flexible representations of the functions $f_j(x_{ij})$ is the *expansion in basis functions*. Hence for example the function $f_j(x_{ij})$ is represented as

$$f_j(x_{ij}) = \sum_{\nu=1}^{K_j} \beta_{j\nu}\, \phi_{j\nu}(x_{ij}) \tag{2}$$

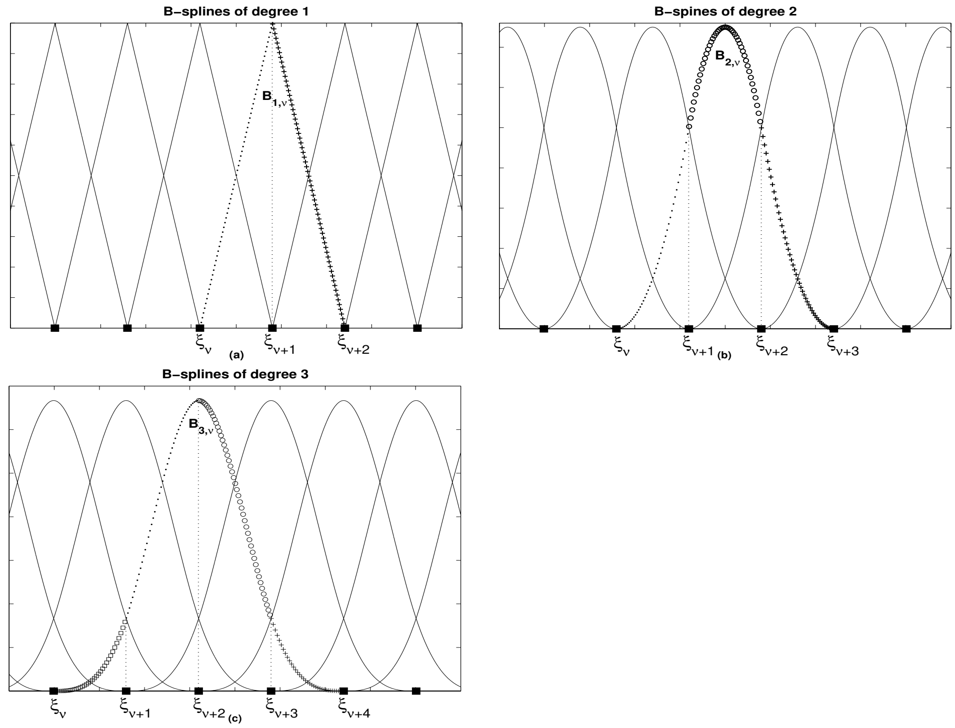


**Figure 1.** *Here B-splines of degree 1, 2 and 3 are shown. In each figure the different polynomials of one B-spline are exemplarily plotted.*

where the $\beta_{j\nu}$ are unknown coefficents and $\{\Phi_{j\nu}(x_{ij}), \nu = 1, \ldots, K_j\}$ is a set of basis functions. Each basis function $\Phi_{j\nu}(x_{ij})$ is characterized by a knot $\xi_{j\nu}$ which is from the range of the $j$th covariate. There are several popular basis functions, e.g. the *truncated power series basis* (Ruppert & Carroll (1997), Wand (2002)), the numerically more stable *B(asic)-spline basis* (Marx & Eilers (1998)), *thin-plate spline basis* (Wood (2002)), *Demmler-Reinsch basis* or *radial basis* functions.

The focus in this paper is on B-splines which are shortly sketched in the following. A more detailed presentation is given in de Boor (1978) or de Boor (1993) and Dierckx (1995). Figure 1(a) shows B-splines of degree 1, respectively order 2. Here at each knot $\xi_\nu, \nu = 1, \ldots, K_j$, a B-spline is generated by joining of two polynomials of degree 1 (piecewise linear functions). Figure 1(b) shows B-splines of degree 2 and order 3 (quadratic B-splines). At the inner knots (for example $\xi_{\nu+1}$ and $\xi_{\nu+2}$) we join together three polynomials of degree 2. For this kind of B-splines the first derivatives are equal at the joining points. This does not hold for the second derivatives. In this paper we mainly use B-splines of degree 3 respectively order 4 (cubic B-splines) which are generated by four polynomials of degree 3 (Figure 1(c)). Again these polynomials are joint at the inner knots. In this case the first and the second derivatives are equal at the joining points.

B-splines of degree $d$ have the following general properties:

- B-splines consist of $d+1$ polynomial pieces, each of degree $d$;
- they have $d$ inner knots where the polynomial pieces become joined;
- B-splines have an overlap by $2d$ neighboring B-splines. Of course the leftmost and the rightmost B-splines have less overlap;
- at the joining points, derivatives up to order $d-1$ are continuous;

- B-splines are positive on a domain spanned by $d+2$ knots; outside of this domain the B-spline is zero.

Note that each interval between two adjacent knots is covered by $d+1$ B-splines of degree $d$. The basis functions $\phi_{j\nu}$ depend on one knot only. When using one knot to identify a specific B-spline we take the leftmost knot at which the spline becomes non-zero. For computation of B-splines the formulae of de Boor (1978), are very helpful. A B-spline $B_{d,\nu}$ (for degree $d \geq 1$), which starts at knot $\xi_\nu$, may be computed by

$$B_{d,\nu}(x) = \frac{x - \xi_\nu}{\xi_{\nu+d} - \xi_\nu} B_{d-1,\nu}(x) + \frac{\xi_{\nu+d+1} - x}{\xi_{\nu+d+1} - \xi_{\nu+1}} B_{d-1,\nu+1}(x) \; . \tag{3}$$

For equidistant knots which are used here (3) simplifies to

$$B_{d,\nu}(x) = \frac{1}{d \cdot d\xi} \left[ (x - \xi_\nu) B_{d-1,\nu}(x) + (\xi_{\nu+d+1} - x) B_{d-1,\nu+1}(x) \right]$$

because $\xi_{\nu+d} - \xi_\nu = \xi_{\nu+d+1} - \xi_{\nu+1} = d \cdot d\xi$ where $d\xi$ is the distance between two adjacent knots.

## 3 Estimation with Penalized Shrinkage

For the additive model (1) parameters are estimated by minimizing the *penalized residual sum of squares (pRSS)*

$$\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \sum_{\nu=1}^{K_j} \beta_{j\nu} \phi_{j\nu}(x_{ij}))^2 + \tau(\{\lambda_{j\nu}\}) \right\} \tag{4}$$

where

$$\tau(\{\lambda_{j\nu}\}) = \sum_{j=1}^{p} \sum_{\nu=k+1}^{K_j} \lambda_{j\nu} (\Delta^k \beta_{j\nu})^2 \tag{5}$$

denoting the penalty term and $\lambda_{j\nu} \geq 0,\ j = 1, \ldots, p,\ \nu = k+1, \ldots, K_j, k = 1, 2, \ldots,$ are *local* smoothing parameters that control the amount of shrinkage: the larger the values of $\lambda_{j\nu}$, the larger the amount of shrinkage (Hastie, Tibshirani & Friedman (2001)). If $\lambda_{j,k+1} = \ldots = \lambda_{j,K_j} = \lambda_j$ we have a *global* smoothing parameter for the $j$th explanatory variable. Although global parameters are more easily to handle, it has been demonstrated by Ruppert & Carroll (2000), that local smoothing parameters yield better performance. For global smoothing parameters the penalization is the same as in Eilers & Marx (1996). They suggested to penalize the difference of adjacent coefficients. Hence in (4) the expression $\Delta^k \beta_{j\nu}, k = 1, 2, \ldots$, denotes the $k$th difference, e.g. the 2th difference has the form

$$\begin{aligned} \Delta^2 \beta_{j\nu} &= \Delta^1 (\beta_{j\nu} - \beta_{j\nu-1}) \\ &= (\beta_{j\nu} - \beta_{j\nu-1}) - (\beta_{j\nu-1} - \beta_{j\nu-2}) \\ &= (\beta_{j\nu} - 2\beta_{j\nu-1} + \beta_{j\nu-2}). \end{aligned}$$

It can be shown (Appendix(A)) that the estimator $\hat{\boldsymbol{\beta}}(\boldsymbol{\Lambda})$ which minimizes (4) has the form

$$\hat{\boldsymbol{\beta}}(\boldsymbol{\Lambda}) = (\mathbf{B}^T \mathbf{B} + \mathbf{D}^T \boldsymbol{\Lambda} \mathbf{D})^{-1} \mathbf{B}^T \mathbf{y}. \tag{6}$$

where $\mathbf{B}$ is a design matrix of dimension $n \times [(K_1 - 1) + \ldots + (K_p - 1)] + 1$, $\mathbf{D}$ is a $[(K_1 - k) + \ldots + (K_p - k)] + 1 \times [(K_1 - 1) + \ldots + (K_p - 1)] + 1$- penalization matrix and $\mathbf{\Lambda} = diag(0, \lambda_{1,k+1}, \ldots, \lambda_{1,K_1}, \lambda_{2,k+1}, \ldots, \lambda_{p,K_p})$ is a smoothing matrix of dimension $[(K_1 - k) + \ldots + (K_p - k)] + 1 \times [(K_1 - k) + \ldots + (K_p - k)] + 1$. The structure of matrices $\mathbf{B}$ and $\mathbf{D}$ are given in detail in Appendix(A).

The performance of the penalized estimate strongly depends on the choice of the smoothing parameters $\lambda_{j\nu}$. A criterion with favourable properties has been proposed by Hurvich & Simonoff (1998), which is given by

$$AIC_{imp} = log\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right] + 1 + 2 \cdot \frac{[tr(\mathbf{H}) + 1]}{n - tr(\mathbf{H}) - 2} \tag{7}$$

where $\mathbf{H} = \mathbf{B}(\mathbf{B}^T\mathbf{B} + \mathbf{D}^T\mathbf{\Lambda}\mathbf{D})^{-1}\mathbf{B}^T$ is the hat matrix. The smoothing parameters have to be chosen such that the criterion becomes minimal. In the simulation study (see section (6)) also alternative selection criteria have been used, e.g. generalized cross validation (GCV) as used by Ruppert & Carroll (2000). However, the results with different selection criteria do not differ very much from each other. Thus, in the representation we restrict ourselves to criterion (7).

## 4 Choice of Smoothing Parameters by Real-coded Genetic Algorithms

The real limit in the choice of smoothing parameters is the dimensionality of the problem. Even if global smoothing parameters are used $p$ smoothing parameters have to be chosen. For local smoothing the number of parameters increases to $K_1 + \ldots + K_p$ which for 30 knots in each dimension results in $30p$ smoothing parameters. A grid search which has been used for simpler problems by Eilers & Marx (1996), cannot be recommended. Ruppert & Carroll (2000) give an iterative method based on a linear interpolation algorithm. In the present paper the use of genetic algorithms is proposed.

Genetic Algorithms (Holland (1975), Goldberg (1989)) are originally based on Darwin's evolution theory (Darwin (1859)) which refers to the principle that better adapted (fitter) individuals win against their competitors under equal external conditions. Like their biological standard, genetic algorithms use biological components (or operators) like selection, crossover, or mutation to model the natural phenomenon of genetic inheritance and Darwinin strife of survival. For some background on the biological processes of genetics and the origin of the terminology see Haupt & Haupt (1998) and Mitchell (1996). In this article we only describe some selected concepts which are important for real-coded genetic algorithms.

The function to be optimized is denoted as fitness-function (short: *fitness*). The optimization problem can be treated as a minimization- or a maximization problem. We consider maximization problems only, because minimizing a function $f$ is equivalent to maximizing the function $-f$.

The smallest units linked to relevant information of a genetic algorithm are called *genes.* The genes are either single units or short blocks of adjacent units and the information is coded in form of numbers, characters, or other symbols. In real-coded genetic algorithms every gene is a single unit which is coded by a real value. Usually several genes are arranged in a linear succession which is called *string* (also *chromosome, individual*). In the context of smoothing parameter selection a string is a vector of the form $(\lambda_{1,k+1}, \ldots, \lambda_{1,K_1}, \lambda_{2,k+1}, \ldots, \lambda_{p,K_p})$ and thus a local smoothing parameter $\lambda_{j\nu}$ correspond to one gene. In the case of global smoothing parameters a string reduces to $(\lambda_1, \ldots, \lambda_p)$. Without loss of generality in this section we assume one covariate $(j = 1)$, only.

Before starting the iterative genetic algorithm the user needs to construct an initial population of several strings. This population usually consists of genes chosen randomly from a uniform distribution on a given interval. The population size (noted as *popsize*) is usually chosen freely. A rating of the quality of the used smoothing paramater combination is given by the *improved AIC-criterion* (Hurvich & Simonoff (1998)). Here, the smoothing parameters have to be chosen such that the criterion becomes minimal. For use of the genetic algorithm it is more suitable to work with a criterion which has to be maximized. This is easily achieved by simple mathematical transformations which are constant during across iterations of the genetic algorithm. For that purpose we subtract a sufficiently large constant from all AIC-values of a population such that all the AIC-values become negative. The simulations (section 6) show that the largeness of the chosen constant has no influence on the results. Following multiplication with $(-1)$ yields a criterion which has to be maximized. We denote the values which characterize the quality of the strings as *fitness values* (short *fitness*).

For the design of powerful genetic algorithms operators like crossover, mutation or selection are important. The genetic algorithm always yields several strings as a potential solution of an optimization problem. This collection of strings is called *population*. If we apply operators to strings we generate a population with new different strings. This new population of strings is called *offspring*. We denote the particular populations as *generations*, or more precisely as parent- respectively offspring generation. Several authors (Herrera, Lozano & Verdegay (1998), Michalewicz (1996)) show that operators have to meet with various purposes during the application of a genetic algorithm. In general there are two conflicting objectives (*exploitation-exploration-dilemma*):

(i) The initial population very rarely includes strings with solutions at (or at least close to) the global optimum. Thus it is helpful to generate offspring which are scattered over the whole search space thereby hoping that at least one of the strings is located near the global optimum. The objective to explore the search space with strings and acquire information about the nature of the space is described as *exploration*.

(ii) After some iteration steps the genetic algorithm may have generated new strings with solutions which are located closer to the global optimum. In this case we are primarily interested in obtaining information near the optimum by utilizing the local possibilities of upgrade close to the parents and by generating fitter offspring there. This stepwise improvement of the stings' fitness by use of local information is called *exploitation*.

The relevance of these two conflicting objectives is differs for particular steps of the algorithm. At the beginning (where we have no idea about the location of the global optimum) exploration is more relevant compared to exploitation and vice versa. Hence a suitable balance between exploration and exploitation is needed during the whole iteration process. To adequately solve these conflicting objectives we require operators which change during the genetic algorithm (*adaptive* or *non-uniform* operators). In the following section the operators, which will be used, are described in brief.

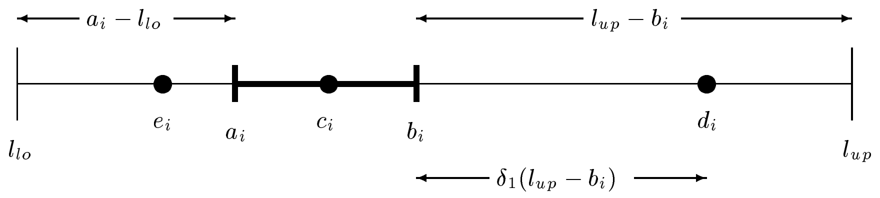


**Figure 2.** *With $a_i$ and $b_i$ representing the parents the IAC operator generates the appropriate genes $c_i$, $d_i$ and $e_i$ of the children. The first offspring $c_i$ is located within the parents' interval $[a_i, b_i]$. The other children are randomly positioned left and right outside the interval $[a_i, b_i]$. Every string only takes values within the range between $l_{lo}$ and $l_{up}$.*

### 4.1 Improved arithmetical crossover (IAC)

In the last decade numerous different types of crossover operators have been suggested (see e.g. Eshelman & Schaffer (1993), Michalewicz (1996), Radcliffe (1991), Wright (1991)). An overview with several simulations can be found in Herrera, Lozano & Verdegay (1998). Here we present a new crossover operator called *improved arithmetical crossover*, short *IAC*.

Suppose we have two real-coded strings (each has $m$ genes) for crossover with values in an intervall $[l_{lo}, l_{up}]$ with lower limit $l_{lo}$ and upper limit $l_{up}$

$$\begin{array}{ll} String\ 1 & (a_1 \dots a_i \dots a_m) \\ String\ 2 & (b_1 \dots b_i \dots b_m). \end{array}$$

The IAC operator is defined by (compare also Figure 2)

$$\begin{aligned} c_i &= \nu a_i + (1-\nu) b_i, \\ d_i &= b_i + \delta_1 (l_{up} - b_i), \\ e_i &= a_i - \delta_2 (a_i - l_{lo}), \end{aligned} \tag{8}$$

with $i = 1, \dots m$, and thus the offspring have the form

$$\begin{array}{ll} Offspring\ 1 & (\nu a_1 + (1-\nu) b_1 \dots \nu a_i + (1-\nu) b_i \dots \nu a_m + (1-\nu) b_m) \\ Offspring\ 2 & (b_1 + \delta_1 (l_{up} - b_1) \dots b_i + \delta_1 (l_{up} - b_i) \dots b_m + \delta_1 (l_{up} - b_m)) \\ Offspring\ 3 & (a_1 - \delta_2 (a_1 - l_{lo}) \dots a_i - \delta_2 (a_i - l_{lo}) \dots a_m - \delta_2 (a_m - l_{lo})) \end{array}$$

where $\nu \in [0, 1]$ can be chosen constant or variable over the number of iterations. The parameters $\delta_i \in [0, 1], i = 1, 2$, are uniformly distributed random numbers. Every string takes values in the default interval $[l_{lo}, l_{up}]$.

A freely chosen crossover probability $p_c$ determines which strings of the parent population are selected for crossover. Therefore we generate a random (float) number $r_i \in [0, 1], i = 1, \dots, popsize$ for every string of the population. A string is used for crossover operation if $r_i < p_c$ holds. In the crossover process we need couple of strings and thus it is necessary to select an even number of parent strings.

The IAC operator generates three new offspring and we select the two best strings, which will replace the parents. Interestingly, the IAC operator yields children which improve exploration and exploitation simultaneously. Figure 2 shows that two offspring ($d_i$ and $e_i$) are located outside the parents' interval $[a_i, b_i]$ and thus regions further apart in the search space can be explored. In addition, one child (here $c_i$) is located within the parents' interval and is primarily responsible for an improvement of exploitation.

In section 6 we compare the quality of IAC operator with that of the arithmetical crossover operator (Michalewicz (1996)). Thereby the arithmetical crossover operator is defined by a weighted linear combination of two parents, i.e.

$$\begin{aligned} c_i &= \nu a_i + (1-\nu) b_i, \qquad && i = 1, \ldots, m \\ d_i &= \nu b_i + (1-\nu) a_i, \qquad && i = 1, \ldots, m \end{aligned} \tag{9}$$

where $\nu \in [0,1]$ can be chosen constant (uniform arithmetical crossover) or variable over the number of iterations (non-uniform arithemical crossover). The arithmetical crossover operator generates two children. Depending on the choice of parameter $\nu$, they only take values in the parents' interval $[a_i, b_i]$. Since the childrens' position is relatively close to their parents, it enhances exploitation. The consequence is missing exploration and thus large parts of the search space remain unconsidered.

### 4.2 Non-uniform mutation

The purpose of the mutation operator is to introduce some extra variability into the population. Several types of mutation operators have been developed (see e.g. Davis (1991), Michalewicz (1996), Mühlenbein & Schlierkamp-Voosen (1993), Voigt & Anheyer (1994)). An overview with various examples of simulations can be found in Herrera, Lozano & Verdegay (1998), and Michalewicz (1996). In our genetic algorithm we use the *non-uniform mutation operator* presented by Michalewicz (1996).

For every gene of a string we generate a random number $r_{gene} \in [0,1]$ and compare $r_{gene}$ with a default probability $p_m$. If $r_{gene} < p_m$, the gene mutates, i.e. it changes its value. Suppose we have a string $(a_1 \ldots a_i \ldots a_m)$ of length $m$ and randomly select the gene $a_i$ for the application of the non-uniform mutation operator. Then we get a vector $(a_1 \ldots a_i' \ldots a_m)$ where

$$a_i' = \begin{cases} a_i + (l_{up} - a_i)(1 - r^{(1-\frac{t}{T})^b}) & if \quad \tau = 0 \\ a_i - (a_i - l_{lo})(1 - r^{(1-\frac{t}{T})^b}) & if \quad \tau = 1. \end{cases} \tag{10}$$

Here $\tau$ is a random number which may have a value of zero or one, $r \in [0,1]$ is an uniform random number, $T$ is the maximum number of generations and $b$ is a user-dependent system parameter which determines the degree of non-uniformity. The function

$$g(t) = (1 - r^{(1-\frac{t}{T})^b}) \tag{11}$$

yields values in the intervall $[0,1]$.

We can distinguish between two extreme cases (compare also Figure (3)): If the generation number $t$ is small, the exponent in (11) yields a value close to one and thus $g(t)$ is primary influenced by a suitable choice of the random number $r$. Because the random number $r$ is uniformly distributed each value $g(t)$ can be (approximatively) accepted with the same probability. Hence each $a_i'$ in (10) has nearly the same probability to be taken. On the other side, if the generation number $t$ becomes large, $g(t)$ in (11) obtains values close to zero for a wide range of random numbers. Thus there is a tendency that the offspring $a_i'$ in (10) is close to its parent $a_i$. In summary the genetic algorithm initially explores the whole search space right of the parents' interval uniformly for an accurate $a_i'$. However at a later stage of the algorithm, we primarily prefer those $a_i'$ which are close to their parent $a_i$.

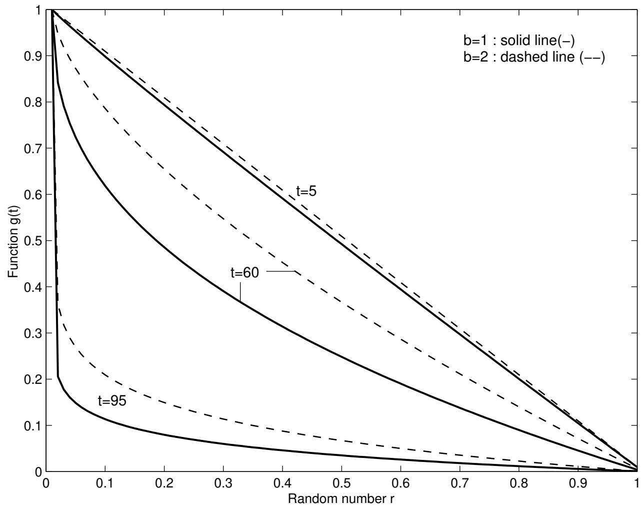


**Figure 3.** *Here is shown the function $g(t)$ in subject to a uniform distributed random number $r$ for three sizes of the generation number $t$. For each generation number $t$ we have plotted two curves with different parameters $b$. The larger parameter $b$, the lower is the degree of non-uniformity.*

### 4.3 Sampling methods

Two important issues exist in genetic search: on the one hand we need several distinguishable strings (that means a large *population diversity*) to search successfully for a global optimum in the search space (exploration). On the other hand a suitable selection of promising strings (we denote this as increase of *selective pressure*) yields a faster convergence of the genetic search (exploitation). These factors are linked closely because an increase of selective pressure decreases the diversity of population and vice versa. Hence strong selective pressure supports premature convergence in a local optimum while a weak selective pressure can make the search ineffective. For a suitable balance we need sampling methods which try to select accurate strings of a population at each iteration step of the algorithm.

In the literature there are many suggestions of sampling methods. The most famous methods are probably *stochastic universal sampling* (Baker, 1987), *rank-based* techniques (Baker (1985), Whitley (1989)) and *tournament selection* (Goldberg, Deb & Korb (1991)). Here we introduce a new modification of the stochastic universal sampling (Baker (1985)). Our modified selection procedure (*modSP*), which includes crossover- and mutation operators, consists of six steps and is illustrated in Figure 4:

- ***Step 1:*** *Suppose that a population $P(t)$ is generated in iteration step $t$. Then delete the worst $u$ percent strings of $P(t)$.*
- ***Step 2:*** *From the remaining strings of step 1 randomly select $r$ strings, which do not necessarily have to be distinct.*
- ***Step 3:*** *From the remaining strings of step 1 randomly select $s$ parent strings. These have not to be distinct from the $r$ selected strings in step 2.*
- ***Step 4:*** *If strings are equal the copies will be mutated. How many genes of a string are randomly selected and mutated is controlled by the probability $p_m > 0$ (at least one gene is mutated). After mutation, there are $r$ different strings. This operation will also be executed for the $s$ parent strings.*

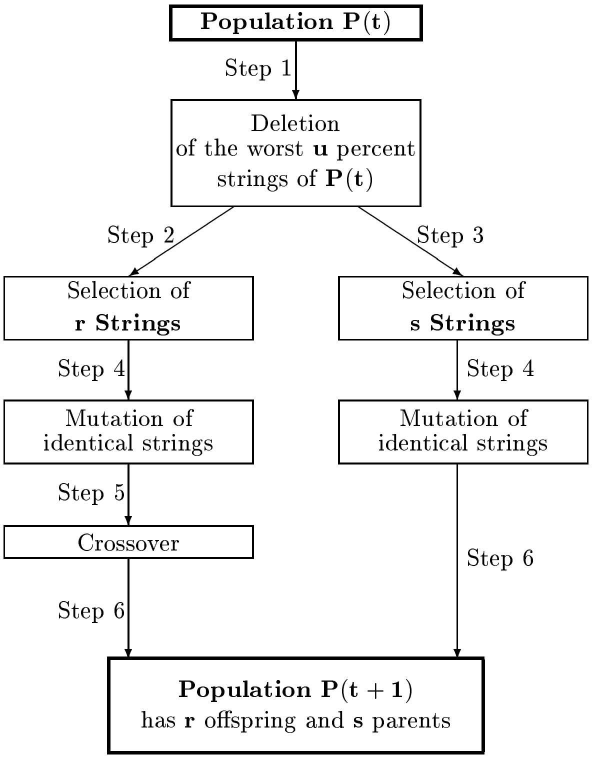


**Figure 4.** *Structure of the modified selection procedure (modSP) given as a flowchart. Details in the text.*

***Step 5:*** *Controlled by the crossover probability $p_c$, apply a crossover operator to the set of the $r$ (distinct) strings and generate $2 \leq i \leq r$ new strings.*

***Step 6:*** *Let $r$ offspring and $s$ parent strings form the new population $P(t+1)$.*

The selection in step 2,3 and 5 is implemented with respect to a probability distribution based on the strings' fitness. The probability for every string to be selected is calculated as follows:

(i) Calculate the fitness value $fit(s_i)$ for every string $s_i$, $i = 1, \ldots, popsize$. The fitness values are calculated by the improved AIC-criterion (section 3). Fitness and AIC-criterion are connected by the mathematical transformations described above.

(ii) Determine the total fitness of the population

$$F = \sum_{i=1}^{pop\_size} fit(s_i) \ .$$

(iii) Calculate the probability $p_i$ and the cumulative probability $q_i$ of a selection for each string $s_i$, $i = 1, \ldots, popsize$ by

$$p_i = \frac{fit(s_i)}{F} \ , \qquad q_i = \sum_{j=1}^{i} p_j \ .$$

To select a single string for the new population, the user first needs to generate *popsize* random (float) numbers $r_i \in [0,1]$ and then check for every $r_i$, $i = 1, \ldots, popsize$:

- If $r_i \leq q_1$ then select the first string $s_1$.
- If $r_i \geq q_i$ then select the $j$-th string $s_i$ such that $q_{j-1} < r_i \leq q_j$ where $j = 2, \ldots, popsize$.

Hence more fit strings have a larger probability to be chosen compared to the less fit strings.

Figure 4 presents the genetic algorithm, used in the simulation studies (section 6). Therefore our genetic algorithm has several build-in steps, which increase the effectiveness compared with many other conventional genetic algorithms:

- Deletion of a default number of worse strings in population $P(t)$ limits the available group of strings for future iterations and thus the selective preassure is high.
- Strings of a population $P(t)$ which have high fitness will enter the new population $P(t+1)$ either as offspring (step 2) or as parent (step 3) with high probability. With step 3 the best solutions of the old population are not forgotten.
- Exact copies of strings are not allowed. Hence there is no danger that a few strings (we call them *super-individuals*) generate many equal copies and thus repress other less fit strings. Mutation of some genes yields new strings of different genotype. The size of a string which will be mutated (and hence the size of lost original information) is controlled by the probability $p_m$.

  Prevention of several equal strings improves the diversity of a population. There will be only a slight increase of selective pressure if we change the genotype of a string by controlled mutation (because most strings maintain their original information).
- The classical mutation-step (Michalewicz (1996)) is canceled. Instead, only step 4 will prevent equal strings.

For termination condition we calculate the average of the $num \in \{2, \ldots, popsize\}$ fittest strings of each population. If the fitness does not change during a default number, $term \in \{2, \ldots, T\}$ of successive iterations ($T$ = maximal iteration number) the genetic algorithm is terminated. All simulations in section 6 have $num = 10$ and $term = 20$.

## 5 Alternative Approaches

This section briefly describes alternative approaches to estimate functions and to select smoothing parameters which are compared to the present approach. The basis of all approaches is the expansion in basis functions with the predictor term

$$\eta(x_i) = \beta_0 + \sum_{j=1}^{p} \sum_{\nu=1}^{K_j} \beta_{j\nu} \phi_{j\nu}(x_{ij}). \tag{12}$$

### 5.1 Mixed models

An approach based on the methodology of mixed models has been used by Parise, Wand, Ruppert & Ryan (2001). The basic concept is to treat the parameters in (12) as random effects. With respect to that strategy and in the context of additive models with truncated power series $\phi_{j\nu}(x) = (x - \xi_{j\nu})_+$ as basis functions one can assume that

$$\beta_{j\nu} \sim \mathcal{N}(0, \sigma_j^2), \qquad \nu = 1, \ldots, K_j,$$

with $\beta_{j1}, \ldots, \beta_{jK_j}$ are independent. The $\sigma_j^2$ express the variability of the parameters: $\sigma_j^2 = \infty$ corresponds to the unrestricted case whereas $\sigma_j^2 \to 0$ implies high restrictions on the parameters. The estimation of structural and smoothing parameters is based on solving the generalized mixed models equation. In the simulation study we assume that adjacent weights of the used truncated power series basis are correlated with a first order autoregressive (AR(1)) structure whose parameter are automatically estimated by the used software package SAS.

### 5.2 Bayesian P-splines

A fully Bayesian approach has been used by Lang & Brezger (2003). In a similar way as in the mixed model approach, the parameters are considered as random. In context of basis functions as B-splines one assumes prior distribution on the parameters. This may be considered as the stochastic analogue to the use of a penalty term in the estimation procedure. For the first order differences one assumes diffuse priors for $\beta_{j1}$ and a first order random walk $\beta_{j\nu} = \beta_{j,\nu-1} + u_{j\nu}$ with Gaussian errors $u_{j\nu} \sim \mathcal{N}(0, \sigma_j^2)$. For full Bayesian inference, hyperpriors are assigned to the parameters $\sigma_j^2$, using highly dispersed inverse Gamma priors, $p(\sigma_j^2) \sim \mathcal{IG}(a_j, b_j)$ with $a_j, b_j$ fixed. Lang & Brezger (2003) use $a_j = 1, b_j = 0.005$. Inference is based on Markov Chain Mote Carlo (MCMC) simulation techniques.

If one uses B-splines which may be constructed from the truncated power series the assumption of independent random effects is replaced by assuming that differences of parameters are normally distributed. In the simpler case one assumes $\beta_{j,\nu+1} - \beta_{j\nu} \sim \mathcal{N}(0, \sigma_j^2)$.

### 5.3 Relevance Vector Machine

The relevance vector machine (Tipping (2000), Tipping (2001)) has been developed in the machine learning community as an improvement of the support vector machine. Tipping also uses a Bayesian framework. Starting with one basis function at each observation the weights $\beta_{j\nu}$ are independent and normally distributed, $\beta_{j,\nu} \sim \mathcal{N}(0, \alpha_i^{-1})$ where the hyperpriors for $\alpha_i$ and $\sigma^2$ are gamma distributions which are optimized by a marginal likelihood approach. The essential difference between Tipping's algorithm and Bayesian approaches is that the number of basis functions initially is equal to the number of observations. Then it reduces to only few remaining basis functions. For the rest the weights become zero. In our simulation study (section 6.1) we use 40 respectively 80 Gaussian kernels as basis functions with different $\sigma_g$, in detail $\sigma_g = 0.15/\sqrt{2}$ (function with $j = 3$) and $\sigma_g = 0.06/\sqrt{2}$ (function with $j = 6$). The hyperpriors are automatically estimated by the software program SAS.

### 5.4 Adaptive Regression

Friedman (1991) proposed *multivariate adaptive regression splines (MARS)*. MARS uses the expansion in basis functions of the form

$$\eta(x_i) = \beta_0 + \sum_{j=1}^{p} \beta_j \phi_j(x_{ij})$$

in a stepwise way where basis functions are constructed successively from products of linear splines $(x_i - \xi_{i\nu})_+$ , $(\xi_{i\nu} - x_i)_+$ where $x_i$ is one component of the vector $\mathbf{x}$ and $\xi_{i\nu}$ are knots which are chosen from the observation of the corresponding component of $\mathbf{x}$. By stepwise inclusion of linear splines a large model (we use a maximum number of 150 basis functions) is obtained for which a backward deletion procedure is often applied. For details see Friedman (1991).

An alternative Bayesian procedure applying adaptive regression splines has been proposed by Biller & Fahrmeir (2002). Like other Bayesian approaches this procedure is not a stepwise approach and is based on a large set of basis functions like B-splines, which are characterized by candidate knots for each variable. In addition to the parameters, the number of knots as well as the specific choice of knots are specified by prior distributions. For estimations, the number of knots are Poisson distributed with a mean number of knots = 20.

### 5.5 Smoothing parameter selection with S-Plus-software

The software package S-Plus offers a restricted possibility of smoothing parameter selection. First one calculates the AIC-criterion for an initial model. Then one has to specify a list with other modelling alternatives. Each covariate can be dropped or integrated in a model as a linear term respectively as a B-spline with a default penalty term. Therefor Eilers & Marx (1996) published a S-Plus-function which allows the expansion of each function $f_j, j = 1, 2, \ldots$ in B-splines with penalty term. Starting with the initial model the implemented function `step` successively calculates the AIC-criterion for all alternative models. If a current model yields a better AIC-value we replace the previous model. Because of its implementation S-Plus can only run a relatively small number of different models. In the simulation study of section 6.2 it has been shown, that for an additive model with 5 functions $f_j, j = 1, \ldots, 5$, where each one is expanded in 20 B-splines, we can select a list from about 17 models (i.e. each covariate can be modelled linearly or as a B-spline with one of 16 different smoothing parameters). However, to present results of the popular statistical software tool we had to make assumptions concerning the choice of smoothing parameters. Hence a grid of 16 smoothing parameters was log-spaced between $10^{-2}$ and $10^{2}$, i.e. their base-10 logarithms were equally spaced between $-2$ and 2. It is obvious that this discrete and restricted smoothing parameter selection yields inexcact solutions and that the optimal choice of smoothing parameters is very rare. Furthermore, we usually do not know the function's true structure and hence the above restrictions are in common not supportable.

### 5.6 Smoothing parameter selection with R-software

The statistic software package `mgcv` (Wood (2001)) running in R yields an automatic smoothing parameter selection, which is based on a method first proposed by Gu & Wahba (1991). The idea is to re-write the multiple smoothing parameter model fitting problem with an extra "overall" smoothing parameter controlling the tradeoff between model fit and overall smoothness. The retained smoothing parameters now control only the relative weights given to the different penalty terms. Then, the approach is to alternate the following steps:

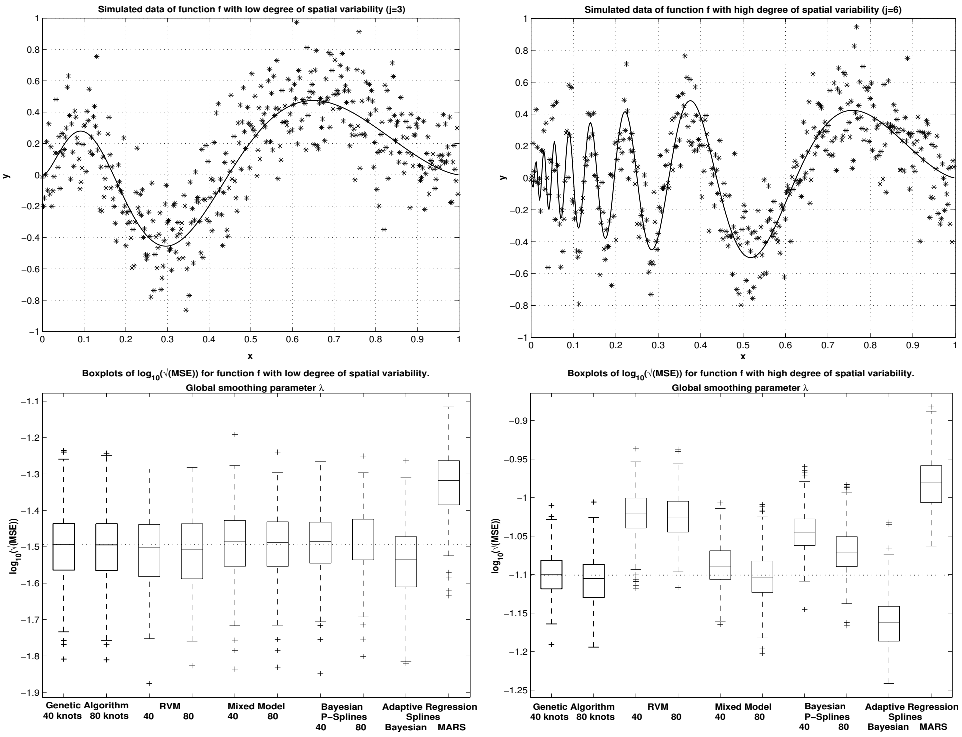


**Figure 5.** *The panels at the top show the true functions with different spatial variability (solid line) for a randomly chosen data set with* $\sigma = 0.2$. *The panels below show boxplots of* $log_{10}(\sqrt{MSE})$ *for various estimators for* $j = 3$ *(left) and* $j = 6$ *(right). The dotted line represents the median of the genetic algorithm with* 40 *knots.*

- Estimation of the overall smoothing parameters using one-dimensional direct search methods.
- Update the relative smoothing parameters simultanously by using the Newton method.

The approach bases on minimizing the Generalized Cross Validation (GCV) as model selection criterion. In the simulation studies we use cubic B-splines whereby the number of knots can be adjusted by hand. For further details see Wood (2000) and Wood (2001).

## 6 Simulations

In the following simulation study the performance of the approach for estimating the smoothing parameters with a genetic algorithm is compared with other related methods in literature. Program packages for the methods in section 5 often do not use the additive structure. Thus these packages are compared in section 6.1 for the single covariate case with functions of rather different spatial variability. The simulations used in this section also show once more the quality of the imposed arithmetical crossover operator (IAC) as described in chapter 4. Section 6.2 compares our approach with other methods in literature by means of simulated additive model.

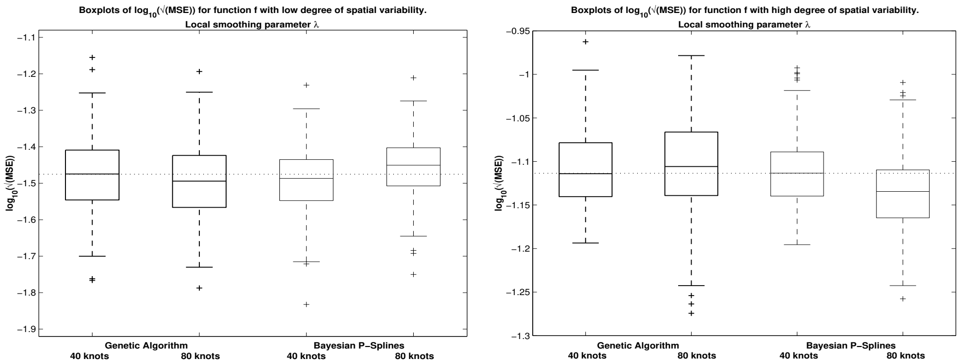


**Figure 6.** *The figures show boxplots of* $log_{10}(\sqrt{MSE})$ *of the genetic algorithm and Bayesian P-splines with local smoothing parameters for* $j = 3$ *(left) and* $j = 6$ *(right). The dotted line represents the median of genetic algorithm with* 40 *knots.*

### 6.1 Estimation of different oscillating functions

In our first simulation study we consider the function

$$f(x) = \sqrt{x(1-x)} \sin\left(\frac{2\pi(1 + 2^{(9-4j)/5})}{x + 2^{(9-4j)/5}}\right).$$

The spatial variability of $f(x)$ can be changed by the parameter $j = 1, 2, \ldots$ (see Ruppert & Carroll (2000)). We simulate 250 data sets for low spatial variability ($j = 3$) or for high spatial variability ($j = 6$), respectively. Each data set consists of 400 independent and uniformly distributed data with $\sigma = 0.2$ (see Figure 5).

For estimating the function, $f(x)$ is expanded in 40 (respectively 80) cubic B-spline basis functions. As penalty we use the third order differences of adjacent coefficients and the smoothing parameter chosen from the interval $[10^{-4}, 10^{4}]$. The default parameters of the used genetic algorithm are: population size (*popsize*) $= 48$ strings, crossover probability $p_c = 0.5$, mutation probability $p_m = 0.25$, deletion of $u = 60$ percent of the worst strings, selection of $r = 30$ and $s = 18$ strings, $\nu = 0.5, T = 1000$ and $b = 1$.

To compare our approach with other methods we computed $log_{10}(\sqrt{MSE})$ with empirical mean squared error given by $MSE(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - \hat{f}(x_i))^2$. For both specifications of spatial variability ($j = 3, 6$) Figure 5 shows boxplots of $log_{10}(\sqrt{MSE})$ for various estimators. Here only one global smoothing parameter is used. From left to right the boxplots refer to genetic algorithm (40 and 80 knots), Relevance Vector Machine (RVM, 40 and 80 knots), mixed model (40 and 80 knots), Bayesian adaptive regression splines and MARS. For better comparison the dotted line represents the median of genetic algorithm with 40 knots.

From Figure 5 we can draw the following conclusions:

- For $j = 3$ most approaches yield similar results. The MARS approach leads to substantial poorer results than all the other methods. For $j = 6$ the performance strongly depends on the method. While mixed models approximately yield the same results as the genetic algorithm, RVM, Bayesian P-splines and MARS show poorer results. Only Bayesian adaptive regression splines lead to better results compared to the genetic algorithm.
- For $j = 3$ doubling the number of basis functions from 40 to 80 knots there are scarcely improves the performance of the estimators if we double. In case

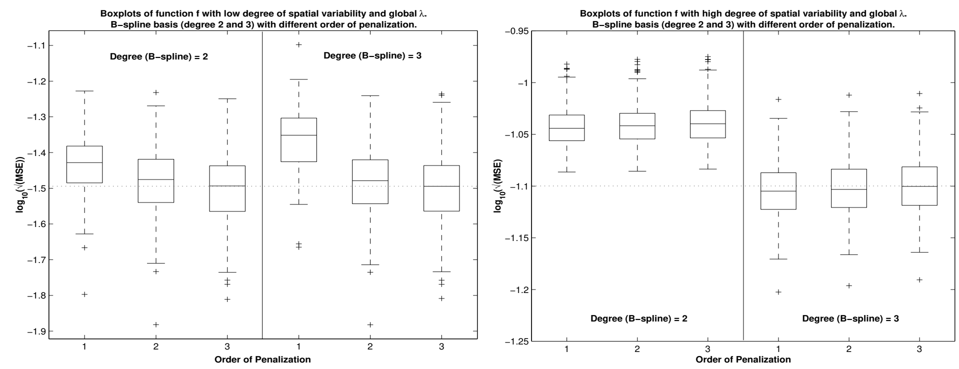


**Figure 7.** *Results for the genetic algorithm with global smoothing parameters but different degrees of B-spline and penalty order for the functions $j = 3$ (left) and $j = 6$ (right).*

of Bayesian P-splines even show a slight deterioration. For $j = 6$ an increasing number of basis functions yields better estimators on average. But the magnitude of the improvements depends on the type of the approache.

For both specifications ($j = 3, 6$) Figure 6 shows the boxplots $log_{10}(\sqrt{MSE})$ of the simulation used above (i.e. equal data sets and default parameters of the genetic algorithm). But now we use local smoothing parameters in the interval $[10^{-4}, 10^{4}]$.

For direct comparison we present the results of our approach and Bayesian P-splines:

- Both specifications ($j = 3, 6$) yield comparably good estimators.
- The use of genetic algorithms with local smoothing parameters shows no improvements compared to genetic algorithms with global smoothing parameters.

To gain better insight into the estimation by means of genetic algorithms with global and local smoothing parameters, we have a look at Figure 7. For both specifications of spatial variability, the simulation was run with 40 B-splines for quadratic (degree $= 2$) and cubic (degree $= 3$) B-splines and different penalty (order $= 1, 2, 3$). The dotted line represents the median of cubic B-splines with penalty order 3 which we used in simulations above.

- For $j = 3$ an increasing penalty order improves the quality of the estimator. Apart from penalty order 1, the degree of B-splines has little influence.
- For $j = 6$ the cubic boxplots show more accurate estimators compared with the quadratic boxplots. In contrast to $j = 3$ the estimators become worse with increasing penalty order.

These results confirm, that B-spline degree and penalty order affect the quality of an estimator. For different oscillating functions, however, choosing a suitable degree of B-splines and penalty order becomes more difficult. For example, our choice of cubic B-splines with penalty order 3 is suitable for low spatial functions. But a lower penalty order seems to be more accurate for highly oscillating functions. Comparable simulations with local smoothing parameters yield similar results as described in Figure 7.

In general, the underlying functions (e.g. in an additive model) are completely unknown and thus we have no idea about the degree of spatial variability. The choice of cubic B-splines with penalty order 2 or 3 should be an adequate solution for all kinds of functions.

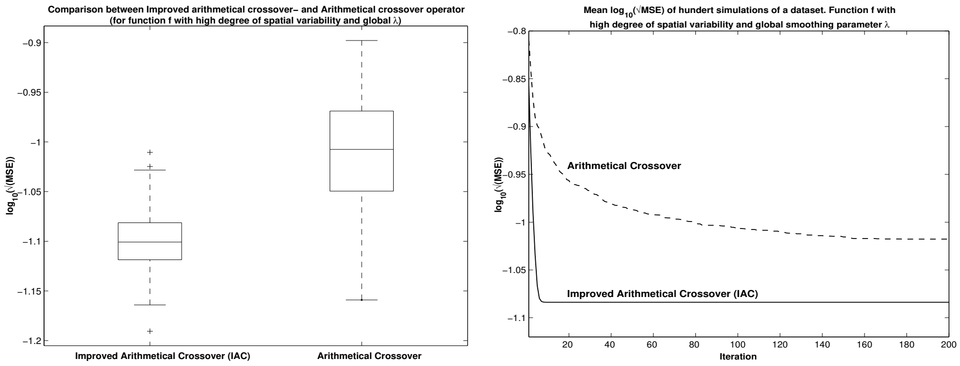


**Figure 8.** *The left figure shows boxplots of $log_{10}(\sqrt{MSE})$ of the genetic algorithm with improved arithmetical crossover and arithmetical crossover. We used 100 data sets of the high oscillating function ($j = 6$) and a global smoothing parameter. The right figure presents the mean $log_{10}(\sqrt{MSE})$ of the data sets for both crossover operators.*

Again we briefly refer to the *improved arithmetical crossover (IAC)*-operator presented in section 4.1. For the simulations above we compare our new operator with the arithmetical crossover operator (section 4.1).

Figure 8 (left) shows boxplots of $log_{10}(\sqrt{MSE})$ with high spatial variability and global smoothing parameters for the simulation described above. The IAC-operator yields estimators which are significantly better than the arithmetical crossover. Moreover, the IAC-operator has faster speed of convergence (see Figure 8 (right)). Here, the mean $log_{10}(\sqrt{MSE})$ of all data sets for both crossover operators is shown. They were chosen by the genetic algorithm up to iteration $t = 200$. In each data set, the current population $P(s = t)$ of iteration $t$ yields the $log_{10}(\sqrt{MSE})$-value only in the case where all former populations $P(s < t)$ have worse fitness. Otherwise, the $log_{10}(\sqrt{MSE})$-value of the former population $P(s = t-1)$ is retained. For the curves in Figure 8 we average across the $log_{10}(\sqrt{MSE})$-values of the 100 data sets and realize a convergence to a minimal value after a few iterations, if using the IAC-operator. However, in general the arithmetical crossover operator does not obtain this minimal value even if we have a larger iteration number.

### 6.2 Estimation for additive models

In this simulation study we choose an additive model, consisting of 5 functions $f_j(x_{ij}), j = 1, \ldots, 5$ (Figure 9). We simulate 250 data sets, where each data set consists of 500 independently and uniformly distributed data with $\sigma_1 = 0.3$ and $\sigma_2 = 0.6$. To estimate the single functions $f_j(x_{ij})$ we expand each function in 20 cubic B-spline basis functions. For penalty we use the third difference of adjacent coefficients. The five global smoothing parameters can be chosen in the interval $[10^{-4}, 10^4]$. The default parameters of the genetic algorithm are the same as in section 6.1.

To compare the results of our approach with other methods we computed $log(MSE)$. Figures 10 and 11 show boxplots for both cases, $\sigma_1 = 0.3$ and $\sigma_2 = 0.6$. The comparison of the estimation performance between the approaches is presented for each single function $f_j$ in an own subplot. The last subplot shows the estimation performance for the total function $f_{total}$ consisting of the 5 components $f_j, j = 1, \ldots, 5$. Each subplot comprises boxplots of the following methods (from left to right): genetic algorithm, R (respectively the R-package "`mgcv`"), S-Plus,

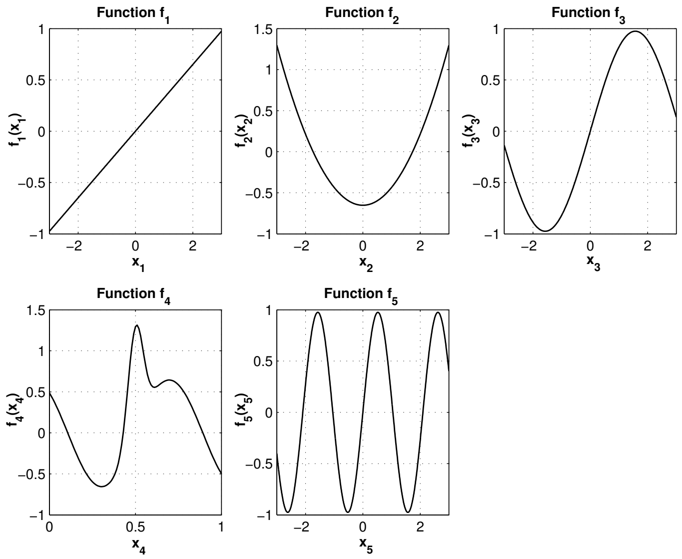


**Figure 9.** *Here the five original functions of the used additive model are shown.*

BayesX and Bayesian adaptive regression splines. For better comparison, the dotted line represents the median of the genetic algorithm.

Figure 10 and 11 show:

- In both cases the linear function $f_1$ is estimated best by S-Plus.
- Function $f_2$ is significantly better estimated for $\sigma_1 = 0.3$ and $\sigma_2 = 0.6$ by the genetic algorithm than all other approaches.
- Apart from S-Plus, in case of $\sigma_1 = 0.3$ function $f_3$ is similarly estimated by all approaches. But in the case $\sigma_2 = 0.6$, the genetic algorithm yields better results compared with the other approaches.
- Bayesian adaptive regression splines significantly yields the best results in estimation of function $f_4$. For $\sigma_1 = 0.3$ the fit of function $f_4$ by the genetic algorithm is worse than that of all other approaches. But for $\sigma_2 = 0.6$ the approaches (except Bayesian adaptive regression splines) yield compareable results.
- Together with S-Plus, the genetic algorithm has the best estimators of function $f_5$ for both specifications ($\sigma_1 = 0.3$ and $\sigma_2 = 0.6$).
- For $\sigma_1 = 0.3$ only S-Plus and Bayesian adaptive regression splines outperform the genetic algorithm in estimation of the total function $f_{total}$. But if we choose $\sigma_2 = 0.6$, the genetic algorithm better estimates $f_{total}$ compared with all other approaches.

The simulation study shows that the results for the estimation of function $f_{total}$ by Bayesian adaptive regression splines are closely connected to the fit of function $f_4$. Although this approach yields average results for all other functions, the excellent estimation of $f_4$ strongly influences the quality of the total function $f_{total}$. The reason is that variable knot selection of Bayesian adaptive regression splines adapt to the different spatial variability of function $f_4$. We notice again that only strict constraints of the smoothing parameter choice (section 5.5) lead to the results of the S-Plus approach. Furthermore, the results of function $f_1$ show the advantage of S-Plus to estimate the function $f_1$ by linear terms.

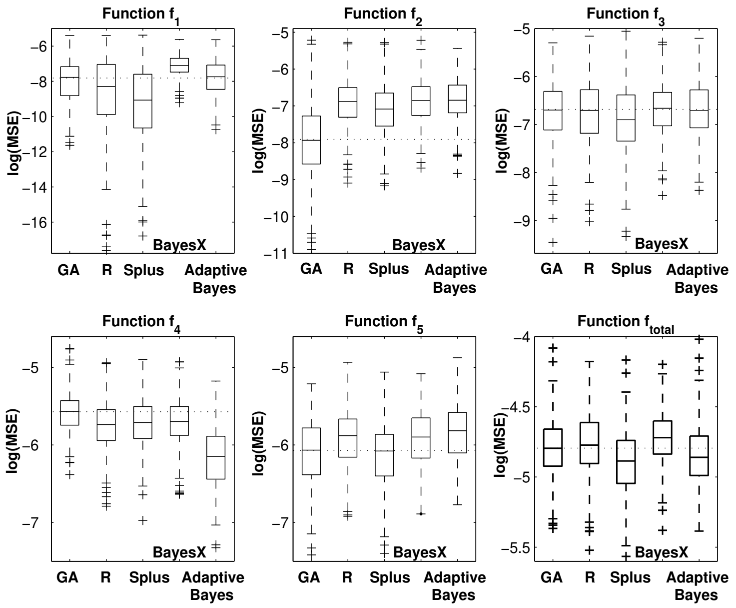


**Figure 10.** *Here boxplots of $log(MSE)$ for the functions $f_j, j = 1, \ldots, 5$ and the total function $f_{total}$ of several estimation approaches with $\sigma_1 = 0.3$ are shown. Details in the text.*

Table 1 shows the average running times per data set of the used programs in the

| Program | Running time in seconds |
|---|---|
| Genetic algorithm | 337 |
| R-package mgcv | 3 |
| S-Plus | 468 |
| BayesX | 49 |
| Adaptive Bayes | 394 |

**Table 1.** *Here average running times for estimation of one data set in the simulation study of an additive model with $\sigma_1 = 0.3$ are shown.*

simulation study (additive model with $\sigma_1$=0.3). S-Plus, BayesX and the R-package mgcv are commercial software and hence optimized with respect to running time. The genetic algorithm and the adaptive Bayesian algorithm are primarily research tools, programmed for comparison of results and thus not optimized for running time. Hence a direct comparison is difficult. But Table 1 shows one obvious fact: the commercial software S-Plus uses much more running time to yield accurate results than all other programs.

## 7 Conclusions

We have presented a new automatic procedure for smoothing parameter choice based on genetic algorithms. In various simulation studies we compared the performance of our approach to other parametric and nonparametric methods given in the literature. The main focus was on estimation of functions with additive models. The simulation studies show that the results between our approach and the

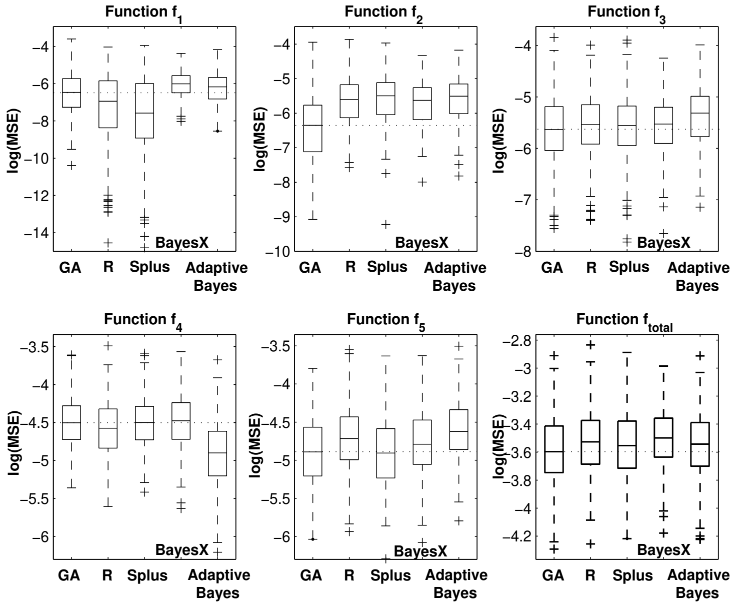


**Figure 11.** *Here boxplots of* $log(MSE)$ *for the functions* $f_j, j = 1, \ldots, 5$ *and the total function* $f_{total}$ *of several estimation approaches with* $\sigma_2 = 0.6$ *are shown. Details in the text.*

other methods are comparable in most cases. Furthermore the genetic algorithm outperforms the other methods in some simulations of section 6.

This paper only refers to models with one-dimensional- or additive structure and uniformly distributed data. Thus it was possible to use several software programs from literature to rank the quality of the new approach. In future we will analyze the genetic algorithm for smoothing parameter choice in models with interactions and other distributions of the reponse.

Finally the question arises whether a genetic algorithm with adaptive choice of knots may further improve Bayesian adaptive regression splines which already yielded promising results in several simulation studies.

## Acknowledgement

We thank Stefan Lang, Thomas Kneib and David Rummel for assistance and the supply of some simulation studies we have used for comparison.

# Appendix

## A Penalized regression splines with constraints

For a detailed derivation of (6) we again start with the penalized residual sum of squares criterion (pRSS)

$$\min_{\tilde{\beta}} \left\{ \sum_{i=1}^{n} (y_i - \tilde{\beta}_0 - \sum_{j=1}^{p} \sum_{\nu=1}^{K_j} \tilde{\beta}_{j\nu} \tilde{\phi}_{j\nu}(x_{ij}))^2 + \sum_{j=1}^{p} \sum_{\nu=k+1}^{K_j} \lambda_{j\nu} (\Delta^k \tilde{\beta}_{j\nu})^2 \right\} \tag{13}$$

where $\lambda_{j\nu} \geq 0,\ j = 1, \ldots, p,\ \nu = k+1, \ldots, K_j, k = 1, 2, \ldots,$ are local smoothing parameters, $\tilde{\beta}_{j\nu}$ are unknown coefficients and $\Delta^k \tilde{\beta}_{j\nu}, k = 1, 2, \ldots$ is the $k$th difference of adjacent coefficients. Writing (13) in matrix form we obtain

$$pRSS(\mathbf{\Lambda}) = (\mathbf{y} - \tilde{\mathbf{B}}\tilde{\boldsymbol{\beta}})^T (\mathbf{y} - \tilde{\mathbf{B}}\tilde{\boldsymbol{\beta}}) + \tilde{\boldsymbol{\beta}}^T \tilde{\mathbf{D}}^T \mathbf{\Lambda} \tilde{\mathbf{D}} \tilde{\boldsymbol{\beta}}. \tag{14}$$

Calculation of the first derivative and set the expression to zero yields an estimator for $\tilde{\boldsymbol{\beta}}$

$$\hat{\tilde{\boldsymbol{\beta}}}(\mathbf{\Lambda}) = (\tilde{\mathbf{B}}^T \tilde{\mathbf{B}} + \tilde{\mathbf{D}}^T \mathbf{\Lambda} \tilde{\mathbf{D}})^{-1} \tilde{\mathbf{B}}^T \mathbf{y}. \tag{15}$$

and thus an estimator for $\mathbf{y}$ is given by

$$\hat{\mathbf{y}} = \tilde{\mathbf{B}} \hat{\tilde{\boldsymbol{\beta}}}(\mathbf{\Lambda}) \tag{16}$$

Here $\tilde{\mathbf{B}}$ is a $n \times (K_1 + \ldots + K_p + 1)$-design matrix

$$\tilde{\mathbf{B}} = [\mathbf{1}, \tilde{\boldsymbol{\phi}}_1, \tilde{\boldsymbol{\phi}}_2, \ldots, \tilde{\boldsymbol{\phi}}_p] = \begin{bmatrix} 1 & \tilde{\phi}_{11}(x_{11}) & \cdots & \tilde{\phi}_{1K_1}(x_{11}) & \tilde{\phi}_{21}(x_{12}) & \cdots & \tilde{\phi}_{pK_p}(x_{1p}) \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & \tilde{\phi}_{11}(x_{n1}) & \cdots & \tilde{\phi}_{1K_1}(x_{n1}) & \tilde{\phi}_{21}(x_{n2}) & \cdots & \tilde{\phi}_{pK_p}(x_{np}) \end{bmatrix}$$

and $\tilde{\mathbf{D}} = diag(\mathbf{0}, \tilde{\mathbf{D}}_1, \ldots, \tilde{\mathbf{D}}_p)$ is a $[(K_1 - k) + \ldots + (K_p - k)] + 1 \times [K_1 + \ldots, K_p] + 1$-penalization matrix of difference order $k$, where each matrix $\tilde{\mathbf{D}}_j$ is of dimension $(K_j - k) \times K_j$. The components in an additive model (1) are not identifiable without further restrictions. A restriction which makes the components unique yields the expression

$$\sum_{\nu=1}^{K_j} \tilde{\beta}_{j\nu} = 0, \qquad j = 1, \ldots, p \tag{17}$$

and thus without loss of generality the last coefficient $\hat{\tilde{\beta}}_{jK_j}$ can be represented by a linear combination of the other coefficients, i.e.

$$\tilde{\beta}_{jK_j} = -\tilde{\beta}_{j1} - \tilde{\beta}_{j2} - \ldots - \tilde{\beta}_{j,K_j-1}, \qquad j = 1, \ldots, p.$$

With regard to this condition we receive from (16) for $y_i, i = 1, \ldots, n$,

$$\begin{aligned}
\hat{y}_i &= \hat{\tilde{\beta}}_0 + \sum_{\nu=1}^{K_1-1} \hat{\tilde{\beta}}_{1\nu} \tilde{\phi}_{1\nu}(x_{i1}) + \hat{\tilde{\beta}}_{1K_1} \tilde{\phi}_{1K_1}(x_{i1}) + \ldots + \\
&\qquad + \sum_{\nu=1}^{K_p-1} \hat{\tilde{\beta}}_{p\nu} \tilde{\phi}_{p\nu}(x_{ip}) + \hat{\tilde{\beta}}_{pK_p} \tilde{\phi}_{pK_p}(x_{ip}) \\
&= \hat{\tilde{\beta}}_0 + \sum_{\nu=1}^{K_1-1} \hat{\tilde{\beta}}_{1\nu} \tilde{\phi}_{1\nu}(x_{i1}) - \sum_{\nu=1}^{K_1-1} \hat{\tilde{\beta}}_{1\nu} \tilde{\phi}_{1K_1}(x_{i1}) + \ldots + \\
&\qquad + \sum_{\nu=1}^{K_p-1} \hat{\tilde{\beta}}_{p\nu} \tilde{\phi}_{p\nu}(x_{ip}) - \sum_{\nu=1}^{K_p-1} \hat{\tilde{\beta}}_{p\nu} \tilde{\phi}_{pK_p}(x_{ip}) \\
&= \hat{\tilde{\beta}}_0 + \sum_{\nu=1}^{K_1-1} \hat{\tilde{\beta}}_{1\nu} \underbrace{\left(\tilde{\phi}_{1\nu}(x_{i1}) - \tilde{\phi}_{1K_1}(x_{i1})\right)}_{\equiv\, \phi_{1\nu}(x_{i1})} + \ldots + \\
&\qquad + \sum_{\nu=1}^{K_p-1} \hat{\tilde{\beta}}_{p\nu} \underbrace{\left(\tilde{\phi}_{p\nu}(x_{ip}) - \tilde{\phi}_{pK_p}(x_{ip})\right)}_{\equiv\, \phi_{p\nu}(x_{ip})}.
\end{aligned}$$

Writing the last expression in matrix notation we obtain

$$\hat{\mathbf{y}} = \mathbf{B}\hat{\boldsymbol{\beta}}(\boldsymbol{\Lambda})$$

where $\mathbf{B}$ is a $n \times [(K_1 - 1) + \ldots + (K_p - 1)] + 1$-design matrix and $\hat{\boldsymbol{\beta}}(\boldsymbol{\Lambda})$ is a $[(K_1-1)+\ldots+(K_p-1)]+1\times 1$-coefficient matrix. Because of the different coefficient vector $\hat{\boldsymbol{\beta}}_j = (\hat{\tilde{\beta}}_{j1}, \ldots, \hat{\tilde{\beta}}_{j,K_j-1}, -\sum_{\nu=1}^{K_j-1} \hat{\tilde{\beta}}_{j\nu})^T$, $j = 1, \ldots, p$, it is necessary to adapt the penalization matrix $\tilde{\mathbf{D}}$. The new blockmatrix $\mathbf{D} = diag(\mathbf{0}, \mathbf{D}_1, \ldots, \mathbf{D}_p)$ has dimension $[(K_1 - k) + \ldots + (K_p - k)] + 1 \times [(K_1 - 1) + \ldots + (K_p - 1)] + 1$. The elements $\mathbf{D}_j, j = 1, \ldots, p$, are computed in the following way:

$$\begin{aligned}
\tilde{\mathbf{D}}_j \hat{\tilde{\boldsymbol{\beta}}}_j &= \begin{bmatrix} \tilde{\mathbf{D}}_{j,\,[(K_j-k)-1\times(K_j-1)]} & \vdots\, \mathbf{0}_{[(K_j-k)-1\times 1]} \\ \cdots\cdots\cdots & \cdots\cdots\cdots \\ \mathbf{0}_{[1\times(K_j-k)-1]} - 1 & \vdots \qquad 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{\tilde{\beta}}_{j1} \\ \vdots \\ \hat{\tilde{\beta}}_{j,K_j-1} \\ -\sum_{\nu=1}^{K_j-1} \hat{\tilde{\beta}}_{j\nu} \end{bmatrix} \\
&= \begin{bmatrix} \tilde{\mathbf{D}}_{j,\,[(K_j-k)-1\times(K_j-1)]} \cdot \hat{\boldsymbol{\beta}}_j \\ -\hat{\tilde{\beta}}_{j,K_j-1} - \sum_{\nu=1}^{K_j-1} \hat{\tilde{\beta}}_{j\nu} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{D}}_{j,\,[(K_j-k)-1\times(K_j-1)]} \\ -\mathbf{1}_{[1\times(K_j-2)]} - 2 \end{bmatrix} \cdot \hat{\boldsymbol{\beta}}_j = \mathbf{D}_j \cdot \hat{\boldsymbol{\beta}}_j
\end{aligned}$$

In the cases we have differences of first ($k = 1$, left matrix below ) or second ($k = 2$, right matrix below) order the matrices $D_j$ have the structure

$$\mathbf{D}_j = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \ldots\ldots & & -1 & 1 \\ -1 & \ldots\ldots & & -1 & -2 \end{bmatrix} \qquad \mathbf{D}_j = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots\ldots & & 1 & -2 & 1 \\ -1 & \ldots\ldots & & -1 & -1 & -2 \end{bmatrix} \tag{18}$$

# References


Baker, J. (1985). Adaptive selection methods for genetic algorithm. In J. Greffenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms*, pp. 101–111. Hillsdale, NJ: Lawrence Erlbaum Associates.

Baker, J. (1987). Reducing bias and inefficiency in the selection algorithm. In J. Greffenstette (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 14–21. Hillsdale, NJ: Lawrence Erlbaum Associates.

Biller, C. and Fahrmeir, L. (2002). Bayesian varying-coefficient models using adaptive regression splines. *Statistical Modelling* **11**, 1–17.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press.

de Boor, C. (1978). *A Practical Guide to Splines*. New York, Heidelberg, Berlin: Springer.

de Boor, C. (1993). B(asic)-spline basics. In L. Piegel (Ed.), *Fundamental Developments of Computer-Aided Geometric Modeling*, pp. 27–49. London: Academic Press.

Chapelle, O. and Vapnik, V. (1999). Model selection for support vector machines. In S. A. Solla, T. K. Leen, & K. R. Müller (Eds.), *Advances in Neural Information Processing Systems 12*. Cambridge, MA: MIT Press.

Darwin, C. (1859). *The origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*. London: Penguin Books.

Davis, L. (1991). *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.

Dierckx, P. (1995). *Curve and Surface Fitting with Splines*. Oxford: Clarendon Press.

Eilers, P. H. C. and Marx, B. D. (1996). Flexible smoothing with b-splines and penalties. *Stat. Science 11*(2), 89–121.

Eshelman, L. and Schaffer, J. (1993). Real-coded genetic algorithms and interval-schemata. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms 2*. San Mateo: Morgan Kaufman Publishers.

Friedman, J. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics 19*(1), 1–141.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E., Deb, K., and Korb, B. (1991). Do not worry, be messy. In R. Belew & L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 24–30. San Mateo, CA: Morgan Kaufmann Publishers.

Gu, C. and Wahba, G. (1991). Minimizing gcv/gml scores with multiple smoothing parameters via the newton methods. *SIAM Journal of Scientific and Statistical Computing* **12**, 383–398.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*. London: Chapman and Hall.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. (2001). *The Elements of Statistical Learning*. New York: Springer.

Haupt, R. L. and Haupt, S. E. (1998). *Practical Genetic Algorithms*. New York: Wiley.

Herrera, F., Lozano, M., and Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review 12*(4), 265–319.

Holland, J. (1975). *Adaption in neural and artificial systems*. Ann Arbor: University of Michigan Press.

Hurvich, C. and Simonoff, J. (1998). Smoothing parameter selection in nonparametric regression using an improved akaike information criterion. *Journal of the Royal Statistical Society B 60*(2), 271–293.

Lang, S. and Brezger, A. (2003). Bayesian p-splines. *Journal of Computational and Graphical Statistics*, to appear.

Marx, B. and Eilers, P. (1998). Direct generalized additive modeling with penalized likelihood. *Computational Statistics and Data Analysis* **28**, 193–209.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Heidelberg: Springer.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge, Massachusetts: MIT Press.

Mühlenbein, H. and Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm 1: Continous parameter optimization. *Evolutionary Computation 1*(1), 25–49.

Parise, H., Wand, M. P., Ruppert, D., and Ryan, L. (2001). Incorporation of historical controls using semiparametric mixed models. *Applied Statistics 50*(1), 31–42.

Radcliffe, N. (1991). Equivalence class analysis of genetic algorithms. *Complex Systems 5*(2), 183–205.

Rawlings, J. O., Pantula, S. G., and Dickey, D. A. (1998). *Applied Regression Analysis*. New York: Springer.

Ruppert, D. and Carroll, R. (1997). Penalized regression splines. *Unpublished manuscript*.

Ruppert, D. and Carroll, R. (2000). Spatially-adaptive penalties for spline fitting. *Australian and New Zealand Journal of Statistics 42*(2), 205–223.

Tipping, M. E. (2000). The relevance vector machine. In T. L. S.A. Solla & K.-R. Müller (Eds.), *In Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* **1**, 211–244.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.

Vapnik, V. (1998). *Statistical learning theory*. New York: John Wiley and Sons.

Voigt, H. and Anheyer, T. (1994). Modal mutations in evolutionary algorithms. *Proc. of the First IEEE International Conference on Evolutionary Computation* **1**, 88–92.

Wand, M. P. (2002). Smoothing and mixed models. *Unpublished manuscript*.

Whitley, D. (1989). The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. Schaffer (Ed.), *Proceedings of* *the Third International Conference on Genetic Algorithms*, pp. 116–121. San Mateo, CA: Morgan Kaufmann Publishers.

Wood, S. (2000). Modelling and smoothing parameter estimation with multiple quadratic penalties. *Journal of the Royal Statistical Society B 62*(2), 413–428.

Wood, S. (2001). mgcv: Gams and generalized ridge regression for R. *R News 1*(2), 20–25.

Wood, S. N. (2002). Thin plate regression splines. *Journal of the Royal Statistical Society B, in press*.

Wright, A. (1991). Genetic algorithms for real parameter optimization. In G. Rawlin (Ed.), *Foundations of Genetic Algorithms 1*. San Mateo: Morgan Kaufman Publishers.