On Algorithms for Ordinary Least Squares Regression Spline Fitting: A Comparative Study*

Thomas C. M. Lee†

October 15, 2001; Revised: January 30, 2002

Abstract

Regression spline smoothing is a popular approach for conducting nonparametric regression. An important issue associated with it is the choice of a "theoretically best" set of knots. Different statistical model selection methods, such as Akaike's information criterion and generalized cross–validation, have been applied to derive different "theoretically best" sets of knots. Typically these best knot sets are defined implicitly as the optimizers of some objective functions. Hence another equally important issue concerning regression spline smoothing is how to optimize such objective functions. In this article different numerical algorithms that are designed for carrying out such optimization problems are compared by means of a simulation study. Both the univariate and bivariate smoothing settings will be considered. Based on the simulation results, recommendations for choosing a suitable optimization algorithm under various settings will be provided.

Keywords: Bivariate Smoothing, Generalized Cross–Validation, Genetic Algorithms, Regression Spline, Stepwise Selection

---

*Running Title: Regression Spline Fitting Algorithms

†Postal: Department of Statistics, Colorado State University, Fort Collins, CO 80523-1877, USA; Email: tlee@stat.colostate.edu; Phone: (970) 491 2185; Fax: (970) 491 7895.

1 Introduction

An increasingly popular approach for performing nonparametric curve estimation is regression spline smoothing. For this approach it is customary to assume that the "true" curve $f(x)$ to be recovered admits the expression

$$f(x) = b_0 + b_1 x + \ldots + b_r x^r + \sum_{j=1}^{m} \beta_j (x - k_j)_+^r, \tag{1}$$

where r is the order of the regression spline (usually chosen *a priori*), k_j is the jth knot, $\boldsymbol{\beta} = (b_0, \ldots, b_r, \beta_1, \ldots, \beta_m)^T$ is a set of coefficients and $(a)_+ = \max(0, a)$. In order to estimate $f(x)$ using (1), one needs to choose the number and the placement of the knots, as well as to estimate $\boldsymbol{\beta}$. As mentioned in Wand (2000), there are two general strategies for carrying out this task. The first strategy is to select a relatively small number of knots and estimate $\boldsymbol{\beta}$ using ordinary least squares (further details will be given below). With this strategy, the choice of the knots is extremely important. Regression spline smoothing procedures following this strategy include Friedman & Silverman (1989), Koo (1997), Kooperberg, Bose & Stone (1997), Lee (2000), Pittman (1999) and Stone, Hansen, Kooperberg & Truong (1997). The second strategy is to use a relatively large number of knots, but do not use ordinary least squares to estimate $\boldsymbol{\beta}$. In contrast to the first strategy, for this second strategy the importance of the choice of knots is relatively minor: the crucial aspect is how $\boldsymbol{\beta}$ is estimated (e.g., by penalized least squares). Recent related references include Denison, Mallick & Smith (1998*a*), DiMatteo, Genovese & Kass (2001), Eilers & Marx (1996), Lindstrom (1999) and Ruppert & Carroll (2000) and Smith & Kohn (1996).

This article focuses on the first strategy. Many regression spline smoothing procedures adopting this strategy are composed of two major components. The first component concerns the use of some sort of statistical model selection principle for *defining* a "theoretically best" set of knots. Quite often, such a "theoretically best" set of knots is defined implicitly as the optimizer of some objective

function. The second component is hence a practical algorithm for performing the corresponding optimization. Typically this optimization is a very hard problem, as (i) the search space is usually huge and (ii) different candidate solutions may have different dimensions. The main purpose of this paper is to provide a study for the performances of some common algorithms that are designed for carrying out such optimization problems. Both the univariate and bivariate settings will be considered (but the question of which is *the* most appropriate principle for defining a "theoretically best" set of knots will not be considered here). Our hope is that, by separating the two issues of "defining the best" and "locating the defined best", and that by performing a focused study on the latter one, useful computational hints can be obtained for reference by future researchers.

Due to the complicated nature of the regression spline optimization algorithms, theoretical comparison seems to be very difficult. Therefore, the study to be presented is entirely based on empirical experiments. Of course it is impossible to exhaust all possible experimental settings, but we shall follow Wand (2000) to use a family approach to alleviate this problem. The idea is to change one experimental factor (e.g., signal–to–noise ratio) at a time so that patterns can be more easily detected.

In (1) it is clear that $f(x)$ is a linear combination of $\{x^j\}_{j=0}^r$ and $\{(x-k_j)_+^r\}_{j=1}^m$. This set of functions is known as the *truncated power basis of degree* r. Other basis functions for regression spline fitting also exist, such as those for B–splines, natural splines and radial basis functions (e.g., see Eilers & Marx 1996 and Green & Silverman 1994). However, as pointed out by Wand (2000), numerical experimental results should not be very sensitive to the choice of basis functions. Therefore for simplicity this article shall concentrate on the truncated power basis.

The rest of this article is organized as follows. In Section 2 further background details on univariate regression spline smoothing and the use of generalized cross–validation (GCV) for defining a "best" estimate will be provided. Then Sections 3 and 4 describe six different optimization algo-

rithms. These six algorithms will be, in Section 5, compared through a simulation study. Finally Section 6 considers the bivariate setting. Conclusions and recommendations for the univariate and the bivariate cases are reported in Sections 5.5 and 6.3 respectively.

2 Regression Spline Smoothing Using GCV

Suppose that n pairs of measurements $\{x_i, y_i\}_{i=1}^n$ satisfying

$$y_i = f(x_i) + \epsilon_i, \qquad \epsilon_i \sim \text{ iid } N(0, \sigma^2),$$

are observed. The goal is to estimate f which is assumed to satisfy (1). In this article it is assumed $r = 3$ and $\min(x_i) < k_1 < \ldots < k_m < \max(x_i)$. Furthermore, following the work of previous authors (e.g., see Friedman & Silverman 1989, Koo 1997 and Smith & Kohn 1996), $\{k_1, \ldots, k_m\}$ is restricted to be a subset of $\{x_1, \ldots, x_n\}$. Such a restriction should not have any serious adverse effect on the quality of the final curve estimate. Since f can be completely specified by $\boldsymbol{\theta} = \{\boldsymbol{k}, \boldsymbol{\beta}\}$, where $\boldsymbol{k} = (k_1, \ldots, k_m)^T$ and $\boldsymbol{\beta} = (b_0, \ldots, b_r, \beta_1, \ldots, \beta_m)^T$, the estimation of f can be achieved via estimating $\boldsymbol{\theta}$. Notice that different estimates $\hat{\boldsymbol{\theta}}$s for $\boldsymbol{\theta}$ may have different dimensions (i.e., different number of parameters). Various methods have been proposed for choosing the dimension of a "best" $\hat{\boldsymbol{\theta}}$. One of the earliest proposals is the use of generalized cross–validation (GCV) (e.g., see Friedman & Silverman 1989, Friedman 1991 and Pittman 1999), in which the "best" estimate of $\boldsymbol{\theta}$, or equivalently, f, is defined as the one that minimizes

$$\text{GCV}(\hat{\boldsymbol{\theta}}) = \frac{1}{n} \sum_{i=1}^{n} \left\{ y_i - \hat{f}(x_i) \right\}^2 \Big/ \left\{ 1 - \frac{d(m)}{n} \right\}^2 . \tag{2}$$

Here $d(m)$ is an increasing function of the number of the knots m. In order to penalize the additional flexibility inherited by the free choice of knot locations, Friedman & Silverman (1989) suggested using $d(m) = 3m + 1$ instead of the conventional GCV choice $d(m) = m + 1$. In the sequel this

GCV choice of $\hat{\boldsymbol{\theta}}$, with $d(m) = 3m + 1$, will be taken as the target that the optimization algorithms should aim at.

Let $\boldsymbol{x} = (x_1, \ldots, x_n)^T$, $\boldsymbol{y} = (y_1, \ldots, y_n)^T$ and $\hat{\boldsymbol{k}} = (\hat{k}_1, \ldots, \hat{k}_{\hat{m}})^T$ be an estimate of $\boldsymbol{k}$. Denote the "design matrix" as $\boldsymbol{X} = (\mathbf{1}, \boldsymbol{x}, \ldots, \boldsymbol{x}^r, (\boldsymbol{x} - \hat{k}_1\mathbf{1})_+^r, \ldots, (\boldsymbol{x} - \hat{k}_{\hat{m}}\mathbf{1})_+^r)$, where $\mathbf{1}$ is a $n \times 1$ vector of ones. If r and $\hat{\boldsymbol{k}}$ are specified beforehand, then the unique maximum likelihood estimate $\hat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ (conditional on r and $\hat{\boldsymbol{k}}$) can be obtained by applying ordinary least squares regression, and admits the closed form expression $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. This means that $\hat{\boldsymbol{\theta}} = \{\hat{\boldsymbol{k}}, \hat{\boldsymbol{\beta}}\}$ is completely determined by $\hat{\boldsymbol{k}}$. Therefore the only effective argument for the minimization of GCV($\hat{\boldsymbol{\theta}}$) (or any other similar criteria) is $\boldsymbol{k}$; i.e., the knots. In the next two sections two classes of knot–based optimization algorithms will be described. Their effectiveness, in terms of minimizing GCV($\hat{\boldsymbol{\theta}}$), will be studied in Section 5 via simulations.

Besides GCV, other model selection principles that have also been applied to derive various "best" knot sets include Akaike's information criterion (AIC) (e.g., Koo 1997 and Kooperberg et al. 1997) and the minimum description length (MDL) principle (e.g., Lee 2000). Some preliminary numerical experiments were also conducted for both AIC and MDL. Empirical conclusions obtained from these experiments are similar to those for GCV. Due to space limitation, results of these experiments are not reported here.

3 Stepwise Knot Selection

The most popular method for searching the minimizer of GCV($\hat{\boldsymbol{\theta}}$) (or any other similar criteria) seems to be stepwise knot selection (e.g., see Friedman 1991, Friedman & Silverman 1989, Hansen, Kooperberg & Sardy 1998, Koo 1997, Kooperberg et al. 1997, Lee 2000 and references given therein). The idea is very similar to stepwise regression for the classical subset selection problem. It begins with an initial model as the current model. Then at each time step a new model is

obtained by either adding a new knot to or removing an existing knot from the current model. This process continues until a certain stopping criterion is met. Amongst all the candidate models that have been visited by the algorithm, the one that gives the smallest value of GCV($\hat{\boldsymbol{\theta}}$) is chosen as the final model. In this article the following four versions of this stepwise method are considered. A comparison of the relative computational speeds amongst these versions is given in Section 5.5.

1. **Forward Addition** (*ForAdd*)**:** Set the initial model as the model with no knots and compute its GCV($\hat{\boldsymbol{\theta}}$) value. Add a new knot to the current model at each time step. The knot is chosen in such a way that, when it is added, it produces the largest reduction (or smallest increase) in the current value of GCV($\hat{\boldsymbol{\theta}}$). Continue this knot addition process until the number of knots added hits a pre–selected limit. Throughout our simulation study reported below, this pre–selected limit was set to $n/3$. Finally, when the knot–adding process finishes, a nested sequence of candidate models are produced and the one that minimizes GCV($\hat{\boldsymbol{\theta}}$) is selected as the final model.

2. **Backward Elimination** (*BackElim*)**:** This version begins with placing a relatively large number of knots in the initial model. One typical strategy for placing these knots is to place a knot at every s (usually $3 \leq s \leq 5$) sorted values of the design points x_i's. In our simulation study we used $s = 3$. Then the algorithm proceeds to remove one knot at a time, until there is no more knots to be removed. At each time step the knot to be removed is chosen in such a way that, when it is removed, it provides the largest reduction (or smallest increase) in the current value of GCV($\hat{\boldsymbol{\theta}}$). Similar to *ForAdd*, at the end of the process a nested sequence of candidate models are obtained. Select the one that gives the smallest GCV($\hat{\boldsymbol{\theta}}$) as the final model.

3. **Addition followed by Elimination** (*AddElim*)**:** This version uses the final model selected

by a first application of *ForAdd* as the initial model for the execution of *BackElim*. The resulting model obtained from such an execution of *BackElim* is taken as the final model.

4. **Elimination followed by Addition** (*ElimAdd*)**:** This version uses the final model selected by a first application of *BackElim* as the initial model for the execution of *ForAdd*. The resulting model obtained from such an execution of *ForAdd* is taken as the final model.

4 Genetic Algorithms

Another class of optimization algorithms that have been applied to regression spline smoothing is genetic algorithms; e.g., see Lee (2002), Pittman (1999), Pittman & Murthy (2000) and references given therein. In below we begin with a brief description of genetic algorithms. As we shall see, an important issue in applying genetic algorithms is how to represent a candidate model as a *chromosome*. Two different chromosome representation methods will be described and compared. For general introductions to genetic algorithms, see for examples Davis (1991), Fogel (2000) and Michalewicz (1996).

4.1 General Description

The use of genetic algorithms for solving optimization problems can be briefly described as follows. An initial set, or population, of possible solutions to an optimization problem is obtained and represented in vector form. Typically these vectors are of the same length and are often called *chromosomes*. They are free to "evolve" in the following way. Firstly parent chromosomes are randomly chosen from the initial population: chromosomes having lower or higher values of the objective criterion to be minimized or maximized, respectively, would have a higher chance of being chosen. Offspring are then reproduced from either applying a *crossover* or a *mutation* operation to these chosen parents. Once a sufficient number of such second generation offspring are produced,

third generation offspring are further produced from these second generation offspring in a similar manner as before. This reproduction process continues for a number of generations. If one believes in Darwin's Natural Selection, the expectation is that the objective criterion values of the offspring should gradually improve over generations; i.e., approaching the optimal value. For the current problem, the objective function is GCV($\hat{\boldsymbol{\theta}}$) and one chromosome represents one $\hat{\boldsymbol{\theta}}$.

In a crossover operation one child chromosome is reproduced from "mixing" two parent chromosomes. The aim is to allow the possibility that the child would receive different best parts from its parents. A typical "mixing" strategy is that every child gene location would have equal chances of either receiving the corresponding father gene or the corresponding mother gene. This crossover operation is the distinct feature that makes genetic algorithms different from other optimization methods.

In a mutation operation one child chromosome is reproduced from one parent chromosome. The child would essentially be the same as its parent except for a small number of genes where randomness is introduced to alter the types of these genes. Such a mutation operation prevents the algorithm being trapped in local optima.

For the reason of preserving the best chromosome of a current generation, an additional step that one may perform is the *elitist* step: replace the worst chromosome of the next generation with the best chromosome of the current generation. Inclusion of this elitist step would guarantee the monotonicity of the algorithm.

4.2 Chromosome Representations

This section describes two methods for representing a $\hat{\boldsymbol{\theta}}$ in a chromosome form: the first method is described in Lee (2002), while the second method can be found in Pittman (1999). First recall that for the current curve fitting problem a possible solution $\hat{\boldsymbol{\theta}}$ can be uniquely specified by $\hat{\boldsymbol{k}}$, and

that $\hat{\boldsymbol{k}}$ is assumed to be a subset of the design points $\{x_1, \ldots, x_n\}$. Thus for the present problem a chromosome only needs to carry information about $\hat{\boldsymbol{k}}$.

Fixed–Length Representation (*GeneFix*)**:** In this representation method all chromosomes are binary vectors with the same length n, the number of data points. A simple example will be used to illustrate its idea. Suppose $n = 15$ and $\hat{\boldsymbol{k}} = \{x_3, x_8, x_{14}\}$; i.e., there are three knots in this candidate model and they are located at x_3, x_8 and x_{14}. If we use "0" to denote a normal gene and "1" to denote a knot gene, then the chromosome for this example is 001000010000010. One advantage of this method is that, it can be easily extended to handle cases when outliers and/or discontinuities are allowed: simply introduces two additional types of genes, outlier genes and discontinuity genes.

Variable–Length Representation (*GeneVar*)**:** For this method the chromosomes are not binary nor having the same length. Here the location of a knot is represented by one integer–valued gene, and hence the length of a chromosome is equal to the number of knots in the candidate model. For the previous example, the corresponding chromosome is 3-8-14 (the hyphens were merely used to separate the genes). Since the chromosomes are generally not of the same length, the optimization is done in the following way. First pre–specify the minimum and the maximum number of knots that any candidate model can have. Denote them as K_{MIN} and K_{MAX} respectively. Then for each $K = K_{\text{MIN}}, \ldots, K_{\text{MAX}}$, apply the algorithm to find the best candidate model amongst only those candidate models that have K knots. That is, for this particular run of the algorithm, all chromosomes are restricted to have the same length K. Then, after the execution of the algorithm for every value of K, a sequence of $K_{\text{MAX}} - K_{\text{MIN}} + 1$ models is obtained. The one that minimizes $\text{GCV}(\hat{\boldsymbol{\theta}})$ will be chosen as the final model.

Please consult Lee (2002) and Pittman (1999) for further details on the implementations of these algorithms.

5 Simulation Study

This section presents results of a series of numerical experiments which were conducted to evaluate the performances of the various optimization algorithms (four stepwise and two genetic) described above. These experiments were designed to study the effects of varying the (i) noise level, (ii) design density, (iii) noise variance function, and (iv) degree of spatial variation.

The simulation was conducted in the following way. For each of the experimental setups described below, 100 artificial noisy data sets, each with 200 design points x_i, were generated. Then, for each of these 100 data sets, the above six algorithms were applied to minimize GCV($\hat{\boldsymbol{\theta}}$) and the corresponding $\hat{f}$ is obtained. For each obtained $\hat{f}$, both the GCV value and the mean–squared error (MSE) value, defined as $\frac{1}{n}\sum_i\{f(x_i) - \hat{f}(x_i)\}^2$, are computed and recorded. Paired Wilcoxon tests were then applied to test if the difference between the median GCV (and MSE) values of any two algorithms is significant or not. The significance level used was $\frac{5}{6}\% = 0.83\%$. If the median GCV (or MSE) value of an algorithm is significantly less than the remaining five, it will be assigned a GCV (or MSE) rank 1. If the median GCV (or MSE) value of an algorithm is significantly larger than one but less than four algorithm, it will be assigned a GCV (or MSE) rank 2, and similarly for ranks 3 to 6. Algorithms having non–significantly different median values will share the same averaged rank. Ranking the algorithms in this manner provides an indicator about the relative merits of the methods (see Wand 2000). Since the main interest of this article is to compare the algorithms in terms of their abilities for conducting numerical minimization, GCV seems to be a more appropriate performance measure here than MSE.

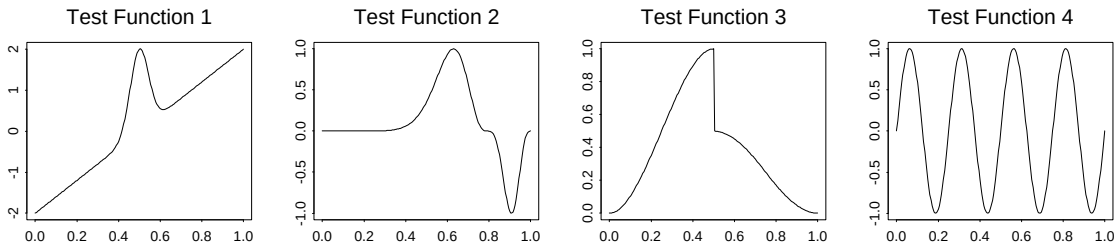

Figure 1: Four Test Functions.

5.1 Varying Noise Level

For this first experiment four test functions were used. They are given in Table 1 and are also plotted in Figure 1. These test functions possess different characteristics and were also used by many previous authors.

The data (x_i, y_i) were generated using $y_i = f(x_i) + e_i$, where f is a test function, x_i is drawn from Unif[0,1], and e_i is a zero–mean Gaussian error with variance σ^2. Three signal–to–noise ratios (SNRs) were used: SNR=2, 4 and 6, where SNR is defined as $\|f\|/\sigma$. Boxplots of the GCV and log(MSE) values for the six algorithms, together with their Wilcoxon test rankings, are given in Figures 2 and 3.

Test Function 1: $$f(x) = (4x-2) + 2\exp\{-16(4x-2)^2\}, \quad x \in [0,1]$$

Test Function 2: $$f(x) = \sin^3(2\pi x^3), \quad x \in [0,1]$$

Test Function 3: $$f(x) = \begin{cases} 4x^2(3-4x), & x \in [0, 0.5) \\ \frac{4}{3}x(4x^2-10x+7) - \frac{3}{2}, & x \in [0.5, 0.75) \\ \frac{16}{3}x(x-1)^2, & x \in [0.75, 1] \end{cases}$$

Test Function 4: $$f(x) = \sin(8\pi x), \quad x \in [0,1]$$

Table 1: Formulae for Test Functions.

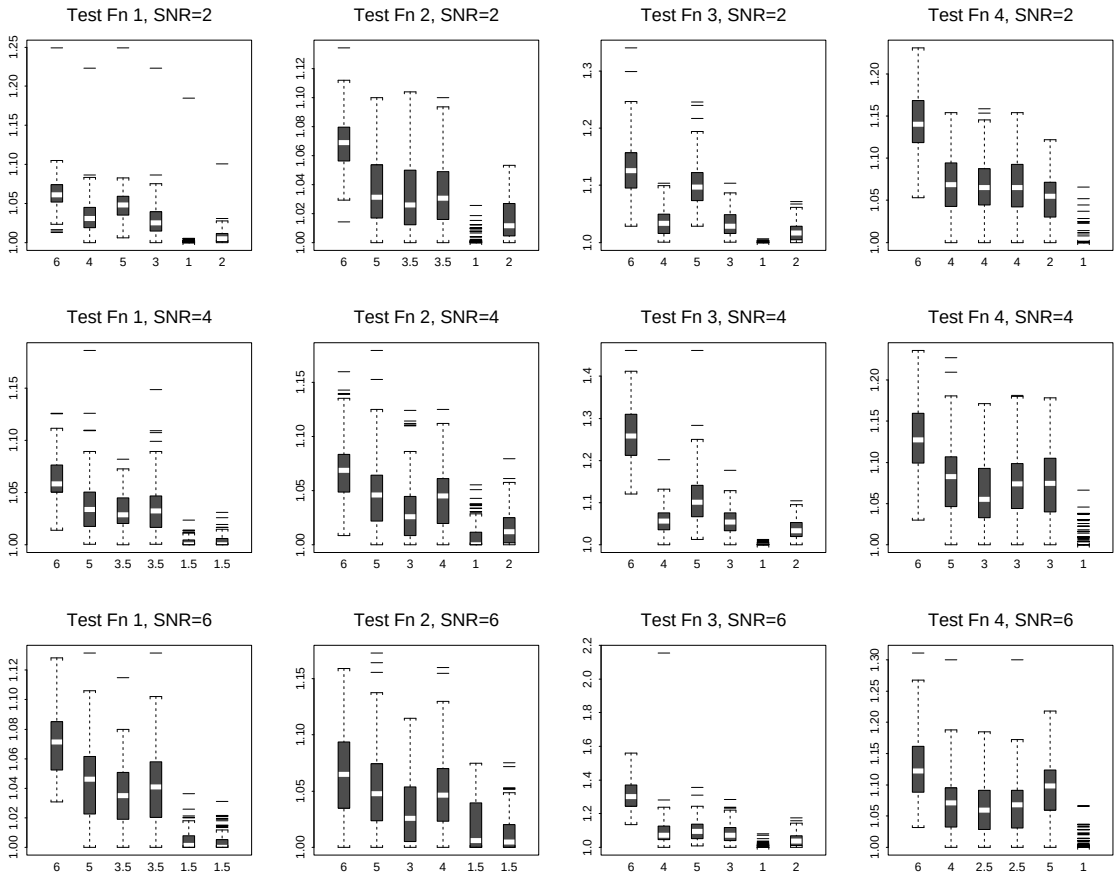

Figure 2: Boxplots of the GCV values for the "Varying Noise Level" experiment. In each panel the boxplots correspond respectively to, from left to right, *ForAdd*, *BackElim*, *AddElim*, *ElimAdd*, *GeneFix* and *GeneVar*. The number below each boxplot is the Wilcoxon test ranking.

5.2 Varying Design Density

The same four test functions as in the *Varying Noise Level* experiment were used. The data (x_i, y_i) were also generated from $y_i = f(x_i)+e_i$ with $\sigma = \|f\|/4$ (i.e., SNR=4), but now the x_i's were drawn from three different densities. The three densities are Beta(1.2, 1.8), Beta(1.5, 1.5) and Beta(1.8, 1.2), and their probability density functions are plotted in Figure 4. Boxplots of the GCV and log(MSE) values, and Wilcoxon test rankings of the algorithms are displayed in Figures 5 and 6 in a similar manner as before.

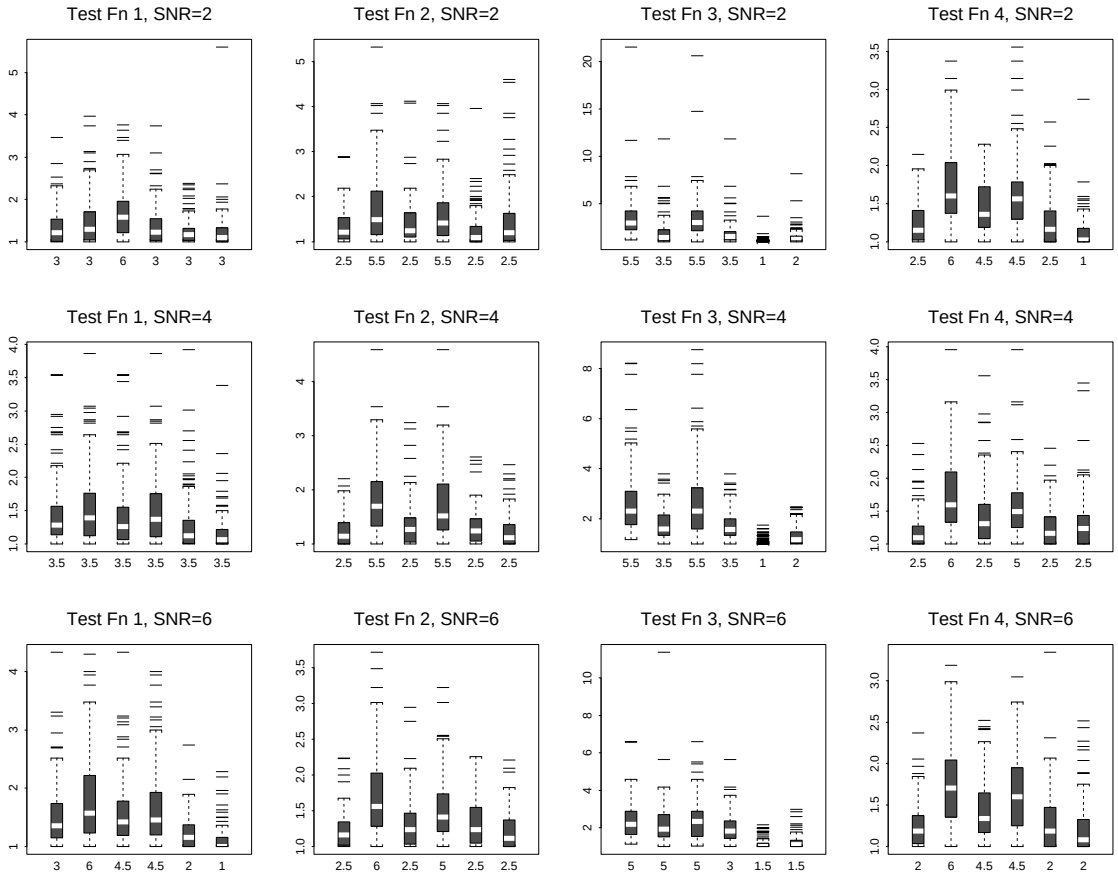

Figure 3: Similar to Figure 2 but for log(MSE).

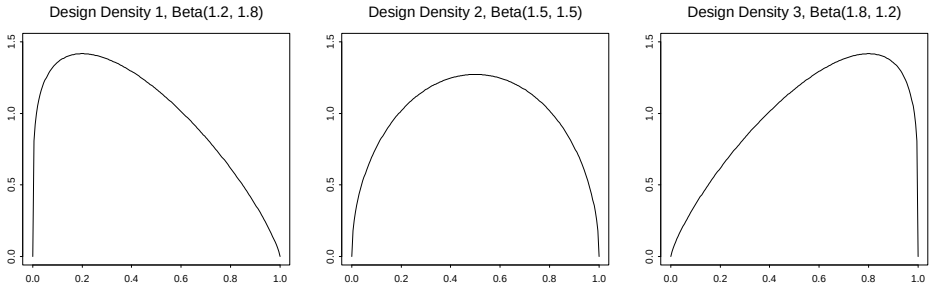

Figure 4: Plots of the three different design densities.

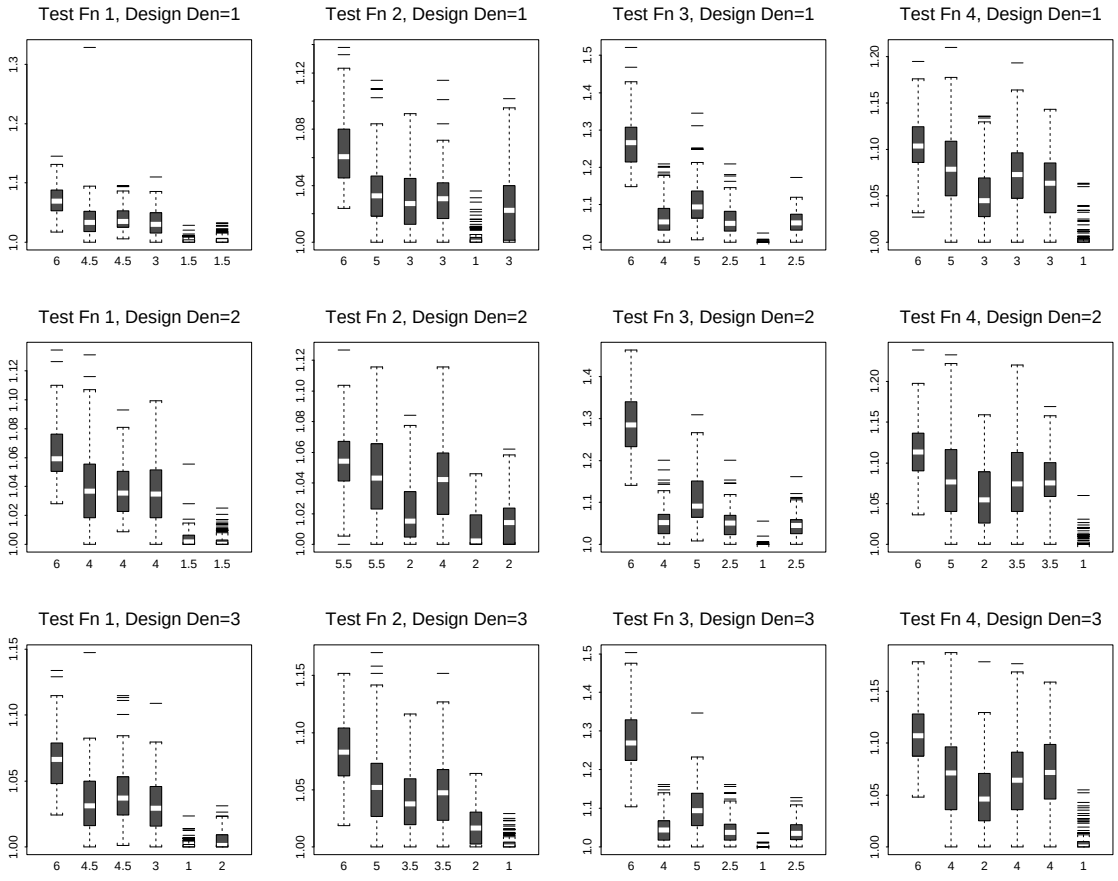

Figure 5: Similar to Figure 2 but for the *Varying Design Density* experiment.

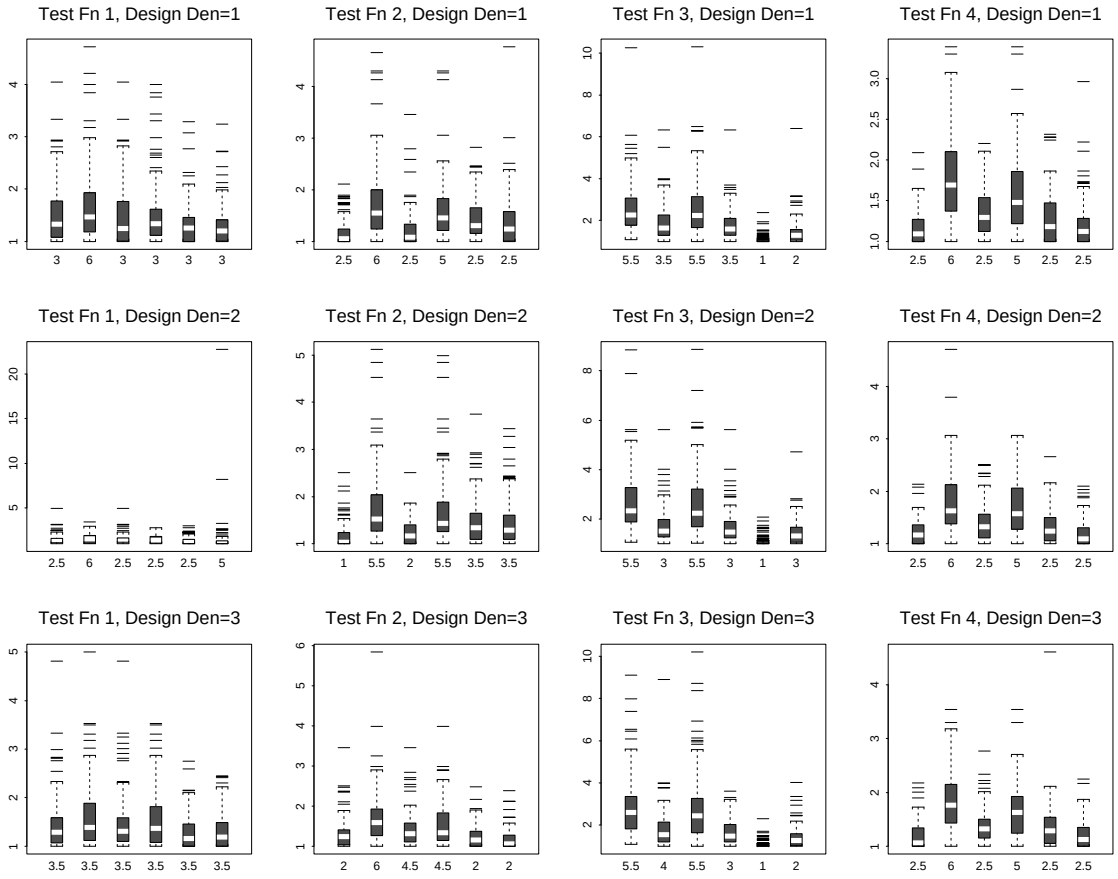

Figure 6: Similar to Figure 5 but for log(MSE).

5.3 Varying Noise Variance Function

Again, the same four test functions as in the *Varying Noise Level* experiment were used, but the variance of the Gaussian noise was not the same for all values of x_i. Rather, the variance was specified by a variance function $v(x)$, and the data (x_i, y_i) were generated from $y_i = f(x_i) + v(x_i)e_i$, where $\sigma = \|f\|/4$ and the x_i's were drawn from Unif[0,1]. The following three variance functions $v(x)$ were used:

$$\text{Variance Function 1} \quad v(x) = 0.5 + x, \quad x \in [0, 1],$$

$$\text{Variance Function 2} \quad v(x) = \begin{cases} 1.3 - 1.2x & x \in [0, 0.5), \\ 0.2 + 1.2x & x \in [0.5, 1], \end{cases}$$

$$\text{Variance Function 3} \quad v(x) = 1.5 - x, \quad x \in [0, 1].$$

These variance functions are plotted in Figure 7. Boxplots of the GCV and log(MSE) values, together with the Wilcoxon test rankings are given in Figures 8 and 9 in a similar manner as before.

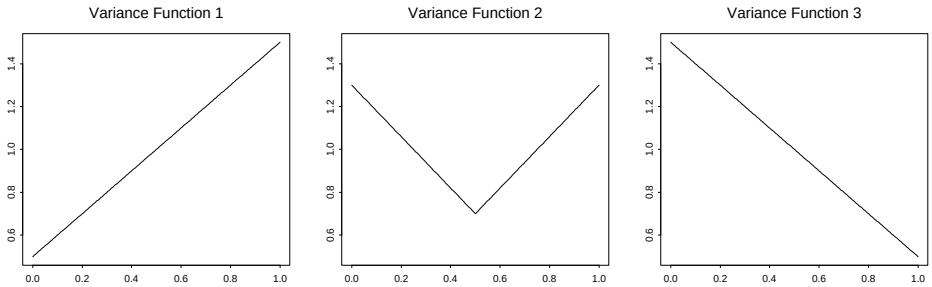

Figure 7: Plots of the Noise Variance Functions.

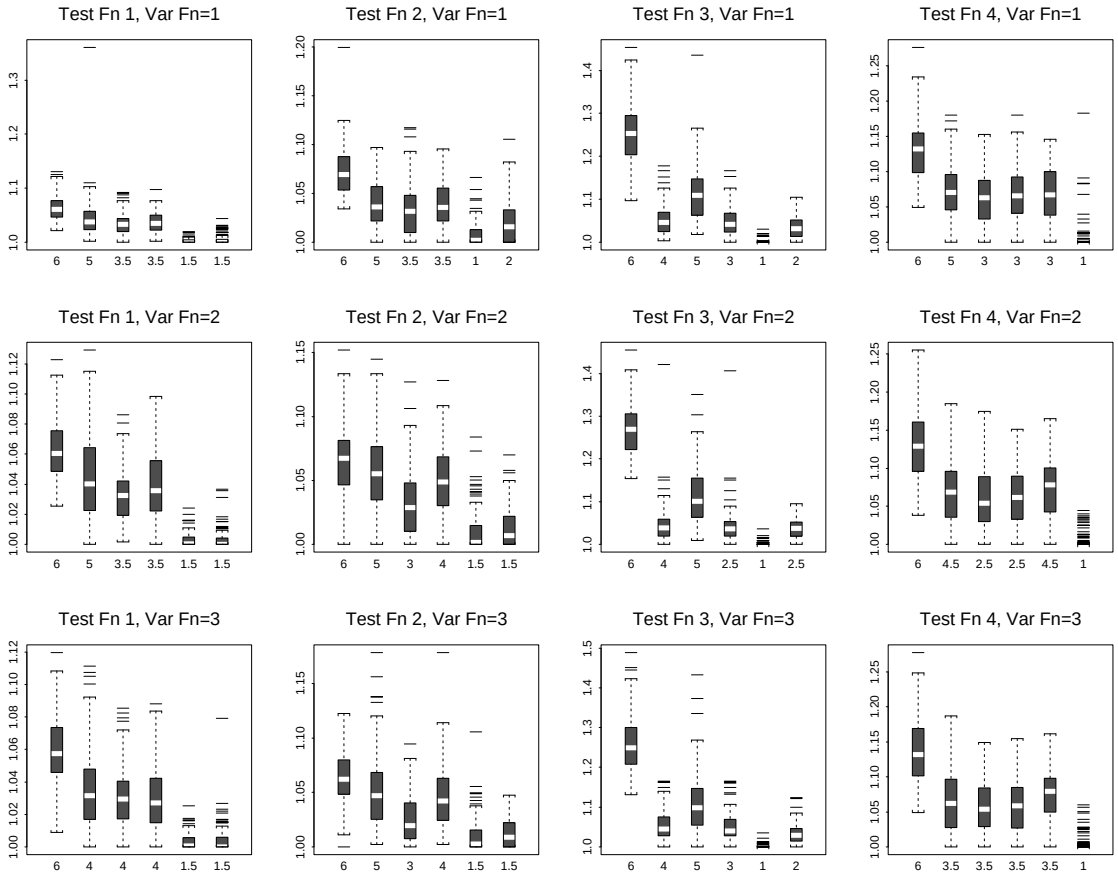

Figure 8: Similar to Figure 2 but for the *Varying Noise Variance Function* experiment.

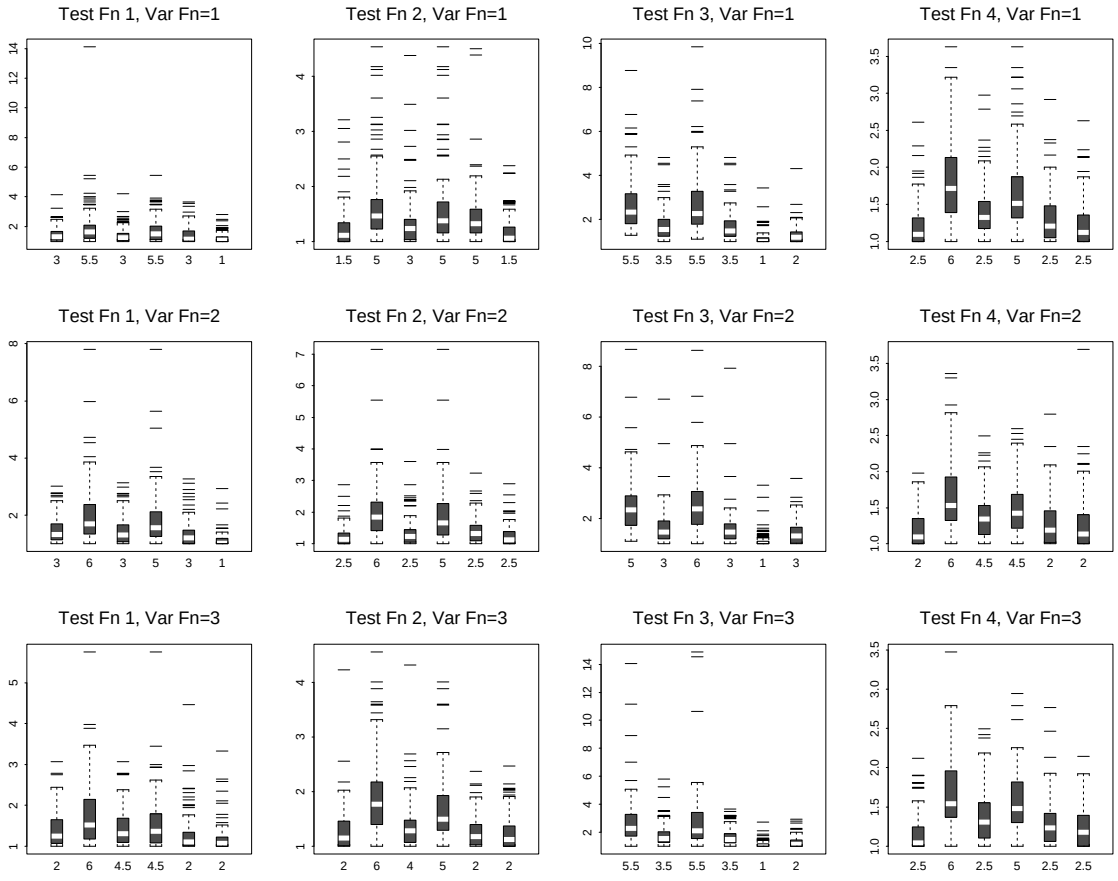

Figure 9: Similar to Figure 8 but for log(MSE).

5.4 Varying Spatial Variation

In this experiment the six test functions were taken from Wand (2000), all have different degrees of spatial variation. They are indexed by a single integer parameter j, and have the form

$$f_j(x) = \sqrt{x(1-x)}\,\sin\left[\frac{2\pi\left\{1+2^{(9-4j)/5}\right\}}{x+2^{(9-4j)/5}}\right], \quad j = 1,\ldots,6.$$

Note that it is of the same structural form of the *Doppler* function introduced by Donoho & Johnstone (1994).

The data (x_i, y_i) were generated from $y_i = f_j(x_i) + e_i$ with $\sigma = \|f_j\|/4$ and the x_i's drawn from Unif[0,1]. The test functions, together with typical simulated data sets, are plotted in Figure 10. In a similar fashion as before, boxplots of GCV and log(MSE) values, and Wilcoxon test rankings of the algorithms are displayed in Figures 11 and 12.

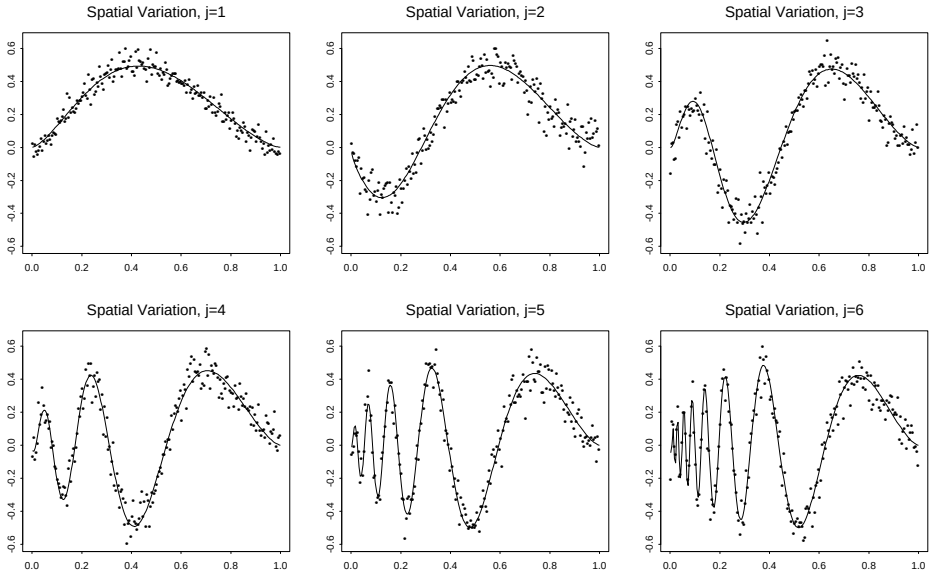

Figure 10: Plots of the Test Functions, together with typical simulated data sets, for the *Varying Spatial Variation* experiment.

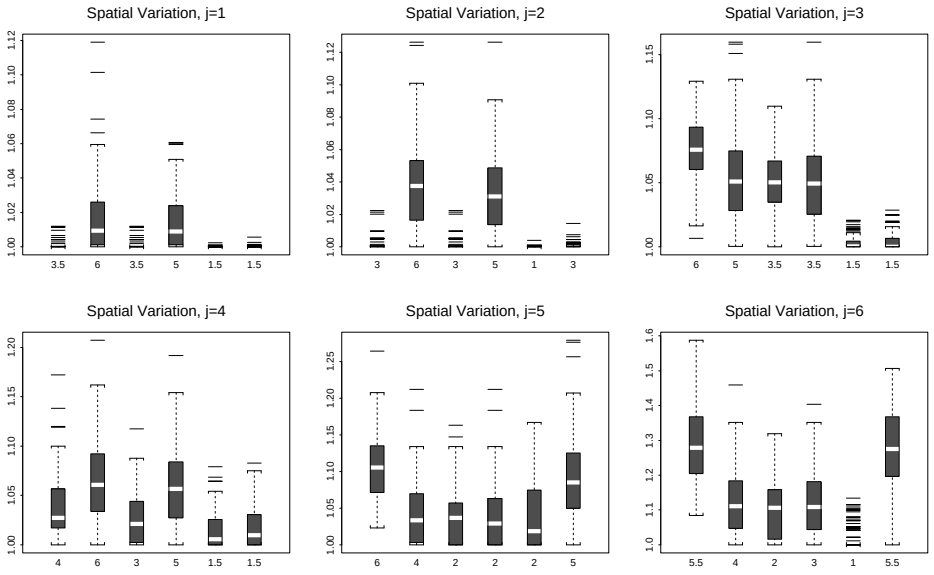

Figure 11: Similar to Figure 2 but for the *Varying Spatial Variation* experiment.

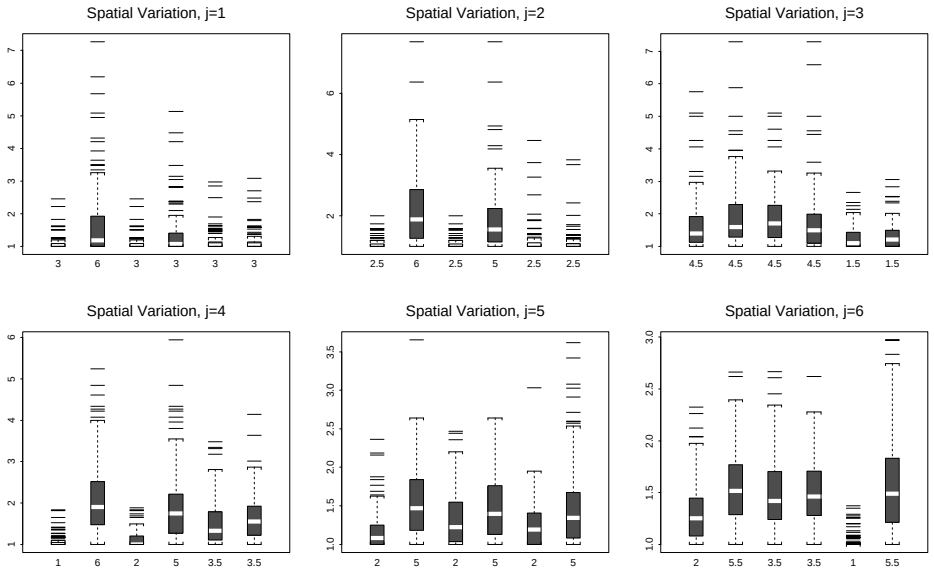

Figure 12: Similar to Figure 11 but for log(MSE).

5.5 Empirical Conclusions and Recommendations

The six algorithms gave fairly consistent performances for the above different experimental settings. The averaged Wilcoxon GCV and MSE test rankings are given in Tables 2 and 3 respectively. Judging from the overall averaged rankings, it seems that the two genetic algorithms are superior to the stepwise procedures, especially in terms of minimizing GCV($\hat{\boldsymbol{\theta}}$).

The computation times taken for the stepwise procedures to finish one run under various settings were reasonably constant. The typical running times on a Sun *Ultra*–60 Workstation for *ForAdd*, *BackElim*, *AddElim* and *ElimAdd* were respectively 15s, 38s, 15s and 53s. These timings are best to be served as upper bounds, as we did not optimize the codes in our implementation. For the genetic algorithms, the execution times were quite variable: for *GeneFix* it ranged from 50s to 70s, while for *GeneVar* it ranged from 50s to 100s. Therefore, if computation time is not an issue, then one should perhaps use the genetic algorithms. However, if time is an important issue, then *AddElim* seems to be a good compromise.

One last interesting observation is that, for some settings the GCV and MSE values are not linearly related; i.e., a $\hat{\boldsymbol{\theta}}$ that has a small GCV value does not guarantee to has a small MSE value, and vice versa. This may suggest that GCV($\hat{\boldsymbol{\theta}}$) is not the best criterion if the goal is to obtain the $\hat{\boldsymbol{\theta}}$ that minimizes MSE.

Algorithm	NoiseL	DesignD	VarFn	SpaVar	Overall
ForAdd	6.00	5.96	6.00	4.67	5.66
BackElim	4.50	4.54	4.50	5.17	4.68
AddElim	3.84	3.62	3.71	2.83	3.50
ElimAdd	3.33	3.21	3.33	3.92	3.45
GeneFix	1.71	1.88	1.88	1.42	1.72
GeneVar	1.63	1.79	1.59	3.00	2.00

Table 2: Average Wilcoxon GCV test rankings for the four univariate experimental settings: Noise Level (NoiseL), Design Density (DesignD), Variance Function (VarFn) and Spatial Variation (SpaVar).

Algorithm	NoiseL	DesignD	VarFn	SpaVar	Overall
ForAdd	3.33	3.21	3.08	2.50	3.03
BackElim	4.96	5.13	5.21	5.50	5.20
AddElim	4.08	3.50	3.88	2.92	3.60
ElimAdd	4.25	4.03	4.54	4.33	4.29
GeneFix	2.21	2.29	2.29	2.25	2.26
GeneVar	2.17	2.83	2.00	3.50	2.63

Table 3: Similar to Table 2 but for the Wilcoxon MSE test rankings.

6 Bivariate Smoothing

This section considers bivariate smoothing. Due to space limitation, the description will be kept minimal. Important references include Friedman (1991) and Kooperberg et al. (1997).

6.1 Background

The data $\{y_i, x_{1i}, x_{2i}\}_{i=1}^n$ are now assumed to satisfy

$$y_i = f(x_{1i}, x_{2i}) + e_i, \quad e_i \sim \text{iid} N(0, \sigma^2), \tag{3}$$

where f is the bivariate "true" surface to be recovered. It is assumed that f can be modeled by

$$f(x_1, x_2) = \sum \beta_j B_j(x_1, x_2),$$

where the basis functions $B_j(x_1, x_2)$'s are of the form 1, x_1, x_2, $(x_1 - k_{1u})_+$, $(x_2 - k_{2v})_+$, $x_1 x_2$, $x_2(x_1 - k_{1u})_+$, $x_1(x_2 - k_{2v})_+$ and $(x_1 - k_{1u})_+(x_2 - k_{2v})_+$. Also, the knots are restricted to be a subset of the marginal values of the design points (x_{1i}, x_{2i}). In our simulation a bivariate version of GCV($\boldsymbol{\theta}$) is used to define a "best" $\hat{f}$.

Driven by the univariate simulation results, two algorithms for minimizing the GCV function are investigated: stepwise knot addition followed by elimination and genetic algorithms with fixed–length chromosomes. These two algorithms are simple straightforward extensions of their univariate counterparts. In fact the stepwise procedure to be compared in our simulation below was taken from the POLYMARS package mainly developed by Charles Kooperberg (see the unpublished manuscript Kooperberg & O'Connor n.d.). Note that in POLYMARS a "no interactions if no main effects" constraint is placed on all the possible candidate models. The same constraint was also imposed for the genetic algorithm.

6.2 Simulation

A simulation study was conducted. Although it was done at a smaller scale when comparing to the univariate setting, we believe that useful empirical conclusions can still be drawn. Five bivariate test functions were used. They are listed in Table 4 and are plotted in Figure 13. These functions have been used by for examples by Denison, Mallick & Smith (1998*b*) and Hwang, Lay, Maechler, Martin & Schimert (1994), and were constructed to have a unit standard deviation and a non–negative range. The data were generated according to (3), with the design points drawn from Unif$[0,1]^2$. The number of design points was 100, and three SNRs were used: 2, 4 and 6. The number of replicates was 100.

For each simulated data set, the two algorithms were applied to minimize the GCV function and obtain the corresponding $\hat{f}$. As for the univariate case, both the GCV and MSE values were computed. The MSE values were computed over a grid of 25×25 grid points. Boxplots of the GCV and log(MSE) values, together with the Wilcoxon test rankings, are provided in Figures 14 and 15. Typically it took less than 5s for the POLYMARS stepwise procedure to finishes, while for the genetic algorithm the computation times were ranged from 50s to 300s.

6.3 Results and Some Suggestions

In terms of minimizing the GCV function, the stepwise procedure performed better for Test Function 1, while the genetic algorithm performed better for the remaining four test functions. However, both procedures gave similar performances in terms of MSE. It seems to suggest that, when the "true" surface is simple in structure (such as Test Function 1), the greedy nature of the stepwise procedure can reliably locate good intermediate search directions during the minimization of GCV. However, when the "true" surface is more complicated, the stepwise procedure may fail to do so, and one may need to switch to the computationally–very–expensive genetic algorithm.

Test Function 1: $f(x_1, x_2) = 10.391\{(x_1 - 0.4)(x_2 - 0.6) + 0.36\}$

Test Function 2: $f(x_1, x_2) = 24.234\{r^2(0.75 - r^2)\}, \quad r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$

Test Function 3: $f(x_1, x_2) = 42.659\{0.1 + \tilde{x}_1(0.05 + \tilde{x}_1^4 - 10\tilde{x}_1^2\tilde{x}_2^2 + 5\tilde{x}_2^4)\}, \quad \tilde{x}_1 = x_1 - 0.5, \tilde{x}_2 = x_2 - 0.5$

Test Function 4: $f(x_1, x_2) = 1.3356[1.5(1 - x_1) + e^{2x_1 - 1}\sin\{3\pi(x_1 - 0.6)^2\} + e^{3(x_2 - 0.5)}\sin\{4\pi(x_2 - 0.9)^2\}]$

Test Function 5: $f(x_1, x_2) = 1.9[1.35 + e^{x_1}\sin\{13(x_1 - 0.6)^2\}e^{-x_2}\sin(7x_2)]$

Table 4: Five Test Functions for the bivariate setting.

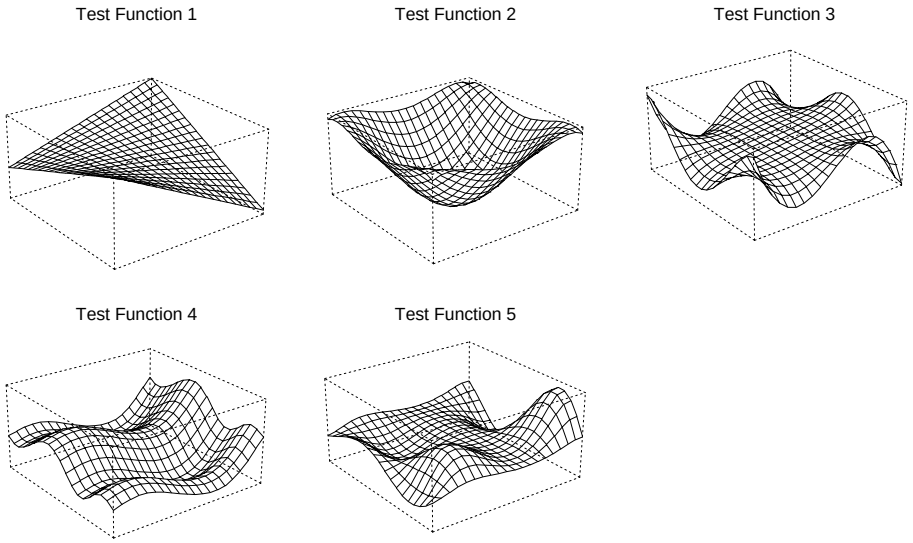

Figure 13: Perspective plots of bivariate Test Functions.

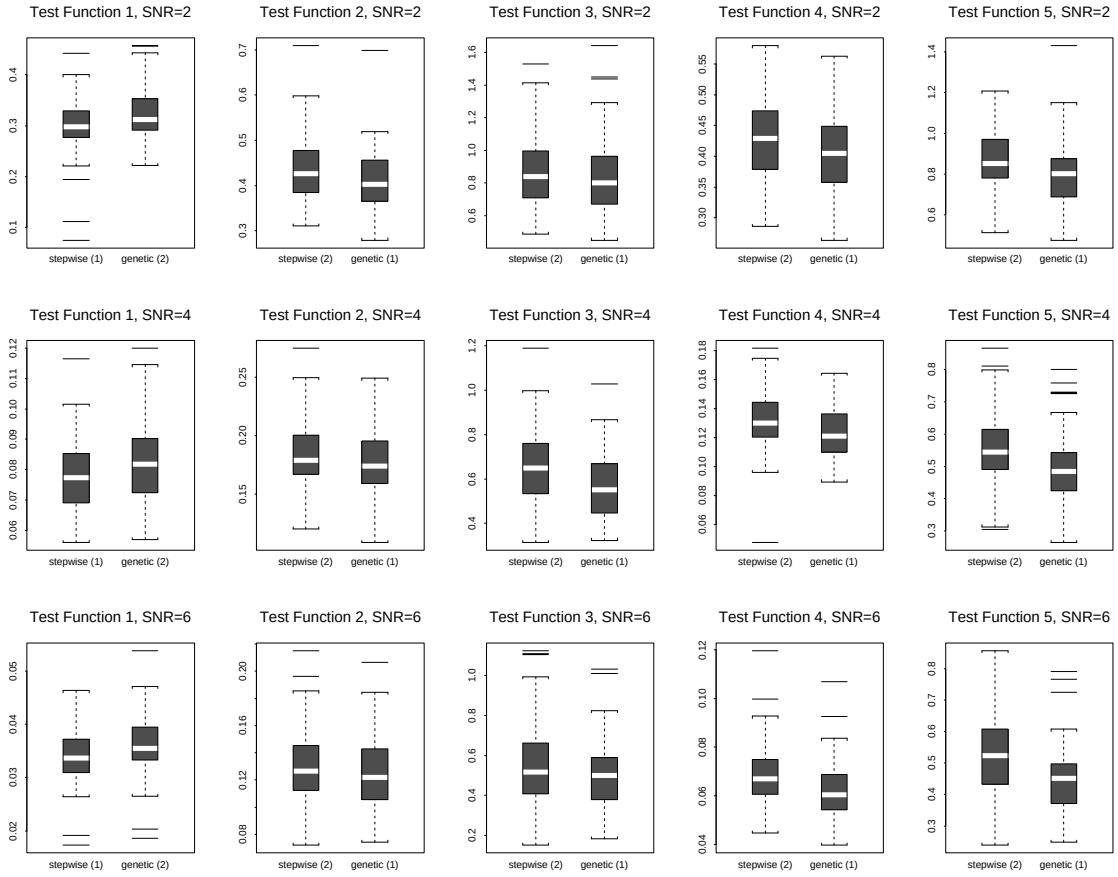

Figure 14: Boxplots of GCV values for the bivariate smoothing experiment. Numbers in parentheses are Wilcoxon test rankings.

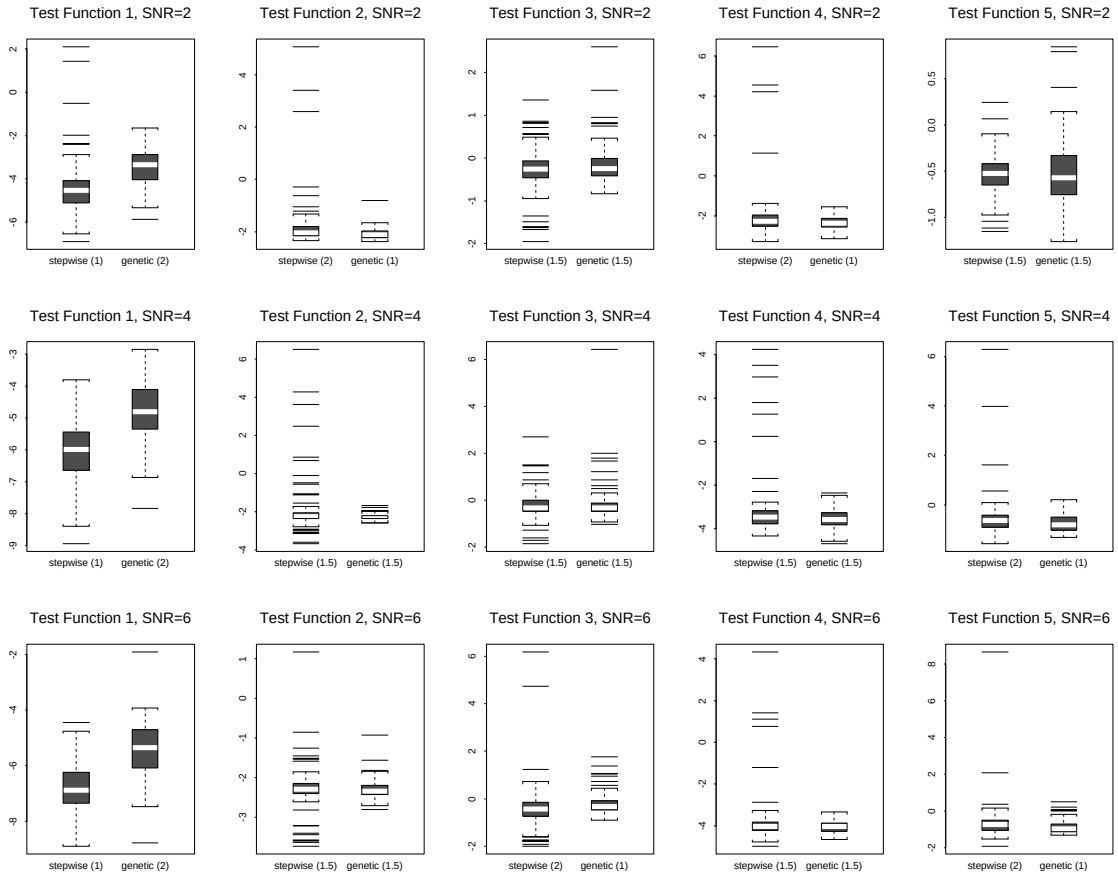

Figure 15: Similar to Figure 14 but for log(MSE).

Acknowledgement

The author would like to thank one referee for his/her constructive comments.

References

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

Denison, D. G. T., Mallick, B. K. & Smith, A. F. M. (1998*a*), 'Automatic Bayesian curve fitting', *Journal of the Royal Statistical Society Series B* **60**, 333–350.

Denison, D. G. T., Mallick, B. K. & Smith, A. F. M. (1998*b*), 'Bayesian MARS', *Statistics and Computing* **8**, 337–346.

DiMatteo, I., Genovese, C. R. & Kass, R. E. (2001), 'Bayesian curve fitting with free-knot splines', *Biometrika* **88**, 1055–1071.

Donoho, D. L. & Johnstone, I. M. (1994), 'Ideal spatial adaptation by wavelet shrinkage', *Biometrika* **81**, 425–455.

Eilers, P. H. C. & Marx, B. D. (1996), 'Flexible parsimonious smoothing and additive modeling (with discussion)', *Statistical Science* **89**, 89–121.

Fogel, D. B. (2000), 'Evolutionary computing', *IEEE Spectrum* pp. 26–32.

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

Friedman, J. H. & Silverman, B. W. (1989), 'Flexible parsimonious smoothing and additive modeling (with discussion)', *Technometrics* **31**, 3–21.

Green, P. J. & Silverman, B. W. (1994), *Nonparametric Regression and Generalized Linear Models*, Chapman and Hall, London.

Hansen, M. H., Kooperberg, C. & Sardy, S. (1998), 'Triogram models', *Journal of the American Statistical Association* **93**, 101–119.

Hwang, J.-N., Lay, S.-R., Maechler, M., Martin, R. D. & Schimert, J. (1994), 'Regression modeling in back–propagation and projection pursuit learning', *IEEE Transactions on Neural Networks* **5**, 342–353.

Koo, J.-Y. (1997), 'Spline estimation of discontinuous regression functions', *Journal of Computational and Graphical Statistics* **6**, 266–284.

Kooperberg, C. & O'Connor, M. (n.d.), 'POLYMARS'. Unpublished manuscript.

Kooperberg, C., Bose, S. & Stone, C. J. (1997), 'Polychotomous regression', *Journal of the American Statistical Association* **92**, 117–127.

Lee, T. C. M. (2000), 'Regression spline smoothing using the minimum description length principle', *Statistics and Probability Letters* **48**, 71–82.

Lee, T. C. M. (2002), 'Automatic smoothing for discontinuous regression functions', *Statistica Sinica*. To appear.

Lindstrom, M. J. (1999), 'Penalized estimation of free–knot splines', *Journal of Computational and Graphical Statistics* **8**, 333–352.

Michalewicz, Z. (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, third, revised and extended edn, Springer-Verlag Berlin Heidelberg.

Pittman, J. (1999), 'Adaptive splines and genetic algorithms', *Journal of Computational and Graphical Statistics*. To appear.

Pittman, J. & Murthy, C. (2000), 'Fitting optimal piecewise linear functions using genetic algorithms', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 701–718.

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

Smith, M. & Kohn, R. (1996), 'Nonparametric regression using Bayesian variable selection', *J. Econometrics* **75**, 317–344.

Stone, C. J., Hansen, M., Kooperberg, C. & Truong, Y. K. (1997), 'Polynomial splines and their tensor products in extended linear modeling (with discussion)', *The Annals of Statistics* **25**, 1371–1470.

Wand, M. P. (2000), 'A comparison of regression spline smoothing procedures', *Computational Statistics* **15**, 443–462.