11. In an experiment comparing two varieties of soybeans, plants were grown on 48 plots and measurements of leaf weight were taken at regular intervals as the plants grew. The `nlme` data frame `Soybean` contains the resulting data and has the following columns:

    `Plot` is a factor variable with levels for each of the 48 plots.

    `weight` is the leaf weight in grammes.

    `Time` is the time in days since planting.

    `Variety` is either `F` or `P` indicating the variety of soybean.

    There is one observation for each variety in each plot at each time. Interest focuses on modelling the growth of soybeans over time and on establishing whether or not this differs between the varieties.
    In this question you may have to increase `niterEM` in the `control` list for `lme` and `glmmPQL`: see `?lmeControl`.

    (a) A possible model for the weights is

    $$w_{ijk} = \alpha_i + \beta_i t_k + \gamma_i t_k^2 + \delta_i t_k^3 + a_j + b_j t_k + \epsilon_{ijk}$$

    where w_{ijk} is the weight measurement for the i^{th} variety in the j^{th} plot at the k^{th} time; $[a_j, b_j]^{\mathsf{T}} \sim N(\mathbf{0}, \boldsymbol{\psi})$ where $\boldsymbol{\psi}$ is a covariance matrix, and $\epsilon_{ijk} \sim N(0, \sigma^2)$. The random effects are independent of the residuals and independent of random effects with different j. The residuals are i.i.d.

    Fit this model using `lme` and confirm that the residual variance appears to increase with the (random effect conditional) mean.

    (b) To deal with the mean variance relationship, it might be appropriate to model the weights as being Gamma distributed, so that the model becomes a GLMM. e.g.

    $$\log(\mu_{ijk}) = \alpha_i + \beta_i t_k + \gamma_i t_k^2 + \delta_i t_k^3 + a_j + b_j t_k,$$

    where $\mu_{ijk} \equiv \mathbb{E}(w_{ijk})$ and $w_{ijk} \sim \text{Gamma}$. Fit this GLMM, using `glmmPQL` from the `MASS` package, and confirm the improvement in residual plots that results.

    (c) Explore whether further improvements to the model could be made by modifications of the random or fixed effects model structures.

Chapter 4

Introducing GAMs

4.1 Introduction

A generalized additive model (Hastie and Tibshirani, 1986, 1990) is a generalized linear model with a linear predictor involving a sum of smooth functions of covariates. In general the model has a structure something like

$$g(\mu_i) = \mathbf{A}_i\boldsymbol{\theta} + f_1(x_{1i}) + f_2(x_{2i}) + f_3(x_{3i}, x_{4i}) + \ldots \tag{4.1}$$

where $\mu_i \equiv \mathbb{E}(Y_i)$ and $Y_i \sim \text{EF}(\mu_i, \phi)$. Y_i is a response variable, $\text{EF}(\mu_i, \phi)$ denotes an exponential family distribution with mean μ_i and scale parameter, ϕ, $\mathbf{A}_i$ is a row of the model matrix for any strictly parametric model components, $\boldsymbol{\theta}$ is the corresponding parameter vector, and the f_j are smooth functions of the covariates, x_k. The model allows for flexible specification of the dependence of the response on the covariates, but by specifying the model only in terms of 'smooth functions', rather than detailed parametric relationships, it is possible to avoid the sort of cumbersome and unwieldy models seen in section 3.3.5, for example. This flexibility and convenience comes at the cost of two new theoretical problems. It is necessary both to represent the smooth functions in some way and to choose how smooth they should be.

This chapter illustrates how GAMs can be represented using basis expansions for each smooth, each with an associated penalty controlling function smoothness. Estimation can then be carried out by penalized regression methods, and the appropriate degree of smoothness for the f_j can be estimated from data using cross validation or marginal likelihood maximization. To avoid obscuring the basic simplicity of the approach with a mass of technical detail, the most complicated model considered here will be a simple GAM with two univariate smooth components. Furthermore, the methods presented will not be those that are most suitable for general practical use, being rather the methods that enable the basic framework to be explained simply. The ideal way to read this chapter is sitting at a computer, working through the statistics, and its implementation in R, side by side. If adopting this approach recall that the help files for R functions can be accessed by typing `?` followed by the function name, at the command line (e.g., `?lm`, for help on the linear modelling function).

4.2 Univariate smoothing

The representation and estimation of component functions of a model is best introduced by considering a model containing one function of one covariate,

$$y_i = f(x_i) + \epsilon_i, \tag{4.2}$$

where y_i is a response variable, x_i a covariate, f a smooth function and the ϵ_i are independent $N(0, \sigma^2)$ random variables.

4.2.1 *Representing a function with basis expansions*

To estimate f, using the methods covered in chapters 1 to 3, requires that f be represented in such a way that (4.2) becomes a linear model. This can be done by choosing a *basis*, defining the space of functions of which f (or a close approximation to it) is an element. Choosing a basis amounts to choosing some *basis functions*, which will be treated as completely known: if $b_j(x)$ is the j^{th} such basis function, then f is assumed to have a representation

$$f(x) = \sum_{j=1}^{k} b_j(x)\beta_j, \tag{4.3}$$

for some values of the unknown parameters, β_j. Substituting (4.3) into (4.2) clearly yields a linear model.

A very simple basis: Polynomials

As a simple example, suppose that f is believed to be a 4^{th} order polynomial, so that the space of polynomials of order 4 and below contains f. A basis for this space is $b_1(x) = 1$, $b_2(x) = x$, $b_3(x) = x^2$, $b_4(x) = x^3$ and $b_5(x) = x^4$, so that (4.3) becomes

$$f(x) = \beta_1 + x\beta_2 + x^2\beta_3 + x^3\beta_4 + x^4\beta_5,$$

and (4.2) becomes the simple model

$$y_i = \beta_1 + x_i\beta_2 + x_i^2\beta_3 + x_i^3\beta_4 + x_i^4\beta_5 + \epsilon_i.$$

Figures 4.1 and 4.2 illustrate a basis function representation of a function, f, using a polynomial basis.

The problem with polynomials

Taylor's theorem implies that polynomial bases will be useful for situations in which interest focuses on properties of f in the vicinity of a single specified point. But when the questions of interest relate to f over its whole domain, polynomial bases are problematic (see exercise 1).

The difficulties are most easily illustrated in the context of interpolation. The middle panel of figure 4.3 illustrates an attempt to approximate the function shown in

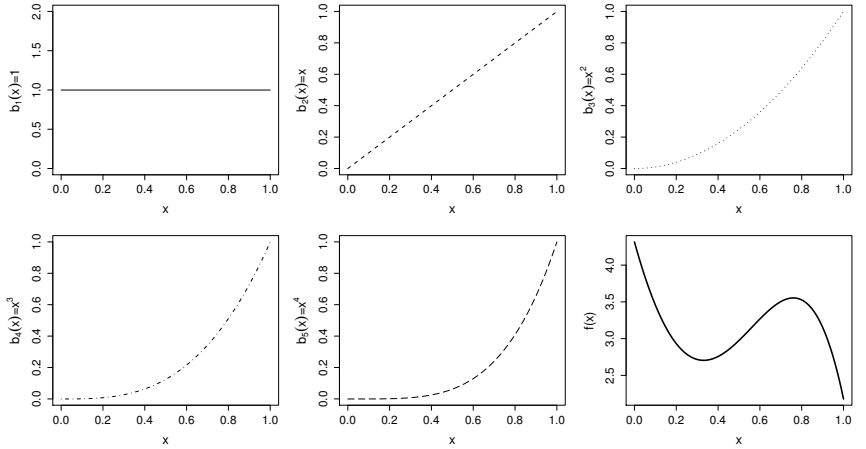

Figure 4.1 *Illustration of the idea of representing a function in terms of basis functions, using a polynomial basis. The first 5 panels (starting from top left) illustrate the 5 basis functions, $b_j(x)$, for a 4th order polynomial basis. The basis functions are each multiplied by a real valued parameter, β_j, and are then summed to give the final curve $f(x)$, an example of which is shown in the bottom right panel. By varying the β_j, we can vary the form of $f(x)$, to produce any polynomial function of order 4 or lower. See also figure 4.2*

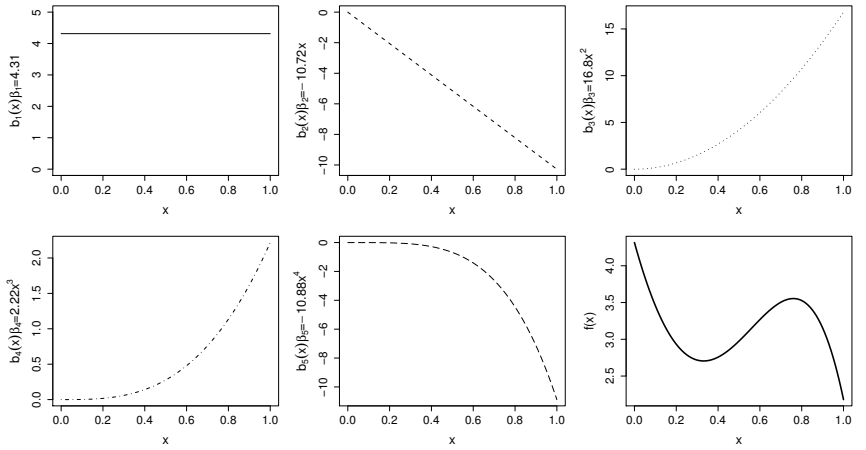

Figure 4.2 *An alternative illustration of how a function is represented in terms of basis functions. As in figure 4.1, a 4th order polynomial basis is illustrated. In this case the 5 basis functions, $b_j(x)$, each multiplied by its coefficient β_j, are shown in the first five figures (starting at top left). Simply summing these 5 curves yields the function, $f(x)$, shown at bottom right.*

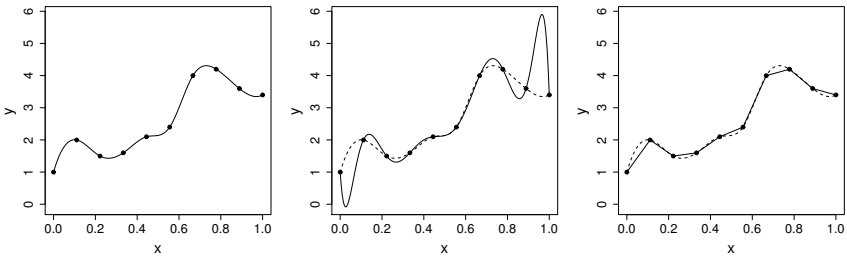

Figure 4.3 *The left panel shows a smooth function sampled at the points shown as black dots. The middle panel shows an attempt to reconstruct the function (dashed curve) by polynomial interpolation (black curve) of the black dots. The right panel shows the equivalent piecewise linear interpolant. The condition that the polynomial should interpolate the data and have all its derivatives continuous leads to quite wild oscillation.*

the left panel of figure 4.3, by polynomial interpolation of the points shown as black dots. The polynomial oscillates wildly in places, in order to accommodate the twin requirements to interpolate the data *and* to have all derivatives w.r.t. x continuous. If we relax the requirement for continuity of derivatives, and simply use a piecewise linear interpolant, then a much better approximation is obtained, as the right hand panel of figure 4.3 illustrates.

It clearly makes sense to use bases that are good at approximating known functions in order to represent unknown functions. Similarly, bases that perform well for interpolating exact observations of a function are also a good starting point for the closely related task of smoothing noisy observations of a function. In subsequent chapters we will see that piecewise linear bases can be improved upon by spline bases having continuity of just a few derivatives, but the piecewise linear case provides such a convenient illustration that we will stick with it for this chapter.

The piecewise linear basis

A basis for piecewise linear functions of a univariate variable x is determined entirely by the locations of the function's derivative discontinuities, that is by the locations at which the linear pieces join up. Let these *knots* be denoted $\{x_j^* : j = 1, \cdots k\}$, and suppose that $x_j^* > x_{j-1}^*$. Then for $j = 2, \ldots, k-1$

$$b_j(x) = \begin{cases} (x - x_{j-1}^*)/(x_j^* - x_{j-1}^*) & x_{j-1}^* < x \leq x_j^* \\ (x_{i+j}^* - x)/(x_{j+1}^* - x_j^*) & x_j^* < x < x_{j+1}^* \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

while

$$b_1(x) = \begin{cases} (x_2^* - x)/(x_2^* - x_1^*) & x < x_2^* \\ 0 & \text{otherwise} \end{cases}$$

and

$$b_k(x) = \begin{cases} (x - x_{k-1}^*)/(x_k^* - x_{k-1}^*) & x > x_{k-1}^* \\ 0 & \text{otherwise} \end{cases}$$

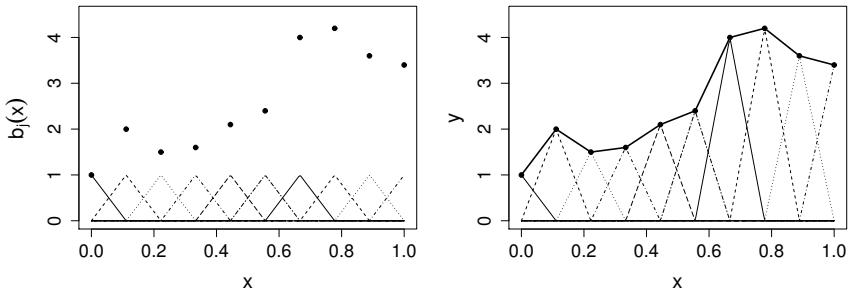

Figure 4.4 *The left panel shows an example tent function basis for interpolating the data shown as black dots. The continuous lines show the tent function basis functions, each of which peaks with value 1 at the x-axis value of one of the data points. The right panel illustrates how the basis functions are each multiplied by a coefficient, before being summed to give the interpolant, shown as the thick black line.*

So $b_j(x)$ is zero everywhere, except over the interval between the knots immediately to either side of x_j^*. $b_j(x)$ increases linearly from 0 at x_{j-1}^* to 1 at x_j^*, and then decreases linearly to 0 at x_{j+1}^*. Basis functions like this, that are non zero only over some finite intervals, are said to have *compact support*. Because of their shape the b_j are often known as *tent functions*. See figure 4.4.

Notice that an exactly equivalent way of defining $b_j(x)$ is as the linear interpolant of the data $\{x_i^*, \delta_i^j : i = 1, \ldots, k\}$ where $\delta_i^j = 1$ if $i = j$ and zero otherwise. This definition makes for very easy coding of the basis in R.

Using this basis to represent $f(x)$, (4.2) now becomes the linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ where $X_{ij} = b_j(x_i)$.

Using the piecewise linear basis

Now consider an illustrative example. It is often claimed, at least by people with little actual knowledge of engines, that a car engine with a larger cylinder capacity will wear out less quickly than a smaller capacity engine. Figure 4.5 shows some data for 19 Volvo engines. The pattern of variation is not entirely clear, so (4.2) might be an appropriate model.

First read the data into R.

```
require(gamair); data(engine); attach(engine)
plot(size,wear,xlab="Engine capacity",ylab="Wear index")
```

Now write an R function defining $b_j(x)$

```
tf <- function(x,xj,j) {
## generate jth tent function from set defined by knots xj
  dj <- xj*0;dj[j] <- 1
  approx(xj,dj,x)$y
}
```

and use it to write an R function that will take a sequence of knots and an array of x values to produce a model matrix for the piecewise linear function.

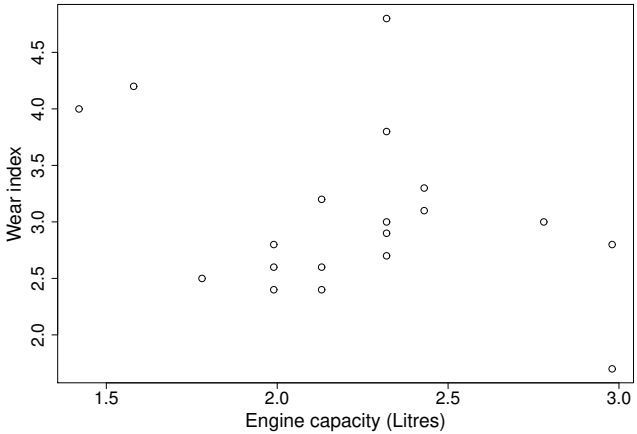

Figure 4.5 *Data on engine wear index versus engine capacity for 19 Volvo car engines, obtained from* `http://www3.bc.sympatico.ca/Volvo_Books/engine3.html`.

```
tf.X <- function(x,xj) {
## tent function basis matrix given data x
## and knot sequence xj
  nk <- length(xj); n <- length(x)
  X <- matrix(NA,n,nk)
  for (j in 1:nk) X[,j] <- tf(x,xj,j)
  X
}
```

All that is required now is to select a set of knots, x_j^*, and the model can be fitted. In the following a rank $k = 6$ basis is used, with the knots spread evenly over the range of the `size` data.

```
sj <- seq(min(size),max(size),length=6) ## generate knots
X <- tf.X(size,sj)                      ## get model matrix
b <- lm(wear ~ X - 1)                   ## fit model
s <- seq(min(size),max(size),length=200)## prediction data
Xp <- tf.X(s,sj)                        ## prediction matrix
plot(size,wear)                         ## plot data
lines(s,Xp %*% coef(b))                 ## overlay estimated f
```

The model fit looks quite plausible (figure 4.6), but the choice of degree of model smoothness, controlled here by the basis dimension, k, was essentially arbitrary. This issue must be addressed if a satisfactory theory for modelling with unknown functions is to be developed.

4.2.2 Controlling smoothness by penalizing wiggliness

One obvious possibility for choosing the degree of smoothing is to try to make use of the hypothesis testing methods from chapter 1, to select k by backwards selection.

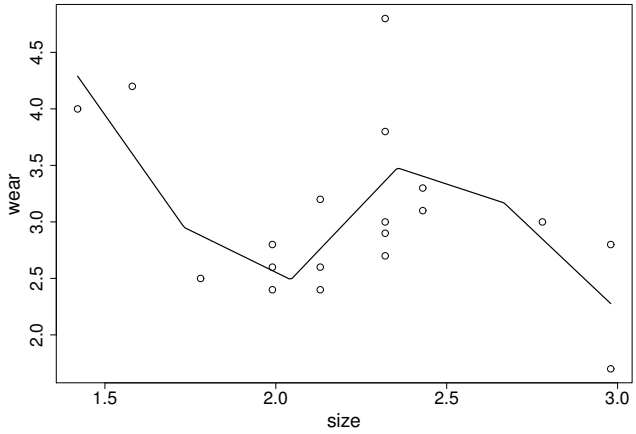

Figure 4.6 *Piecewise linear regression fit (continuous line) to data (∘) on engine wear index versus engine capacity (size) for 19 Volvo car engines.*

However, such an approach is problematic, since a model based on $k-1$ evenly spaced knots will not generally be nested within a model based on k evenly spaced knots. It is possible to start with a fine grid of knots and simply drop knots sequentially, as part of backward selection, but the resulting uneven knot spacing can itself lead to poor model performance. Furthermore, the fit of such regression models tends to depend quite strongly on the locations chosen for the knots.

An alternative is to keep the basis dimension fixed at a size a little larger than it is believed could reasonably be necessary, but to control the model's smoothness by adding a 'wiggliness' penalty to the least squares fitting objective. For example, rather than fitting the model by minimizing

$$\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2,$$

it could be fitted by minimizing

$$\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2+\lambda\sum_{j=2}^{k-1}\{f(x_{j-1}^*)-2f(x_j^*)+f(x_{j+1}^*)\}^2,$$

where the summation term measures wiggliness as a sum of squared second differences of the function at the knots (which crudely approximates the integrated squared second derivative penalty used in cubic spline smoothing: see section 5.1.2, p. 198). When f is very wiggly the penalty will take high values and when f is 'smooth' the penalty will be low.* If f is a straight line then the penalty is actually zero. So the penalty has a *null space* of functions that are un-penalized: the straight lines in this

*Note that even knot spacing has been assumed: uneven knot spacing would usually require some re-weighting of the penalty terms.

case. The dimension of the penalty null space is 2, since the basis for straight lines is 2-dimensional.

The *smoothing parameter*, λ, controls the trade-off between smoothness of the estimated f and fidelity to the data. $\lambda \to \infty$ leads to a straight line estimate for f, while $\lambda = 0$ results in an un-penalized piecewise linear regression estimate.

For the basis of tent functions, it is easy to see that the coefficients of f are simply the function values at the knots, i.e., $\beta_j = f(x_j^*)$. This makes it particularly straightforward to express the penalty as a quadratic form, $\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$, in the basis coefficients (although in fact linearity of f in the basis coefficients is all that is required for this). Firstly note that

$$\begin{bmatrix} \beta_1 - 2\beta_2 + \beta_3 \\ \beta_2 - 2\beta_3 + \beta_4 \\ \beta_3 - 2\beta_4 + \beta_5 \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & -2 & 1 & 0 & . & . & . \\ 0 & 1 & -2 & 1 & 0 & . & . \\ 0 & 0 & 1 & -2 & 1 & 0 & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ . \\ . \end{bmatrix}$$

so that writing the right hand side as $\mathbf{D}\boldsymbol{\beta}$, by definition of $(k-2) \times k$ matrix $\mathbf{D}$, the penalty becomes

$$\sum_{j=2}^{k-1} (\beta_{j-1} - 2\beta_j + \beta_{j+1})^2 = \boldsymbol{\beta}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}}\mathbf{D}\boldsymbol{\beta} = \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta} \tag{4.5}$$

where $\mathbf{S} = \mathbf{D}^{\mathsf{T}}\mathbf{D}$ ($\mathbf{S}$ is obviously rank deficient by the dimension of the penalty null space). Hence the penalized regression fitting problem is to minimize

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta} \tag{4.6}$$

w.r.t. $\boldsymbol{\beta}$. The problem of estimating the degree of smoothness for the model is now the problem of estimating the smoothing parameter λ. But before addressing λ estimation, consider $\boldsymbol{\beta}$ estimation given λ.

It is fairly straightforward to show (see exercise 3) that the formal expression for the minimizer of (4.6), the penalized least squares estimator of $\boldsymbol{\beta}$, is

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}. \tag{4.7}$$

Similarly the influence (hat) matrix, $\mathbf{A}$, for the model can be written

$$\mathbf{A} = \mathbf{X}\left(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S}\right)^{-1}\mathbf{X}^{\mathsf{T}}.$$

Recall that $\hat{\boldsymbol{\mu}} = \mathbf{A}\mathbf{y}$. As with the un-penalized linear model, these expressions are not the ones to use for computation, for which the greater numerical stability offered by orthogonal matrix methods is to be preferred. For practical computation, therefore, note that

$$\left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{D} \end{bmatrix} \boldsymbol{\beta} \right\|^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}.$$

Obviously any matrix square root such that $\mathbf{D}^{\mathsf{T}}\mathbf{D} = \mathbf{S}$ could be substituted for the original $\mathbf{D}$ here, but at the moment there is no reason to use an alternative. The sum of squares term, on the left hand side, is just a least squares objective for a model in which the model matrix has been augmented by a square root of the penalty matrix, while the response data vector has been augmented with $k - 2$ zeros. Fitting the augmented linear model will therefore yield $\hat{\boldsymbol{\beta}}$.

To see a penalized regression spline in action, first note that $\mathbf{D}$ can be obtained in R using `diff(diag(k),differences=2)`, which applies second order differencing to each column of the rank k identity matrix. Now it is easy to write a simple function for fitting a penalized piecewise linear smoother.

```
prs.fit <- function(y,x,xj,sp) {
  X <- tf.X(x,xj)       ## model matrix
  D <- diff(diag(length(xj)),differences=2) ## sqrt penalty
  X <- rbind(X,sqrt(sp)*D) ## augmented model matrix
  y <- c(y,rep(0,nrow(D))) ## augmented data
  lm(y ~ X - 1)    ## penalized least squares fit
}
```

To use this function, we need to choose the basis dimension, k, the (evenly spaced) knot locations, x_j^*, and a value for the smoothing parameter, λ. Provided that k is large enough that the basis is more flexible than we expect to *need* to represent $f(x)$, then neither the exact choice of k, nor the precise selection of knot locations, has a great deal of influence on the model fit. Rather it is the choice of λ that now plays the crucial role in determining model flexibility, and ultimately the shape of $\hat{f}(x)$. In the following example $k = 20$ and the knots are evenly spread out over the range of observed engine sizes. It is the smoothing parameter, $\lambda = 2$, which really controls the behaviour of the fitted model.

```
sj <- seq(min(size),max(size),length=20) ## knots
b <- prs.fit(wear,size,sj,2) ## penalized fit
plot(size,wear)   ## plot data
Xp <- tf.X(s,sj)  ## prediction matrix
lines(s,Xp %*% coef(b)) ## plot the smooth
```

By changing the value of the smoothing parameter, λ, a variety of models of different smoothness can be obtained. Figure 4.7 illustrates this, but begs the question, which value of λ is 'best'?

4.2.3 Choosing the smoothing parameter, λ, by cross validation

If λ is too high then the data will be over-smoothed, and if it is too low then the data will be under-smoothed: in both cases this will mean that the estimate $\hat{f}$ will not be close to the true function f. Ideally, it would be good to choose λ so that $\hat{f}$ is as close as possible to f. A suitable criterion might be to choose λ to minimize

$$M = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i - f_i)^2,$$

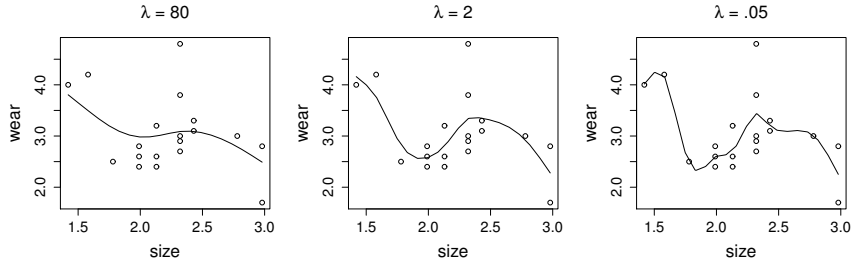

Figure 4.7 *Penalized piecewise linear fits to the engine wear versus capacity data, using three different values for the smoothing parameter, λ. Notice how penalization produces quite smooth estimates, despite the piecewise linear basis.*

where the notation $\hat{f}_i \equiv \hat{f}(x_i)$ and $f_i \equiv f(x_i)$ has been adopted for conciseness. Since f is unknown, M cannot be used directly, but it is possible to derive an estimate of $\mathbb{E}(M) + \sigma^2$, which is the expected squared error in predicting a new variable. Let $\hat{f}^{[-i]}$ be the model fitted to all data except y_i, and define the *ordinary cross validation* score

$$\mathcal{V}_o = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - y_i)^2.$$

This score results from leaving out each datum in turn, fitting the model to the remaining data and calculating the squared difference between the missing datum and its predicted value: these squared differences are then averaged over all the data. Substituting $y_i = f_i + \epsilon_i$,

$$\begin{aligned}\mathcal{V}_o &= \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - f_i - \epsilon_i)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - f_i)^2 - 2(\hat{f}_i^{[-i]} - f_i)\epsilon_i + \epsilon_i^2.\end{aligned}$$

Since $\mathbb{E}(\epsilon_i) = 0$, and ϵ_i and $\hat{f}_i^{[-i]}$ are independent, the second term in the summation vanishes if expectations are taken:

$$\mathbb{E}(\mathcal{V}_o) = \frac{1}{n}\mathbb{E}\left(\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - f_i)^2\right) + \sigma^2.$$

Now, $\hat{f}^{[-i]} \approx \hat{f}$ with equality in the large sample limit, so $\mathbb{E}(\mathcal{V}_o) \approx \mathbb{E}(M) + \sigma^2$ also with equality in the large sample limit. Hence choosing λ in order to minimize $\mathcal{V}_o$ is a reasonable approach if the ideal would be to minimize M. Choosing λ to minimize $\mathcal{V}_o$ is known as ordinary cross validation.

Ordinary cross validation is a reasonable approach, in its own right, even without a mean square (prediction) error justification. If models are judged only by their ability to fit the data from which they were estimated, then complicated models are

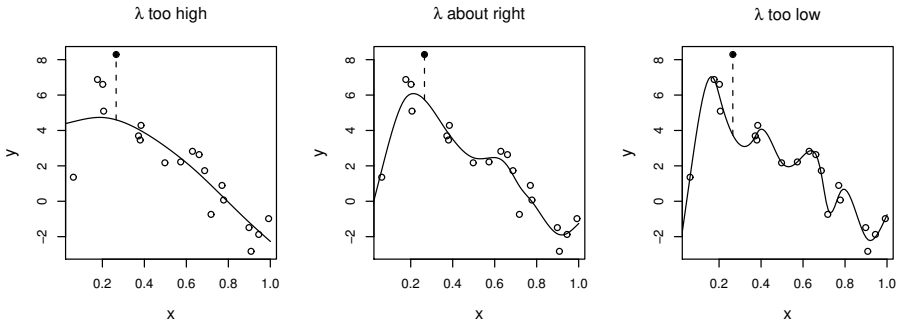

Figure 4.8 *Illustration of the principle behind cross validation. The fifth datum (•) has been omitted from fitting and the continuous curve shows a penalized regression spline fitted to the remaining data (○). When the smoothing parameter is too high the spline fits many of the data poorly and does no better with the missing point. When λ is too low the spline fits the noise as well as the signal and the consequent extra variability causes it to predict the missing datum poorly. For intermediate λ the spline is fitting the underlying signal quite well, but smoothing through the noise: hence, the missing datum is reasonably well predicted. Cross validation leaves out each datum from the data in turn and considers the average ability of models fitted to the remaining data to predict the omitted data.*

always selected over simpler ones. Choosing a model in order to maximize the ability to predict data to which the model was not fitted, does not suffer from this problem, as figure 4.8 illustrates.

It is computationally costly to calculate $\mathcal{V}_o$ by leaving out one datum at a time, refitting the model to each of the n resulting data sets, but it can be shown that

$$\mathcal{V}_o = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_i)^2/(1 - A_{ii})^2,$$

where $\hat{f}$ is the estimate from fitting to all the data, and $\mathbf{A}$ is the corresponding influence matrix (see section 6.2.2, p. 256). In practice the A_{ii} are often replaced by their mean, $\mathrm{tr}\,(\mathbf{A})/n$, resulting in the *generalized cross validation* score

$$\mathcal{V}_g = \frac{n\sum_{i=1}^{n}(y_i - \hat{f}_i)^2}{[n - \mathrm{tr}\,(\mathbf{A})]^2}.$$

GCV has computational advantages over OCV, and it also has advantages in terms of invariance (see Wahba, 1990, p.53 or sections 6.2.2 and 6.2.3, p. 258). In any case, it can also be shown to minimize $\mathbb{E}(M)$ in the large sample limit.

Returning to the engine wear example, a simple direct search for the GCV optimal smoothing parameter can be made as follows.

```
rho = seq(-9,11,length=90)
n <- length(wear)
V <- rep(NA,90)
```

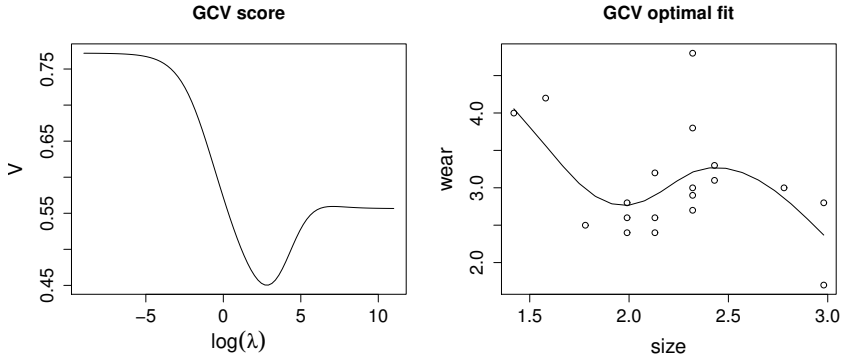

Figure 4.9 *Left panel: the GCV function for the engine wear example against log smoothing parameter. Right panel: the fitted model which minimizes the GCV score.*

```
for (i in 1:90) { ## loop through smoothing params
  b <- prs.fit(wear,size,sj,exp(rho[i])) ## fit model
  trF <- sum(influence(b)$hat[1:n])      ## extract EDF
  rss <- sum((wear-fitted(b)[1:n])^2)    ## residual SS
  V[i] <- n*rss/(n-trF)^2                ## GCV score
}
```

Note that the `influence()` function returns a list of diagnostics including `hat`, an array of the elements on the leading diagonal of the influence/hat matrix of the augmented model. The first n of these are the leading diagonal of the influence matrix of the penalized model (see exercise 4).

For the example, `V[54]` is the lowest GCV score, so that the optimal smoothing parameter, from those tried, is $\hat{\lambda} \approx 18$. Plots of the GCV score and the optimal model are easily produced

```
plot(rho,V,type="l",xlab=expression(log(lambda)),
                    main="GCV score")
sp <- exp(rho[V==min(V)])      ## extract optimal sp
b <- prs.fit(wear,size,sj,sp) ## re-fit
plot(size,wear,main="GCV optimal fit")
lines(s,Xp %*% coef(b))
```

The results are shown in figure 4.9.

4.2.4 The Bayesian/mixed model alternative

At some level we introduce smoothing penalties because we believe that the truth is more likely to be smooth than wiggly. We might as well formalise this belief in a Bayesian way, and specify a prior distribution on function wiggliness. Perhaps the simplest choice is an exponential prior

$$\propto \exp(-\lambda \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S} \boldsymbol{\beta} / \sigma^2)$$

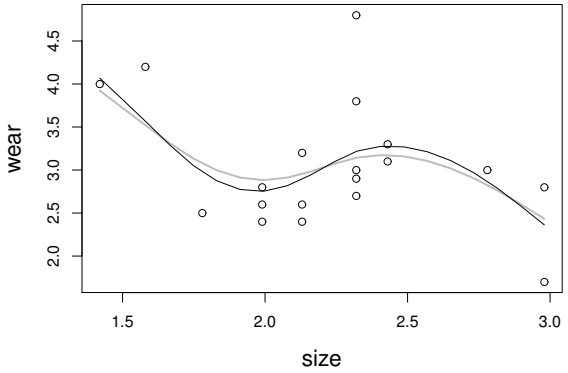

Figure 4.10 *Smooth model fits to the engine wear data with smoothing parameters estimated using marginal likelihood maximization (grey) or REML (black).*

(where scaling by σ^2 is introduced merely for later convenience), but this is immediately recognisable as being equivalent to an improper multivariate normal prior $\boldsymbol{\beta} \sim N(\mathbf{0}, \sigma^2 \mathbf{S}^-/\lambda)$. That is, the prior precision matrix is proportional to $\mathbf{S}$: because $\mathbf{S}$ is rank deficient by the dimension of the penalty null space, the prior covariance matrix is proportional to the pseudo-inverse[†] $\mathbf{S}^-$.

This Bayesian interpretation of the smoothing penalty gives the model the structure of a linear mixed model as discussed in chapter 2, and in consequence the MAP estimate of $\boldsymbol{\beta}$ is the solution (4.7) to (4.6), while

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{S})^{-1}\sigma^2)$$

— the Bayesian posterior distribution of $\boldsymbol{\beta}$ (this is equivalent to (2.17), p. 80). Also, having given the model this extra structure opens up the possibility of estimating σ^2 and λ using marginal likelihood maximization or REML.

In this section we will re-parameterize slightly to get the smooth model into a form such that its marginal likelihood can be evaluated using the simple routine `llm` from section 2.4.4 (p. 81). R routine `optim` can be used to fit the model. The same re-parameterization allows the model to be easily estimated using `lme` (see section 2.5, p. 86). As we will see in chapter 6, this re-parameterization is not necessary: it just simplifies matters for the moment, and perhaps makes the relationship between fixed effects and the penalty null space clearer than might otherwise be the case.

The re-parameterization is to re-write the model in terms of parameters, $\boldsymbol{\beta}' = \mathbf{D}_+\boldsymbol{\beta}$ where

$$\mathbf{D}_+ = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0} \\ & \mathbf{D} \end{bmatrix}.$$

So we now have $\mathbf{X}\boldsymbol{\beta} = \mathbf{X}\mathbf{D}_+^{-1}\boldsymbol{\beta}'$ and $\boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta} = \sum_{i=3}^{k} \beta_i'^2$. If we write the first 2

[†] Consider eigen-decomposition $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\mathsf{T}$. Let $\boldsymbol{\Lambda}^-$ denote the diagonal matrix of the inverse of the non-zero eigenvalues, with zeroes in place of the inverse for any zero eigenvalues. Then $\mathbf{S}^- = \mathbf{U}\boldsymbol{\Lambda}^-\mathbf{U}^\mathsf{T}$.

elements of $\boldsymbol{\beta}'$ as $\boldsymbol{\beta}^*$ and the remainder as $\mathbf{b}$, the Bayesian smoothing prior becomes $\mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma^2/\lambda)$ (which is proper). $\boldsymbol{\beta}^*$ is completely unpenalized, so we treat this as a vector of fixed effects. To make the connection to a standard mixed model completely clear, let $\mathbf{X}^*$ now denote the first 2 columns of $\mathbf{X}\mathbf{D}_+^{-1}$, while $\mathbf{Z}$ is the matrix of the remaining columns. Then the smooth model has become

$$\mathbf{y} = \mathbf{X}^*\boldsymbol{\beta}^* + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}, \quad \mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma^2/\lambda), \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}\sigma^2)$$

which is self-evidently in the standard form of a linear mixed model given in section 2.3, (p.77). Here is the code to re-parameterize the model and estimate it using `optim` and `llm` from section 2.4.4 (p. 81):

```
X0 <- tf.X(size,sj)          ## X in original parameterization
D <- rbind(0,0,diff(diag(20),difference=2))
diag(D) <- 1                    ## augmented D
X <- t(backsolve(t(D),t(X0)))  ## re-parameterized X
Z <- X[,-c(1,2)]; X <- X[,1:2] ## mixed model matrices
## estimate smoothing and variance parameters...
m <- optim(c(0,0),llm,method="BFGS",X=X,Z=Z,y=wear)
b <- attr(llm(m$par,X,Z,wear),"b") ## extract coefficients
## plot results...
plot(size,wear)
Xp1 <- t(backsolve(t(D),t(Xp))) ## re-parameterized pred. mat.
lines(s,Xp1 %*% as.numeric(b),col="grey",lwd=2)
```

The resulting plot is shown in figure 4.10.

Estimation using REML via `lme` is also easy. In `lme` terms all the data belong to a single group, so to use `lme` we must create a dummy grouping variable enforcing this. A covariance matrix proportional to the identity matrix is then specified via the `pdIdent` function.

```
library(nlme)
g <- factor(rep(1,nrow(X)))         ## dummy factor
m <- lme(wear ~ X - 1, random=list(g = pdIdent(~ Z-1)))
lines(s,Xp1 %*% as.numeric(coef(m))) ## and to plot
```

The curve of the estimated smooth is shown in black in figure 4.10. Notice how the REML based estimate (black) is more variable than the ML based estimate (grey), as expected from section 2.4.5 (p. 83).

4.3 Additive models

Now suppose that two explanatory variables, x and v, are available for a response variable, y, and that a simple additive model structure,

$$y_i = \alpha + f_1(x_i) + f_2(v_i) + \epsilon_i, \tag{4.8}$$

is appropriate. α is an intercept parameter, the f_j are smooth functions, and the ϵ_i are independent $N(0, \sigma^2)$ random variables.

There are two points to note about this model. Firstly, the assumption of additive effects is a fairly strong one: $f_1(x) + f_2(v)$ is a quite restrictive special case of

the general smooth function of two variables $f(x, v)$. Secondly, the fact that the model now contains more than one function introduces an identifiability problem: f_1 and f_2 are each only estimable to within an additive constant. To see this, note that any constant could be simultaneously added to f_1 and subtracted from f_2, without changing the model predictions. Hence identifiability constraints have to be imposed on the model before fitting.

Provided that the identifiability issue is addressed, the additive model can be represented using penalized regression splines, estimated by penalized least squares and the degree of smoothing selected by cross validation or (RE)ML, in the same way as for the simple univariate model.

4.3.1 Penalized piecewise regression representation of an additive model

Each smooth function in (4.8) can be represented using a penalized piecewise linear basis. Specifically, let

$$f_1(x) = \sum_{j=1}^{k_1} b_j(x)\delta_j$$

where the δ_j are unknown coefficients, while the $b_j(x)$ are basis functions of the form (4.4), defined using a sequence of k_1 knots, x_j^*, evenly spaced over the range of x. Similarly

$$f_2(v) = \sum_{j=1}^{k_2} \mathcal{B}_j(v)\gamma_j$$

where the γ_j are the unknown coefficients and the $\mathcal{B}_j(v)$ are basis functions of the form (4.4), defined using a sequence of k_2 knots, v_j^*, evenly spaced over the range of v. Defining n-vector $\mathbf{f}_1 = [f_1(x_1), \ldots, f_1(x_n)]^\mathsf{T}$, we have $\mathbf{f}_1 = \mathbf{X}_1\boldsymbol{\delta}$ where $b_j(x_i)$ is element i, j of $\mathbf{X}_1$. Similarly, $\mathbf{f}_2 = \mathbf{X}_2\boldsymbol{\gamma}$, where $\mathcal{B}_j(v_i)$ is element i, j of $\mathbf{X}_2$.

A penalty of the form (4.5) is also associated with each function: $\boldsymbol{\delta}^\mathsf{T}\mathbf{D}_1^\mathsf{T}\mathbf{D}_1\boldsymbol{\delta} = \boldsymbol{\delta}^\mathsf{T}\bar{\mathbf{S}}_1\boldsymbol{\delta}$ for f_1 and $\boldsymbol{\gamma}^\mathsf{T}\mathbf{D}_2^\mathsf{T}\mathbf{D}_2\boldsymbol{\gamma} = \boldsymbol{\gamma}^\mathsf{T}\bar{\mathbf{S}}_2\boldsymbol{\gamma}$ for f_2.

Now it is necessary to deal with the identifiability problem. For estimation purposes, almost any linear constraint that removed the problem could be used, but most choices lead to uselessly wide confidence intervals for the constrained functions. The best constraints from this viewpoint are sum-to-zero constraints, such as

$$\sum_{i=1}^{n} f_1(x_i) = 0, \quad \text{or equivalently } \mathbf{1}^\mathsf{T}\mathbf{f}_1 = 0,$$

where $\mathbf{1}$ is an n vector of 1's. Notice how this constraint still allows f_1 to have exactly the same shape as before constraint, with exactly the same penalty value. The constraint's only effect is to shift f_1, vertically, so that its mean value is zero.

To apply the constraint, note that we require $\mathbf{1}^\mathsf{T}\mathbf{X}_1\boldsymbol{\delta} = 0$ for all $\boldsymbol{\delta}$, which implies that $\mathbf{1}^\mathsf{T}\mathbf{X}_1 = \mathbf{0}$. To achieve this latter condition the column mean can be subtracted from each column of $\mathbf{X}_1$. That is, we define a column centred matrix

$$\tilde{\mathbf{X}}_1 = \mathbf{X}_1 - \mathbf{1}\mathbf{1}^\mathsf{T}\mathbf{X}_1/n$$

and set $\tilde{\mathbf{f}}_1 = \tilde{\mathbf{X}}_1\boldsymbol{\delta}$. It's easy to check that this constraint imposes no more than a shift in the level of $\mathbf{f}_1$:

$$\tilde{\mathbf{f}}_1 = \tilde{\mathbf{X}}_1\boldsymbol{\delta} = \mathbf{X}_1\boldsymbol{\delta} - \mathbf{1}\mathbf{1}^\mathsf{T}\mathbf{X}_1\boldsymbol{\delta}/n = \mathbf{X}_1\boldsymbol{\delta} - \mathbf{1}c = \mathbf{f}_1 - c$$

by definition of the scalar $c = \mathbf{1}^\mathsf{T}\mathbf{X}_1\boldsymbol{\delta}/n$. Finally note that the column centring reduces the rank of $\tilde{\mathbf{X}}_1$ to $k_1 - 1$, so that only $k_1 - 1$ elements of the k_1 vector $\boldsymbol{\delta}$ can be uniquely estimated. A simple identifiability constraint deals with this problem: a single element of $\boldsymbol{\delta}$ is set to zero, and the corresponding column of $\tilde{\mathbf{X}}_1$ and $\mathbf{D}$ is deleted.[‡] The column centred rank reduced basis will automatically satisfy the identifiability constraint. In what follows the tildes will be dropped, and it is assumed that the $\mathbf{X}_j$, $\mathbf{D}_j$, etc. are the constrained versions.

Here is an R function which produces constrained versions of $\mathbf{X}_j$ and $\mathbf{D}_j$.

```
tf.XD <- function(x,xk,cmx=NULL,m=2) {
## get X and D subject to constraint
  nk <- length(xk)
  X <- tf.X(x,xk)[,-nk]                    ## basis matrix
  D <- diff(diag(nk),differences=m)[,-nk] ## root penalty
  if (is.null(cmx)) cmx <- colMeans(X)
  X <- sweep(X,2,cmx)         ## subtract cmx from columns
  list(X=X,D=D,cmx=cmx)
}
```

`tf.XD` calls the functions producing the unconstrained basis and square root penalty matrices, given knot sequence `xk` and covariate values `x`. It drops a column of each resulting matrix and centres the remaining columns of the basis matrix. `cmx` is the vector of values to subtract from the columns of the `X`. For setting up a basis `cmx` should be `NULL`, in which case it is set to the column means of the basis matrix `X`. However, when using `tf.XD` to produce a basis matrix for *predicting* at new covariate values, it is essential that the basis matrix columns are centred using the same constants used for the *original* basis setup, so these must be supplied. Later code will clarify this.

Having set up constrained bases for the f_j it is now straightforward to re-express (4.8) as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\mathbf{X} = (\mathbf{1}, \mathbf{X}_1, \mathbf{X}_2)$ and $\boldsymbol{\beta}^\mathsf{T} = (\alpha, \boldsymbol{\delta}^\mathsf{T}, \boldsymbol{\gamma}^\mathsf{T})$. Largely for later notational convenience it is useful to express the penalties as quadratic forms in the full coefficient vector $\boldsymbol{\beta}$, which is easily done by simply padding out $\bar{\mathbf{S}}_j$ with zeroes, as appropriate. For example,

$$\boldsymbol{\beta}^\mathsf{T}\mathbf{S}_1\boldsymbol{\beta} = (\alpha, \boldsymbol{\delta}^\mathsf{T}, \boldsymbol{\gamma}^\mathsf{T}) \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{S}}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \alpha \\ \boldsymbol{\delta} \\ \boldsymbol{\gamma} \end{bmatrix} = \boldsymbol{\delta}^\mathsf{T}\bar{\mathbf{S}}_1\boldsymbol{\delta}.$$

[‡]The recipe given here is applicable to any basis which includes the constant function in its span, and has a penalty that is zero for constant functions. However, for bases that explicitly include a constant function, it is not sufficient to set any coefficient to zero: the coefficient for the constant is the one to constrain.

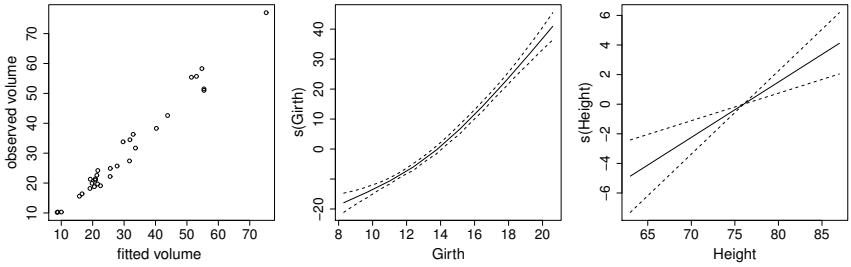

Figure 4.11 *The best fit two term additive model for the* `tree` *data. The left panel shows actual versus predicted tree volumes. The middle panel is the estimate of the smooth function of girth. The right panel is the estimate of the smooth function of height.*

4.3.2 Fitting additive models by penalized least squares

The coefficient estimates $\hat{\boldsymbol{\beta}}$ of the model (4.8) are obtained by minimization of the penalized least squares objective

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_1 \boldsymbol{\beta} + \lambda_2 \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_2 \boldsymbol{\beta},$$

where the smoothing parameters λ_1 and λ_2 control the weight to be given to the objective of making f_1 and f_2 smooth, relative to the objective of closely fitting the response data. For the moment, assume that these smoothing parameters are given.

Similarly to the single smooth case we have

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda_1 \mathbf{S}_1 + \lambda_2 \mathbf{S}_2\right)^{-1} \mathbf{X}^{\mathsf{T}} \mathbf{y} \text{ and } \mathbf{A} = \mathbf{X}\left(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda_1 \mathbf{S}_1 + \lambda_2 \mathbf{S}_2\right)^{-1} \mathbf{X}^{\mathsf{T}},$$

but again these expressions are sub-optimal with regard to computational stability and it is better to re-write the objective as

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_1 \boldsymbol{\beta} + \lambda_2 \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_2 \boldsymbol{\beta} = \left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \mathbf{B} \end{bmatrix} \boldsymbol{\beta} \right\|^2, \tag{4.9}$$

where

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} & \sqrt{\lambda_1}\mathbf{D}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sqrt{\lambda_2}\mathbf{D}_2 \end{bmatrix}$$

(or any other matrix such that $\mathbf{B}^{\mathsf{T}}\mathbf{B} = \lambda_1 \mathbf{S}_1 + \lambda_2 \mathbf{S}_2$).

As in the single smooth case, the right hand side of (4.9) is simply the unpenalized least squares objective for an augmented version of the model and corresponding response data. Hence, the model can be fitted by standard linear regression using stable orthogonal matrix based methods.

Here is a function to set up and fit a simple two term additive model, assuming the same number of knots for each smooth.

```
am.fit <- function(y,x,v,sp,k=10) {
  ## setup bases and penalties...
  xk <- seq(min(x),max(x),length=k)
  xdx <- tf.XD(x,xk)
  vk <- seq(min(v),max(v),length=k)
  xdv <- tf.XD(v,vk)
  ## create augmented model matrix and response...
  nD <- nrow(xdx$D)*2
  sp <- sqrt(sp)
  X <- cbind(c(rep(1,nrow(xdx$X)),rep(0,nD)),
             rbind(xdx$X,sp[1]*xdx$D,xdv$D*0),
             rbind(xdv$X,xdx$D*0,sp[2]*xdv$D))
  y1 <- c(y,rep(0,nD))
  ## fit model..
  b <- lm(y1 ~ X - 1)
  ## compute some useful quantities...
  n <- length(y)
  trA <- sum(influence(b)$hat[1:n]) ## EDF
  rsd <- y - fitted(b)[1:n] ## residuals
  rss <- sum(rsd^2)         ## residual SS
  sig.hat <- rss/(n-trA)       ## residual variance
  gcv <- sig.hat*n/(n-trA)     ## GCV score
  Vb <- vcov(b)*sig.hat/summary(b)$sigma^2 ## coeff cov matrix
  ## return fitted model...
  list(b=coef(b),Vb=Vb,edf=trA,gcv=gcv,fitted=fitted(b)[1:n],
       rsd=rsd,xk=list(xk,vk),cmx=list(xdx$cmx,xdv$cmx))
}
```

In addition to the quantities that we met in the single smooth case, `am.fit` also returns an estimate of the Bayesian covariance matrix for the model coefficients:

$$\hat{\mathbf{V}}_\beta = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda_1\mathbf{S}_1 + \lambda_2\mathbf{S}_2)^{-1}\hat{\sigma}^2$$

where $\hat{\sigma}^2$ is taken as the residual sum of squares for the fitted model, divided by the effective residual degrees of freedom. Following section 4.2.4 the posterior distribution for $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, \mathbf{V}_\beta), \tag{4.10}$$

and this result can be used for further inference about $\boldsymbol{\beta}$ (see section 6.10, p. 293).

Let us use the routine to estimate an additive model for the data in R data frame `trees`. The data are `Volume`, `Girth` and `Height` for 31 felled cherry trees. Interest lies in predicting `Volume`, and we can try estimating the model

$$\texttt{Volume}_i = \alpha + f_1(\texttt{Girth}_i) + f_2(\texttt{Height}_i) + \epsilon_i.$$

Now that we have two smoothing parameters, grid searching for the GCV optimal values starts to become inefficient. Instead R function `optim` can be used to minimize the GCV score. The function to be optimised has to be in a particular form for use with `optim`: the optimization parameter vector must be the first argument, and the function must be real valued. A simple wrapper for `am.fit` suffices:

```
am.gcv <- function(lsp,y,x,v,k) {
## function suitable for GCV optimization by optim
  am.fit(y,x,v,exp(lsp),k)$gcv
}
```

Using log smoothing parameters for optimization ensures that the estimated smoothing parameters are non-negative. Fitting the model is now straightforward

```
## find GCV optimal smoothing parameters...
fit <- optim(c(0,0), am.gcv, y=trees$Volume, x=trees$Girth,
             v=trees$Height,k=10)
sp <- exp(fit$par) ## best fit smoothing parameters
## Get fit at GCV optimal smoothing parameters...
fit <- am.fit(trees$Volume,trees$Girth,trees$Height,sp,k=10)
```

Now let's plot the smooth effects. The following function will do this.

```
am.plot <- function(fit,xlab,ylab) {
## produces effect plots for simple 2 term
## additive model
  start <- 2 ## where smooth coeffs start in beta
  for (i in 1:2) {
    ## sequence of values at which to predict...
    x <- seq(min(fit$xk[[i]]), max(fit$xk[[i]]), length=200)
    ## get prediction matrix for this smooth...
    Xp <- tf.XD(x, fit$xk[[i]], fit$cmx[[i]])$X
    ## extract coefficients and cov matrix for this smooth
    stop <- start + ncol(Xp)-1; ind <- start:stop
    b <- fit$b[ind];Vb <- fit$Vb[ind,ind]
    ## values for smooth at x...
    fv <- Xp %*% b
    ## standard errors of smooth at x....
    se <- rowSums((Xp %*% Vb) * Xp)^.5
    ## 2 s.e. limits for smooth...
    ul <- fv + 2 * se; ll <- fv - 2 * se
    ## plot smooth and limits...
    plot(x, fv, type="l", ylim=range(c(ul,ll)), xlab=xlab[i],
         ylab=ylab[i])
    lines(x, ul, lty=2); lines(x, ll, lty=2)
    start <- stop + 1
  }
}
```

Calling it with the fitted tree model

```
par(mfrow=c(1,3))
plot(fit$fitted,trees$Vol,xlab="fitted volume ",
     ylab="observed volume")
am.plot(fit,xlab=c("Girth","Height"),
        ylab=c("s(Girth)","s(Height)"))
```

gives the result in figure 4.11. Notice that the smooth of `Height` is estimated to be a straight line, and as a result its confidence interval has zero width at some point.

The zero width point in the interval occurs because the sum to zero constraint exactly determines where the straight line must pass through zero.

As with the one dimensional smooth, the additive model could also be estimated as a linear mixed model, but let us move on.

4.4 Generalized additive models

Generalized additive models (GAMs) follow from additive models, as generalized linear models follow from linear models. That is, the linear predictor now predicts some known smooth monotonic function of the expected value of the response, and the response may follow any exponential family distribution, or simply have a known mean variance relationship, permitting the use of a quasi-likelihood approach. The resulting model has a general form something like (4.1) in section 4.1.

As an illustration, suppose that we would like to model the `trees` data using a GAM of the form:

$$\log\{\mathbb{E}(\texttt{Volume}_i)\} = f_1(\texttt{Girth}_i) + f_2(\texttt{Height}_i), \quad \texttt{Volume}_i \sim \text{ gamma.}$$

This model is perhaps more natural than the additive model, as we might expect volume to be the product of some function of girth and some function of height, and it is reasonable to expect the variance in volume to increase with mean volume.

Whereas the additive model was estimated by penalized least squares, the GAM will be fitted by penalized likelihood maximization, and in practice this will be achieved by penalized iterative least squares (PIRLS).[§] For given smoothing parameters, the following steps are iterated to convergence.

1. Given the current linear predictor estimate, $\hat{\boldsymbol{\eta}}$, and corresponding estimated mean response vector, $\hat{\boldsymbol{\mu}}$, calculate:

$$w_i = \frac{1}{V(\hat{\mu}_i)g'(\hat{\mu}_i)^2} \quad \text{and} \quad z_i = g'(\hat{\mu}_i)(y_i - \hat{\mu}_i) + \hat{\eta}_i$$

   where $\text{var}(Y_i) = V(\mu_i)\phi$, as in section 3.1.2, and g is the link function.

2. Defining $\mathbf{W}$ as the diagonal matrix such that $W_{ii} = w_i$, minimize

$$\|\sqrt{\mathbf{W}}\mathbf{z} - \sqrt{\mathbf{W}}\mathbf{X}\boldsymbol{\beta})\|^2 + \lambda_1\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_1\boldsymbol{\beta} + \lambda_2\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_2\boldsymbol{\beta}$$

   w.r.t. $\boldsymbol{\beta}$ to obtain new estimate $\hat{\boldsymbol{\beta}}$, and hence updated estimates $\hat{\boldsymbol{\eta}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ and $\hat{\mu}_i = g^{-1}(\hat{\eta}_i)$.

The penalized least squares problem at step 2 is exactly the problem already solved for the simple additive model. Note the link to section 3.4.1 (p. 148).

For the trees GAM, the link function, g, is the log function, so $g'(\mu_i) = \mu_i^{-1}$, while for the gamma distribution, $V(\mu_i) = \mu_i^2$ (see table 3.1, p. 104). Hence, for the log-link gamma model, we have:

$$w_i = 1 \quad \text{and} \quad z_i = (y_i - \hat{\mu}_i)/\hat{\mu}_i + \hat{\eta}_i.$$

[§] There is no simple trick to produce an unpenalized GLM whose likelihood is equivalent to the penalized likelihood of the GAM that we wish to fit.

So, given λ_1 and λ_2 it will be straightforward to obtain $\hat{\boldsymbol{\beta}}$, but what should be used as the GCV score for this model? A natural choice is to use the GCV score for the final linear model in the PIRLS iteration (although this choice is poor for binary data: see section 6.2, p. 255 for better performing alternatives). It is easy to show that this GCV score is equivalent to the usual GCV score, but with the Pearson statistic replacing the residual sum of squares. Obviously we could also estimate the smoothing parameters by exploiting the Bayesian/mixed model connection of section 4.2.4, and estimating the model as a generalized linear mixed model using the methods of section 3.4 (p. 147).

The following function implements the PIRLS loop for the log-gamma model, and returns the required GCV score in its return list.

```
gam.fit <- function(y,x,v,sp,k=10) {
## gamma error log link 2 term gam fit...
  eta <- log(y) ## get initial eta
  not.converged <- TRUE
  old.gcv <- -100 ## don't converge immediately
  while (not.converged) {
    mu <- exp(eta)  ## current mu estimate
    z <- (y - mu)/mu + eta ## pseudodata
    fit <- am.fit(z,x,v,sp,k) ## penalized least squares
    if (abs(fit$gcv-old.gcv)<1e-5*fit$gcv) {
      not.converged <- FALSE
    }
    old.gcv <- fit$gcv
    eta <- fit$fitted ## updated eta
  }
  fit$fitted <- exp(fit$fitted) ## mu
  fit
}
```

Again a simple wrapper is needed in order to optimize the GCV score using `optim`

```
gam.gcv <- function(lsp,y,x,v,k=10) {
  gam.fit(y,x,v,exp(lsp),k=k)$gcv
}
```

Now fitting and plotting proceeds exactly as in the simple additive case.

```
fit <- optim(c(0,0),gam.gcv,y=trees$Volume,x=trees$Girth,
             v=trees$Height,k=10)
sp <- exp(fit$par)
fit <- gam.fit(trees$Volume,trees$Girth,trees$Height,sp)
par(mfrow=c(1,3))
plot(fit$fitted,trees$Vol,xlab="fitted volume ",
     ylab="observed volume")
am.plot(fit,xlab=c("Girth","Height"),
        ylab=c("s(Girth)","s(Height)"))
```

The resulting plots are shown in figure 4.12.

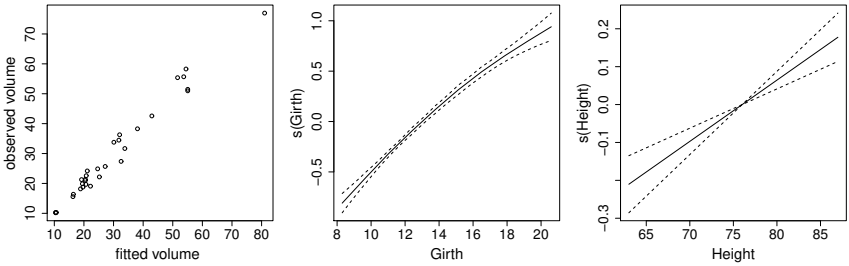

Figure 4.12 *The best fit two term generalized additive model for the* `tree` *data. The left panel shows actual versus predicted tree volumes. The middle panel is the estimate of the smooth function of girth. The right panel is the estimate of the smooth function of height.*

4.5 Summary

The preceding sections have illustrated how models based on smooth functions of predictor variables can be represented, and estimated, once a basis and wiggliness penalty have been chosen for each smooth in the model. Estimation is by penalized versions of the least squares and maximum-likelihood methods used for linear models and GLMs. Indeed technically GAMs are simply GLMs estimated subject to smoothing penalties. The most substantial difficulty introduced by this penalization is the need to select the degree of penalization, that is, to estimate the smoothing parameters. As we have seen, GCV provides one quite reasonable solution, and marginal likelihood provides an alternative.

The rest of this book will stick to the basic framework presented here, simply adding refinements to it. We will consider a variety of basis-penalty smoothers somewhat preferable to the piecewise linear basis given here, and some alternatives to GCV for smoothness estimation. More efficient and reliable computational methods will be developed, and the theoretical basis for inference will be more fully expounded. The link between smooths and random effects will also be developed, as will models based on linear functionals of smooths. However, throughout, functions are represented using penalized basis expansions, estimation of coefficients is by penalized likelihood maximisation and estimation of smoothing parameters uses a separate criterion, such as GCV or REML.

4.6 Introducing package `mgcv`

Before considering smoothers and GAM theory in more detail, it is worth briefly introducing the `mgcv` package. The `gam` function from `mgcv` is very much like the `glm` function covered in chapter 3. The main difference is that the `gam` model formula can include smooth terms, `s()` and `te()` (as well as the `te` variants `ti` and `t2`). Also there are a number of options available for controlling automatic smoothing parameter estimation, or for directly controlling model smoothness (summarized in table 4.1).

The cherry tree data provide a simple example with which to introduce the modelling functions available in R package mgcv.

```
library(mgcv)    ## load the package
data(trees)
ct1 <- gam(Volume ~ s(Height) + s(Girth),
           family=Gamma(link=log),data=trees)
```

This fits the generalized additive model

$$\log(\mathbb{E}[\texttt{Volume}_i]) = f_1(\texttt{Height}_i) + f_2(\texttt{Girth}_i) \quad \text{where } \texttt{Volume}_i \sim \text{gamma}$$

and the f_j are smooth functions. By default, the degree of smoothness of the f_j (within certain limits) is estimated by GCV. The results can be checked by typing the name of the fitted model object to invoke the `print.gam` print method, and by plotting the fitted model object. For example

```
> ct1

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth)

Estimated degrees of freedom:
1.00 2.42  total = 4.42

GCV score: 0.008082356
> plot(ct1,residuals=TRUE)
```

The resulting plot is displayed in the upper two panels of figure 4.13. Notice that the default print method reports the model distribution family, link function and formula, before displaying the effective degrees of freedom for each term (in the order that the terms appear in the model formula) and the whole model: in this case a nearly straight line, corresponding to about one degree of freedom, is estimated for the effect of height, while the effect of girth is estimated as a smooth curve with 2.4 degrees of freedom; the total degrees of freedom is the sum of these two, plus one degree of freedom for the model intercept. Finally, the GCV score for the fitted model is reported.

The plots show the estimated effects as solid lines/curves, with 95% confidence limits (strictly Bayesian credible intervals; see section 6.10, p. 293), based on (4.10), shown as dashed lines. The coincidence of the confidence limits and the estimated straight line, at the point where the line passes through zero on the vertical axis, is a result of the identifiability constraints applied to the smooth terms.[¶] The points shown on the plots are *partial residuals*. These are simply the Pearson residuals

[¶]The identifiability constraint is that the sum of the values of each curve, at the observed covariate values, must be zero: for a straight line, this condition exactly determines where the line must pass through zero, so there can be no uncertainty about this point.

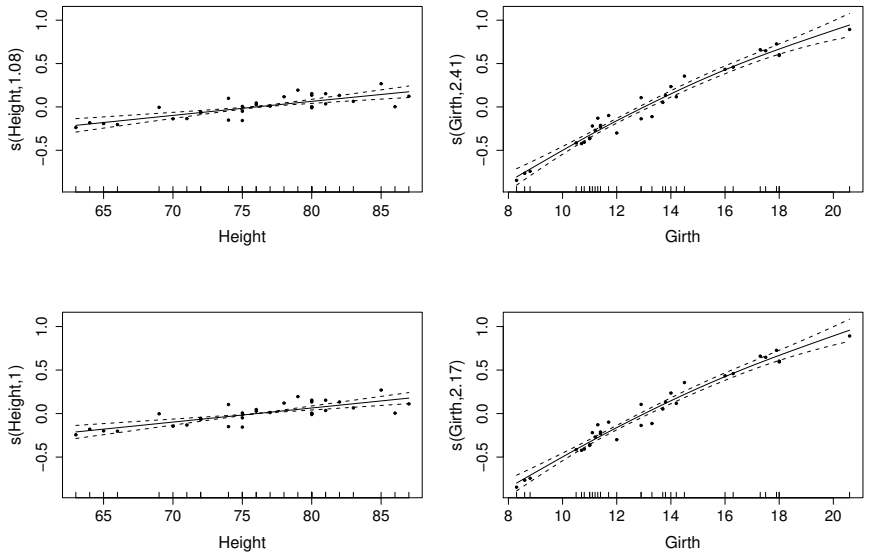

Figure 4.13 *Components of GAM model fits to the cherry tree data. The upper two panels are from* `ct1` *and the lower 2 from* `ct4`.

added to the smooth terms evaluated at the appropriate covariate values. For example, the residuals plotted in the top left panel of figure 4.13 are given by

$$\hat{\epsilon}_{1i}^{\text{partial}} = f_1(\texttt{Height}_\texttt{i}) + \hat{\epsilon}_i^p$$

plotted against $\texttt{Height}_i$. For a well fitting model the partial residuals should be evenly scattered around the curve to which they relate. The 'rug plots', along the bottom of each plot, show the values of the covariates of each smooth. The number in each y-axis caption is the effective degrees of freedom of the term being plotted.

4.6.1 Finer control of `gam`

The simple form of the `gam` call producing `ct1` hides a number of options that have been set to default values. The first of these is the choice of basis used to represent the smooth terms. The default is to use thin plate regression splines (section 5.5.1, p. 215), which have some appealing properties, but can be somewhat computationally costly for large data sets. The full range of available smoothers is covered in chapter 5. In the following, penalized cubic regression splines are selected using `s(...,bs="cr")`.

```
> ct2 <- gam(Volume ~ s(Height,bs="cr") + s(Girth,bs="cr"),
+            family=Gamma(link=log),data=trees)
> ct2
```

scale	The value of the scale parameter, or a negative value if it is to be estimated. For method="GCV.Cp" then scale > 0 implies Mallows' C_p/UBRE/AIC is used. scale $< 0 \Rightarrow$ implies GCV is used. scale $= 0 \Rightarrow$ UBRE/AIC for Poisson or binomial, otherwise GCV.
gamma	This multiplies the model degrees of freedom in the GCV or UBRE/AIC criteria. Hence as gamma is increased from 1 the 'penalty' per degree of freedom increases in the GCV or UBRE/AIC criterion and increasingly smooth models are produced. Increasing gamma to around 1.5 can usually reduce over-fitting, without much degradation in prediction error performance.
sp	An array of supplied smoothing parameters. When this array is non-null, a negative element signals that a smoothing parameter should be estimated, while a non-negative value is used as the smoothing parameter for the corresponding term. This is useful for directly controlling the smoothness of some terms.
method	Selects the smoothing parameter selection criterion: GCV.Cp, GACV, ML or REML.

Table 4.1 *Main arguments to* gam *for controlling the smoothness estimation process.*

```
Family: Gamma
Link function: log

Formula:
Volume ~ s(Height, bs = "cr") + s(Girth, bs = "cr")

Estimated degrees of freedom:
 1.000126 2.418591   total =  4.418718

GCV score:  0.008080546
```

As you can see, the change in basis has made very little difference to the fit. Plots are almost indistinguishable to those for ct1. This is re-assuring: it would be unfortunate if the model depended very strongly on details like the exact choice of basis. However, larger changes to the basis, such as using P-splines (section 5.3.3, p. 204), can make an appreciable difference.

Another choice, hidden in the previous two model fits, is the *dimension*, k, of the basis used to represent smooth terms. In the previous two fits, the (arbitrary) default, $k = 10$, was used. The choice of basis dimensions amounts to setting the *maximum* possible degrees of freedom allowed for each model term. The actual effective degrees of freedom for each term will usually be estimated from the data, by GCV or another smoothness selection criterion, but the upper limit on this estimate is $k - 1$: the basis dimension minus one degree of freedom due to the identifiability constraint on each smooth term. The following example sets k to 20 for the smooth of Girth (and illustrates, by the way, that there is no problem in mixing different bases).

```
> ct3 <- gam(Volume ~ s(Height) + s(Girth,bs="cr",k=20),
+            family=Gamma(link=log),data=trees)
> ct3

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth, bs = "cr", k = 20)

Estimated degrees of freedom:
 1.000003 2.424226   total =  4.424229

GCV score:  0.00808297
```

Again, this change makes boringly little difference in this case, and the plots (not shown) are indistinguishable from those for `ct1`. This insensitivity to basis dimension is not universal, of course, and checking of this choice is covered in section 5.9 (p. 242). One quite subtle point is worth being aware of. This is that a space of functions of dimension 20 will contain a larger subspace of functions with effective degrees of freedom 5, than will a function space of dimension 10 (the particular numbers being arbitrary here). Hence it is often the case that increasing k will change the effective degrees of freedom estimated for a term, even though both old and new estimated degrees of freedom are lower than the original $k - 1$.

Another choice is the parameter `gamma` which can be used to multiply the model effective degrees of freedom in the GCV or UBRE scores in order to (usually) increase the amount of smoothing selected. The default value is 1, but GCV is known to have some tendency to overfitting on occasion, and it has been suggested that using $\gamma \approx 1.5$ can somewhat correct this without compromising model fit (e.g., Kim and Gu, 2004). See section 6.2.4 for one justification. Applying this idea to the current model results in the bottom row of figure 4.13 and the following output.

```
> ct4 <- gam(Volume ~ s(Height) + s(Girth),
+            family=Gamma(link=log),data=trees,gamma=1.4)
> ct4

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth)

Estimated degrees of freedom:
 1.00011 2.169248   total =  4.169358

GCV score:  0.00922805

> plot(ct4,residuals=TRUE)
```

The heavier penalty on each degree of freedom in the GCV score has resulted in

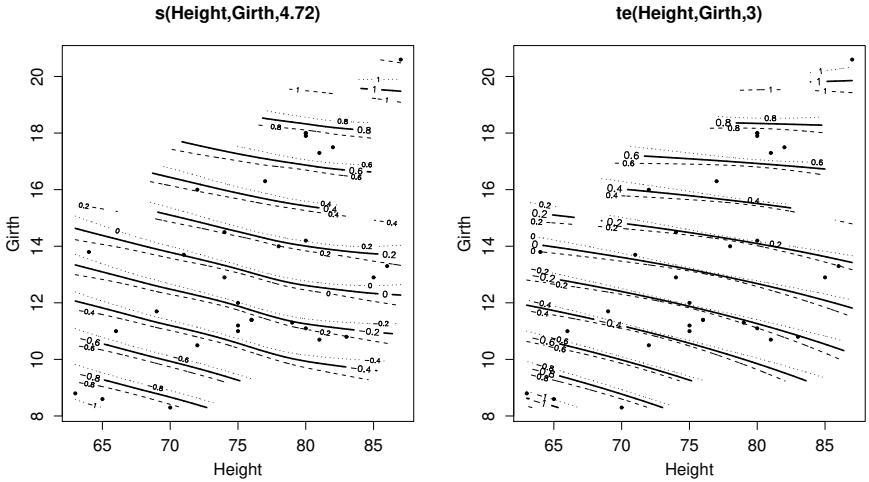

Figure 4.14 *Smooth functions of height and girth fitted to the cherry tree data, with degree of smoothing chosen by GCV. The left hand panel shows a thin plate regression spline fit (*`ct5`*), while the right panel shows a tensor product spline fit (*`ct6`*). For both plots the bold contours show the estimate of the smooth; the dashed contours show the smooth plus the standard error of the smooth and the dotted contours show the smooth less its standard error. The symbols show the locations of the covariate values on the height–girth plane. Parts of the smooths that are far away from covariate values have been excluded from the plots using the* `too.far` *argument to* `plot.gam`.

a model with fewer degrees of freedom, but the figure indicates that the change in estimates that this produces is barely perceptible.

4.6.2 Smooths of several variables

`gam` is not restricted to models containing only smooths of one predictor. In principle, smooths of any number of predictors are possible via two types of smooth. Within a model formula, `s()` terms, using the `"tp"`, `"ds"` or `"gp"` bases,[‖] produce isotropic smooths of multiple predictors, while `te()` terms produce smooths of multiple predictors from tensor products of *any* singly penalized bases available for use with `s()` (including mixtures of different bases). The tensor product smooths are invariant to linear rescaling of covariates, and can be quite computationally efficient. Alternative versions `t2()` and `ti()` are available for different sorts of functional ANOVA decomposition. Section 5.7 (p. 237) compares isotropic and tensor product smoothers.

[‖] Or indeed `"sos"` or `"so"` bases.

By way of illustration, the following code fragments both fit the model

$$\log(\mathbb{E}[\texttt{Volume}_i]) = f(\texttt{Height}_i, \texttt{Girth}_i) \text{ where } \texttt{Volume}_i \sim \text{gamma},$$

and f is a smooth function. Firstly an isotropic thin plate regression spline is used:

```
> ct5 <- gam(Volume ~ s(Height,Girth,k=25),
+            family=Gamma(link=log),data=trees)
> ct5

Family: Gamma
Link function: log

Formula:
Volume ~ s(Height, Girth, k = 25)

Estimated degrees of freedom:
 4.668129   total =  5.668129

GCV score:  0.009358786

> plot(ct5,too.far=0.15)
```

yielding the left hand panel of figure 4.14. Secondly a tensor product smooth is used. Note that the `k` argument to `te` specifies the dimension for each marginal basis: if different dimensions are required for the marginal bases then `k` can also be supplied as an array. The basis dimension of the tensor product smooth is the product of the dimensions of the marginal bases.

```
> ct6 <- gam(Volume ~ te(Height,Girth,k=5),
+            family=Gamma(link=log),data=trees)
> ct6

Family: Gamma
Link function: log

Formula:
Volume ~ te(Height, Girth, k = 5)

Estimated degrees of freedom:
 3.000175   total =  4.000175

GCV score:  0.008197151

> plot(ct6,too.far=0.15)
```

Notice how the tensor product model has fewer degrees of freedom and a lower GCV score than the TPRS smooth. In fact, with just 3 degrees of freedom, the tensor product smooth model amounts to

$$\log(\mathbb{E}[\texttt{Volume}_i]) = \beta_0 + \beta_1 \texttt{Height}_i + \beta_2 \texttt{Girth}_i + \beta_3 \texttt{Height}_i \texttt{Girth}_i,$$

the 'wiggly' components of the model having been penalized away altogether.

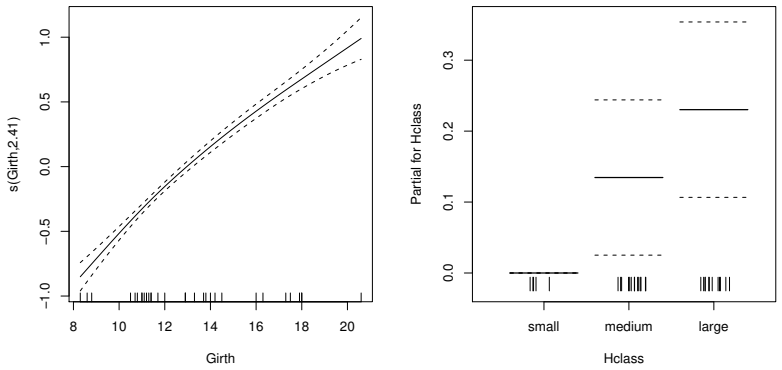

Figure 4.15 *Plot of model* `ct7`*, a* semi-parametric *model of cherry tree volume, with a factor for height and a smooth term for the dependence on girth. The left plot shows the smooth of girth, with 95% confidence interval, while the right panel shows the estimated effect, for each level of factor* `Hclass`*. The effect of being in the* `small` *height class is shown as zero, because the default contrasts have been used here, which set the parameter for the first level of each factor to zero.*

4.6.3 Parametric model terms

So far, only models consisting of smooth terms have been considered, but there is no difficulty in mixing smooth and parametric model components. For example, given that the model `ct1` smooth of height is estimated to be a straight line, we might as well fit the model:

```
gam(Volume ~ Height+s(Girth),family=Gamma(link=log),data=trees)
```

but to make the example more informative, let us instead suppose that the `Height` is actually only measured as a categorical variable. This can easily be arranged, by creating a factor variable which simply labels each tree as small, medium or large:

```
trees$Hclass <- factor(floor(trees$Height/10)-5,
                  labels=c("small","medium","large"))
```

Now we can fit a generalized additive model to these data, using the `Hclass` variable as a factor variable, and plot the result (figure 4.15).

```
ct7 <- gam(Volume ~ Hclass + s(Girth),
           family=Gamma(link=log), data=trees)
par(mfrow=c(1,2)); plot(ct7,all.terms=TRUE)
```

Often, more information about a fitted model is required than is supplied by plots or the default print method, and various utility functions exist to provide this. For example the `anova` function can be used to investigate the approximate significance of model terms.

```
> anova(ct7)

Family: Gamma
```

```
Link function: log

Formula:
Volume ~ Hclass + s(Girth)

Parametric Terms:
       df     F p-value
Hclass  2 7.076 0.00358

Approximate significance of smooth terms:
           edf Est.rank     F  p-value
s(Girth) 2.414    9.000 54.43 1.98e-14
```

Clearly there is quite strong evidence that both height and girth matter (see section 6.12, for information on the p-value calculations for the smooth terms). Similarly, an approximate AIC value can be obtained for the model (see section 6.11, p. 301):

```
> AIC(ct7)
[1] 154.9411
```

The `summary` method provides considerable detail.

```
> summary(ct7)

Family: Gamma
Link function: log

Formula:
Volume ~ Hclass + s(Girth)

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.12693    0.04814  64.949  < 2e-16 ***
Hclassmedium  0.13459    0.05428   2.479 0.020085 *
Hclasslarge   0.23024    0.06137   3.752 0.000908 ***

Approximate significance of smooth terms:
           edf Est.rank     F  p-value
s(Girth) 2.414    9.000 54.43 1.98e-14 ***

R-sq.(adj) =  0.967   Deviance explained = 96.9%
GCV score = 0.012076   Scale est. = 0.0099671  n = 31
```

Notice that, in this case, the significance of individual parameters of the parametric terms is given, rather than whole term significance. Other measures of fit are also reported, such as the adjusted r^2 and percentage deviance explained, along with the GCV score, an estimate of the scale parameter of the model, and the number of data fitted.

4.6.4 The mgcv *help pages*

mgcv has quite extensive help pages, both documenting functions and attempting to provide overviews of a topic. The easiest way to access the pages is via the HTML versions, by typing help.start() in R, then navigating to the mgcv pages and browsing. Several pages are well worth knowing about:

- mgcv-package offers an overview of the package and what it offers.
- family.mgcv gives an overview of the distributions available.
- smooth.terms gives an overview of the smooths types available.
- random.effects is an overview of random effects in mgcv.
- gam.models reviews some aspects of model specification; gam.selection covers model selection options.
- gam, bam, gamm and jagam cover the main modelling functions.

4.7 Exercises

1. This question is about illustrating the problems with polynomial bases. First run

```
set.seed(1)
x<-sort(runif(40)*10)^.5
y<-sort(runif(40))^0.1
```

   to simulate some apparently innocuous x, y data.

   (a) Fit 5^{th} and 10^{th} order polynomials to the simulated data using, e.g., lm(y~poly(x,5)).

   (b) Plot the x, y data, and overlay the fitted polynomials. (Use the predict function to obtain predictions on a fine grid over the range of the x data: only predicting at the data fails to illustrate the polynomial behaviour adequately).

   (c) One particularly simple basis for a cubic regression spline is $b_1(x) = 1$, $b_2(x) = x$ and $b_{j+2}(x) = |x - x_j^*|^3$ for $j = 1 \ldots q-2$, where q is the basis dimension, and the x_j^* are knot locations. Use this basis to fit a rank 11 cubic regression spline to the x, y data (using lm and evenly spaced knots).

   (d) Overlay the predicted curve according to the spline model, onto the existing x, y plot, and consider which basis you would rather use.

2. Polynomial models of the data from question 1 can also provide an illustration of why orthogonal matrix methods are preferable to fitting models by solution of the 'normal equations' $\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^\mathsf{T}\mathbf{y}$. The bases produced by poly are actually orthogonal polynomial bases, which are a numerically stable way of representing polynomial models, but if a naïve basis is used then a numerically badly behaved model can be created.

```
form<-paste("I(x^",1:10,")",sep="",collapse="+")
form <- as.formula(paste("y~",form))
```

   produces the model formula for a suitably ill-behaved model. Fit this model using lm, extract the model matrix from the fitted model object using model.matrix, and re-estimate the model parameters by solving the 'normal equations' given

above (see `?solve`). Compare the estimated coefficients in both cases, along with the fits. It is also instructive to increase the order of the polynomial by one or two and examine the results (and to decrease it to 5, say, in order to confirm that the QR and normal equations approaches agree if everything is 'well behaved'). Finally, note that the singular value decomposition (see B.10) provides a reliable way of diagnosing the linear dependencies that can cause problems when model fitting. `svd(X)$d` obtains the singular values of a matrix `X`. The largest divided by the smallest gives the 'condition number' of the matrix — a measure of how ill-conditioned computations with the matrix are likely to be.

3. Show that the $\boldsymbol{\beta}$ minimizing (4.6), in section 4.2.2, is given by (4.7).
4. Let $\mathbf{X}$ be an $n \times p$ model matrix, $\mathbf{S}$ a $p \times p$ penalty matrix, and $\mathbf{B}$ any matrix such that $\mathbf{B}^\mathsf{T}\mathbf{B} = \mathbf{S}$. If
$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{B} \end{bmatrix}$$
is an augmented model matrix, show that the sum of the first n elements on the leading diagonal of $\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^\mathsf{T}\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\mathsf{T}$ is $\mathrm{tr}\left(\mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X}+\mathbf{S})^{-1}\mathbf{X}^\mathsf{T}\right)$.
5. The 'obvious' way to estimate smoothing parameters is by treating smoothing parameters just like the other model parameters, $\boldsymbol{\beta}$, and to choose λ to minimize the residual sum of squares for the fitted model. What estimate of λ will such an approach always produce?
6. Show that for any function f, which has a basis expansion
$$f(x) = \sum_j \beta_j b_j(x),$$
it is possible to write
$$\int f''(x)^2 dx = \boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta},$$
where the coefficient matrix $\mathbf{S}$ can be expressed in terms of the known basis functions b_j (assuming that these possess at least two (integrable) derivatives). As usual $\boldsymbol{\beta}$ is a parameter vector with β_j in its j^{th} element.
7. Show that for any function f which has a basis expansion
$$f(x,z) = \sum_j \beta_j b_j(x,z),$$
it is possible to write
$$\int \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial z}\right)^2 + \left(\frac{\partial f^2}{\partial z^2}\right)^2 dxdz = \boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta},$$
where the coefficient matrix $\mathbf{S}$ can be expressed in terms of the known basis functions b_j (assuming that these possess at least two (integrable) derivatives w.r.t. x and z). Again, $\boldsymbol{\beta}$ is a parameter vector with β_j in its j^{th} element.
8. The additive model of section 4.3 can equally well be estimated as a mixed model.

(a) Write a function which converts the model matrix and penalty returned by `tf.XD` into mixed model form. Hint: because of the constraints the penalty null space is of dimension 1 now, leading to a slight modification of $\mathbf{D}_+$.

(b) Using your function from part (a) obtain the model matrices required to fit the two term additive tree model, and estimate it using `lme`. Because there are now two smooths, two `pdIdent` terms will be needed in the `random` list supplied to `lme`, which will involve two dummy grouping variables (which can just be differently named copies of the same variable).

(c) Produce residual versus fitted volume and raw volume against fitted volume plots.

(d) Produce plots of the two smooth effect estimates with partial residuals.

9. Following on from question 8, we can also estimate a GAM as a GLMM. This is particularly easy to implement using the PQL method of section 3.4.2 (p. 149).

(a) Modify the function `gam.fit` from section 4.4, so that in place of the call to `am.fit` there is an appropriate call to `lme` to estimate the coefficients and smoothing parameters of a working linear mixed model. The modified function should take a response vector and the model matrices from the previous question as inputs, and return the `lme` fitted model object for the working model at convergence.

(b) Use your function to fit the Gamma additive model of section 4.4 to the `trees` data.

(c) Produce plots of measured volume against predicted volume, and of residuals against the linear predictor of the model.

(d) Plot the smooth effects with partial residuals.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Chapter 5

Smoothers

The piecewise linear smoother of the previous chapter offers a perfectly reasonable way of representing the smooth functions in additive models, but substantial improvement is possible. In particular, if we represent the smooth model terms using *spline* bases, then it is possible to obtain substantially reduced function approximation error for a given dimension of smoothing basis. This chapter covers spline smoothers, and the equivalence between quadratically penalized smoothers and Gaussian random effects/fields.

The chapter starts by considering one dimensional spline smoothing, and then computationally efficient reduced rank penalized regression splines. Several one dimensional penalized regression smoothers are covered, including adaptive smoothers. Constraints, the 'natural' parameterization of a smooth and effective degrees of freedom are discussed next, followed by multidimensional smoothers. First isotropic smoothers: thin-plate and other Duchon splines, splines on the sphere and soap film smoothers. Isotropy is usually inappropriate when the arguments of a smooth have different units. Scale invariant 'tensor product' smoothers are therefore constructed next, and consideration given to the notion of a smooth interaction, and to decompositions involving smooth main effects and smooth interactions. There follows a discussion of the duality between smooths and Gaussian random effects/fields and of Gaussian Markov random field smoothers and Gaussian process smoothers. A slightly more advanced final section gives a brief introduction to the reproducing kernel Hilbert space approach to spline theory.

5.1 Smoothing splines

Almost all the smooths considered in this book are based in some way on splines, so it is worth spending a little time on the theoretical properties that make these functions so appealing for penalized regression. Rather than attempt full generality, the flavour of the theoretical ideas can be gleaned by considering some properties of cubic splines, first in the context of interpolation, and then of smoothing.

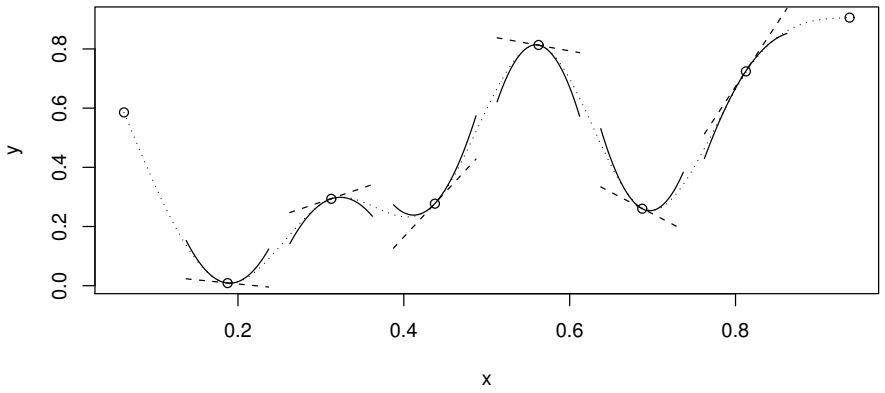

Figure 5.1 *A cubic spline is a curve constructed from sections of cubic polynomial joined together so that the curve is continuous up to second derivative. The spline shown (dotted curve) is made up of 7 sections of cubic. The points at which they are joined (∘) (and the two end points) are known as the* knots *of the spline. Each section of cubic has different coefficients, but at the knots it will match its neighbouring sections in value and first two derivatives. Straight dashed lines show the gradients of the spline at the knots and the continuous curves are quadratics matching the first and second derivatives at the knots: these illustrate the continuity of first and second derivatives across the knots. This spline has zero second derivatives at the end knots: a* 'natural *spline'. Note that there are many alternative ways of representing such a cubic spline using basis functions: although all are equivalent, the link to the piecewise cubic characterization is not always transparent.*

5.1.1 Natural cubic splines are smoothest interpolators

Consider a set of points $\{x_i, y_i : i = 1, \ldots, n\}$ where $x_i < x_{i+1}$. The *natural cubic spline*, $g(x)$,* interpolating these points, is a function made up of sections of cubic polynomial, one for each $[x_i, x_{i+1}]$, which are joined together so that the whole spline is continuous to second derivative, while $g(x_i) = y_i$ and $g''(x_1) = g''(x_n) = 0$. Figure 5.1 illustrates such a cubic spline.

Of all functions that are continuous on $[x_1, x_n]$, have absolutely continuous first derivatives and interpolate $\{x_i, y_i\}$, $g(x)$ is the one that is smoothest in the sense of minimizing:

$$J(f) = \int_{x_1}^{x_n} f''(x)^2 dx.$$

Green and Silverman (1994) provide a neat proof of this, based on the original work of Schoenberg (1964). Let $f(x)$ be an interpolant of $\{x_i, y_i\}$, other than $g(x)$,

*In this chapter g denotes a generic smooth function, rather than the link function.

and let $h(x) = f(x) - g(x)$. We seek an expression for $J(f)$ in terms of $J(g)$.

$$\begin{aligned}\int_{x_1}^{x_n} f''(x)^2 dx &= \int_{x_1}^{x_n} \{g''(x) + h''(x)\}^2 dx \\ &= \int_{x_1}^{x_n} g''(x)^2 dx + 2\int_{x_1}^{x_n} g''(x)h''(x)dx + \int_{x_1}^{x_n} h''(x)^2 dx\end{aligned}$$

and integrating the second term on the second line, by parts, yields

$$\begin{aligned}\int_{x_1}^{x_n} g''(x)h''(x)dx &= g''(x_n)h'(x_n) - g''(x_1)h'(x_1) - \int_{x_1}^{x_n} g'''(x)h'(x)dx \\ &= -\int_{x_1}^{x_n} g'''(x)h'(x)dx = -\sum_{i=1}^{n-1} g'''(x_i^+)\int_{x_i}^{x_{i+1}} h'(x)dx \\ &= -\sum_{i=1}^{n-1} g'''(x_i^+)\{h(x_{i+1}) - h(x_i)\} = 0,\end{aligned}$$

where equality of lines 1 and 2 follows from the fact that $g''(x_1) = g''(x_n) = 0$. The equalities in line 2 result from the fact that $g(x)$ is made up of sections of cubic polynomial, so that $g'''(x)$ is constant over any interval (x_i, x_{i+1}); x_i^+ denotes an element of such an interval. The final equality to zero follows from the fact that both $f(x)$ and $g(x)$ are interpolants, and are hence equal at x_i, implying that $h(x_i) = 0$.

So we have shown that

$$\int_{x_1}^{x_n} f''(x)^2 dx = \int_{x_1}^{x_n} g''(x)^2 dx + \int_{x_1}^{x_n} h''(x)^2 dx \geq \int_{x_1}^{x_n} g''(x)^2 dx$$

with equality only if $h''(x) = 0$ for $x_1 < x < x_n$. However, $h(x_1) = h(x_n) = 0$, so in fact we have equality if and only if $h(x) = 0$ on $[x_1, x_n]$. In other words, any interpolant that is not identical to $g(x)$ will have a higher integrated squared second derivative. So there is a well defined sense in which the cubic spline is the smoothest possible interpolant through any set of data.

The smoothest interpolation property is not the only good property of cubic spline interpolants. In de Boor (1978, Chapter 5) a number of results are presented showing that cubic spline interpolation is optimal, or at least very good, in various respects. For example, if a 'complete' cubic spline, g, is used to approximate a function, $\tilde{f}$, by interpolating a set of points $\{x_i, \tilde{f}(x_i) : i = 1, \ldots, n\}$ and matching $\tilde{f}'(x_1)$ and $\tilde{f}'(x_n)$ then if $\tilde{f}(x)$ has 4 continuous derivatives:

$$\max|\tilde{f} - g| \leq \frac{5}{384}\max(x_{i+1} - x_i)^4 \max|\tilde{f}''''|. \tag{5.1}$$

The inequality is 'sharp' (meaning that 5/384 can not be improved upon).

These properties of spline interpolants suggest that splines ought to provide a good basis for representing smooth terms in statistical models. Whatever the true underlying smooth function is, a spline ought to be able to approximate it closely, and if we want to construct models from smooth functions of covariates, then representing those functions from smoothest approximations is intuitively appealing.

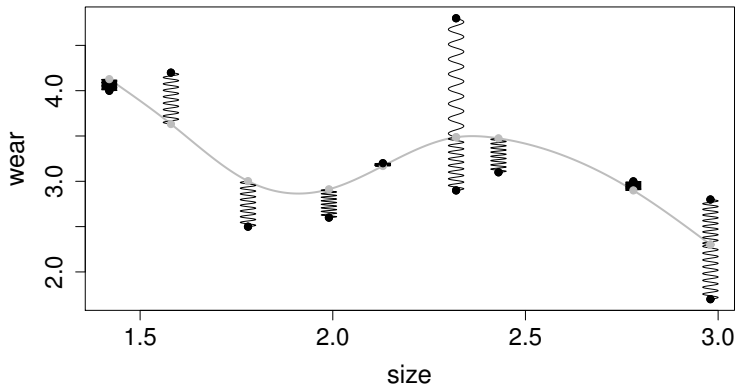

Figure 5.2 *Physical analogy for a cubic smoothing spline. The black points represent x, y data, while the grey curve is a thin flexible strip (e.g., of wood). If the strip is hooked up to the data by springs then the thin strip can be viewed as a* smoothing spline*. The springs are idealised here, having zero length under zero load, and only extending in the y direction. Also, strictly the analogy only holds if the vertical deflection of the grey curve is 'small'.*

5.1.2 Cubic smoothing splines

In statistical work y_i is usually measured with noise, and it is generally more useful to smooth x_i, y_i data, rather than interpolating them. To this end, rather than setting $g(x_i) = y_i$, it might be better to treat the $g(x_i)$ as n free parameters of the cubic spline, and to estimate them in order to minimize

$$\sum_{i=1}^{n} \{y_i - g(x_i)\}^2 + \lambda \int g''(x)^2 dx,$$

where λ is a tunable parameter, used to control the relative weight to be given to the conflicting goals of matching the data and producing a smooth g. The resulting $g(x)$ is a *smoothing spline* (Reinsch, 1967). Figure 5.2 illustrates the basic idea. In fact, of *all functions*, f, that are continuous on $[x_1, x_n]$, and have absolutely continuous first derivatives, $g(x)$ is the function minimizing:

$$\sum_{i=1}^{n} \{y_i - f(x_i)\}^2 + \lambda \int f''(x)^2 dx. \tag{5.2}$$

The proof is easy. Suppose that some other function, $f^*(x)$, minimized (5.2). In that case we could interpolate $\{x_i, f^*(x_i)\}$ using a cubic spline, $g(x)$. Now $g(x)$ and $f^*(x)$ have the same sum of squares term in (5.2), but by the properties of interpolating splines, $g(x)$ must have the lower integrated squared second derivative. Hence $g(x)$ yields a lower (5.2) than $f^*(x)$, and a contradiction, unless $f^* = g$.

Notice the obvious corollary that the same result holds substituting for the residual sum of squares any measure of fit dependent on f only via $f(x_1), \ldots f(x_n)$. In particular we might substitute a log likelihood.

So, rather than being chosen in advance, the cubic spline basis arises naturally from the specification of the smoothing objective (5.2), in which what is meant by model fit and by smoothness are precisely defined in a basis independent way.

Smoothing splines seem to be somewhat ideal smoothers. The only substantial problem is the fact that they have as many free parameters as there are data to be smoothed. This is wasteful, given that, in practice, λ will almost always be high enough that the resulting spline is much smoother than n degrees of freedom would suggest. Indeed, in section 5.4.2 we will see that many degrees of freedom of a spline are often suppressed completely by the penalty. For univariate smoothing with cubic splines the large number of parameters turns out not to be problematic (e.g. De Hoog and Hutchinson, 1987, give a stable $O(n)$ algorithm), but as soon as we try to deal with more covariates the computational expense becomes severe.

5.2 Penalized regression splines

An obvious compromise between retaining the good properties of splines and computational efficiency, is to use penalized regression splines, as introduced in Chapter 4. At its simplest, this involves constructing a spline basis (and associated penalties) for a much smaller data set than the one to be analysed, and then using that basis (plus penalties) to model the original data set. The covariate values in the smaller data set should be arranged to nicely cover the distribution of covariate values in the original data set. This penalized regression spline idea is presented in Wahba (1980) and Parker and Rice (1985), for example. An alternative approach is based on eigen-approximation and is covered in section 5.5.1.

Use of penalized regression splines raises the question of how many basis functions to use. No answer to this is generally possible without knowing the true functions that we are trying to estimate, but it is possible to say something about how the basis dimension should scale with sample size, n, as $n \to \infty$.

Consider the case of cubic interpolating spline approximation of a function $g(x)$, where the spline has k knots, spaced evenly along the x axis. We saw in section 5.1.1 that the approximation error of such a spline is $O(k^{-4})$. See figure 5.3(a). Now suppose that instead of interpolating g we have n/k noisy observations of g at each knot, as illustrated in figure 5.3(b). Without penalisation, the spline estimate of g is just the spline interpolant of the mean observation at each knot. Hence the standard error of the spline estimate is $O(\sqrt{k/n})$. For fixed k, in the $n/k \to \infty$ limit, the standard error vanishes, but the approximation error bias remains at $O(k^{-4})$. To allow the approximation error to also vanish in the limit we require k to grow with n. If we want neither the $O(\sqrt{k/n})$ standard error, nor the $O(k^{-4})$ bias to dominate in the $n \to \infty$ limit, then we should set $k = O(n^{1/9})$ to obtain a mean square error rate of $O(n^{-8/9})$.

Obviously the situation in which all the observations are at the knots is not the one that we usually face, so consider the regression spline with k evenly spaced knots, but where the observations are spread over the domain (in a non-pathological manner so that the standard error at any particular point within the range of the knots is still $O(\sqrt{k/n})$). Let $\hat{g}_\infty$ again denote the spline interpolating g at the knots. From

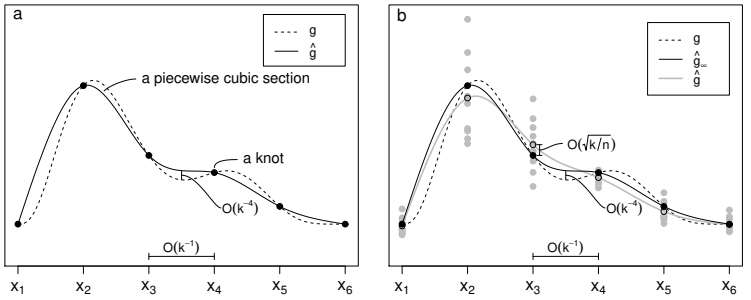

Figure 5.3 *(a) Cubic interpolating spline approximation $\hat{g}$ to function g observed at k knots x_j without error. The x-axis knot spacing is $O(k^{-1})$ so that the approximation error is $O(k^{-4})$. (b) Unpenalized cubic regression spline estimate, $\hat{g}$, to function g noisily observed n/k times at each x_j. The standard error of $\hat{g}$ is $O(\sqrt{k/n})$. In the $n/k \to \infty$ limit, the spline, $\hat{g}_\infty$, coincides with the interpolating spline in a, and has approximation error $O(k^{-4})$. To avoid approximation error or sampling error dominating in the $n \to \infty$ limit requires $k = O(n^{1/9})$.*

(5.1) the elements of $\mathbf{g} - \hat{\mathbf{g}}_\infty$ are bounded above by a constant $O(k^{-4})$, and so $n^{-1}\|\mathbf{g} - \hat{\mathbf{g}}_\infty\|$ must also be bounded above by a constant $O(k^{-4})$. Now in the large sample limit, the regression spline, $\hat{\mathbf{g}}$, minimises $n^{-1}\|\mathbf{g} - \hat{\mathbf{g}}\|$ (see section 1.8.7, p. 54). This must also be bounded above by a constant $0(k^{-4})$ since otherwise we could reduce $n^{-1}\|\mathbf{g} - \hat{\mathbf{g}}\|$ by substituting $\hat{g}_\infty$ for $\hat{g}$. $\hat{g}$ can not have a lower order bias term than this in general, since for some designs $\hat{g} = \hat{g}_\infty$ which has $O(k^{-4})$ bias. In the general regression spline case the average variance at the observation locations is $\text{tr}(\mathbf{A})\sigma^2/n = \sigma^2 k/n$, so again $k = O(n^{1/9})$ is required for optimality. See Agarwal and Studden (1980) for the original result.

Under penalization the situation is slightly more complicated. We could use $k = O(n^{1/9})$ and achieve an asymptotic mean square error rate of $O(n^{-8/9})$, but under smoothing parameter selection, this will often imply having no penalization in the large sample limit. In reality, when smoothing parameters are estimated, we would usually view a λ estimate near zero as a possible indication that k is too low, and therefore increase it. Hence asymptotics in which penalization persists in the large sample limit are more useful. For the cubic spline under REML smoothing parameter estimation this turns out to require that $k = O(n^{1/5})$, in which case the penalized cubic regression spline can achieve a mean square error of $O(n^{-4/5})$. This is somewhat below the optimal rate achievable by a non-parametric estimator of a 4^{th} order continuous function which Cox (1983) shows is $O(n^{-8/9})$. This optimal rate can be achieved by cubic smoothing splines (Stone, 1982; Speckman, 1985) and penalized cubic regression splines (Hall and Opsomer, 2005), under particular assumptions about the rate at which the smoothing parameters change with n. Unfortunately there is little work on the rates actually achieved when smoothing parameters are estimated: Kauermann et al. (2009) is a notable exception, considering estimation under REML smoothing parameter selection, but assuming $k = O(n^{1/9})$ in the cu-

Basis functions for a cubic spline

$$a_j^-(x) = (x_{j+1} - x)/h_j \qquad c_j^-(x) = [(x_{j+1} - x)^3/h_j - h_j(x_{j+1} - x)]/6$$

$$a_j^+(x) = (x - x_j)/h_j \qquad c_j^+(x) = [(x - x_j)^3/h_j - h_j(x - x_j)]/6$$

Non-zero matrix elements — non-cyclic spline

$$D_{i,i} = 1/h_i \qquad D_{i,i+1} = -1/h_i - 1/h_{i+1} \qquad D_{i,i+2} = 1/h_{i+1}$$
$$B_{i,i} = (h_i + h_{i+1})/3 \qquad\qquad i = 1 \ldots k-2$$

$$B_{i,i+1} = h_{i+1}/6 \qquad B_{i+1,i} = h_{i+1}/6 \qquad i = 1 \ldots k-3$$

Non-zero matrix elements — cyclic spline

$$\tilde{B}_{i-1,i} = \tilde{B}_{i,i-1} = h_{i-1}/6 \qquad \tilde{B}_{i,i} = (h_{i-1} + h_i)/3$$
$$\tilde{D}_{i-1,i} = \tilde{D}_{i,i-1} = 1/h_{i-1} \qquad \tilde{D}_{i,i} = -1/h_{i-1} - 1/h_i \qquad i = 2 \ldots k-1$$

$$\tilde{B}_{1,1} = (h_{k-1} + h_1)/3 \qquad \tilde{B}_{1,k-1} = h_{k-1}/6 \qquad \tilde{B}_{k-1,1} = h_{k-1}/6$$
$$\tilde{D}_{1,1} = -1/h_1 - 1/h_{k-1} \qquad \tilde{D}_{1,k-1} = 1/h_{k-1} \qquad \tilde{D}_{k-1,1} = 1/h_{k-1}$$

Table 5.1 *Definitions of basis functions and matrices used to define a cubic regression spline.* $h_j = x_{j+1} - x_j$.

bic regression spline case, which corresponds to not penalizing in the large sample limit. Claeskens et al. (2009) explicitly deal with the asymptotically penalized and unpenalized regimes separately, but not under smoothing parameter estimation.

Despite the obvious deficiencies in the theory, the upshot of this sort of analysis is that we really only need the basis dimension to grow rather slowly with sample size in order to achieve statistical performance asymptotically indistinguishable from that of a full smoothing spline.

5.3 Some one-dimensional smoothers

This section covers the explicit representation of some basis-penalty smoothers useful for smoothing with respect to a single predictor variable. Cubic penalized regression splines are covered first, along with their cyclic version. P-splines are then discussed, and used to create an adaptive smoother. Note that the thin plate regression splines of section 5.5.1 can also be used for one-dimensional smoothing.

5.3.1 Cubic regression splines

There are many equivalent bases that can be used to represent cubic splines. One approach is to parameterize the spline in terms of its values at the knots. Consider

defining a cubic spline function, $f(x)$, with k knots, $x_1 \ldots x_k$. Let $\beta_j = f(x_j)$ and $\delta_j = f''(x_j)$. Then the spline can be written as

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)\delta_j + c_j^+(x)\delta_{j+1} \text{ if } x_j \leq x \leq x_{j+1}, \quad (5.3)$$

where the basis functions a_j^-, a_j^+, c_j^- and c_j^+ are defined in table 5.1. The conditions that the spline must be continuous to second derivative, at the x_j, and should have zero second derivative at x_1 and x_k, can be shown to imply (exercise 1) that

$$\mathbf{B}\boldsymbol{\delta}^- = \mathbf{D}\boldsymbol{\beta}, \quad (5.4)$$

where $\boldsymbol{\delta}^- = (\delta_2, \ldots, \delta_{k-1})^\mathsf{T}$, $\delta_1 = \delta_k = 0$ and $\mathbf{B}$ and $\mathbf{D}$ are defined in table 5.1.

Defining $\mathbf{F}^- = \mathbf{B}^{-1}\mathbf{D}$ and

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} \\ \mathbf{F}^- \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{0}$ is a row of zeros, we have that $\boldsymbol{\delta} = \mathbf{F}\boldsymbol{\beta}$. Hence, the spline can be rewritten entirely in terms of $\boldsymbol{\beta}$ as

$$f(x) = a_j^-(x)\beta_j + a_j^+(x)\beta_{j+1} + c_j^-(x)\mathbf{F}_j\boldsymbol{\beta} + c_j^+(x)\mathbf{F}_{j+1}\boldsymbol{\beta} \text{ if } x_j \leq x \leq x_{j+1},$$

which can be re-written, once more, as

$$f(x) = \sum_{i=1}^{k} b_i(x)\beta_i$$

by implicit definition of new basis functions $b_i(x)$: figure 5.4 illustrates the basis. Hence, given a set of x values at which to evaluate the spline, it is easy to obtain a model matrix mapping $\boldsymbol{\beta}$ to the evaluated spline. It can further be shown (e.g., Lancaster and Šalkauskas, 1986, or exercise 2) that

$$\int_{x_1}^{x_k} f''(x)^2 dx = \boldsymbol{\beta}^\mathsf{T}\mathbf{D}^\mathsf{T}\mathbf{B}^{-1}\mathbf{D}\boldsymbol{\beta},$$

i.e. $\mathbf{S} \equiv \mathbf{D}^\mathsf{T}\mathbf{B}^{-1}\mathbf{D}$ is the penalty matrix for this basis.

In addition to having directly interpretable parameters, this basis does not require any re-scaling of the predictor variables before it can be used to construct a GAM, although, as with the chapter 3 basis, we do have to choose the locations of the knots, x_j. See Lancaster and Šalkauskas (1986) for more details. In `mgcv`, model terms like `s(x,bs="cr",k=15)` use this basis (basis dimension `k` defaults to 10 if not supplied).

5.3.2 *A cyclic cubic regression spline*

It is quite often appropriate for a model smooth function to be 'cyclic', meaning that the function has the same value and first few derivatives at its upper and lower

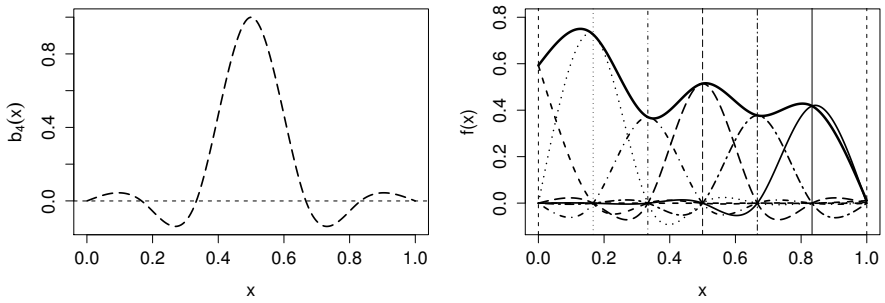

Figure 5.4 *The left hand panel illustrates one basis function, $b_4(x)$, for a cubic regression spline of the type discussed in section 5.3.1: this basis function takes the value one at one knot of the spline, and zero at all other knots (such basis functions are sometimes called 'cardinal basis functions'). The right hand panel shows how such basis functions are combined to represent a smooth curve. The various curves of medium thickness show the basis functions, $b_j(x)$, of a cubic regression spline, each multiplied by its associated coefficient β_j: these scaled basis functions are summed to get the smooth curve illustrated by the thick continuous curve. The vertical thin lines show the knot locations.*

boundaries. For example, in most applications, it would not be appropriate for a smooth function of time of year to change discontinuously at the year end. The penalized cubic regression spline of the previous section can be modified to produce such a smooth. The spline can still be written in the form (5.3), but we now have that $\beta_1 = \beta_k$ and $\delta_1 = \delta_k$. In this case then, we define vectors $\boldsymbol{\beta}^\mathsf{T} = (\beta_1, \ldots, \beta_{k-1})$ and $\boldsymbol{\delta}^\mathsf{T} = (\delta_1, \ldots, \delta_{k-1})$. The conditions that the spline must be continuous to second derivative at each knot, and that $f(x_1)$ must match $f(x_k)$ up to second derivative, are equivalent to

$$\tilde{\mathbf{B}}\boldsymbol{\delta} = \tilde{\mathbf{D}}\boldsymbol{\beta}$$

where $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{D}}$ are defined in table 5.1. Similar reasoning to that employed in the previous section implies that the spline can be written as

$$f(x) = \sum_{i=1}^{k-1} \tilde{b}_i(x)\beta_i,$$

by appropriate definition of the basis functions $\tilde{b}_i(x)$: figure 5.5 illustrates this basis. A second derivative penalty also follows:

$$\int_{x_1}^{x_k} f''(x)^2 dx = \boldsymbol{\beta}^\mathsf{T}\tilde{\mathbf{D}}^\mathsf{T}\tilde{\mathbf{B}}^{-1}\tilde{\mathbf{D}}\boldsymbol{\beta}.$$

In `mgcv`, model terms like `s(x,bs="cc")` specify this basis. By default the spline will match at the smallest and largest x values, since these are the default locations of the outermost knots. Often this is inappropriate, and knots should be supplied by the user, via the `knots` argument to `gam`. If only two knots are supplied then

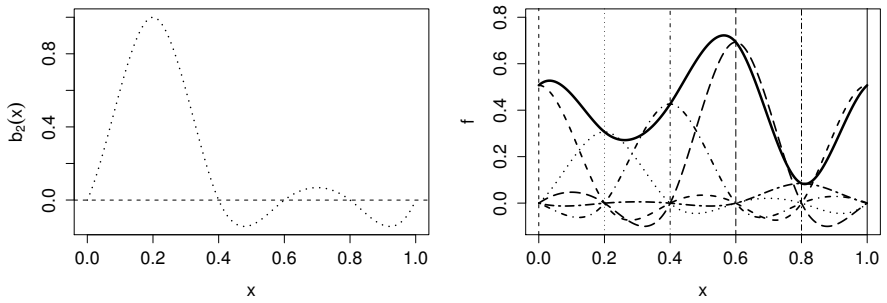

Figure 5.5 *The left hand panel illustrates one basis function, $b_2(x)$, for a cyclic cubic regression spline of the type discussed in section 5.3.2: this basis function takes the value one at one knot of the spline, and zero at all other knots — notice how the basis function values and first two derivatives match at $x = 0$ and $x = 1$. The right hand panel shows how such basis functions are combined to represent a smooth curve. The various curves of medium thickness show the basis functions, $b_j(x)$, of a cyclic cubic regression spline, each multiplied by its associated coefficient β_j: these scaled basis functions are summed to get the smooth curve illustrated by the thick continuous curve. The vertical thin lines show the knot locations.*

they are taken to be the outermost knots, and the remaining `k-2` knots are placed automatically between these.

5.3.3 P-splines

Yet another way to represent cubic splines (and splines of higher or lower order) is by use of the B-spline basis. The B-spline basis is appealing because the basis functions are strictly local — each basis function is only non-zero over the intervals between $m + 3$ adjacent knots, where $m + 1$ is the order of the basis (e.g., $m = 2$ for a cubic spline[†]). To define a k parameter B-spline basis, first define $k + m + 2$ knots, $x_1 < x_2 < \ldots < x_{k+m+2}$, where the interval over which the spline is to be evaluated lies within $[x_{m+2}, x_{k+1}]$ (so the first and last $m + 1$ knot locations are essentially arbitrary). An $(m + 1)^{\text{th}}$ order spline can then be represented as

$$f(x) = \sum_{i=1}^{k} B_i^m(x)\beta_i,$$

where the B-spline basis functions are most conveniently defined recursively as follows:

$$B_i^m(x) = \frac{x - x_i}{x_{i+m+1} - x_i} B_i^{m-1}(x) + \frac{x_{i+m+2} - x}{x_{i+m+2} - x_{i+1}} B_{i+1}^{m-1}(x) \quad i = 1, \ldots k$$

[†]The somewhat inconvenient definition of order is for compatibility with the notation usually used for smoothing splines.

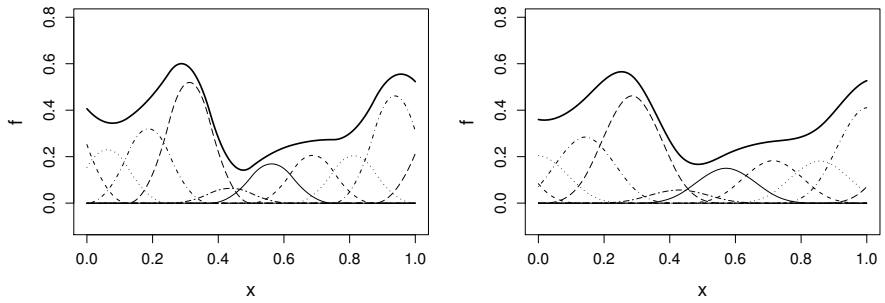

Figure 5.6 *Illustration of the representation of a smooth curve by rank 10 B-spline bases. The left plot shows a B-spline basis with* $m = 1$*. The thin curves show B-spline basis functions multiplied by their associated coefficients; each is non-zero over only 3 intervals. The sum of the coefficients multiplied by the basis functions gives the spline itself, represented by the thicker continuous curve. The right panel is the same, but for a basis for which* $m = 2$*: in this case each basis function is non-zero over 4 adjacent intervals. In both panels the knot locations are where each basis function peaks.*

and

$$B_i^{-1}(x) = \begin{cases} 1 & x_i \leq x < x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

(see, e.g., de Boor, 1978; Lancaster and Šalkauskas, 1986). For example, the following R code can be used to evaluate single B-spline basis functions at a series of x values:

```
bspline <- function(x,k,i,m=2)
# evaluate ith B-spline basis function of order m at the
# values in x, given knot locations in k
{ if (m==-1) { # base of recursion
    res <- as.numeric(x<k[i+1]&x>=k[i])
  } else {     # construct from call to lower order basis
    z0 <- (x-k[i])/(k[i+m+1]-k[i])
    z1 <- (k[i+m+2]-x)/(k[i+m+2]-k[i+1])
    res <- z0*bspline(x,k,i,m-1)+ z1*bspline(x,k,i+1,m-1)
  }
  res
}
```

Figure 5.6 illustrates the representation of functions using B-spline bases of two different orders. In R the function `splineDesign` from the `splines` package can be used to generate B-spline bases, while `cSplineDes` from package `mgcv` will generate cyclic B-spline bases.

B-splines were developed as a very stable basis for large scale spline interpolation (see de Boor, 1978, for further details), but for most statistical work with low rank penalized regression splines you would have to be using very poor numerical methods before the enhanced stability of the basis became noticeable. The real sta-

tistical interest in B-splines has resulted from the work of Eilers and Marx (1996) in using them to develop what they term *P-splines*.

P-splines are low rank smoothers using a B-spline basis, usually defined on evenly spaced knots, with a *difference penalty* applied directly to the parameters, β_i, to control function wiggliness. How this works is best seen by example. If we decide to penalize the squared difference between adjacent β_i values then the penalty is

$$\mathcal{P} = \sum_{i=1}^{k-1} (\beta_{i+1} - \beta_i)^2 = \boldsymbol{\beta}^{\mathsf{T}} \mathbf{P}^{\mathsf{T}} \mathbf{P} \boldsymbol{\beta}$$

where

$$\mathbf{P} = \begin{bmatrix} -1 & 1 & 0 & . & . \\ 0 & -1 & 1 & 0 & . \\ . & . & . & . & . \\ . & . & . & . & . \end{bmatrix} \text{ so that } \begin{bmatrix} \beta_2 - \beta_1 \\ \beta_3 - \beta_2 \\ . \\ . \end{bmatrix} = \mathbf{P}\boldsymbol{\beta},$$

and hence

$$\mathcal{P} = \boldsymbol{\beta}\mathbf{P}^{\mathsf{T}}\mathbf{P}\boldsymbol{\beta} = \boldsymbol{\beta}^{\mathsf{T}} \begin{bmatrix} 1 & -1 & 0 & . & . \\ -1 & 2 & -1 & . & . \\ 0 & -1 & 2 & . & . \\ . & . & . & . & . \\ . & . & . & . & . \end{bmatrix} \boldsymbol{\beta}.$$

Such penalties are very easily generated in R. For example,

```
k <- 6                              # example basis dimension
P <- diff(diag(k),differences=1)    # sqrt of penalty matrix
S <- t(P)%*%P                       # penalty matrix
```

Higher order penalties are produced by increasing the `differences` parameter. The only lower order penalty is the identity matrix.

P-splines are extremely easy to set up and use and allow a good deal of flexibility, in that any order of penalty can be combined with any order of B-spline basis, as the user sees fit. Their disadvantage is that the simplicity is somewhat diminished if uneven knot spacing is required, and that, relative to the more usual spline penalties, the discrete penalties are less easy to interpret in terms of the properties of the fitted smooth. They can be especially useful for MCMC based Bayesian inference, where the sparsity of the basis and the penalty can offer substantial computational savings.

In `mgcv`, terms like `s(x,bs="ps",m=c(2,3))` specify P-splines (`bs="cp"` for a cyclic version), with the `m` parameter specifying the order of the spline basis and penalty. Basis order 2 gives cubic P-splines, while penalty order 3 would specify a penalty built on third differences. If only a single `m` is specified then it is used for both. `m` defaults to 2.

5.3.4 P-splines with derivative based penalties

Actually it is not very difficult to use derivative based penalties with any B-spline basis, and the P-spline advantages of sparse basis and penalty and the ability to 'mix-and-match' basis and penalty orders carry over to this case as well. The penalty

matrix setup requires a few lines of code, rather than two lines needed for P-splines, but it is still relatively straightforward (Wood, 2016b).

In this section let m_1 denote the order of basis with 3 corresponding to cubic, and let m_2 denote the order of differentiation required in the penalty

$$J = \int_a^b f^{[m_2]}(x)^2 dx = \boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta}.$$

We seek $\mathbf{S}$ and possibly its 'square root'. Let $x_1, x_2 \ldots x_{k-m+1}$ be the (ordered) 'interior knots' defining the B-spline basis: that is, the knots within whose range the spline and its penalty are to be evaluated (so $a = x_1$ and $b = x_{k-m+1}$). Let the inter-knot distances be $h_j = x_{j+1} - x_j$, for $0 < j \leq k - m$.

1. For each interval $[x_j, x_{j+1}]$, generate $p + 1$ evenly spaced points within the interval. For $p = 0$ the point should be at the interval centre; otherwise the points always include the end points x_j and x_{j+1}. Let $\mathbf{x}'$ contain the unique x values so generated, in ascending order.
2. Obtain the matrix $\mathbf{G}$ mapping the spline coefficients to the m_2^{th} derivative of the spline at the points $\mathbf{x}'$.
3. If $p = 0$, $\mathbf{W} = \text{diag}(\mathbf{h})$.
4. It $p > 0$, let $p+1 \times p+1$ matrices $\mathbf{P}$ and $\mathbf{H}$ have elements $P_{ij} = (-1+2(i-1)/p)^j$ and $H_{ij} = (1 + (-1)^{i+j-2})/(i + j - 1)$ (i and j start at 1). Then compute matrix $\tilde{\mathbf{W}} = \mathbf{P}^{-\mathsf{T}}\mathbf{H}\mathbf{P}^{-1}$. Now compute $\mathbf{W} = \sum_q \mathbf{W}^q$ where each $\mathbf{W}^q$ is zero everywhere except at $W^q_{i+pq-p,j+pq-p} = h_q\tilde{W}_{ij}/2$, for $i = 1, \ldots, p + 1$, $j = 1, \ldots, p + 1$. $\mathbf{W}$ is banded with $2p + 1$ non-zero diagonals.
5. The diagonally banded penalty coefficient matrix is $\mathbf{S} = \mathbf{G}^\mathsf{T}\mathbf{W}\mathbf{G}$.
6. Optionally, compute the diagonally banded Cholesky decomposition $\mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{W}$, and form diagonally banded matrix $\mathbf{D} = \mathbf{R}\mathbf{G}$, such that $\mathbf{S} = \mathbf{D}^\mathsf{T}\mathbf{D}$.

`splines:splineDesign` in R can be used for step 2, while the single rank $p+1$ matrix inversion of $\mathbf{P}$ at step 4 can be pre-formed using `solve`. $\mathbf{P}$ is somewhat ill-conditioned for $p \geq 20$. However it is difficult to imagine a sensible application for which $p > 10$, and for $p \leq 10$, $\mathbf{P}$'s condition number is $< 2 \times 10^4$. Note that $\mathbf{W}$ is formed without explicitly forming the $\mathbf{W}^q$ matrices. Step 6 can be accomplished by a banded Cholesky decomposition (for example `bandchol` in `mgcv`) but for applications with k less than 1000 or so, a dense Cholesky decomposition might be deemed efficient enough. Step 6 is preferable to construction of $\mathbf{D}$ by decomposition of $\mathbf{S}$, since $\mathbf{W}$ is positive definite by construction, while, for $m_2 > 0$, $\mathbf{S}$ is only positive semi-definite. In `mgcv`, terms like `s(x,bs="bs",m=c(3,2))` create such smooths (the example creates a cubic spline with second derivative penalty).

5.3.5 Adaptive smoothing

A nice illustration of the flexibility of P-splines is provided by adaptive smoothing. Suppose that we would like the amount of smoothing to vary with the smoothing

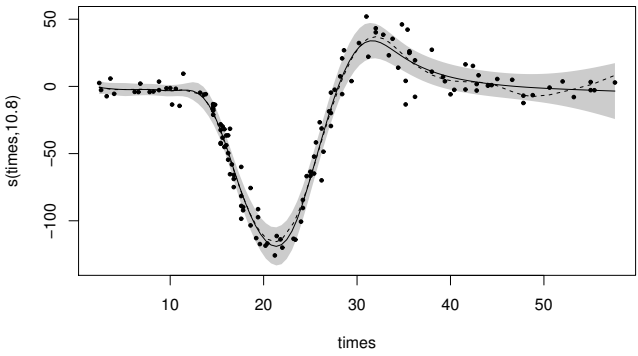

Figure 5.7 *An adaptive smooth of the acceleration of the head of a crash test dummy in a simulated motorcycle crash (*`mcycle` *data from* `MASS` *library). The continuous black curve is an adaptive smooth described in section 5.3.5, with 95% CI shown in grey. The dashed curve is a conventional cubic spline smoother. Smoothing parameter selected by GCV in both cases.*

covariate, x, say. For example consider a second order P-spline penalty in which the squared differences are now weighted

$$\mathcal{P}_a = \sum_{i=2}^{k-1} \omega_i (\beta_{i-1} - 2\beta_i + \beta_{i+1})^2 = \boldsymbol{\beta}^{\mathsf{T}} \mathbf{D}^{\mathsf{T}} \mathsf{diag}(\boldsymbol{\omega}) \mathbf{D} \boldsymbol{\beta}$$

$$\text{where } \mathbf{D} = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdot \\ 0 & 1 & -2 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

Now let the weights ω_i vary smoothly with i and hence with x. An obvious way to do this is to use a B-spline basis expansion $\boldsymbol{\omega} = \mathbf{B}\boldsymbol{\lambda}$, where $\boldsymbol{\lambda}$ is the vector of basis coefficients. Then, writing $\mathbf{B}_{\cdot j}$ for the j^{th} column of $\mathbf{B}$ we have

$$\boldsymbol{\beta}^{\mathsf{T}} \mathbf{D}^{\mathsf{T}} \mathsf{diag}(\boldsymbol{\omega}) \mathbf{D} \boldsymbol{\beta} = \sum_j \lambda_j \boldsymbol{\beta}^{\mathsf{T}} \mathbf{D}^{\mathsf{T}} \mathsf{diag}(\mathbf{B}_{\cdot j}) \mathbf{D} \boldsymbol{\beta} = \sum_j \lambda_j \boldsymbol{\beta}^{\mathsf{T}} \mathbf{S}_j \boldsymbol{\beta}.$$

Notice the B-spline basis property that the elements of $\mathbf{B}$ are non-negative, which ensures that the $\mathbf{S}_j$ are positive semi-definite. The point here is that the adaptive smoother can be represented using a multiply penalized B-spline basis. In `mgcv` a term such as `s(x,bs="ad",k=30,m=4)` would specify an adaptive smoother with a basis dimension of 30 and 4 smoothing parameters (the defaults are 40 and 5, respectively). `?adaptive.smooth` describes how cyclic and two dimensional versions are also possible. Figure 5.7 illustrates the application of such an adaptive smoother.

5.3.6 SCOP-splines

Another interesting application of the P-spline approach is shape constrained smoothing. Pya and Wood (2015) propose one method for constructing shape constrained splines in this way. Consider creating a monotonically increasing smoother,

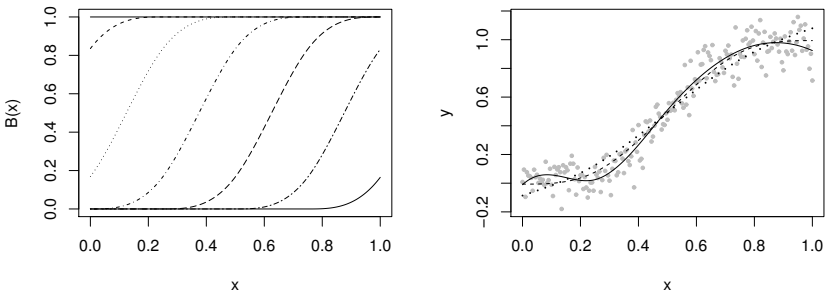

Figure 5.8 *Left. A 7-dimensional SCOP spline basis based on cubic B-splines; each curve shows one basis function. Right. Data sampled from a monotonic truth in grey, and some model fits shown as curves. The continuous curve is the un-penalized fit with a 7-dimensional B-spline basis. The dashed curve is SCOP-spline fit with negligible penalization. The dotted curve is a more heavily penalized SCOP-spline fit.*

represented using an order m B spline basis expansion (where $m = 3$ here denotes a cubic basis):

$$f(x) = \sum_{j=1}^{k} \beta_j B_{m,j}(x).$$

From (de Boor, 2001, p. 116), we have

$$f'(x) = \sum_j \beta_j B'_{m,j}(x) = (m-1) \sum_j \frac{\beta_j - \beta_{j-1}}{x_{j+m} - x_j} B_{m-1,j}(x),$$

where x_j are the (increasing) knot locations. Since the B-spline basis functions of any order are non-negative, a sufficient condition for $f'(x) > 0$ is that the β_j form an increasing sequence. Similarly a decreasing, concave or convex β_j sequence yields, respectively, a decreasing, concave or convex $f(x)$ (and combinations of monotonicity and convexity restrictions are possible).

Employing the convention $\sum_{i=2}^{1} x_i = 0$, suppose that we re-parameterize so that

$$\beta_j = \gamma_1 + \sum_{i=2}^{j} \exp(\gamma_i)$$

where the γ_j are unrestricted coefficients. Clearly the β_j sequence is increasing (in finite precision arithmetic, non-decreasing). Now suppose that we impose the quadratic penalty

$$J = \sum_{i=2}^{k-1} (\gamma_{j+1} - \gamma_j)^2$$

on $\boldsymbol{\gamma}$. Clearly this tends to force the γ_i towards equality, which in turn means that we

are smoothing towards a straight line. So we have a means for producing a monotonically increasing penalized spline. The price paid is that the spline is no longer linear in its coefficients, but the non-linearity is rather benign.

Further insight into the basis can be obtained as follows. Let $\tilde{\gamma}_1 = \gamma_1$ and $\tilde{\gamma}_i = \exp(\gamma_i)$ if $i > 2$, then $\boldsymbol{\beta} = \mathbf{B}\tilde{\boldsymbol{\gamma}}$ where $B_{ij} = 1$ if $i \geq j$ and 0 otherwise. So if $\mathbf{X}$ is the usual B-spline model matrix for $f(x)$ then we have $\mathbf{f} = \mathbf{XB}\tilde{\boldsymbol{\gamma}}$. Hence we could view the monotonic basis as being that evaluated in $\mathbf{XB}$, with basis functions

$$M_{m,j}(x) = \sum_{i=j}^{k} B_{m,i}(x).$$

Figure 5.8 illustrates such basis functions in its left panel, while the right panel shows some monotonic fits using the basis. If sum-to-zero constraints are required for a SCOP-spline then they should be implemented in the manner described in section 4.3.1 (p. 175). That is, the first (constant) column of $\mathbf{XB}$ should be dropped, and each remaining column should have its mean subtracted from each of its elements (i.e., the columns should be 'centred'). Pya and Wood (2015) provides the equivalent $\mathbf{B}$ matrices for monotonic decrease, and a variety of other constraints (as well as a complete framework for shape constrained additive modelling).

`mgcv`'s `"ps"` smooth constructor allows monotonic setup

```
ssp <- s(x,bs="ps",k=k); ssp$mono <- 1
sm <- smoothCon(ssp,data.frame(x))[[1]]
```

(`ssp$mono <- -1` would have set things up for monotonic decrease). The `sm` smooth object contains a model matrix `X` and penalty matrix `S` that we can then use in a Newton loop for fitting. The penalized least squares fits shown in the right hand panel of figure 5.8 were actually produced by the following very simple Newton loop.

```
X <- sm$X; XX <- crossprod(X); sp <- .5
gamma <- rep(0,k); S <- sm$S[[1]]
for (i in 1:20) {
  gt <- c(gamma[1],exp(gamma[2:k]))
  dg <- c(1,gt[2:k])
  g <- -dg*(t(X)%*%(y-X%*%gt)) + sp*S%*%gamma
  H <- dg*t(dg*XX)
  gamma <- gamma - solve(H+sp*S,g)
}
```

where `sp` is the smoothing parameter.[‡]

5.4 Some useful smoother theory

Before covering smoothing with respect to multiple covariates, it is worth considering two forms of re-parameterization of smooths. The first is the re-parameterization needed to absorb identifiability constraints into the smooth model matrix and penalty

[‡]Obviously in more serious applications a Newton loop should perform step length control, and should test for convergence, rather than just assuming that 20 iterations will do.

in practice, while the second is a reparameterization that helps understand the notion of the effective degrees of freedom of a smooth. It is also worth considering the construction of penalties for the otherwise un-penalized components of the smooth.

5.4.1 *Identifiability constraints*

Incorporating smooths into additive models involves the imposition of identifiability constraints, and it is convenient if these are dealt with by re-parameterization. The constraints are necessary because it is clearly not possible to estimate an intercept term for each smooth. The constraints that lead to sharpest inference about the constrained components are those that orthogonalise the smooth to the intercept term, i.e., $\sum_j f(x_j) = 0$ which states that the smooth sums to zero over the observed values of x_j.[§] If $\mathbf{X}$ is the model matrix for the smooth then the constraint becomes $\mathbf{1}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} = 0$. Section 4.3.1 (p. 175) presents one method for incorporating such constraints, but an elegant alternative uses the general QR based approach of section 1.8.1 (p. 47).

The idea is to create a $k \times (k-1)$ matrix $\mathbf{Z}$ such that if $\boldsymbol{\beta} = \mathbf{Z}\boldsymbol{\beta}_z$, then $\mathbf{1}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} = 0$ for any $\boldsymbol{\beta}_z$. Then we re-parameterize the smooth by setting its model matrix to $\mathbf{XZ}$ and its penalty matrix to $\mathbf{Z}^\mathsf{T}\mathbf{S}\mathbf{Z}$. In section 1.8.1 $\mathbf{Z}$ is constructed using a QR decomposition of $\bar{\mathbf{x}} = \mathbf{X}^\mathsf{T}\mathbf{1}$, but actually the orthogonal factor in the QR decomposition consists of a single Householder reflection (see section B.5), making the formation of $\mathbf{XZ}$ and $\mathbf{Z}^\mathsf{T}\mathbf{S}\mathbf{Z}$ particularly straightforward and efficient. In particular

$$\bar{\mathbf{x}} = \mathbf{H}\bar{\mathbf{x}}' = [\mathbf{D} : \mathbf{Z}]\bar{\mathbf{x}}' \text{ where } \bar{\mathbf{x}}' = \begin{bmatrix} \|\mathbf{x}\| \\ \mathbf{0} \end{bmatrix}, \mathbf{H} = (\mathbf{I} - 2\mathbf{u}\mathbf{u}^\mathsf{T}/\mathbf{u}^\mathsf{T}\mathbf{u}) \text{ and } \mathbf{u} = \bar{\mathbf{x}} - \bar{\mathbf{x}}'.$$

Multiplication by $\mathbf{H}$ is a factor of p more efficient in terms of floating point operations and storage costs if it is done as shown in section B.5, rather than by forming $\mathbf{H}$ explicitly. This means that rather than form $\mathbf{Z}$ explicitly, we should form $\mathbf{XZ}$ by computing $\mathbf{XH}$ and dropping its first column. The same approach can be used for forming $\mathbf{Z}^\mathsf{T}\mathbf{S}\mathbf{Z}$ having first computed $\mathbf{Z}^\mathsf{T}\mathbf{S}$ by computing $\mathbf{HS}$ and then dropping its first row. Note also that if $\boldsymbol{\beta}'_z = (0, \boldsymbol{\beta}_z^\mathsf{T})^\mathsf{T}$ then $\mathbf{Z}\boldsymbol{\beta}_z = \mathbf{H}\boldsymbol{\beta}'_z$.

5.4.2 *'Natural' parameterization, effective degrees of freedom and smoothing bias*

Penalized smoothers, with a single penalty, can always be parameterized in such a way that the parameter estimators are independent with unit variance in the absence of the penalty, and the penalty matrix is diagonal. This 'natural' or Demmler and Reinsch (1975) parameterization is particularly helpful for understanding the way in which the penalty suppresses model degrees of freedom.

Consider a smooth with model matrix $\mathbf{X}$, parameter vector $\boldsymbol{\beta}$, wiggliness penalty coefficient matrix $\mathbf{S}$, and smoothing parameter λ. Suppose that the model is to be

[§] If f is estimated to be a straight line then the constraint determines exactly where it must pass through zero — in consequence confidence intervals for f will have zero width at this point

estimated by penalized least squares from data with variance σ^2. Forming the QR decomposition

$$\mathbf{X} = \mathbf{QR},$$

we can re-parameterize in terms of $\boldsymbol{\beta}'' = \mathbf{R}\boldsymbol{\beta}$, so that the model matrix is now $\mathbf{Q}$, and the penalty matrix becomes $\mathbf{R}^{-\mathsf{T}}\mathbf{S}\mathbf{R}^{-1}$. Eigen-decomposing the penalty matrix yields

$$\mathbf{R}^{-\mathsf{T}}\mathbf{S}\mathbf{R}^{-1} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}},$$

where $\mathbf{U}$ is an orthogonal matrix, the columns of which are the eigenvectors of $\mathbf{R}^{-\mathsf{T}}\mathbf{S}\mathbf{R}^{-1}$, while $\mathbf{D}$ is a diagonal matrix of the corresponding eigenvalues, arranged in decreasing order. Reparameterization, via a rotation/reflection of the parameter space, now yields parameters $\boldsymbol{\beta}' = \mathbf{U}^{\mathsf{T}}\boldsymbol{\beta}''$, and correspondingly a model matrix $\mathbf{QU}$ and penalty matrix $\mathbf{D}$. If the penalty is not applied then the covariance matrix for these parameters is $\mathbf{I}\sigma^2$, since $\mathbf{U}$ is orthogonal, and the columns of $\mathbf{Q}$ are columns of an orthogonal matrix.

If the penalty is applied, then the Bayesian covariance matrix of the parameters (see sections 4.2.4 and 5.8) is simply the diagonal matrix $(\mathbf{I}+\lambda\mathbf{D})^{-1}\sigma^2$, from which the role of the penalty in limiting parameter variance is rather clear. The frequentist equivalent is equally simple: $(\mathbf{I} + \lambda\mathbf{D})^{-2}\sigma^2$.

Now in the natural parameterization, $\hat{\boldsymbol{\beta}}' = (\mathbf{I} + \lambda\mathbf{D})^{-1}\mathbf{U}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{y}$, and hence the corresponding unpenalized estimate is $\tilde{\boldsymbol{\beta}}' = \mathbf{U}^{\mathsf{T}}\mathbf{Q}^{\mathsf{T}}\mathbf{y}$. That is the penalized parameter estimates are simply shrunken versions of the unpenalized coefficient estimates:

$$\hat{\beta}'_i = (1 + \lambda D_{ii})^{-1}\tilde{\beta}'_i.$$

The shrinkage factors $(1 + \lambda D_{ii})^{-1}$ lie in the interval $(0, 1]$. Since the unpenalized coefficients have one degree of freedom each, the shrinkage factor, $(1 + \lambda D_{ii})^{-1}$, can be viewed as the 'effective degrees of freedom' of $\hat{\beta}'_i$. This means that the total effective degrees of freedom of the smooth is

$$\sum_i (1 + \lambda D_{ii})^{-1} = \mathrm{tr}(\mathbf{F}) \text{ where } \mathbf{F} = (\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{X}.$$

Notice how this is bounded between the number of zero eigenvalues of the penalty (when $\lambda \to \infty$), and the total number of coefficients (when $\lambda = 0$). Also notice how this notion of effective degrees of freedom corresponds to that given for mixed effects models in section 2.4.6 (p. 83).

The fact that the unpenalized estimators are unbiased also means that $\mathbb{E}(\hat{\beta}'_i) = (1 + \lambda D_{ii})^{-1}\beta_i$: the shrinkage factors determine the relative *smoothing bias*. In the original parameterization this becomes $\mathbb{E}(\hat{\boldsymbol{\beta}}) = \mathbf{F}\boldsymbol{\beta}$.

So, in the natural parameterization, all parameters have the same degree of associated variability when un-penalized, and the penalty acts on each parameter independently (i.e., the degree of penalization of one parameter has no effect on the other parameters). In this parameterization the relative magnitudes of different D_{ii} terms directly indicate the relative degree of penalization of different components of the model space. $\mathbf{D}$ is uniquely defined — no matter what parameterization you start out with, the elements of $\mathbf{D}$ are always the same.

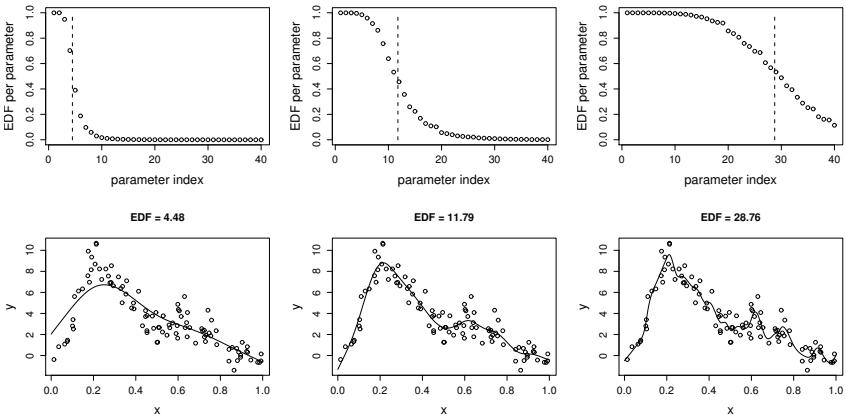

Figure 5.9 *How parameters are shrunk to zero by the penalty in the natural parameterization. All plots relate to a rank 40 cubic regression spline fit to the data shown in the bottom row plots. The top row plots the degrees of freedom associated with each parameter against parameter index using the natural parameterization, for 3 levels of penalization. On each plot the effective degrees of freedom (i.e., the effective number of free parameters) is shown by a vertical dashed line: note that this is simply the sum of the plotted degrees of freedom per parameter. The lower row shows the smooth fit corresponding to each of the top row plots. For this smooth the first two 'natural' parameters are never penalized. Notice how some potentially penalized parameters are effectively un-penalized, some are effectively suppressed completely, while some suffer intermediate penalization. Clearly, it is only approximately true that a penalized fit is equivalent to an un-penalized fit with the number of parameters given by the penalized fit effective degrees of freedom.*

Figure 5.9 illustrates how parameters get shrunk by the penalty, using the natural parameterization. The figures illustrate that at most levels of penalization some subspace of the model space is almost completely suppressed, while some other subspace is left almost unpenalized. This lends some support to the idea that a penalized fit is somehow equivalent to an unpenalized fit with degrees of freedom close to the effective degrees of freedom of the penalized model. However, the fact that many parameters have intermediate penalization means that this support is only limited.

The natural parameterization also makes the penalized smoothers behave more like full spline models than is otherwise immediately apparent. For example, for a full spline smoother the EDF matrix is simply the influence matrix, $\mathbf{A}$, which also defines the Bayesian posterior covariance matrix $\mathbf{A}\sigma^2$ and the equivalent frequentist matrix $\mathbf{A}^2\sigma^2$. In other words, the relationship between these matrices that holds for smoothing splines also holds for general penalized regression smoothers with the natural parameterization.

The penalty's action in effectively suppressing some dimensions of the model space is also readily apparent in the natural parameterization. For most smoothers the penalty assesses the 'wiggliness' of the fitted model, in such a way that smoother functions are generally not simply closer to the zero function. In this circumstance,

if increased penalization is to lead to continuously smoother models then, in the natural parameterization, it is clear that the elements of $\mathbf{D}$ must have a spread of (non-negative) values, with the higher values penalizing coefficients corresponding to more wiggly components of the model function. If this is not clear, consider the converse situation in which all elements on the leading diagonal of $\mathbf{D}$ are the same. In that case increased penalization would amount to simply multiplying each model coefficient by the same constant, thereby shrinking the fitted function towards zero without changing its shape.

5.4.3 Null space penalties

One feature of most smoothing penalties is that they treat some *null* space of functions as being 'completely smooth' and hence having zero penalty. For example the usual cubic spline penalty, $\int f''(x)^2 dx$, is zero for any straight line, $f(x) = \alpha + \beta x$. This means that as $\lambda \to \infty$ the estimated smoother tends to a function in the penalty null space, rather than to the zero function. Hence, although smoothing parameter selection methods effectively perform a great deal of model selection, in selecting f from a wide class of possibilities of differing complexity, λ selection is not usually able to remove a smooth term from a model altogether.

One possibility is to add an extra penalty to a smooth term, which penalizes only functions in the smoothing penalty null space. Such penalties are easily constructed. Consider the eigen-decompositon of the penalty matrix, $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$, and let $\tilde{\mathbf{U}}$ denote the columns of $\mathbf{U}$ (eigenvectors of $\mathbf{S}$), corresponding to zero eigenvalues on the leading diagonal of $\mathbf{\Lambda}$. Then $\lambda' \boldsymbol{\beta}^\mathsf{T} \tilde{\mathbf{U}} \tilde{\mathbf{U}}^\mathsf{T} \boldsymbol{\beta}$ can be used as an extra penalty on the null space of the original penalty. This construction ensures that if all smoothing parameters for a term $\to \infty$ then the term will tend to the zero function. Note that the new penalty has no effect on the components of the smooth not in the original penalty null space. For terms with multiple penalties the same construction applies, but an eigen-decomposition of $\sum_j \mathbf{S}_j$ is used. Marra and Wood (2011) provide more details. The `select` argument of `gam` can be used to impose such penalties in `mgcv`.

A variant on this idea is to replace the zero eigenvalues in the $\mathbf{\Lambda}$ with constants somewhat smaller than the smallest positive entry in $\mathbf{\Lambda}$. Call the result $\tilde{\mathbf{\Lambda}}$. Then the original penalty is replaced by $\lambda \boldsymbol{\beta}^\mathsf{T} \mathbf{U} \tilde{\mathbf{\Lambda}} \mathbf{U}^\mathsf{T} \boldsymbol{\beta}$, and $\lambda \to \infty$ again causes the smooth term to tend to the zero function. This idea does not work so well with multiple penalties, and results are somewhat dependent on how we choose the 'somewhat smaller' constant. In `mgcv` the 'shrinkage' bases, `"cs"` and `"ts"`, implement this idea.

5.5 Isotropic smoothing

This section considers smooths of one or more variables, in particular smooths that, in the multiple covariate case, produce identical predictions of the response variable under any rotation or reflection of the covariates.[¶]

[¶] In the literature 'isotropic' is sometimes used for smooths that are not rotationally invariant, but simply smooth 'equally' with respect to each covariate. This is a slight abuse of language.

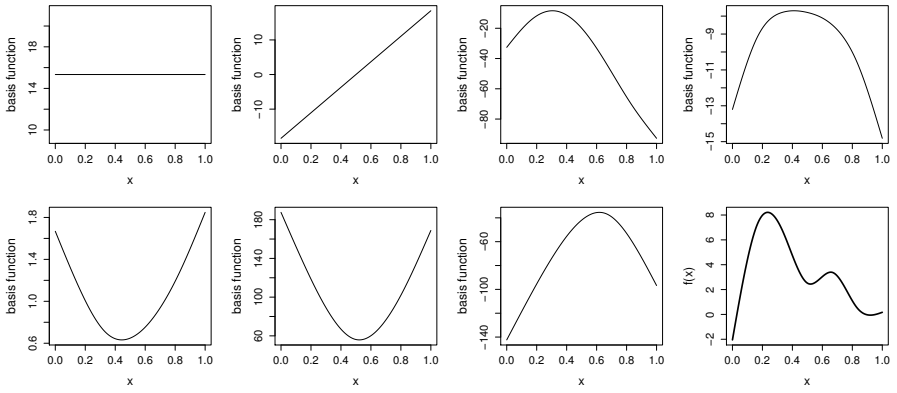

Figure 5.10 *Illustration of a thin plate spline basis for representing a smooth function of one variable fitted to 7 data with penalty order $m = 2$. The first 7 panels (starting at top left) show the basis functions, multiplied by coefficients, that are summed to give the smooth curve in the lower right panel. The first two basis functions span the space of functions that are completely smooth, according to the wiggliness measure. The remaining basis functions represent the wiggly component of the smooth curve: these latter functions are shown after absorption of the thin plate spline constraints $\mathbf{T}^{\mathsf{T}}\boldsymbol{\delta} = \mathbf{0}$ into the basis.*

5.5.1 Thin plate regression splines

The bases covered so far are each useful in practice, but are open to some criticisms.

1. It is necessary to choose knot locations, in order to use each basis: this introduces an extra degree of subjectivity into the model fitting process.
2. The bases are only useful for representing smooths of one predictor variable.
3. It is not clear to what extent the reduced rank bases are better or worse than any other reduced rank basis that might be used.

In this section, an approach is developed which goes some way to addressing these issues, by producing knot free bases for smooths of any number of predictors, that are in a certain limited sense 'optimal': the thin plate regression splines.

Thin plate splines

Thin plate splines (Duchon, 1977) are a very elegant and general solution to the problem of estimating a smooth function of multiple predictor variables, from noisy observations of the function at particular values of those predictors. Consider the problem of estimating the smooth function $g(\mathbf{x})$, from n observations $(y_i, \mathbf{x}_i)$ such that

$$y_i = g(\mathbf{x}_i) + \epsilon_i$$

where ϵ_i is a random error term and where $\mathbf{x}$ is a d-vector. Thin-plate spline smoothing estimates g by finding the function $\hat{f}$ minimizing

$$\|\mathbf{y} - \mathbf{f}\|^2 + \lambda J_{md}(f), \tag{5.5}$$

where $\mathbf{y}$ is the vector of y_i data and $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)]^\mathsf{T}$. $J_{md}(f)$ is a penalty functional measuring the 'wiggliness' of f, and λ is a smoothing parameter, controlling the tradeoff between data fitting and smoothness of f. The wiggliness penalty is defined as

$$J_{md} = \int_{\Re^d} \sum_{\nu_1+\ldots+\nu_d=m} \frac{m!}{\nu_1!\ldots\nu_d!} \left(\frac{\partial^m f}{\partial x_1^{\nu_1} \ldots \partial x_d^{\nu_d}} \right)^2 dx_1 \ldots dx_d.^{\|} \tag{5.6}$$

Further progress is only possible if m is chosen so that $2m > d$, and in fact for 'visually smooth' results it is preferable that $2m > d+1$. Subject to the first of these restrictions, it can be shown that the function minimizing (5.5) has the form

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \delta_i \eta_{md}(\|\mathbf{x} - \mathbf{x}_i\|) + \sum_{j=1}^{M} \alpha_j \phi_j(\mathbf{x}), \tag{5.7}$$

where $\boldsymbol{\delta}$ and $\boldsymbol{\alpha}$ are vectors of coefficients to be estimated, $\boldsymbol{\delta}$ being subject to the linear constraints that $\mathbf{T}^\mathsf{T}\boldsymbol{\delta} = \mathbf{0}$ where $T_{ij} = \phi_j(\mathbf{x}_i)$. The $M = \binom{m+d-1}{d}$ functions ϕ_i are linearly independent polynomials spanning the space of polynomials in $\Re^d$ of degree less than m. The ϕ_i span the space of functions for which J_{md} is zero, i.e., the 'null space' of J_{md}: those functions that are considered 'completely smooth'. For example, for $m = d = 2$ these functions are $\phi_1(\mathbf{x}) = 1$, $\phi_2(\mathbf{x}) = x_1$ and $\phi_3(\mathbf{x}) = x_2$. The remaining basis functions used in (5.7) are defined as

$$\eta_{md}(r) = \begin{cases} \frac{(-1)^{m+1+d/2}}{2^{2m-1}\pi^{d/2}(m-1)!(m-d/2)!} r^{2m-d}\log(r) & d \text{ even} \\ \frac{\Gamma(d/2-m)}{2^{2m}\pi^{d/2}(m-1)!} r^{2m-d} & d \text{ odd.} \end{cases}$$

Now defining matrix $\mathbf{E}$ by $E_{ij} \equiv \eta_{md}(\|\mathbf{x}_i - \mathbf{x}_j\|)$, the thin plate spline fitting problem becomes,

$$\text{minimize } \|\mathbf{y} - \mathbf{E}\boldsymbol{\delta} - \mathbf{T}\boldsymbol{\alpha}\|^2 + \lambda\boldsymbol{\delta}^\mathsf{T}\mathbf{E}\boldsymbol{\delta} \text{ subject to } \mathbf{T}^\mathsf{T}\boldsymbol{\delta} = \mathbf{0}, \tag{5.8}$$

with respect to $\boldsymbol{\delta}$ and $\boldsymbol{\alpha}$. Wahba (1990) or Green and Silverman (1994) provide further information about thin plate splines, and figure 5.10 illustrates a thin plate spline basis in one dimension.

The thin plate spline, $\hat{f}$, is something of an ideal smoother: it has been constructed by defining exactly what is meant by smoothness, exactly how much weight to give to the conflicting goals of matching the data and making $\hat{f}$ smooth, and finding the *function* that best satisfies the resulting smoothing objective. Notice that in

$^{\|}$The general form of the penalty is somewhat intimidating, so an example is useful. In the case of a smooth of two predictors with wiggliness measured using second derivatives, we have

$$J_{22} = \int\int \left(\frac{\partial^2 f}{\partial x_1^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x_1 \partial x_2}\right)^2 + \left(\frac{\partial^2 f}{\partial x_2^2}\right)^2 dx_1 dx_2.$$

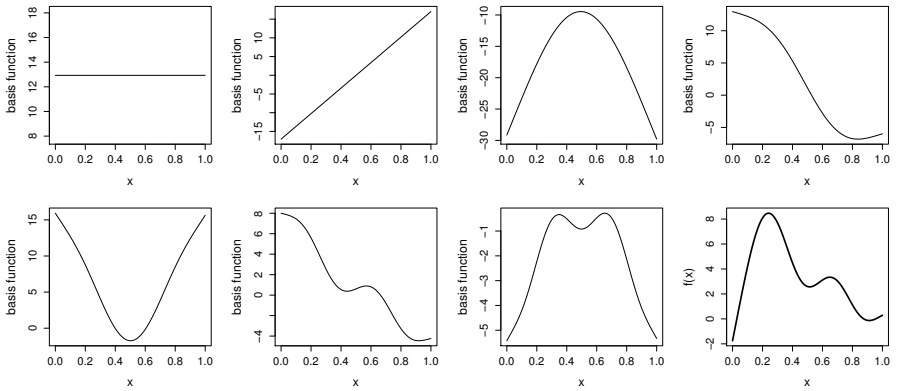

Figure 5.11 *Illustration of a rank 7 thin plate regression spline basis for representing a smooth function of one variable, with penalty order* $m = 2$*. The first 7 panels (starting at top left) show the basis functions, multiplied by coefficients, that are summed to give the smooth curve in the lower right panel. The first two basis functions span the space of functions that are completely smooth, according to the wiggliness measure. The remaining basis functions represent the wiggly component of the smooth curve: notice how these functions become successively more wiggly while generally tending to contribute less and less to the overall fit.*

doing this we did not have to choose knot positions or select basis functions: both of these emerged naturally from the mathematical statement of the smoothing problem. In addition, thin plate splines can deal with any number of predictor variables, and allow the user some flexibility to select the order of derivative used in the measure of function wiggliness. So, at first sight it might seem that the problems listed at the start of this section are all solved, and thin plate spline bases and penalties should be used to represent all the smooth terms in the model.

The difficulty with thin plate splines is computational cost: these smoothers have as many unknown parameters as there are data (strictly, number of unique predictor combinations), and, except in the single predictor case, the computational cost of model estimation is proportional to the cube of the number of parameters. This is a very high price to pay for using such smooths. Given the discussion in section 5.2, and the fact that the effective degrees of freedom estimated for a model term is usually a small proportion of n, it seems wasteful to use so many parameters to represent the term. This begs the question of whether a low rank approximation could be produced which is as close as possible to the thin plate spline smooth, without incurring prohibitive computational cost.

Thin plate regression splines

Thin plate regression splines are based on the idea of truncating the space of the wiggly components of the thin plate spline (the components with parameters $\boldsymbol{\delta}$), while leaving the components of 'zero wiggliness' unchanged (the $\boldsymbol{\alpha}$ components). Let $\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}}$ be the eigen-decomposition of $\mathbf{E}$, where $\mathbf{D}$ is a diagonal matrix of

eigenvalues of $\mathbf{E}$ arranged so that $|D_{i,i}| \geq |D_{i-1,i-1}|$ and the columns of $\mathbf{U}$ are the corresponding eigenvectors. Now let $\mathbf{U}_k$ denote the matrix consisting of the first k columns of $\mathbf{U}$ and let $\mathbf{D}_k$ denote the top left $k \times k$ submatrix of $\mathbf{D}$. Restricting $\boldsymbol{\delta}$ to the column space of $\mathbf{U}_k$, by writing $\boldsymbol{\delta} = \mathbf{U}_k\boldsymbol{\delta}_k$, means that (5.8) becomes

$$\text{minimise } \|\mathbf{y} - \mathbf{U}_k\mathbf{D}_k\boldsymbol{\delta}_k - \mathbf{T}\boldsymbol{\alpha}\|^2 + \lambda\boldsymbol{\delta}_k^{\mathsf{T}}\mathbf{D}_k\boldsymbol{\delta}_k \text{ subject to } \mathbf{T}^{\mathsf{T}}\mathbf{U}_k\boldsymbol{\delta}_k = \mathbf{0}$$

w.r.t. $\boldsymbol{\delta}_k$ and $\boldsymbol{\alpha}$. The constraints can be absorbed in the usual manner, described in section 1.8.1 (p. 47). We first find any orthogonal column basis, $\mathbf{Z}_k$, such that $\mathbf{T}^{\mathsf{T}}\mathbf{U}_k\mathbf{Z}_k = \mathbf{0}$. One way to do this is to form the QR decomposition of $\mathbf{U}_k^{\mathsf{T}}\mathbf{T}$: the final $n - M$ columns of the orthogonal factor give a $\mathbf{Z}_k$ (see sections 1.8.1 and B.6). Restricting $\boldsymbol{\delta}_k$ to this space, by writing $\boldsymbol{\delta}_k = \mathbf{Z}_k\tilde{\boldsymbol{\delta}}$, yields the unconstrained problem that must be solved to fit the rank k approximation to the smoothing spline:

$$\text{minimise } \|\mathbf{y} - \mathbf{U}_k\mathbf{D}_k\mathbf{Z}_k\tilde{\boldsymbol{\delta}} - \mathbf{T}\boldsymbol{\alpha}\|^2 + \lambda\tilde{\boldsymbol{\delta}}^{\mathsf{T}}\mathbf{Z}_k^{\mathsf{T}}\mathbf{D}_k\mathbf{Z}_k\tilde{\boldsymbol{\delta}}$$

with respect to $\tilde{\boldsymbol{\delta}}$ and $\boldsymbol{\alpha}$. This has a computational cost of $O(k^3)$. Having fitted the model, evaluation of the spline at any point is easy: simply evaluate $\boldsymbol{\delta} = \mathbf{U}_k\mathbf{Z}_k\tilde{\boldsymbol{\delta}}$ and use (5.7).

Now, the main problem is how to find $\mathbf{U}_k$ and $\mathbf{D}_k$ sufficiently cheaply. A full eigen-decomposition of $\mathbf{E}$ requires $O(n^3)$ operations, which would somewhat limit the utility of the TPRS approach. Fortunately the method of Lanczos iteration can be employed to find $\mathbf{U}_k$ and $\mathbf{D}_k$ at the substantially lower cost of $O(n^2k)$ operations. See Appendix B, section B.11, for a suitable Lanczos algorithm. If n is large then even $O(n^2k)$ becomes prohibitive. In that case we can randomly select n_r covariate values from the full set of data and then form the thin plate regression spline basis from this random selection, at $O(n_r^2k)$ computational cost. The idea is to choose $n \gg n_r \gg k$, to gain efficiency without losing much in the way of practical performance.

Properties of thin plate regression splines

It is clear that thin plate regression splines avoid the problem of knot placement, are relatively cheap to compute, and can be constructed for smooths of any number of predictor variables, but what of their optimality properties? The thin plate splines are optimal in the sense that no smooth function will better minimize (5.5), but to what extent is that optimality inherited by the TPRS approximation? To answer this it helps to think about what would make a good approximation. An ideal approximation would probably result in the minimum possible perturbation of the fitted values of the spline, at the same time as making the minimum possible change to the 'shape' of the fitted spline. It is difficult to see how both these aims could be achieved, for all possible response data, without first fitting the full thin plate spline. But if the criteria are loosened somewhat to minimizing the worst possible changes in shape and fitted value then progress can be made, as follows.

The basis change and truncation can be thought of as replacing $\mathbf{E}$, in the norm in (5.8), by the matrix $\hat{\mathbf{E}} = \mathbf{E}\mathbf{U}_k\mathbf{U}_k^{\mathsf{T}}$, while replacing $\mathbf{E}$, in the penalty term of (5.8), by $\tilde{\mathbf{E}} = \mathbf{U}_k^{\mathsf{T}}\mathbf{U}_k\mathbf{E}\mathbf{U}_k\mathbf{U}_k^{\mathsf{T}}$. Now since the fitted values of the spline are given

by $\mathbf{E}\hat{\boldsymbol{\delta}} + \mathbf{T}\boldsymbol{\alpha}$, and $\mathbf{T}$ remains unchanged, the worst possible change in fitted values could be measured by:

$$\hat{e}_k = \max_{\boldsymbol{\delta}\neq\mathbf{0}} \frac{\|(\mathbf{E} - \hat{\mathbf{E}}_k)\boldsymbol{\delta}\|}{\|\boldsymbol{\delta}\|}.$$

(Dividing by $\|\boldsymbol{\delta}\|$ is necessary, since the upper norm otherwise has a maximum at infinity.) The 'shape' of the spline is measured by the penalty term in (5.8), so a suitable measure of the worst possible change in the shape of the spline, caused by the truncation, might be:

$$\tilde{e}_k = \max_{\boldsymbol{\delta}\neq\mathbf{0}} \frac{\boldsymbol{\delta}^{\mathsf{T}}(\mathbf{E} - \tilde{\mathbf{E}}_k)\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|^2}.$$

It turns out to be quite easy to show that $\hat{e}_k$ and $\tilde{e}_k$ are simultaneously minimized by the choice of $\mathbf{U}_k$ as the truncated basis for $\boldsymbol{\delta}$. That is, there is no matrix of the same dimension as $\mathbf{U}_k$ which would lead to lower $\hat{e}_k$ or $\tilde{e}_k$, if used in place of $\mathbf{U}_k$ (see Wood, 2003).

Note that $\hat{e}_k$ and $\tilde{e}_k$ are really formulated in too large a space. Ideally we would impose the constraints $\mathbf{T}^{\mathsf{T}}\boldsymbol{\delta} = \mathbf{0}$ on both, but in that case different bases minimize the two criteria. This in turn leads to the question of whether it would not be better to concentrate on just one of the criteria, but this is unsatisfactory, as it leads to results that depend on how the original thin plate spline problem is parameterized. Furthermore, these results can be extremely poor for some parameterizations. For example, if the thin plate spline is parameterized in terms of the fitted values, then the $\hat{e}_k$ optimal approximation is not smooth. Similarly, very poor fitted values result from an $\tilde{e}_k$ optimal approximation to a thin plate spline, if that thin plate spline is parameterized so that the penalty matrix is an identity matrix, with some leading diagonal entries zeroed.

To sum up: thin plate regression splines are probably the best that can be hoped for in terms of approximating the behaviour of a thin plate spline using a basis of any given low rank. They have the nice property of avoiding having to choose 'knot locations', and are reasonably computationally efficient if Lanczos iteration is used to find the truncated eigen-decomposition of $\mathbf{E}$. They also retain the rotational invariance (isotropy) of full thin plate spline. Figures 5.11 and 5.12 provide examples of the basis functions that result from adopting a TPRS approach.

Knot-based approximation

If one is prepared to forgo optimality and choose knot locations, then a simpler approximation is available, which avoids the truncated eigen-decomposition. If knot locations $\{\mathbf{x}_i^* : i = 1 \ldots k\}$ are chosen, then the thin plate spline can be approximated by

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{k} \delta_i \eta_{md}(\|\mathbf{x} - \mathbf{x}_i^*\|) + \sum_{j=1}^{M} \alpha_j \phi_j(\mathbf{x}) \tag{5.9}$$

where $\boldsymbol{\delta}$ and $\boldsymbol{\alpha}$ are estimated by minimizing

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta} \text{ subject to } \mathbf{C}\boldsymbol{\beta} = \mathbf{0}$$

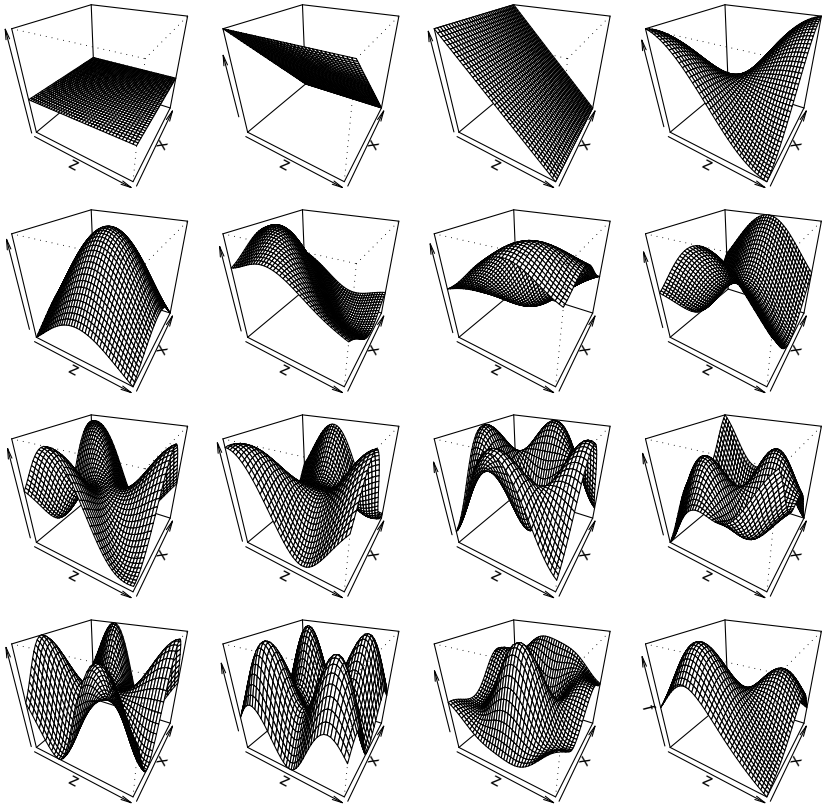

Figure 5.12 *Illustration of a rank 15 thin plate regression spline basis for representing a smooth function of two variables, with penalty order $m = 2$. The first 15 panels (starting at top left) show the basis functions, multiplied by coefficients, that are summed to give the smooth surface in the lower right panel. The first three basis functions span the space of functions that are completely smooth, according to the wiggliness measure, J_{22}. The remaining basis functions represent the wiggly component of the smooth curve: notice how these functions become successively more wiggly.*

w.r.t. $\boldsymbol{\beta}^{\mathsf{T}} = (\boldsymbol{\delta}^{\mathsf{T}}, \boldsymbol{\alpha}^{\mathsf{T}})$. $\mathbf{X}$ is an $n \times (k + M)$ matrix such that

$$X_{ij} = \begin{cases} \eta_{md}(\|\mathbf{x}_i - \mathbf{x}_j^*\|) & j = 1, \ldots, k \\ \phi_{j-k}(x_i) & j = k+1, \ldots, k+M. \end{cases}$$

$\mathbf{S}$ is a $(k + M) \times (k + M)$ matrix with zeroes everywhere except in its upper left $k \times k$ block where $S_{ij} = \eta_{md}(\|\mathbf{x}_i^* - \mathbf{x}_j^*\|)$. Finally, $\mathbf{C}$ is an $M \times (k + M)$ matrix such that

$$C_{ij} = \begin{cases} \phi_i(\mathbf{x}_j^*) & j = 1, \ldots, k \\ 0 & j = k+1, \ldots, k+M. \end{cases}$$

This approximation goes back at least to Wahba (1980). Some care is required to choose the knot locations carefully. In one dimension it is usual to choose quantiles

of the empirical distribution of the predictor, or even spacing, but in more dimensions matters are often more difficult. One possibility is to take a random sample of the observed predictor variable combinations; another is to take a 'spatially stratified' sample of the predictor variable combinations. Even spacing is sometimes appropriate, or more sophisticated space filling schemes can be used: Ruppert et al. (2003) provide a useful discussion of the alternatives.

5.5.2 Duchon splines

Thin plate splines have been widely adopted in the statistics literature, but are actually only one of a broader class of splines proposed in Duchon (1977). The larger class is particularly useful if we have reason to base the spline penalty on an order of derivative, m, that does not meet the condition $m > d/2(+1/2)$ for the spline to exist (and be smooth). To see why this might be important, the following table shows how the penalty null space dimension, M, has to grow with the number of covariates of the smooth d, if the spline is to have no first derivative discontinuities.

d	1	2	3	4	5	6	7	8	9	10
m	2	2	3	3	4	4	5	5	6	6
M	2	3	10	15	56	84	330	495	2002	3003

Clearly in the rare cases that it is appropriate to smooth isotropically in high dimensions, the null space dimension makes the thin plate spline rather impractical.

Let $\mathcal{F}$ denote Fourier transform, and $\boldsymbol{\tau}$ frequency. By Plancherel's theorem the thin plate spline penalty (5.6) can be expressed

$$J_{md} = \int_{\Re^d} \sum_{\nu_1+\ldots+\nu_d=m} \frac{m!}{\nu_1!\ldots\nu_d!} \left(\mathcal{F}\frac{\partial^m f}{\partial x_1^{\nu_1}\ldots\partial x_d^{\nu_d}}(\boldsymbol{\tau})\right)^2 d\boldsymbol{\tau}$$

and Duchon (1977) actually considers the more general penalty

$$J_{mds} = \int_{\Re^d} \|\boldsymbol{\tau}\|^{2s} \sum_{\nu_1+\ldots+\nu_d=m} \frac{m!}{\nu_1!\ldots\nu_d!} \left(\mathcal{F}\frac{\partial^m f}{\partial x_1^{\nu_1}\ldots\partial x_d^{\nu_d}}(\boldsymbol{\tau})\right)^2 d\boldsymbol{\tau} \quad (5.10)$$

which is only the familiar thin plate spline penalty when $s = 0$. Under the restrictions $-d/2 < s < d/2$ and $m + s > d/2$ ($+1/2$ if we want first derivative continuity), Duchon (1977) shows that the spline with this penalty can be expressed as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \delta_i \eta_{2m-2s-d}(\|\mathbf{x} - \mathbf{x}_i\|) + \sum_{j=1}^{M} \alpha_j \phi_j(\mathbf{x}).$$

As for the thin plate spline, the ϕ_j span the space of polynomials of order less than m, but the η_q are now given by

$$\eta_q(t) = \begin{cases} (-1)^{(q+1)/2}|t|^q & q \text{ odd} \\ (-1)^{q/2}|t|^q \log|t| & q \text{ even.} \end{cases}$$

The Duchon spline fitting problem has exactly the form (5.8), but with $E_{ij} =$

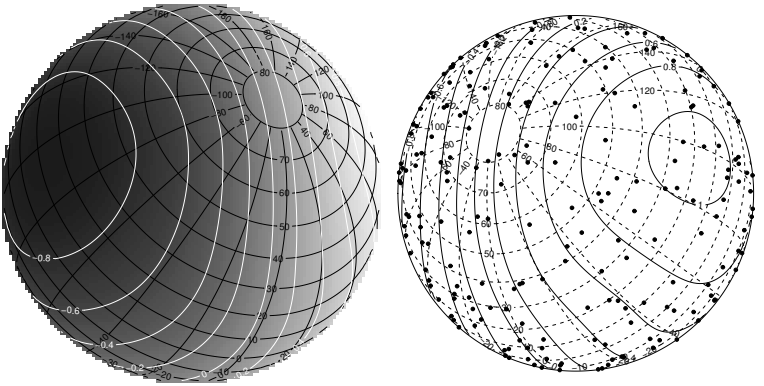

Figure 5.13 *Two visualisations of a spline on the sphere estimated using the* `"sos"` *basis in* `mgcv`. *Left shows a 'heat map' of the estimates on the surface of the sphere, with contours overlaid. Right shows a view from a different direction without the heat map but showing the data locations.*

$\eta_{2m-2s-d}(\|\mathbf{x}_j - \mathbf{x}_i\|)$. It follows that exactly the same rank reduction approaches can be taken as in the normal thin plate spline case. `s(x,z,bs="ds",m=c(1,.5))` specifies a Duchon spline in `mgcv` (here with $m = 1$, $s = 1/2$ and $d = 2$). Notice that if we set $s = (d-1)/2$, then $m = 1$ can be used for any dimension d, with the consequence that the null space basis dimension will be $M = d + 1$.

5.5.3 Splines on the sphere

Sometimes data arrive on the sphere (e.g., satellite data) and it is appropriate to smooth in a way that recognises this[**]. Let θ = latitude and ϕ = longitude. The analogue of the second order thin plate spline penalty on the sphere is

$$J(f) = \int_0^{2\pi} \int_0^{\pi} \left(\frac{f_{\phi\phi}}{\sin^2\theta} + \frac{\{\sin(\theta f_\theta)\}_\theta}{\sin\theta} \right)^2 \sin\theta d\theta d\phi \tag{5.11}$$

where subscripts denote differentiation (note the annoying range for θ here). The operator used in the squared bracket is the Laplace-Beltrami operator.

Let $\mathbf{p} = (\phi, \theta)^\mathsf{T}$ and $\gamma(\mathbf{p}, \mathbf{p}') = \arccos\{\sin\theta \sin\theta' + \cos\theta\cos\theta' cos(\phi - \phi')\}$ be the angle between points $\mathbf{p}$ and $\mathbf{p}'$. Wendelberger (1981) shows that the spline with penalty (5.11) has the representation

$$f(\mathbf{p}) = \alpha + \sum_i^n \gamma_i R(\mathbf{p}, \mathbf{p}_i),$$

where if $x = \cos\{\gamma(\mathbf{p}, \mathbf{p}')\}$ then $R(\mathbf{p}, \mathbf{p}') = r(x)$,

$$r(x) = \begin{cases} 1 - \pi^2/6 + \mathrm{Li}_2(1/2 + x/2) & -1 < x \leq 0 \\ 1 - \log(1/2 + x/2)\log(1/2 - x/2) - \mathrm{Li}_2(1/2 - x/2) & 1 \geq x > 0, \end{cases}$$

[**]I am very grateful to Grace Wahba and Jean Duchon for their help with the material in this section.

and $\mathrm{Li}_2 = \sum_{k=1}^{\infty} x^k/k^2$, $|x| \leq 1$ (the dilogarithm). Since we only require $\mathrm{Li}_2(x)$ for $|x| \leq 1/2$ then in double precision this series converges in 50 terms or fewer.

The estimates of $\boldsymbol{\gamma}$ and α minimize

$$\|\mathbf{y} - \mathbf{R}\boldsymbol{\gamma} - \mathbf{1}\alpha\|^2 + \lambda \boldsymbol{\gamma}^{\mathsf{T}} \mathbf{R} \boldsymbol{\gamma} \text{ s.t. } \mathbf{1}^{\mathsf{T}} \boldsymbol{\gamma} = 0$$

where $\mathbf{R}$ is the matrix with i, j^{th} element $R(\mathbf{p}_i, \mathbf{p}_j)$. Again this is essentially the same structure as the thin plate spline problem, and the same eigen or 'knot' based rank reduction is possible.

Wahba (1981) avoids evaluation of the dilogarithm by modifying the penalty a little so that the corresponding R has the simpler form,

$$R(\mathbf{p}, \mathbf{p}') = \frac{1}{2\pi}\{q_2(1-x)/4 - 1/6\}$$

where $q_2(z) = \log\left(1 + \sqrt{2/z}\right)\left(3z^2 - 2z\right) - 6z^{3/2}/\sqrt{2} + 3z + 1$. She also considers different orders of penalty.

Duchon has also proposed smoothing with respect to the 3-dimensional coordinates of the points on the sphere's surface, but using a Duchon spline with $m = 2$, $s = 1/2$ and $d = 3$. In limited simulation testing the three approaches have similar performance, with the mean square error for the Duchon approach being slightly better than the Wendelberger (1981) method, which is slightly better than the Wahba (1981) method. In `mgcv` the `"sos"` basis implements all three variants. See figure 5.13.

5.5.4 *Soap film smoothing over finite domains*

Sometimes data are gathered within a domain having a complicated boundary, and it is important not to 'smooth across' boundary features. Figure 5.14 provides an example: the top left figure is a remote sensed image of chlorophyll concentration in the Aral Sea. If we want to smooth the chlorophyll concentrations, it is important not to 'smooth across' the peninsula separating the two arms of the sea. The middle panel in the top row shows an attempt to smooth using a thin plate spline: notice how the densities on the eastern edge of the western arm are too high, as an artefact of smoothing, and the high concentrations at the western edge of the eastern arm. If data coverage is uneven then this effect can become much worse, as the thinned version of the image on the lower row emphasises: the thin plate spline artefacts are then very large. The right panels of the figure show the much better results of smoothing with a 'soap film' smoother (Wood et al., 2008), constructed to avoid such artefacts. This section explains the construction.

Figure 5.15 illustrates the basic idea. Consider a loop of wire (black curve) following the boundary (grey loop) of the region, Ω, of the $x - y$ plane over which we are interested in smoothing. The z coordinate of the loop gives the known function values at the boundary. One definition of a 'completely smooth' function over this domain can be obtained by considering a soap film supported by the boundary wire (in zero gravity): see the surface in the middle panel of figure 5.15. Assuming a

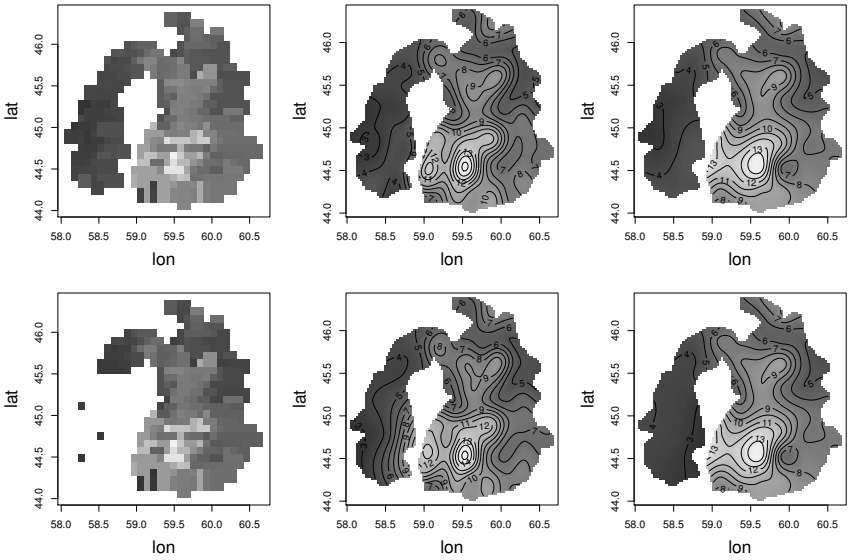

Figure 5.14 *Top row. The left panel is raw remote sensed chlorophyll levels in the Aral Sea. The middle panel is a smoothed version of the left panel data, using a thin plate spline — notice the tendency to 'smooth across' the peninsula. The right panel is a soap film smoother, which avoids smoothing across the peninsula. Bottom row. As top row, but with most of the data in the western arm of the sea dropped (at random), the thin plate spline now shows severe 'leakage' artefacts, while the soap film behaves more reasonably.*

'small' z range for the wire, the height of the soap film inside the boundary is given by the function f satisfying

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

and the boundary conditions: i.e., the soap film adopts a minimum surface tension configuration. In order to smooth noisily observed z data over the domain, the soap film should distort smoothly from its minimum energy configuration by moving vertically towards the data (right panel of figure 5.15). An appropriate measure of the total degree of distortion might be:

$$J_\Omega(f) = \int_\Omega \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^2 dxdy.$$

This differs from a thin plate spline penalty functional in three respects: it is only integrated over Ω, rather than the whole x, y plane, there is no mixed second derivative term, and the sum of the second derivative terms is squared, rather than the terms being separately squared. Hence the second derivatives with respect to x and y can be traded off against each other, so that the space of functions for which $J_\Omega(f)$ is zero is infinite dimensional, and functions with zero penalty can be curved enough to meet any smooth boundary condition.

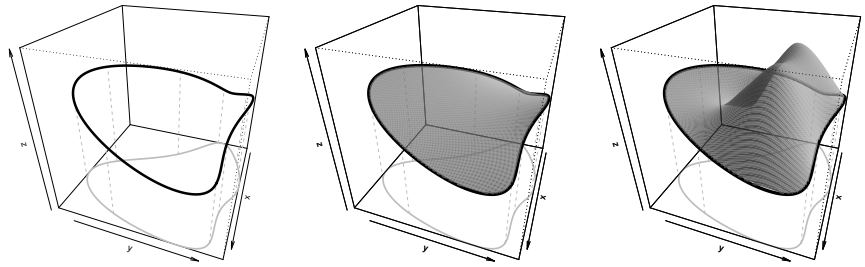

Figure 5.15 *Soap film smoothing idea. Left: the boundary of a finite region in the x, y plane is shown in grey. Suppose the smooth has a known value on the boundary, shown as the thick black curve. Middle: the smoothest surface satisfying the boundary condition is a soap film matching the boundary. Right: the soap film is allowed to distort smoothly towards noisy z data observed over the domain. In reality the boundary condition is usually unknown, so we represent it using a cyclic spline.*

If we have n data points, z_k, which are noisy observations of $h(x_k, y_k)$ where h is a smooth function over the domain, we might seek to estimate h by minimizing (subject to the known boundary conditions)

$$\sum_{i=1}^{n} \{z_i - f(x_i, y_i)\}^2 + \lambda J_\Omega(f) \tag{5.12}$$

w.r.t. f. Here λ is a tunable smoothing parameter. This problem was first considered by Ramsay (2002) (using a different motivation), but the resulting finite element fitting methods do not provide the simple basis-penalty representation that is convenient in the GAM context. To this end the following theorem and lemma from Wood et al. (2008) are helpful

Theorem 2 (Soap film interpolation). *Consider a smooth function $f^*(x, y)$ over the x, y plane. Let B be a collection of closed loops in the x, y plane, such that no two loops intersect and one 'outer' loop encloses all the others. Let Ω be the region made up of all x, y points which are interior to an odd total number of these loops. Suppose that $f^*(x, y)$ is known exactly on B, and that $z_k = f^*(x_k, y_k)$ are observations of f^* at n locations x_k, y_k within Ω. Let $f(x, y)$ be the function which*

1. *interpolates the known f^* values on B and the z_ks at the n points (x_k, y_k);*
2. *satisfies $\partial^2 f/\partial x^2 + \partial^2 f/\partial y^2 = 0$ on B; and*
3. *minimizes*

$$J_\Omega(f) = \int_\Omega \left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right)^2 dxdy.$$

Then f is the function, meeting condition 1, such that

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \rho, \tag{5.13}$$

where

$$\frac{\partial^2 \rho}{\partial x^2} + \frac{\partial^2 \rho}{\partial y^2} = 0 \tag{5.14}$$

except at the x_k, y_k points, and $\rho = 0$ on B. Equations (5.13) and (5.14) are the Poisson and Laplace equations, respectively.

Proof. See Wood et al. (2008) □

Lemma 1 (Soap film smoothing). *Let the setup be as above, except that the z_k are now measured with error, and $\mathbf{f} = [f(x_1, y_1), f(x_2, y_2), \ldots, f(x_n, y_n)]^{\mathsf{T}}$. The function, $f(x, y)$, minimizing*

$$\|\mathbf{z} - \mathbf{f}\|^2 + \lambda_f J_\Omega(f) \tag{5.15}$$

subject to the known conditions on B must satisfy (5.13) and (5.14).

Proof. If the minimizing function were $\tilde{f}$, different to f, then we could interpolate the values in $\mathbf{f}$ using a soap film interpolant, thereby leaving $\|\mathbf{z} - \mathbf{f}\|^2$ unchanged but reducing J_Ω, by the minimum J_Ω interpolant property of the soap film interpolant. That is, there is a contradiction unless $\tilde{f} = f$. □

Characterization of the soap film smoother in terms of the solution of (5.13) and (5.14) is the key to evaluation of the basis functions and penalty. Let $\rho_k(x, y)$ denote the function which is zero on B, satisfies (5.14) everywhere in Ω except at the single point x_k, y_k (the k^{th} data location) and satisfies $\int_\Omega \rho_k(x, y) dx dy = 1$. Then any function which satisfies (5.14) everywhere in Ω, except at the set of points $\{x_k, y_k : k = 1, \ldots, n\}$, can be written as

$$\rho(x, y) = \sum_{k=1}^{n} \gamma_k \rho_k(x, y),$$

where the γ_k are coefficients. It is then straightforward to confirm that

$$J_\Omega(f) = \boldsymbol{\gamma}^{\mathsf{T}} \mathbf{S} \boldsymbol{\gamma},$$

where $\mathbf{S}$ is a matrix of fixed coefficients given by

$$\mathbf{S}_{i,j} = \int_\Omega \rho_i(x, y) \rho_j(x, y) dx dy.$$

The function f can also be written in terms of the γ_k. Let $a(x, y)$ be the solution to (5.13) with $\rho(x, y) = 0 \; \forall \; x, y$, subject to the known boundary condition on B. Now define $g_i(x, y)$ as the solution to (5.13) with $\rho(x, y) = \rho_i(x, y)$ and the boundary condition that $f = 0$ on B; linearity of (5.13) implies that the soap film smoother can be written as

$$f(x, y) = a(x, y) + \sum_{k=1}^{n} \gamma_k g_k(x, y). \tag{5.16}$$

To obtain the basis functions ρ_j, g_j and a, the required partial differential equations can be efficiently solved by multigrid methods (see, e.g., Press et al., 2007). Alternatively, simple finite difference discretization of the derivatives followed by direct solution of the resulting sparse matrix systems using the `Matrix` package in R seems to be even more efficient (Bates and Maechler, 2015; Davis, 2006).

We do not usually want to work with n basis functions for n data, and instead choose a representative set of x, y 'knots' and compute the basis functions only for these. It is also rare to know the value of the smooth on B, and instead we can let the boundary condition be given by a cyclic spline around the boundary. In this case let $a_j(x, y)$ be the solution to (5.13) with $\rho(x, y) = 0 \; \forall \; x, y$, subject to the boundary condition given by setting all the cyclic boundary spline coefficients, β_i, to zero, except for the j^{th} which is set to 1. Then we can substitute $a(x, y) = \sum_j \beta_j a_j(x, y)$ in (5.16). Terms like `s(x,y,bs="so",xt=list(bnd=list(bnd)))` in `mgcv` implement soap film smooths. The list supplied in the `xt` argument defines the boundary. See Wood et al. (2008) and `?soap` for more details. Wang and Ranalli (2007) or Miller and Wood (2014) provide alternative approaches.

5.6 Tensor product smooth interactions

The isotropy of the smooths considered above is often considered to be desirable when modelling something as a smooth function of geographic coordinates,[††] but it has some disadvantages. Chief among them is the difficulty of knowing how to scale predictors relative to one another, when both are arguments of the same smooth but they are measured in fundamentally different units. For example, consider a smooth function of a single spatial coordinate and time: the implied relative importance of smoothness in time versus smoothness in space is very different between a situation in which the units are metres and hours, compared to that in which the units are light-years and nanoseconds. One pragmatic approach is to scale all predictors into the unit square, as is often done in LOESS smoothing, but this is essentially arbitrary. A more satisfactory approach uses *tensor product smooths*. Whereas the thin plate spline generalizes from one dimension to multiple dimensions by exchanging a flexible strip for a flexible sheet, the tensor product smooth can be interpreted as exchanging a single flexible strip for a lattice work of strips (as is illustrated in figure 5.17), with the component strips having different stiffness in different directions.

5.6.1 Tensor product bases

The basic approach of this section is to start from smooths of single covariates, represented using any basis with an associated quadratic penalty measuring 'wiggliness' of the smooth. From these 'marginal smooths' a 'tensor product' construction is used to build up smooths of several variables. See de Boor (1978) for an important early reference on tensor product spline bases, and Wood (2006a) for this construction.

The methods developed here can be used to construct smooth functions of *any*

[††] Although it's possible to overstate the case for doing this: in many applications at many locations North-South is not the same as East-West.

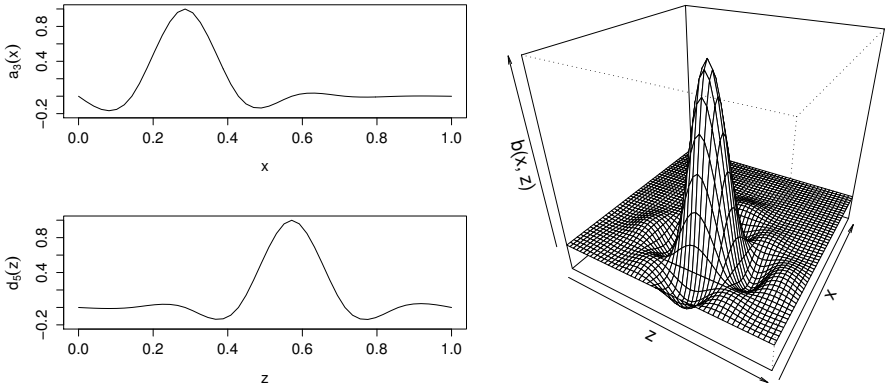

Figure 5.16 *How the product of two marginal basis functions for smooth functions of x and z, separately, results in a basis function for a smooth function of x and z together. The two left panels show the 3rd and 5th basis functions for rank 8 cubic regression spline smooths of x and z, respectively. The right hand plot shows $a_3(x)d_5(z)$, one of 64 similar basis functions of the tensor product smooth derived from these two marginal smooths.*

number of covariates, but the simplest introduction is via the construction of a smooth function of 3 covariates, x, z and v, the generalization then being trivial. The process starts by assuming that we have low rank bases available for representing smooth functions f_x, f_z and f_v of each of the covariates. That is, we can write:

$$f_x(x) = \sum_{i=1}^{I} \alpha_i a_i(x), \quad f_z(z) = \sum_{l=1}^{L} \delta_l d_l(z) \quad \text{and} \quad f_v(v) = \sum_{k=1}^{K} \beta_k b_k(v),$$

where the α_i, δ_l and β_k are parameters, and the $a_i(x)$, $d_l(z)$ and $b_k(v)$ are known basis functions.

Now consider how f_x, the smooth function of x, could be converted into a smooth function of x and z. What is required is for f_x to vary smoothly with z, and this can be achieved by allowing its parameters, α_i, to vary smoothly with z. Using the basis already available for representing smooth functions of z we could write:

$$\alpha_i(z) = \sum_{l=1}^{L} \delta_{il} d_l(z)$$

which immediately gives

$$f_{xz}(x, z) = \sum_{i=1}^{I} \sum_{l=1}^{L} \delta_{il} d_l(z) a_i(x).$$

Figure 5.16 illustrates this construction. Continuing in the same way, we could now

create a smooth function of x, z and v by allowing f_{xz} to vary smoothly with v. Again, the obvious way to do this is to let the parameters of f_{xz} vary smoothly with v, and following the same reasoning as before we get

$$f_{xzv}(x,z,v) = \sum_{i=1}^{I}\sum_{l=1}^{L}\sum_{k=1}^{K}\beta_{ilk}b_k(v)d_l(z)a_i(x).$$

For any particular set of observations of x, z and v, there is a simple relationship between the model matrix, $\mathbf{X}$, evaluating the tensor product smooth at these observations, and the model matrices $\mathbf{X}_x$, $\mathbf{X}_z$ and $\mathbf{X}_v$ that would evaluate the marginal smooths at the same observations. If $\odot$ is the row-wise Kronecker product (see section B.4), then it is easy to show that, given appropriate ordering of the β_{ilk} into a vector $\boldsymbol{\beta}$,

$$\mathbf{X} = \mathbf{X}_x \odot \mathbf{X}_z \odot \mathbf{X}_v.$$

Clearly, (i) this construction can be continued for as many covariates as are required; (ii) the result is independent of the order in which we treat the covariates and (iii) the covariates can themselves be vector covariates. Figure 5.17 attempts to illustrate the tensor product construction for a smooth of two covariates.

5.6.2 Tensor product penalties

Having derived a 'tensor product' basis for representing smooth functions, it is also necessary to have some way of measuring function 'wiggliness', if the basis is to be useful for representing smooth functions in a GAM context. Again, it is possible to start from wiggliness measures associated with the marginal smooth functions, and again the three covariate case provides sufficient illustration. Suppose that each marginal smooth has an associated functional that measures function wiggliness and can be expressed as a quadratic form in the marginal parameters. That is

$$J_x(f_x) = \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{S}_x\boldsymbol{\alpha}, \quad J_z(f_z) = \boldsymbol{\delta}^{\mathsf{T}}\mathbf{S}_z\boldsymbol{\delta} \text{ and } J_v(f_v) = \mathcal{B}^{\mathsf{T}}\mathbf{S}_v\mathcal{B}.$$

The $\mathbf{S}_\bullet$ matrices contain known coefficients, and $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$ and $\mathcal{B}$ are vectors of coefficients of the marginal smooths. An example of a penalty functional is the cubic spline penalty, $J_x(f_x) = \int f_x''(x)^2dx$. Now let $f_{x|zv}(x)$ be $f_{xvz}(x,z,v)$ considered as a function of x only, with z and v held constant, and define $f_{z|xv}(z)$ and $f_{v|xz}(v)$ similarly. A natural way of measuring wiggliness of f_{xzv} is to use:

$$J(f_{xzv}) = \lambda_x\int_{z,v} J_x(f_{x|zv})dzdv + \lambda_z\int_{x,v} J_z(f_{z|xv})dxdv + \lambda_v\int_{x,z} J_v(f_{v|xz})dxdz$$

where the $\lambda_\bullet$ are smoothing parameters controlling the tradeoff between wiggliness in different directions, and allowing the penalty to be invariant to the relative scaling of the covariates. As an example, if cubic spline penalties were used as the marginal penalties, then

$$J(f) = \int_{x,z,v}\lambda_x\left(\frac{\partial^2 f}{\partial x^2}\right)^2 + \lambda_z\left(\frac{\partial^2 f}{\partial z^2}\right)^2 + \lambda_v\left(\frac{\partial^2 f}{\partial v^2}\right)^2 dxdzdv.$$

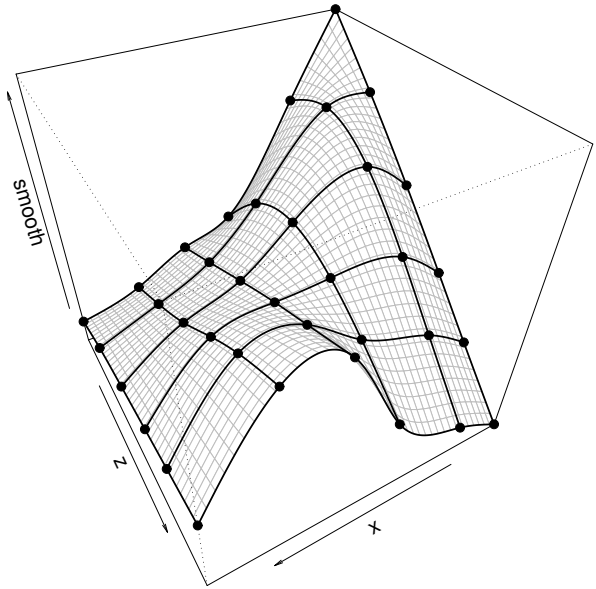

Figure 5.17 *Illustration of a tensor product smooth of two variables x and z, constructed from two rank 6 marginal bases. Following section 5.6.2 a tensor product smooth can always be parameterized in terms of the values of the function at a set of 'knots' spread over the function domain on a regular mesh: i.e. in terms of the heights of the •'s shown. The basis construction can be thought of as follows: start with a smooth of x parameterized in terms of function values at a set of 'knots'; to make the smooth of x vary smoothly with z, simply allow each of its parameters to vary smoothly with z. This can be done by representing each parameter using a smooth of z, also parameterized in terms of function values at a set of 'knots'. Exactly the same smooth arises if we reverse the roles of x and z in this construction. The tensor product smooth penalty in the x direction, advocated in section 5.6.2, is simply the sum of the marginal wiggliness measure for the smooth of x, applied to the thick black curves parallel to the x axes. The z penalty is similarly defined in terms of the marginal penalty of the smooth of z applied to the thick black curves parallel to the z-axis.*

Hence, if the marginal penalties are easily interpretable, in terms of function shape, then so is the induced penalty. Numerical evaluation of the integrals in J is straightforward. As an example consider the penalty in the x direction. The function $f_{x|zv}(x)$ can be written as

$$f_{x|zv}(x) = \sum_{i=1}^{I} \alpha_i(z,v) a_i(x),$$

and it is always possible to find the matrix of coefficients $\mathbf{M}_{z,v}$ such that $\boldsymbol{\alpha}(z,v) = \mathbf{M}_{zv}\boldsymbol{\beta}$ where $\boldsymbol{\beta}$ is the vector of β_{ilk} arranged in some appropriate order. Hence

$$J_x(f_{x|zv}) = \boldsymbol{\alpha}(z,v)^{\mathsf{T}}\mathbf{S}_x\boldsymbol{\alpha}(z,v) = \boldsymbol{\beta}^{\mathsf{T}}\mathbf{M}_{zv}^{\mathsf{T}}\mathbf{S}_x\mathbf{M}_{zv}\boldsymbol{\beta}$$

and so

$$\int_{z,v} J_x(f_{x|zv})dzdv = \boldsymbol{\beta}^{\mathsf{T}} \int_{z,v} \mathbf{M}_{zv}^{\mathsf{T}} \mathbf{S}_x \mathbf{M}_{zv} dzdv \boldsymbol{\beta}.$$

The last integral can be performed numerically (in fact Reiss et al. (2014) show how to obtain it directly), and it is clear that the same approach can be applied to all components of the penalty. However, a simple re-parameterization can be used to provide a straightforward approximation to the terms in the penalty, which performs well in practice.

To see how the approach works, consider the marginal smooth f_x. Let $\{x_i^* : i = 1, \ldots, I\}$ be a set of values of x spread evenly through the range of the observed x values. In this case we can always re-parameterize f_x in terms of new parameters

$$\alpha_i' = f_x(x_i^*).$$

Clearly under this re-parameterization $\boldsymbol{\alpha}' = \boldsymbol{\Gamma}\boldsymbol{\alpha}$ where $\Gamma_{ij} = a_i(x_j^*)$. Hence the marginal model matrix becomes $\mathbf{X}_x' = \mathbf{X}_x\boldsymbol{\Gamma}^{-1}$ and the penalty coefficient matrix becomes $\mathbf{S}_x' = \boldsymbol{\Gamma}^{-\mathsf{T}}\mathbf{S}_x\boldsymbol{\Gamma}^{-1}$.

Now suppose that the same sort of re-parameterization is applied to the marginal smooths f_v and f_z. In this case we have that

$$\int_{z,v} J_x(f_{x|zv})dzdv \approx h \sum_{lk} J_x(f_{x|z_l^* v_k^*}),$$

where h is some constant of proportionality related to the spacing of the z_l^* and v_k^*. Similar expressions hold for the other integrals making up J. If $\otimes$ is the Kronecker product (see section B.4), it is straightforward to show that the summation in the above approximation is:

$$J_x^*(f_{xzv}) = \boldsymbol{\beta}^{\mathsf{T}}\tilde{\mathbf{S}}_x\boldsymbol{\beta} \text{ where } \tilde{\mathbf{S}}_x = \mathbf{S}_x' \otimes \mathbf{I}_L \otimes \mathbf{I}_K$$

and $\mathbf{I}_L$ is the rank L identity matrix. Exactly similar definitions hold for the other components of the penalty so that

$$J_z^*(f_{xzv}) = \boldsymbol{\beta}^{\mathsf{T}}\tilde{\mathbf{S}}_z\boldsymbol{\beta} \text{ where } \tilde{\mathbf{S}}_z = \mathbf{I}_I \otimes \mathbf{S}_z' \otimes \mathbf{I}_K$$

and

$$J_v^*(f_{xzv}) = \boldsymbol{\beta}^{\mathsf{T}}\tilde{\mathbf{S}}_v\boldsymbol{\beta} \text{ where } \tilde{\mathbf{S}}_v = \mathbf{I}_I \otimes \mathbf{I}_L \otimes \mathbf{S}_v'.$$

Hence

$$J(f_{xzv}) \approx J^*(f_{xzv}) = \lambda_x J_x^*(f_{xzv}) + \lambda_z J_z^*(f_{xzv}) + \lambda_v J_v^*(f_{xzv}),$$

where any constants, h, have been absorbed into the λ_j. Again, this penalty construction clearly generalizes to any number of covariates. Figure 5.17 attempts to illustrate what the penalties actually measure, for a smooth of two variables.

Given its model matrix and penalties, the coefficients and smoothing parameters of a tensor product smooth can be estimated as GAM components using the methods

of Chapter 6. These smooths have the nice property of being invariant to rescaling of the covariates, provided only that the marginal smooths are similarly invariant (which is always the case in practice).

Note that it is possible to omit the re-parameterization of the marginal smooths, in terms of function values, and to work with penalties of the form

$$\boldsymbol{\beta}^{\mathsf{T}}\bar{\mathbf{S}}_z\boldsymbol{\beta} \text{ where } \bar{\mathbf{S}}_z = \mathbf{I}_I \otimes \mathbf{S}_z \otimes \mathbf{I}_K,$$

for example; Eilers and Marx (2003) successfully used this approach to smooth with respect to two variables using tensor products of B-splines. A potential problem is that the penalties no longer have the interpretation in terms of (averaged) function shape that is inherited from the marginal smooths when re-parameterization is used. Another proposal in the literature is to use single penalties of the form:

$$\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta} \text{ where } \mathbf{S} = \mathbf{S}_1 \otimes \mathbf{S}_2 \otimes \cdots \otimes \mathbf{S}_d,$$

but this often leads to severe under-smoothing. The rank of $\mathbf{S}$ is the product of the ranks of the $\mathbf{S}_j$, and in practice this is often far too low for practical work. For example, consider a smooth of 3 predictors constructed as a tensor product smooth of 3 cubic spline bases, each of rank 5. The resulting smooth would have 125 free parameters, but a penalty matrix of rank 27. This means that varying the weight given to the penalty would only result in the effective degrees of freedom for the smooth varying between 98 and 125: not a very useful range. By contrast, for the same marginal bases, the multiple term penalties would have rank 117, leading to a much more useful range of effective degrees of freedom of between 8 and 125.

In `mgcv`, tensor product smooths are specified using `te` terms in model formulae. Any single smoothing parameter smooth that can be specified using an `s` term can be used as a marginal smooth for the term. For example `te(x,z)` would specify a tensor product of two (default) 5-dimensional `"cr"` basis smoothers, resulting in a 25-dimensional basis with two penalties (one for each direction). A more complicated specification is `s(x,z,t,d=c(2,1),bs=c("tp","cr"),k=c(20,6))`:a tensor product smooth constructed from a rank 20 thin plate spline of the 2 variables x and z and a rank 6 cubic regression spline of the single variable t. The result has a basis dimension of 120, and two penalties.

5.6.3 ANOVA decompositions of smooths

Sometimes it is useful to include model components such as

$$f_1(x) + f_2(z) + f_3(x, z), \tag{5.17}$$

where f_1 and f_2 are smooth 'main effects' and f_3 is a smooth 'interaction'. Whether nature is really so kind as to arrange matters in a manner so tidily comprehensible is a moot point, but the decomposition can be useful for testing whether additivity ($f_1(x) + f_2(z)$) is appropriate, for example.

In principle, terms like (5.17) can be estimated by using any bases we like for the f_j terms, testing for linear dependencies between them, and imposing constraints

to remove any that are found. `mgcv` will do such testing and constraint by default (see below), but such an approach does not leave $f_1 + f_2$ orthogonal to f_3, so that interpretation of the resulting fits is complicated. Ideally we would rather have f_3 constructed to include no components of the form $f_1 + f_2$.

Actually, constructing such interaction terms is very easy if we recognise that the tensor product basis construction is exactly the same as the construction used for any interaction in a linear model, as described in section 1.6.3. It follows that if we subject the marginal smooths of a tensor product to sum-to-zero identifiability constraints *before* constructing the tensor product basis, then the resulting interaction smooths do not include the corresponding main effects (exactly as would occur with any interaction in a linear model). What happens is that the sum-to-zero constraints remove the unit function from the span of the marginals, with the result that the tensor product basis will not include the main effect that results from the product of a marginal basis with the unit functions in the other marginal bases. The sum-to-zero constraints have no effect on the penalty, of course.

Exactly as with other linear model interactions the approach is general: for example the three way interaction terms constructed from three sum-to-zero constrained marginals will exclude not only the bases of the three main effects, but also the bases for the three 2-way interactions (i.e., the terms resulting from the product of the constant function and two other marginal bases). This approach is so simple, obvious and general that it only seems to appear in passing in the literature (in contrast to much more complicated constructions). In `mgcv`, `ti` terms specify such pure interactions, with the syntax being the same as for `te` terms.

Numerical identifiability constraints for nested terms

For completeness this section provides a method for imposing constraints to make model components like (5.17) identifiable, whatever bases are chosen. However, in almost all practical cases use of `ti` type terms is a better choice. The principle is as follows. Working through all smooths, starting from smooths of two variables and working up through smooths of more variables:

1. Identify all smooths of fewer or the same number of variables sharing covariates with the current smooth.
2. Numerically identify any linear dependence of the basis for the current smooth on the other smooths identified as sharing its covariates, and constrain the current smooth to remove this.

Let $\mathbf{X}_1$ be the combined model matrix for all the lower order model terms sharing covariates with the smooth of interest, and $\mathbf{X}_2$ be the model matrix of the smooth of interest. Provided $\mathbf{X}_1$ and $\mathbf{X}_2$ are each of full column rank, we can test for dependence between them by forming their QR decomposition in two steps. We seek

$$[\mathbf{X}_1 : \mathbf{X}_2] = \mathbf{QR},$$

where the columns of $\mathbf{Q}$ are columns of an orthogonal matrix, and $\mathbf{R}$ is full rank upper triangular if $[\mathbf{X}_1 : \mathbf{X}_2]$ is of full column rank, and reduced rank if $\mathbf{X}_2$ depends

on $\mathbf{X}_1$. First form the QR decomposition

$$\mathbf{X}_1 = \mathbf{Q}_1 \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix}$$

and let $\mathbf{B}$ be $\mathbf{Q}_1^\mathsf{T}\mathbf{X}_2$ with the first r rows removed, where r is the number of columns of $\mathbf{X}_1$. Now form a second QR decomposition *with pivoting*

$$\mathbf{B} = \mathbf{Q}_2 \begin{bmatrix} \mathbf{R}_2 \\ \mathbf{0} \end{bmatrix}.$$

If some columns of $\mathbf{X}_2$ depend on columns of $\mathbf{X}_1$ then there will be a zero block at the lower right corner of $\mathbf{R}_2$, and columns of $\mathbf{X}_2$ responsible for this block will be identifiable from the record of which columns of $\mathbf{X}_2$ have been pivoted to these final columns. These dependent columns can be removed, to make the model identifiable, or equivalently, the corresponding parameters can be constrained to zero.

If the basis of this algorithm is unclear, note that the implied QR decomposition is

$$[\mathbf{X}_1 : \mathbf{X}_2] = \mathbf{Q}_1 \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 & \bar{\mathbf{B}} \\ \mathbf{0} & \mathbf{R}_2 \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where $\bar{\mathbf{B}}$ is the first r rows of $\mathbf{Q}_1^\mathsf{T}\mathbf{X}_2$.

5.6.4 Tensor product smooths under shape constraints

The section 5.3.6 SCOP-spline approach can be combined with the tensor product construction of section 5.6. If we require a shape restriction along a margin of a tensor product smooth then the penalty and model matrix ($\mathbf{XB}$) for that marginal are set up as in section 5.3.6, the other marginals are set up as normal, and the tensor product basis and penalties are set up from the marginals in the usual way (but without the section 5.6 reparameterization). If the j^{th} basis function for the tensor product involves a basis function from the shape constrained margin for which the corresponding coefficient must be positive, then the coefficient, β_j, of the tensor product must also be positive. So we would represent it as $\beta_j = \exp(\gamma_j)$ where γ_j is unrestricted. Sum-to-zero constraints can be imposed by the column centring approach of section 4.3.1 (p. 175).

Again Pya and Wood (2015) provide fuller details, but there is an important restriction on the type of smooths that can be used as unconstrained margins: the basis functions must be non-negative over the region where the marginal monotonicity restriction is to hold. Unusually this restriction is best understood by stating a theorem (not appearing in Pya and Wood, 2015) which establishes the conditions under which tensor product smooths with SCOP-spline monotonic marginals will be monotonic in that margin. The theorem is pretty much proved by stating it.

Theorem 3. Marginal monotonicity. *Let* $t \in \Re$, $\mathbf{x} \in \Re^d$, *for* $d \geq 1$ *and* $f(t, \mathbf{x}) = \sum_j \alpha_j(t) a_j(\mathbf{x})$ *where* $\alpha'_j(t) \geq 0 \; \forall \; j, t$. *Let* $M = \{j : \alpha_j(t) > 0 \text{ for some } t\}$. *If* $a_j(\mathbf{x}) \geq 0 \; \forall \; \mathbf{x}, j \in M$ *then*

$$\frac{\partial f(t, \mathbf{x})}{\partial t} \geq 0 \; \forall \; \mathbf{x}.$$

Proof. $\partial f(t, \mathbf{x})/\partial t = \sum_j \alpha'_j(t) a_j(\mathbf{x}) \geq 0$ by the conditions of the theorem. □

Corollary. Now for all $j \in M$ let a_j be basis functions multiplied by non-negative coefficients. If any basis function, a_k, is not strictly positive, then there exist coefficients for which $\partial f(t, \mathbf{x})/\partial t < 0$ for some t.

An obvious example is when all coefficients are zero, apart from the k^{th}. So the positive basis function condition is both necessary and sufficient to guarantee that the tensor product smooth is monotonic with respect to the margin t.

Notice that the same argument that justifies column centring for imposing sum-to-zero constraints in section 4.3.1, appears to justify adding constants to basis functions in order to ensure that they meet the positivity restriction over the region of interest, but care is needed. For example if a thin plate spline marginal in x includes basis functions $b_1(x) = 1$ and the linear function $b_2(x) = x$, then we would inadvertently force an overall positive trend in x over much of the tensor product by adopting this approach. Using a B-spline basis does not suffer from this problem, basically because it looks the same under reversal of the ordering of x.

5.6.5 An alternative tensor product construction

An alternative tensor product construction is due to Fabian Scheipl (`mgcv` 1.7-0, 2010, Wood et al., 2013, online 2012). It is a direct reduced rank analogue of the smoothing spline ANOVA methods of Gu (2013) and Wahba (1990), and has the advantage that the penalty terms are non-overlapping sub-matrices of the identity matrix, making estimation with standard mixed modelling software quite easy (the earlier alternative of Lee and Durbán (2011) does not produce non-overlapping penalties). The smooths are constructed in such a way that the main effects, low order interactions, etc., are explicitly present in the basis and can simply be 'read off' from a fitted term.

We start with any set of (unconstrained) marginal smooths, exactly as for the construction already discussed. The next step is to re-parameterize each marginal so that the penalty becomes the identity matrix, with M leading diagonal entries zeroed, where M is the dimension of the penalty null space. Section 5.4.2 can be used to do this, with the addition of a simple linear rescaling of the parameters in order to equate the positive elements of the penalty matrix to 1 (see also section 5.8). Now let us split up each reparameterized model matrix into columns $\mathbf{X}$ corresponding to the zero elements on the leading diagonal of the penalty and columns $\mathbf{Z}$ corresponding to the unit entries. So

$$\mathbf{f} = \mathbf{X}\boldsymbol{\delta} + \mathbf{Z}\mathbf{b}.$$

For maximum interpretability it helps to have the constant function explicitly

present in each marginal basis, but the section 5.4.2 method does not guarantee this. For bases in which the null space basis corresponding to $\mathbf{X}$ is already separated out before re-parameterization it is easy to modify the re-parameterization to ensure that $\mathbf{X}$ has a constant column, but the following simple recipe provides a general automatic re-parameterization method for all cases.

If g is a function in the penalty null space then the additional penalty $\mathcal{P}_N = \sum_i \{g(x_i) - \bar{g}\}^2$ would shrink g towards the space of constant functions. We can write $\mathcal{P}_N = \boldsymbol{\delta}^\mathsf{T}\mathbf{D}^\mathsf{T}\mathbf{D}\boldsymbol{\delta}$ where $\mathbf{D} = \mathbf{X} - \mathbf{1}\mathbf{1}^\mathsf{T}\mathbf{X}/n$ and form the eigen-decomposition

$$\mathbf{D}^\mathsf{T}\mathbf{D} = \boldsymbol{\mathcal{U}}\boldsymbol{\Omega}\boldsymbol{\mathcal{U}}^\mathsf{T}.$$

Reparameterizing so that the null space model matrix is now $\mathbf{X}\boldsymbol{\mathcal{U}}$ ensures that the final column of the new model matrix is constant, provided that the null space of the original penalty includes the constant function in its span.

Having re-parameterized the d marginal smooths so that the j^{th} has unpenalized model matrix $\mathbf{X}^j$ and penalized model matrix $\mathbf{Z}^j$ (with corresponding penalty $\mathbf{I}$), we then initialise a set of matrices $\gamma = \{\mathbf{X}^1, \mathbf{Z}^1\}$, or $\gamma = \{[\mathbf{X}^1], \mathbf{Z}^1\}$ where $[\mathbf{X}^j]$, denotes the set of columns of $\mathbf{X}^j$, each treated as a separate matrix. The following steps are then repeated for i in 2 to d...

1. Form row-wise Kronecker products of $\mathbf{X}^i$ (or of all the columns $[\mathbf{X}^i]$) with all elements of γ.
2. Form row-wise Kronecker products of $\mathbf{Z}^i$ with all elements of γ.
3. Append the matrices resulting from the previous two steps to the set γ.

The model matrix, $\mathbf{X}$, for the tensor product smooth is formed by appending all the elements of γ, columnwise. Each element of γ has an associated identity penalty with an associated smoothing parameter, except for the element(s) which involve(s) no $\mathbf{Z}^j$ term, which is unpenalized. The variant in which $[\mathbf{X}^j]$ replaces $\mathbf{X}^j$ ensures that the smooth is strictly invariant w.r.t. linear rescaling of the covariates, but at the cost of requiring extra penalty terms. In practice the difference between the variants usually seems to be small.

The simplest form of sum-to-zero constraint for such a smooth consists of dropping the constant column from $\mathbf{X}$ while subtracting their means from the remaining columns. Such constraints leave the penalties diagonal and penalizing separate subsets of coefficients. Note that if producing a prediction matrix when using such constraints it is important to subtract the column means of the *original* model matrix used in fitting, and not the column means of the prediction matrix itself. There are also arguments for simply dropping the constant column for fitting, and transforming the coefficients to satisfy the sum to zero constraints after model estimation (see Wood et al., 2013).

What is being penalized?

The construction given above describes the mechanics of setting up the alternative tensor product smoother, without revealing what the corresponding penalties are actually penalizing. As an example, consider the case in which the marginals have

penalties based on integrals of squared derivatives. There are two cases: the penalty on the product of the penalty range spaces of two marginals, and the penalty on the product of a range space and a null space.

Consider marginal smooths $h(\mathbf{x})$ and $g(\mathbf{z})$, with penalties

$$\int \sum_i (D_i h)^2 d\mathbf{x} \quad \text{and} \quad \int \sum_j (\Delta_j g)^2 d\mathbf{z}$$

where the D_i and Δ_i are differential operators.

1. Let δ_k denote the k^{th} element of the operator given by $\sum_i D_i \sum_j \Delta_j$. The penalty on the product of the range spaces of the marginal smooths is

$$\int \sum_k (\delta_k f)^2 d\mathbf{x} d\mathbf{z}.$$

2. The product of the null space basis of g's penalty with the basis for h can be written

$$\sum_j \gamma_j(\mathbf{x}) \tilde{c}_j(\mathbf{z})$$

where the $\tilde{c}_j(\mathbf{z})$ are basis functions for the null space of the penalty on g and $\gamma_j(\mathbf{x})$ is a smooth 'coefficient function' represented using the basis for h. The penalty on the product of the null space of g with the range space of h is then

$$\sum_j \int \sum_i (D_i \gamma_j)^2 d\mathbf{x}.$$

Construction of penalties for products of more than two marginals simply applies these rules iteratively (see Wood et al., 2013, for more detail).

For example, when using cubic spline marginals and the $[\mathbf{X}^j]$ variant construction, with penalties $\int h''(x)^2 dx$, then $f(x, z)$ decomposes into the following separately penalized components:

$$f_x(x) + f_z(z) + f_{x1}(x)z + f_{z1}(z)x + f_{xz}(x, z).$$

The penalties on the four univariate terms are the usual cubic spline penalty inherited from the marginal, while the penalty on f_{xz} is $\int \partial^4 f_{xz}/\partial^2 x \partial^2 z dx dz$.

`t2` terms in `mgcv` model formulae are used to specify tensor product smooths using this construction. Use is the same as `te` except that there is an extra logical argument `full` to indicate whether or not to use the $[\mathbf{X}^j]$ variant construction.

5.7 Isotropy versus scale invariance

Thin plate splines and Duchon splines adapt well to irregularly scattered data and have the advantage of only having one smoothing parameter to estimate. They are often used for spatial data as a result. However, the isotropic smoothness assumption is a strong one: isotropic smoothers are sensitive to linear rescaling of single

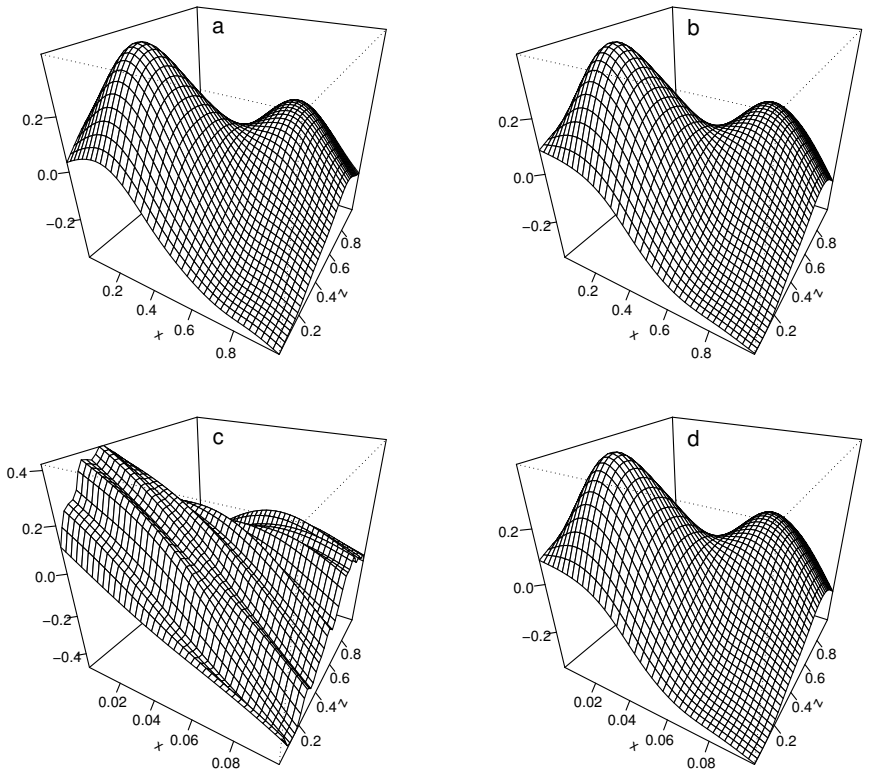

Figure 5.18 *Isotropy versus scale invariance.* (a) *Thin plate spline smooth of noisy observations of a truth looking much like the reconstruction.* (b) *Tensor product smooth of the same data as in a.* (c) *and* (d) *are as a and b, with the single difference that the x values have been divided by 10. Notice how the tensor product smooth (b and d) is invariant to the rescaling, while the isotropic thin plate spline (a and c) is highly sensitive to it.*

covariates (such as occurs under a change of measurement units) in a way that tensor product smooths are not.

Figure 5.18 illustrates this point with four smooths of essentially the same data set. The top row plots show a thin plate spline and a tensor product smooth of the same data, when the smoothing domain is nicely scaled relative to the variability of the true function from which the data were generated. The smooths are very similar (and closely resemble the true function used for data generation). The lower row again shows a thin plate and a tensor product smooth of the same data, but with the x co-ordinates divided by 10. The tensor product smooth is invariant to the change, but the thin plate spline tries to achieve the same smoothness per unit change in x and z, with the result shown in the lower left plot.

5.8 Smooths, random fields and random effects

As we saw in section 4.2.4 (p. 172), one view of the smoothing process is that the smoothing penalty, $\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}$, is employed during fitting in order to impose the belief that the true function is more likely to be smooth than wiggly. In that case we might as well express the belief in a Bayesian manner, by defining a prior distribution on function wiggliness $f(\boldsymbol{\beta}) \propto \exp(-\lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}/2)$, which is equivalent to

$$\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}^{-}/\lambda), \tag{5.18}$$

where $\mathbf{S}^{-}$ is a pseudoinverse of $\mathbf{S}$ (since $\mathbf{S}$ is rank deficient by the dimension of the penalty null space). Given the smoothing basis, smooth model $y_i = f(x_i) + \epsilon_i$, $\epsilon_i \sim N(0, \sigma^2)$ can be expressed as $\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \mathbf{I}\sigma^2)$. Exactly as in section 2.4 (p. 78), we can then seek the posterior mean/mode, $\hat{\boldsymbol{\beta}}$, to maximize $f(\mathbf{y}|\boldsymbol{\beta})f(\boldsymbol{\beta})$. Taking logs, multiplying by -1 and discarding irrelevant constants we find that

$$\hat{\boldsymbol{\beta}} = \underset{\beta}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2/\sigma^2 + \lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}.$$

It is also common to absorb σ^2 into λ so that it does not appear explicitly in the objective function. Notice how the smooth has the mathematical structure of a random effect in a linear mixed model. This is useful computationally, but requires the qualification that the smooth is really best thought of in a Bayesian manner. This is because we do not usually expect $\boldsymbol{\beta}$ to be re-drawn from (5.18) on each replication of the data, as we usually would for a frequentist random effect.

We can also follow section 2.4.1 (p. 79), to obtain the posterior of $\boldsymbol{\beta}$,

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, (\mathbf{X}^{\mathsf{T}}\mathbf{X}/\sigma^2 + \lambda\mathbf{S})^{-1}),$$

and, unsurprisingly, the machinery of section 3.4 (p. 147) also allows us to include smooths in generalized linear mixed models in the same way as random effects.

If $\mathbf{f}$ is the vector containing the values of the smooth evaluated at the observed covariate values, then (5.18) implies the prior $\mathbf{f} \sim N(\mathbf{0}, \mathbf{X}\mathbf{S}^{-}\mathbf{X}^{\mathsf{T}})$. This emphasises that we can view the smooth as a Gaussian random field, with posterior mode $\hat{\mathbf{f}}$.

On occasion it is useful to express the smooth more explicitly in mixed model form so that, if $\mathbf{f}$ is the vector such that $f_i = f(x_i)$, then

$$\mathbf{f} = \mathbf{X}'\boldsymbol{\beta}' + \mathbf{Z}\mathbf{b}$$

where $\mathbf{b} \sim N(\mathbf{0}, \mathbf{I}\sigma_b^2)$. Here the columns of $\mathbf{X}'$ form a basis for the null space of the smoothing penalty, while the columns of $\mathbf{Z}$ form a basis for its range space. This is particularly useful if smooths are to be estimated using software designed for estimating linear or generalized linear mixed models. Section 5.4.2 provides one obvious way of computing the required matrices. Following that section's notation, partition $\mathbf{U} = [\mathbf{U}^{+} : \mathbf{U}^{0}]$ where $\mathbf{U}^{+}$ are the eigenvectors corresponding to positive eigenvalues of the smoothing penalty matrix, and $\mathbf{U}^{0}$ are the remaining eigenvectors. Then $\mathbf{X}' = \mathbf{Q}\mathbf{U}^{0}$ and $\mathbf{Z} = \mathbf{Q}\mathbf{U}^{+}\mathbf{D}^{-1/2}$. There are also computationally cheaper ways of

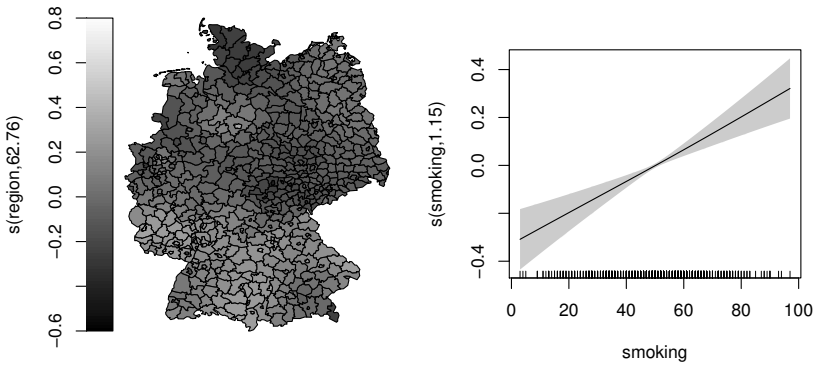

Figure 5.19 *Smooths from a log Poisson additive model of cancer of the larynx recorded by health district in Germany. Left is a GMRF smoother over districts, and right is a cubic spline smooth of smoking level. The near zero width confidence interval in the middle of the smoking variable range is a consequence of the smooth being near linear and the sum to zero identifiability constraint.*

computing the required matrices: for example use a pivoted Cholesky decomposition of the $\mathbf{S}$ matrix, or see section 4.2.4 (p. 172).

This equivalence between smooths, random effects and Gaussian random fields, is discussed in Kimeldorf and Wahba (1970) and Silverman (1985), for example. One consequence is that the computations required to estimate random effects can be used to estimate smooths (e.g., Verbyla et al., 1999; Ruppert et al., 2003), and conversely methods aimed at the smoothing problem can also be used to estimate Gaussian random effects.

5.8.1 Gaussian Markov random fields

Suppose that a region is divided into m distinct districts, indexed j, each with a corresponding parameter γ_j, giving the level of some quantity within the district. Let $\text{nei}(j)$ denote the indices of the districts neighbouring district j, and $\overline{\text{nei}}(j)$ denote the elements of $\text{nei}(j)$ for which $i > j$. Now suppose that we would like neighbouring districts to have similar γ_j estimates. One way to achieve this is to impose the penalty

$$J(\boldsymbol{\gamma}) = \sum_{j=1}^{m} \sum_{i \in \overline{\text{nei}}(j)} (\gamma_j - \gamma_i)^2.$$

That is we sum the squared difference between γ_j values for all pairs of neighbouring districts, with each pair entering the summation only once. In that case

$$J(\boldsymbol{\gamma}) = \boldsymbol{\gamma}^{\mathsf{T}} \mathbf{S} \boldsymbol{\gamma},$$

where $S_{ij} = -1$ if $i \in \text{nei}(j)$ and $S_{jj} = n_j$ where n_j is the number of districts neighbouring district j (not including district j itself). As with any of the smoothers

covered here, the penalty can be viewed as being induced by an improper Gaussian prior

$$\boldsymbol{\gamma} \sim N(\mathbf{0}, \tau \mathbf{S}^-)$$

where τ is some precision parameter. So the $\boldsymbol{\gamma}$ and the neighbourhood structure can be viewed as an (intrinsic) *Gaussian Markov random field* (GMRF), with precision matrix $\mathbf{S}$. It's a *Markov* random field because the precision matrix is sparse.

If we observe n data by district then the GMRF can be viewed as a smoother with model matrix given by an $n \times m$ matrix $\mathbf{X}$ such that $X_{ij} = 1$ if observation i relates to district j and 0 otherwise. The natural parameterization of 5.4.2 can be truncated to produce a reduced rank version.

Terms like `s(loc,bs="mrf",xt=xt)` invoke such smooths in `mgcv`. If `xt` contain a list of polygons defining the district boundaries, then the neighbour relationships can be inferred from these, and they can be used for term plotting, but there are other alternatives for specifying the neighbour relations, and indeed the exact form of the penalty: see `?mrf`. Figure 5.19 illustrates such a smooth, for data on cancer of the larynx: see `?larynx` in the `gamair` package for fitting code. Rue and Held (2005) provide much more on modelling with GMRFs.

5.8.2 Gaussian process regression smoothers

Consider again the model $y_i = f(\mathbf{x}_i) + \epsilon_i$, where f is a random field (smooth function). Let $C(\mathbf{x}, \mathbf{x}_i) = c(\|\mathbf{x} - \mathbf{x}_i\|)$ be a non negative function such that $c(0) = 1$ and $c(d) \to 0$ monotonically as $d \to \infty$. Now consider representing f as

$$f(\mathbf{x}) = (1, \mathbf{x}^\mathsf{T})\boldsymbol{\beta} + \sum_i b_i C(\mathbf{x}, \mathbf{x}_i), \text{ where } \mathbf{b} \sim N(\mathbf{0}, (\lambda \mathbf{C})^{-1}).$$

$C_{ij} = C(\mathbf{x}_j, \mathbf{x}_i)$ and $\boldsymbol{\beta}$ is a vector of parameters. So if $\mathbf{f}$ is the vector such that $f_i = f(\mathbf{x}_i)$ and $\mathbf{B}$ is the matrix with i^{th} row $(1, \mathbf{x}_i^\mathsf{T})$ then $\mathbf{f} = \mathbf{B}^\mathsf{T}\boldsymbol{\beta} + \mathbf{C}\mathbf{b}$, and from basic properties of the transformation of covariance matrices the covariance matrix of $\mathbf{f}$ is $\mathbf{C}/\lambda$. Hence C is interpretable as the correlation function of the random field f, and the posterior mode of f is found by finding $\hat{\boldsymbol{\beta}}$, $\hat{\mathbf{b}}$ to minimize

$$\|\mathbf{y} - \mathbf{B}\boldsymbol{\beta} - \mathbf{C}\mathbf{b}\|^2/\sigma^2 + \lambda \mathbf{b}^\mathsf{T}\mathbf{C}\mathbf{b}. \tag{5.19}$$

Clearly (5.19) has exactly the structure required to use the rank reduction approaches of section 5.5.1, so these can be applied here too.

This approach is known as Gaussian process regression (or Kriging). Extending Handcock et al. (1994), Kammann and Wand (2003) propose $C(\mathbf{x}, \mathbf{x}_i) = c_0(\|\mathbf{x} - \mathbf{x}_i\|)$ where c_0 is the simplified Matérn correlation function

$$c_0(d) = (1 + d/\rho)\exp(-d/\rho).$$

Kammann and Wand (2003) suggest using $\rho = \max_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|$.[‡‡] In `mgcv` such

[‡‡] Although for efficiency reasons, if using a rank reduced version, one should base this on the 'knots' used in basis creation, rather than the whole dataset.

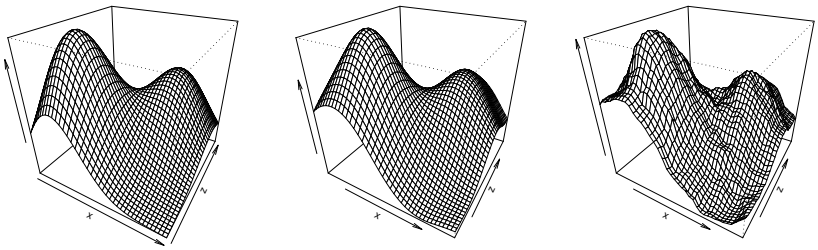

Figure 5.20 *Gaussian process reconstructions of the function shown on the left panel (range 0–0.7), from 300 noisy observations ($\sigma = 0.05$) randomly scattered over the $x - z$ plane. The middle panel uses a Matérn correlation function as suggested by Kammann and Wand (2003). The rougher right panel uses a spherical correlation function with $\rho = .05$.*

'Gaussian process smooths' are specified using terms like `s(x,z,bs="gp")`. The `m` argument can be used to control the form of c. `m[1]` selects from the following

1. The spherical correlation function

$$c(d) = \begin{cases} 1 - 1.5d/\rho + 0.5(d/\rho)^2 & \text{if } d \leq \rho \\ 0 & \text{otherwise.} \end{cases}$$

2. $c(d) = \exp\{(-d/\rho)^\kappa\}$ where $0 < \kappa \leq 2$. The 'power exponential'.
3. $c(d) = (1 + d/\rho)\exp(-d/\rho)$. Matérn with $\kappa = 1.5$.
4. $c(d) = \{1 + d/\rho + (d/\rho)^2/3\}\exp(-d/\rho)$. Matérn with $\kappa = 2.5$.
5. $c(d) = \{1 + d/\rho + 2(d/\rho)^2/5 + (d/\rho)^3/15\}\exp(-d/\rho)$. Matérn with $\kappa = 3.5$.

See Fahrmeir et al. (2013), section 8.1.3, for more details. If present and positive `m[2]` supplies ρ; otherwise the Kammann and Wand (2003) suggestion is used. `m[3]` supplies κ in the power exponential, if present; otherwise this defaults to 1. If `m` is not supplied then option 3 is used. Figure 5.20 illustrates options 3 and 1: the discontinuity in the spherical correlation function leads to quite rough results.

5.9 Choosing the basis dimension

When using penalized regression splines the modeller chooses the basis dimension as part of the model building process. Typically, this substantially reduces the computational burden of modelling, relative to full spline methods. It also recognizes the fact that, usually, something is seriously wrong if a statistical model really *requires* as many coefficients as there are data. Indeed we saw in section 5.2 that asymptotically the basis dimension need grow only rather slowly with sample size: for example $k = O(n^{1/5})$ for cubic regression splines estimated by REML. But, of course, the asymptotic results tell us nothing about what size of basis to use in any particular application: to know what is optimal we would have to know the 'truth' that we are trying to estimate.

In practice then, choice of basis dimension has to remain a part of model specification. But it is important to note that all the choice is doing is to set an upper limit on the flexibility of a term: the actual flexibility is controlled by the smoothing parameters. Hence, provided that we do not make the basis dimension too restrictively small, its exact value should have only a small effect on the fitted model. However, there is one subtle caveat: a function space with basis dimension 20 will contain a larger space of functions with EDF 5 than will a function space of dimension 10 (the numbers being arbitrary), so that model fit tends to retain some sensitivity to basis dimension, even if the appropriate EDF for a term is well below its basis dimension.

Fortunately informal checking that the basis dimension for a particular term is appropriate is quite easy. Suppose that a smooth term is a function of a covariate $\mathbf{x}$ (which may be vector in general). Let $\text{nei}(i)$ denote the set of indices of the m nearest neighbours of $\mathbf{x}_i$ according to an appropriate measure $\|\mathbf{x}_i - \mathbf{x}_j\|$, and compute the mean absolute or squared difference between the deviance residual ϵ_i and $\{\epsilon_j : j \in \text{nei}(i)\}$. Average this difference over i to get a single measure Δ of the difference between residuals and their neighbours. Now repeat the calculation several hundred times for randomly reshuffled residuals. If there is no residual pattern with respect to covariate $\mathbf{x}$ then Δ should look like an ordinary observation from the distribution of Δ under random re-shuffling, but if the original Δ is unusually small this means that residuals appear positively correlated with their neighbours, suggesting that we may have under-smoothed. In that case if the EDF is also close to the basis dimension then it may be appropriate to increase the basis dimension. Note, however, that un-modelled residual auto-correlation and an incorrectly specified mean variance relationship can both lead to the same result, even when the basis dimension is perfectly reasonable. This residual randomization test is easily automated and computationally efficient (given an efficient nearest neighbour algorithm). It is implemented by default in `gam.check` in `mgcv`.

An alternative to this simple check is to extract the deviance residuals for the full model and try smoothing them w.r.t. the covariates of the term being checked, *but using an increased basis dimension* (e.g., doubled). Smoothing can be done using an identity link, constant variance model. If the original term is really missing variation w.r.t. the covariates, then this approach will pick it up, as the estimated smooth of the residuals will be relatively complicated, rather than very smooth and nearly zero.

In summary, the modeller needs to decide roughly how large a basis dimension is fairly certain to provide just adequate flexibility, in any particular application, and use that. Informal and automatic residual checking can then help to check this assumption.

5.10 Generalized smoothing splines

Much smoothing spline theory is built around some very elegant and general methods based on the theory of reproducing kernel Hilbert spaces. The aim of this section is to give a brief introduction to this theory.

To reduce the level of abstraction, a little, let us revisit the cubic smoothing spline. To this end, first construct a space of functions of x, say, which are 'wiggly' accord-

ing to the cubic spline penalty. That is, consider a space of functions, $\mathcal{F}$, where the inner product of two functions, f and $g \in \mathcal{F}$, is $\langle g, f\rangle = \int g''(x)f''(x)dx$, and consequently a norm on the space is the cubic spline penalty, $\int f''(x)^2dx$: except for the zero function, functions for which this norm is zero are currently excluded from $\mathcal{F}$.

There is a rather remarkable theorem, the Riesz representation theorem, which says that there exists a function $R_z \in \mathcal{F}$, such that $f(z) = \langle R_z, f\rangle$, for any $f \in \mathcal{F}$: so if we want to evaluate f at some particular value z, then one way of doing it is to take the inner product of the function, f, with the 'representor of evaluation' function, R_z.

Suppose further that we construct a function of two variables $R(z, x)$, such that for any z, $R(z, x) = R_z(x)$. This function is known as the *reproducing kernel* of the space, since $\langle R(z, \cdot), R(\cdot, t)\rangle = R(z, t)$, i.e.,

$$\int \frac{\partial^2 R(z,x)}{\partial x^2}\frac{\partial^2 R(x,t)}{\partial x^2}dx = R(z,t),$$

by the Riesz representation theorem. Basically, if you take an appropriately defined inner product of R, with itself, you get back R: hence the terminology.

Now, consider the cubic smoothing spline problem of finding the *function* minimizing:

$$\sum_{i=1}^{n}\{y_i - f(x_i)\}^2 + \lambda\int f''(x)^2dx. \tag{5.20}$$

It turns out that the minimizer is in the space of functions that can be represented as

$$\hat{f}(x) = \beta_0 + \beta_1 x + \sum_{i=1}^{n}\delta_i R(x_i, x),$$

where the β_j and δ_i are coefficients to be estimated. The first two terms on the r.h.s. simply span the space of functions for which $\int f''(x)^2dx = 0$, the null space of the penalty: clearly β_0 and β_1 can be chosen to minimize the sum of squares term without worrying about the effect on the penalty. The terms in the summation represent the part of $\hat{f}(x)$ that is in $\mathcal{F}$. It is clear that not all functions in $\mathcal{F}$ can be represented as $\sum_{i=1}^{n}\delta_i R(x, x_i)$, so how do we know that the minimizer of 5.20 can?

The answer lies in writing the minimizer $\hat{f}$'s component in $\mathcal{F}$ as $r + \eta$ where $r = \sum_{i=1}^{n}\delta_i R(x_i, x)$, and η is any function in the part of $\mathcal{F}$ which is orthogonal to r. Orthogonality means that each inner product $\langle R(x_i, x), \eta(x)\rangle = 0$, but these inner products evaluate $\eta(x_i)$, so we have that $\eta(x_i) = 0$ for each i: so $\eta(x)$ can have no effect on the sum of squares term in (5.20). In the case of the penalty term, orthogonality of r and η means that:

$$\int \hat{f}''(x)^2dx = \int r''(x)^2dx + \int \eta''(x)^2dx.$$

Obviously, the η which minimizes this is the zero function, $\eta(x) = 0$.

Having demonstrated what form the minimizer has, we must now actually find the minimizing β_j and δ_i. Given that we now have a basis, this is not difficult, although it