J. R. Statist. Soc. B (2006)
68, *Part* 2, *pp.* 259–280

Generalized linear array models with applications to multidimensional smoothing

I. D. Currie,

Heriot-Watt University, Edinburgh, UK

M. Durban

Universidad Carlos III de Madrid, Spain

and P. H. C. Eilers

Leiden University Medical Centre, the Netherlands

[Received October 2004. Revised October 2005]

Summary. Data with an array structure are common in statistics, and the design or regression matrix for analysis of such data can often be written as a Kronecker product. Factorial designs, contingency tables and smoothing of data on multidimensional grids are three such general classes of data and models. In such a setting, we develop an arithmetic of arrays which allows us to define the expectation of the data array as a sequence of nested matrix operations on a coefficient array. We show how this arithmetic leads to low storage, high speed computation in the scoring algorithm of the generalized linear model. We refer to a generalized linear array model and apply the methodology to the smoothing of multidimensional arrays. We illustrate our procedure with the analysis of three data sets: mortality data indexed by age at death and year of death, spatially varying microarray background data and disease incidence data indexed by age at death, year of death and month of death.

Keywords: Arrays; *B*-splines; Generalized linear models; Kronecker products; Mixed models; Penalties; Smoothing; Yates's algorithm

1. Introduction

Data arise naturally in the form of arrays: factorial designs and contingency tables are two general classes of such data. There are also many specific examples: mortality data consist of two data matrices, one for deaths and one for exposures, image data and multidimensional optical spectra often consist of data matrices of millions of points, and in health studies we may have disease incidence data classified by age, year and month. Gower (1982) gave an example of a $2 \times 3 \times 4$ factorial design in his paper on Yates's algorithm. In this example, the data are arranged in a $2 \times 3 \times 4$ array. A regression analysis involves flattening the data array to a vector, declaring three factor variables each of length 24 and fitting the model with the standard regression algorithm. This is an attractive recipe since all such models can be fitted with the same procedure, regardless of the number of dimensions in the data array. However, two things are lost. First, the array structure of the data and of the model have been discarded; second, the computational method fails to take advantage of the array structure of the data and the model.

Address for correspondence: I. D. Currie, Department of Actuarial Mathematics and Statistics, Heriot-Watt University, Riccarton, Edinburgh, EH14 4AS, UK.
E-mail: I.D.Currie@hw.ac.uk

 1369–7412/06/68259

It was this second point that Yates (1937) exploited when he proposed his efficient method of calculating the treatment effects.

In this paper we propose a new approach to generalized linear models (GLMs) when data are in array form and the regression or design matrix can be expressed as a Kronecker product, as in the case of a factorial design. We express the linear predictor as a sequence of nested matrix operations on an array of coefficients, thus preserving the array structure of the data throughout. With this approach, Yates's algorithm results naturally, not just as a computational device, but as the way to calculate the treatment effects. However, our methods have a very much wider application than Yates's original proposal: our algorithms can be applied in the setting of GLMs and in particular the smoothing of data in multidimensional arrays, which was our original motivation for this work.

One common way to extend one-dimensional smoothing to higher dimensions is to construct a basis in higher dimensions as the Kronecker or tensor product of the marginal basis for each variable. Smoothness can then be achieved by marginal penalization; see, for example, the R package `mgcv` as implemented by Wood (2000, 2004). However, the Kronecker product is very memory hungry and so storage and computational time quickly become important with this approach; the use of increasingly lower dimensional marginal bases is not a satisfactory solution to the problem. The approach that is described in the present paper offers a way round this problem.

We shall refer to a GLM where the data are in the form of an array and where the model matrix can be expressed as a Kronecker product as a generalized linear array model (GLAM). To be precise, we suppose that we have data $\mathbf{Y}$ that are arranged in a d-dimensional array, $n_1 \times n_2 \times \ldots \times n_d$. The corresponding data vector $\mathbf{y}$ of length $n = n_1 n_2 \ldots n_d$ consists of the elements of $\mathbf{Y}$ arranged in a vector with the first dimension of $\mathbf{Y}$ varying fastest, the second dimension varying next fastest, etc.; this corresponds to how R stores an array (R Development Core Team, 2004). We suppose that the model matrix $\mathbf{X}$ can be written as $\mathbf{X}_d \otimes \ldots \otimes \mathbf{X}_1$ where $\mathbf{X}_i$, $n_i \times c_i$, is the model matrix for the ith marginal variable and $\otimes$ denotes the Kronecker product. The model can be fitted as a standard GLM with data vector $\mathbf{y}$ and model matrix $\mathbf{X}$ by repeated evaluation of the scoring algorithm

$$\mathbf{X}'\tilde{\mathbf{W}}_\delta \mathbf{X}\hat{\boldsymbol{\theta}} = \mathbf{X}'\tilde{\mathbf{W}}_\delta \tilde{\mathbf{z}} \tag{1.1}$$

where the tilde represents an approximate solution, as in $\tilde{\boldsymbol{\theta}}$, and $\hat{\boldsymbol{\theta}}$ is an improved approximation; here the weight matrix $\mathbf{W}_\delta$ is diagonal with elements

$$w_{ii}^{-1} = (\partial \eta_i / \partial \mu_i)^2 \operatorname{var}(y_i),$$

$\mathbf{z} = \boldsymbol{\eta} + \mathbf{W}_\delta^{-1}(\mathbf{y} - \boldsymbol{\mu})$ is the working variable, $\boldsymbol{\eta} = g(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\theta}$ is the linear predictor, $g(\cdot)$ is the link function and $\boldsymbol{\mu} = E(\mathbf{y})$. However, unless the dimensions of the component model matrices $\mathbf{X}_i$ are small there are computational challenges in this approach: we must evaluate

(a) linear functions, $\mathbf{X}\boldsymbol{\theta}$ and $\mathbf{X}'\mathbf{W}_\delta\mathbf{z}$, and
(b) inner products $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$;

in addition we usually require

(c) variances of $\mathbf{X}\hat{\boldsymbol{\theta}}$ which need the diagonal elements of $\mathbf{X}(\mathbf{X}'\mathbf{W}_\delta\mathbf{X})^{-1}\mathbf{X}'$.

Conceptually and computationally our approach is different. Conceptually, we maintain the array structure of the problem throughout: we replace $\mathbf{y}$ by the data array $\mathbf{Y}$, the coefficient vector $\boldsymbol{\theta}$ by a coefficient array $\boldsymbol{\Theta}$, $c_1 \times c_2 \times \ldots \times c_d$, whose dimensions are induced by the marginal matrices $\mathbf{X}_i$, the diagonal weight matrix $\mathbf{W}_\delta$ by a weight array $\mathbf{W}$ and the working variable $\mathbf{z}$ by a working array $\mathbf{Z}$; both $\mathbf{W}$ and $\mathbf{Z}$ have the same dimensions as $\mathbf{Y}$. Computationally, we eval-

uate the linear functions and inner products that are required to evaluate algorithm (1.1) by a sequence of nested matrix operations on a d-dimensional array, i.e. without the construction of the model matrix. The sequential approach solves the storage problem; it is also extremely fast with gains in speed of orders of magnitude over a direct evaluation. The same sequential algorithm also gives the variances of fitted values, thus unifying the calculation of linear functions, weighted inner products and variances.

It is straightforward to apply these ideas to the smoothing of multidimensional arrays. We take the marginal model matrix $\mathbf{X}_i$ to be $\mathbf{B}_i$, the smoother matrix for the ith marginal variable. In this paper we take $\mathbf{B}_i$ to be a regression matrix of B-splines but other bases, such as truncated polynomials, are equally possible. The overall smoother matrix $\mathbf{B}$ is then constructed as the Kronecker product of the $\mathbf{B}_i$. We assume that the number of B-splines in the basis is sufficiently 'rich' that the fitted surface is overfitted. Smoothness is then achieved by penalization. We shall discuss the detail of this in the remainder of the paper but for the moment we observe that if $\mathbf{P}$ is the penalty matrix then algorithm (1.1) becomes

$$(\mathbf{B}'\tilde{\mathbf{W}}_\delta\mathbf{B}+\mathbf{P})\hat{\boldsymbol{\theta}}=\mathbf{B}'\tilde{\mathbf{W}}_\delta\tilde{\mathbf{z}}, \tag{1.2}$$

the penalized scoring algorithm (Eilers and Marx, 1996). None of our remarks on the structure of a GLAM or on the computations that are required to solve algorithm (1.1) is affected by the insertion of $\mathbf{P}$ in equation (1.2).

Eilers *et al.* (2006) contains some of the computational ideas in the present paper; however, the formulation in terms of arrays is new and both simplifies and unifies this earlier work.

Efficient computation with Kronecker products has been the subject of intense activity. Yates (1937) is an early paper in this area, as Gower (1982) pointed out. de Boor (1979) gave a general algorithm for the efficient computation of $\mathbf{Xy}$ and in particular the solution of $\mathbf{X}\boldsymbol{\theta}=\mathbf{y}$. Van Loan (2000) is an excellent general reference to the Kronecker product and its role in fast computation.

The plan of the paper is as follows. In Section 2.1 we introduce the ideas of a GLAM by considering one dimension; we shall see that even here there are some important differences from the usual approach. In Section 2.2 we consider two dimensions and use some mortality data to illustrate our methods. Section 3 is the heart of the paper and we describe the general procedure in d-dimensions for computing linear functions, weighted inner products and variances of fitted values; proofs are provided in Section 4 and associated software details in Section 5. In Section 6 we show that a GLAM has a mixed model representation which retains the array features, i.e. it can be regarded as a generalized linear array mixed model. In Section 7 we give two further examples:

(a) an additive model for microarray data in which each component has the GLAM structure and
(b) a model for a large data set on disease incidence indexed by age at death, year of death and month of death.

This second example illustrates interpolation and extrapolation in the GLAM setting. The paper concludes with a short discussion.

2. Generalized linear array model in one and two dimensions

In this section we consider a GLAM in one and two dimensions. In one dimension an array is a vector and in two dimensions it is a matrix so in these cases we can develop the basic ideas of a GLAM with ordinary matrix multiplication alone and without the general array arithmetic

that is required in higher dimensions. In one dimension we define the function which sets up our sequential approach to computation and in two dimensions we use a well-known Kronecker product identity to motivate our sequential formulae. We discuss the one- and two-dimensional cases in turn.

2.1. Generalized linear array model in one dimension

We start with a GLAM in one dimension. At first sight, this may seem odd but a GLAM in one dimension is not the same as an ordinary GLM although they do share the common feature that in both cases a linear function is computed as $\mathbf{X}\boldsymbol{\theta}$, a single matrix transformation of the vector of coefficients. We mention two differences: first, in a GLM we have a diagonal weight matrix $\mathbf{W}_\delta$ which, in a GLAM, becomes a one-dimensional array or vector of weights $\mathbf{w}$; second, neither the inner product $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ nor the diagonal elements of $\mathbf{X}(\mathbf{X}'\mathbf{W}_\delta\mathbf{X})^{-1}\mathbf{X}'$ are computed in a sequential fashion but require successive operations involving $\mathbf{X}$. We now show how each of these expressions can be rewritten as a single matrix transformation of a vector, in the same fashion as the linear function.

We start with the inner products $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$. Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_c)$, $n \times c$, and $\mathbf{w}$ be the vector of weights, i.e. $\mathbf{w}$ is the vector consisting of the diagonal elements of $\mathbf{W}_\delta$. Then the (j,k)th element of $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ is given by

$$\begin{aligned}(\mathbf{X}'\mathbf{W}_\delta\mathbf{X})_{jk} &= \sum_{i=1}^{n} w_i x_{ij} x_{ik} \\ &= (x_{1j}x_{1k}, \ldots, x_{nj}x_{nk})\mathbf{w} = (\mathbf{x}_j * \mathbf{x}_k)'\mathbf{w} \end{aligned} \tag{2.1}$$

where the asterisk denotes element-by-element multiplication. Expression (2.1) suggests an alternative way of evaluating $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$. We need to define a $c^2 \times n$ matrix whose (j,k)th row consists of $(\mathbf{x}_j * \mathbf{x}_k)'$. This is not well defined but a convenient ordering is given in terms of the following definition.

Definition 1. The *row tensor* of a matrix $\mathbf{X}$ with c columns is defined as

$$G(\mathbf{X}) = (\mathbf{X} \otimes \mathbf{1}') * (\mathbf{1}' \otimes \mathbf{X}) \tag{2.2}$$

where $\mathbf{1}$ is a vector of 1s of length c.

The operation (2.2) is such that row i of $G(\mathbf{X})$ is the Kronecker product of row i of $\mathbf{X}$ with itself; the corresponding operation on the columns of $\mathbf{X}$ is known as the Khatri–Rao product (Brewer, 1978). For our purpose it is easy to see that the columns of $G(\mathbf{X})$ consist of the pairwise element-by-element products $\mathbf{x}_j * \mathbf{x}_k$ of the columns of $\mathbf{X}$, and so, from equation (2.1), $G(\mathbf{X})$ has the property that

$$\mathbf{X}'\mathbf{W}_\delta\mathbf{X},\ c \times c \equiv G(\mathbf{X})'\mathbf{w},\ c^2 \times 1, \tag{2.3}$$

as desired; here '$\equiv$' means that both sides have the same elements, although their dimensions, as indicated, are different. Moreover, since

$$\mathrm{vec}(\mathbf{X}'\mathbf{W}_\delta\mathbf{X}) = G(\mathbf{X})'\mathbf{w},$$

our target matrix $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ can be recovered simply by redimensioning $G(\mathbf{X})'\mathbf{w}$ (here $\mathrm{vec}(\mathbf{M})$ means stack the columns of a matrix $\mathbf{M}$ on top of each other to give a vector).

We note that the nature of the calculations $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ and $G(\mathbf{X})'\mathbf{w}$ is very different: first, $G(\mathbf{X})'\mathbf{w}$ computes the inner product as a single matrix transformation of a vector, as required; second, in the direct calculation, the products of the columns of $\mathbf{X}$ must be obtained at each iteration of algorithm (1.1), whereas in the array calculation the products can be stored before iteration. We

do not make any claim that a GLAM is a sensible way of fitting a GLM since, although computational times are similar, the method is space inefficient since $G(\mathbf{X})$, $n \times c^2$, must be stored. However, in two and more dimensions a GLAM is superior in both computational efficiency and storage requirements.

We close the discussion of the one-dimensional case by considering computation of the variances of the fitted values $\mathbf{X}\hat{\boldsymbol{\theta}}$. Let $\mathbf{S}_m = \text{var}(\hat{\boldsymbol{\theta}}) = (\mathbf{X}'\hat{\mathbf{W}}_\delta\mathbf{X})^{-1}$, $c \times c$. Then it is easy to show that

$$\begin{aligned}\text{diag}\{\text{var}(\mathbf{X}\hat{\boldsymbol{\theta}})\} &= \text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}') \\ &= G(\mathbf{X})\mathbf{s}\end{aligned} \tag{2.4}$$

where $\mathbf{s} = \text{vec}(\mathbf{S}_m)$, $c^2 \times 1$, and $G(\cdot)$ is the row tensor function in equation (2.2).

We emphasize that the three formulae $\mathbf{X}\boldsymbol{\theta}$, $G(\mathbf{X})'\mathbf{w}$ and $G(\mathbf{X})\mathbf{s}$ for the linear function, for the inner product and for the diagonal function respectively all have the same form, a single matrix transformation of a one-dimensional array (vector).

2.2. Generalized linear array model in two dimensions

Let $\mathbf{Y}$, $n_1 \times n_2$, be a data matrix and let $\mathbf{y} = \text{vec}(\mathbf{Y})$ be its vector equivalent. We consider a GLM for $\mathbf{y}$ with model matrix $\mathbf{X}$ and suppose that $\mathbf{X} = \mathbf{X}_2 \otimes \mathbf{X}_1$ where $\mathbf{X}_i$ is $n_i \times c_i$, $i = 1, 2$. Let $\boldsymbol{\theta}$, $c_1c_2 \times 1$, be the vector of regression coefficients. We use a well-known identity of the Kronecker product (Searle (1982), page 333) to write the linear predictor as

$$(\mathbf{X}_2 \otimes \mathbf{X}_1)\boldsymbol{\theta},\ n_1n_2 \times 1 \equiv \mathbf{X}_1\boldsymbol{\Theta}\mathbf{X}_2',\ n_1 \times n_2, \tag{2.5}$$

where $\boldsymbol{\Theta}$, $c_1 \times c_2$, is the coefficient matrix corresponding to $\boldsymbol{\theta}$, i.e. $\text{vec}(\boldsymbol{\Theta}) = \boldsymbol{\theta}$. Conceptually, expression (2.5) allows us to think of the linear predictor as $\mathbf{X}_1\boldsymbol{\Theta}\mathbf{X}_2'$, a matrix which corresponds to data $\mathbf{Y}$. Computationally, expression (2.5) has two important properties: first, the right-hand side avoids the computation of the potentially large matrix $\mathbf{X}$ and, second, the number of multiplications on the right-hand side is very much less than that on the left-hand side. For example if $\mathbf{X}_1$ and $\mathbf{X}_2$ are both $10^3 \times 10^2$ then the left-hand side requires 10^{10} multiplications whereas the right-hand side requires just over 10^8 multiplications. We can then recover the left-hand side by redimensioning, which is a very efficient operation. Expression (2.5) also enables efficient computation of $\mathbf{X}'\tilde{\mathbf{W}}_\delta\tilde{\mathbf{z}}$ in algorithm (1.1).

We continue with the computation of the inner products $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$. We define $\mathbf{W}$ to be the $n_1 \times n_2$ matrix of weights, i.e. $\text{vec}(\mathbf{W}) = \text{diag}(\mathbf{W}_\delta)$. The following formula is suggested by expressions (2.3) and (2.5):

$$(\mathbf{X}_2 \otimes \mathbf{X}_1)'\mathbf{W}_\delta(\mathbf{X}_2 \otimes \mathbf{X}_1),\ c_1c_2 \times c_1c_2 \equiv G(\mathbf{X}_1)'\mathbf{W}\,G(\mathbf{X}_2),\ c_1^2 \times c_2^2, \tag{2.6}$$

where $G(\cdot)$ is the row tensor function that is defined in equation (2.2). This achieves efficient computation of the entries in $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$. A proof is given in a general setting in Section 4 and details of the rearranging and redimensioning of the right-hand side that are required to give the left-hand side are given in Section 5. Some insight into expression (2.6) is provided by noting that the columns of $G(\mathbf{X}_2 \otimes \mathbf{X}_1)$ are the same as the columns of $G(\mathbf{X}_2) \otimes G(\mathbf{X}_1)$ but in a different order. Hence

$$\begin{aligned}(\mathbf{X}_2 \otimes \mathbf{X}_1)'\mathbf{W}_\delta(\mathbf{X}_2 \otimes \mathbf{X}_1) &\equiv G(\mathbf{X}_2 \otimes \mathbf{X}_1)'\mathbf{w} && \text{by expression (2.3)} \\ &\equiv (G(\mathbf{X}_2) \otimes G(\mathbf{X}_1))'\,\mathbf{w} \\ &\equiv G(\mathbf{X}_1)'\mathbf{W}\,G(\mathbf{X}_2) && \text{by expression (2.5)}\end{aligned}$$

and this is the right-hand side of expression (2.6).

Finally, variances are computed by using the two-dimensional version of equation (2.4). First, we note that in equation (2.4) the elements of $(\mathbf{X}'\mathbf{W}_\delta\mathbf{X})^{-1}$, a $c \times c$ matrix, are reorganized into a $c^2 \times 1$ array (or vector). In a similar fashion, in two dimensions we reorganize the elements of $\mathbf{S}_m = (\mathbf{X}'\mathbf{W}_\delta\mathbf{X})^{-1}$, a $c_1c_2 \times c_1c_2$ matrix, into $\mathbf{S}$, a $c_1^2 \times c_2^2$ matrix; details of how the reorganization is achieved are given in Section 5. Then

$$\mathrm{diag}\{\mathrm{var}(\mathbf{X}\hat{\boldsymbol{\theta}})\} = \mathrm{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}'),\ n_1n_2 \times 1 \equiv G(\mathbf{X}_1)\mathbf{S}\,G(\mathbf{X}_2)',\ n_1 \times n_2. \tag{2.7}$$

An attractive point is that this $n_1 \times n_2$ matrix has exactly the same structure as the data matrix $\mathbf{Y}$ and so all variances are in their correct place. We note that the linear function in expression (2.5), the inner product in expression (2.6) and the variances in expression (2.7) are each computed by two successive matrix transformations of a two-dimensional array, which is a natural extension of the remark on the one-dimensional computation at the end of the previous section. In the next section we extend these formulae to d-dimensions. First we illustrate the GLAM in two dimensions.

2.3. Example: smoothing mortality data

We consider data on the mortality of assured lives in the UK insurance market. The data set consists of the number of policy claims (deaths) and the number of years lived (exposure) for each age, 12–96 years, and each calendar year, 1951–1999; see Currie *et al.* (2004) for more information on these data. The claims and exposures are arranged respectively in matrices $\mathbf{Y}$ and $\mathbf{E}$, with rows indexed by age and columns by year. We model $\mathbf{Y}$ with a two-dimensional GLAM with Poisson errors and log-link

$$\begin{aligned} E(\mathbf{Y}) &= \mathbf{E} * \boldsymbol{\Gamma}, \\ \log(\mathbf{E} * \boldsymbol{\Gamma}) &= \log(\mathbf{E}) + \mathbf{B}_a\boldsymbol{\Theta}\mathbf{B}_y', \end{aligned} \tag{2.8}$$

where $\mathbf{B}_a = \mathbf{B}_a(\mathbf{x}_a)$ and $\mathbf{B}_y = \mathbf{B}_y(\mathbf{x}_y)$ are marginal model matrices of B-splines for age $\mathbf{x}_a = (12, \ldots, 96)'$ and year $\mathbf{x}_y = (1951, \ldots, 1999)'$; the term $\log(\mathbf{E})$ is a matrix offset. We have used cubic B-splines and one knot every 4 years in defining both $\mathbf{B}_a$ and $\mathbf{B}_y$ so $\mathbf{B}_a$ is 85×24 and $\mathbf{B}_y$ is 49×15. As a GLM, the regression matrix $\mathbf{B} = \mathbf{B}_y \otimes \mathbf{B}_a$ is 4165×360 and has nearly 1.5×10^6 entries. Fig. 1 gives an idea of what a Kronecker product basis of B-splines looks like. The full basis of 360 functions results in a rather crowded plot so only a very small subset of basis functions is shown. Smoothness of the fitted surface is ensured by penalization but note that smoothness at the edges of the data region is ensured by the penalty as opposed to modification of the B-spline basis, as in natural splines for example (Green and Silverman (1994), page 12). Each regression coefficient in $\boldsymbol{\Theta}$ is associated with the peak of a basis function so a smooth surface results if the coefficients are themselves smooth; this is the P-spline method of Eilers and Marx (1996). In two dimensions smoothness of the fitted surface is achieved by placing penalties on the columns and rows of $\boldsymbol{\Theta}$. Let $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{c_y})$ and $\mathbf{D}_a$ be a difference matrix acting on a column of $\boldsymbol{\Theta}$. Then an appropriate penalty on the columns of $\boldsymbol{\Theta}$ is

$$\sum_{j=1}^{c_y} \boldsymbol{\theta}_j'\mathbf{D}_a'\mathbf{D}_a\boldsymbol{\theta}_j = \boldsymbol{\theta}'(\mathbf{I}_{c_y} \otimes \mathbf{D}_a'\mathbf{D}_a)\boldsymbol{\theta} \tag{2.9}$$

where $\boldsymbol{\theta} = \mathrm{vec}(\boldsymbol{\Theta})$. A similar argument shows that the corresponding penalty on the rows of $\boldsymbol{\Theta}$ can be written $\boldsymbol{\theta}'(\mathbf{D}_y'\mathbf{D}_y \otimes \mathbf{I}_{c_a})\boldsymbol{\theta}$ and so the penalty matrix is given by

$$\mathbf{P} = \lambda_a\mathbf{I}_{c_y} \otimes \mathbf{D}_a'\mathbf{D}_a + \lambda_y\mathbf{D}_y'\mathbf{D}_y \otimes \mathbf{I}_{c_a} \tag{2.10}$$

where λ_a and λ_y are the smoothing parameters for columns and rows respectively.

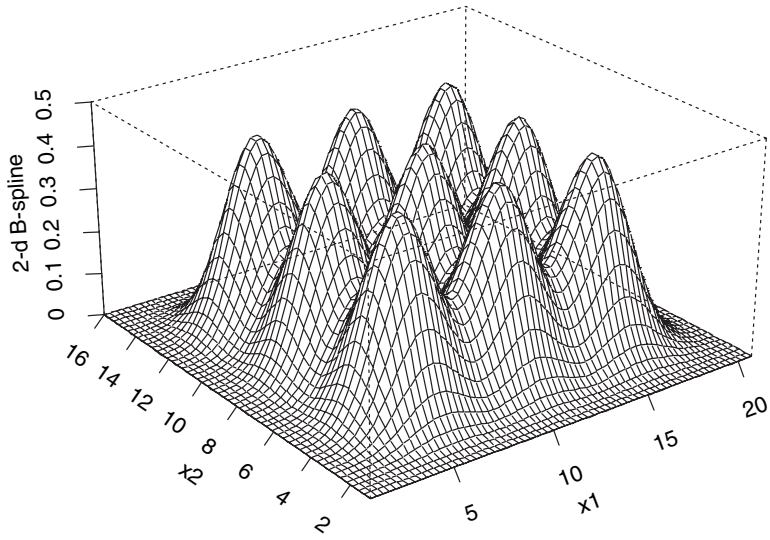

Fig. 1. Subset of a two-dimensional Kronecker product B-spline basis

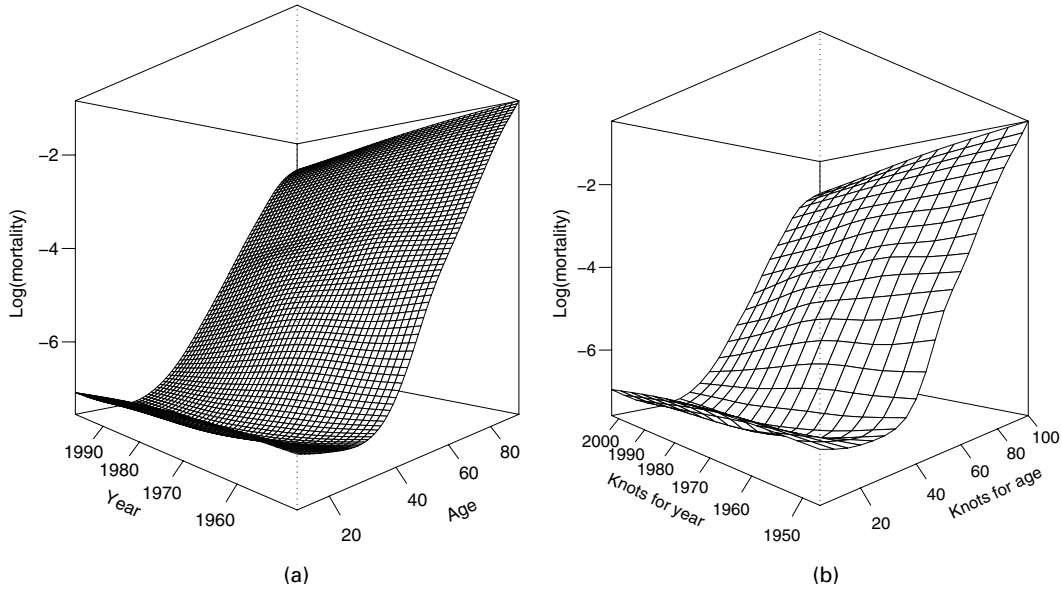

Fig. 2. (a) Fitted log-mortality surface and (b) fitted regression coefficients plotted at knot positions

There are several criteria for choosing the smoothing parameters: the Akaike information criterion (Akaike, 1973), the Bayesian information criterion (Schwarz, 1978) and generalized cross-validation (Craven and Wahba, 1979) are three possibilities. The present paper describes efficient model fitting for a certain class of models and so can be applied whichever criterion is adopted. Fig. 2(a) shows the fitted surface that results when the Bayes information criterion is used; Fig. 2(b) shows the smooth surface of regression coefficients, the result of the penalization. The sharp decline in mortality over the last 50 years that is shown in Fig. 3 has major consequences for insurance companies and policy-holders alike. Using a Bayesian approach, Wahba

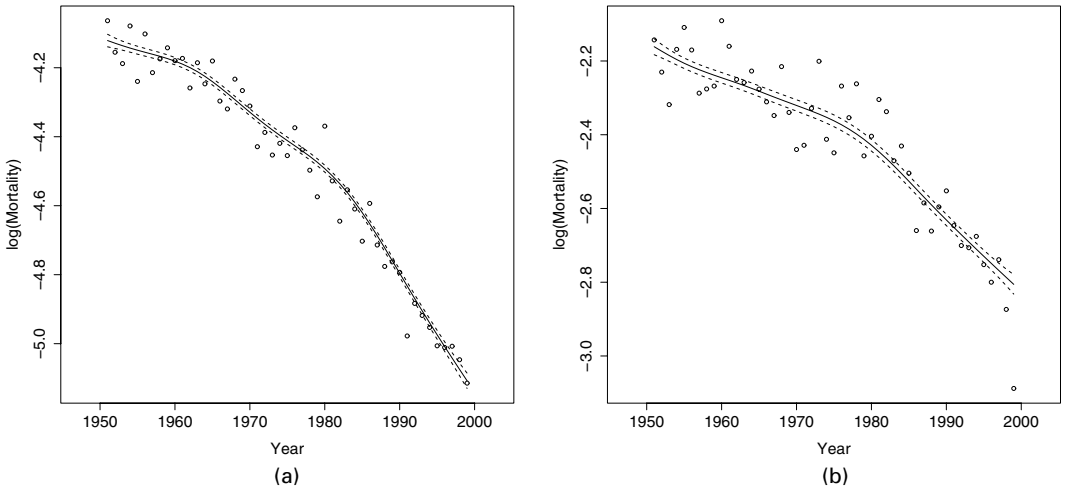

Fig. 3. Fitted log-mortality with 95% probability intervals: (a) age = 60 years; (b) age = 80 years

(1983) and Silverman (1985) estimated a posterior covariance for the vector of parameters, $\boldsymbol{\theta}$, by

$$\mathbf{S}_m = (\mathbf{B}\hat{\mathbf{W}}_\delta\mathbf{B} + \mathbf{P})^{-1}$$

so approximate probability intervals for the curves in Fig. 3 were computed from expression (2.7) and are also shown.

3. Generalized linear array model in *d*-dimensions

We consider a GLAM in d dimensions with data $\mathbf{Y}$, $n_1 \times \ldots \times n_d$, and model matrix

$$\mathbf{X} = \mathbf{X}_d \otimes \ldots \otimes \mathbf{X}_1 \tag{3.1}$$

where the marginal model matrix $\mathbf{X}_i$ is $n_i \times c_i$, $i = 1, \ldots, d$. The dimensions of the $\mathbf{X}_i$ induce an array structure on the vector of regression coefficients, $\boldsymbol{\theta}$, which becomes the d-dimensional array $\boldsymbol{\Theta}$, $c_1 \times \ldots \times c_d$. Suppose that the link function is $g(\cdot)$. We need to generalize the one-dimensional expression, $g(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\theta}$, $E(\mathbf{y}) = \boldsymbol{\mu}$, and the two-dimensional expression, $g(\mathbf{M}) = \mathbf{X}_1\boldsymbol{\Theta}\mathbf{X}_2'$, $E(\mathbf{Y}) = \mathbf{M}$. The idea is to transform $\boldsymbol{\Theta}$ successively by the marginal model matrices $\mathbf{X}_i$, $i = 1, \ldots, d$, and for this we need to define premultiplication of d-dimensional arrays, such as $\boldsymbol{\Theta}$, by a matrix.

Definition 2. The *H-transform* of the d-dimensional array $\mathbf{A}$ of size $c_1 \times c_2 \times \ldots \times c_d$ by the matrix $\mathbf{X}$ of size $r \times c_1$ is denoted $H(\mathbf{X}, \mathbf{A})$ and defined as follows: let $\mathbf{A}^*$ be the $c_1 \times c_2 c_3 \ldots c_d$ matrix that is obtained by flattening dimensions 2–d of $\mathbf{A}$; form the matrix product $\mathbf{X}\mathbf{A}^*$ of size $r \times c_2 c_3 \ldots c_d$; then $H(\mathbf{X}, \mathbf{A})$ is the d-dimensional array of size $r \times c_2 \times \ldots \times c_d$ that is obtained from $\mathbf{X}\mathbf{A}^*$ by reinstating dimensions 2–d of $\mathbf{A}$.

In one dimension $\mathbf{A} = \mathbf{a}$, so $H(\mathbf{X}, \mathbf{a}) = \mathbf{X}\mathbf{a}$, whereas in two dimensions $H(\mathbf{X}, \mathbf{A}) = \mathbf{X}\mathbf{A}$. Thus, the H-transform generalizes premultiplication of vectors and matrices by a matrix. Expression (2.5) also suggests that we need to generalize the transpose of a matrix.

Definition 3. The *rotation* of the d-dimensional array $\mathbf{A}$ of size $c_1 \times c_2 \times \ldots \times c_d$ is the d-dimensional array $R(\mathbf{A})$ of size $c_2 \times c_3 \times \ldots \times c_d \times c_1$ that is obtained by permuting the indices of $\mathbf{A}$.

Table 1. Comparison of computation in GLMs and one- and two-dimensional GLAMs

Function	*GLM*	*1-dimensional GLAM*	*2-dimensional GLAM*
Linear	$\mathbf{X}\boldsymbol{\theta}$	$\mathbf{X}_1\boldsymbol{\theta}$	$(\mathbf{X}_2(\mathbf{X}_1\boldsymbol{\Theta})')'$
Inner product	$\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$	$G(\mathbf{X}_1)'\mathbf{w}$	$(G(\mathbf{X}_2)'(G(\mathbf{X}_1)'\mathbf{W})')'$
Diagonal	$\text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$	$G(\mathbf{X}_1)\mathbf{s}$	$(G(\mathbf{X}_2)(G(\mathbf{X}_1)\mathbf{S})')'$

It is convenient to combine these two definitions in the following definition.

Definition 4. The *rotated H-transform* of the array $\mathbf{A}$ by the matrix $\mathbf{X}$ is given by

$$\rho(\mathbf{X}, \mathbf{A}) = R\{H(\mathbf{X}, \mathbf{A})\}.$$

We can now write down the GLAM for $\mathbf{Y}$ in d dimensions. If $E(\mathbf{Y}) = \mathbf{M}$ then

$$g(\mathbf{M}) = \rho[\mathbf{X}_d, \ldots, \rho\{\mathbf{X}_2, \rho(\mathbf{X}_1, \boldsymbol{\Theta})\}\ldots], \tag{3.2}$$

where $g(\cdot)$ is the link function. In addition, we require an array computation of

(a) the inner product $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ so that the scoring algorithm may be evaluated and
(b) the diagonal of $\mathbf{X}\mathbf{S}_m\mathbf{X}'$ so that variances of fitted values may be found.

Table 1 summarizes the results from Sections 2.1 and 2.2. In two dimensions we operate successively on a two-dimensional array either with the marginal regression matrices $\mathbf{X}_i$ or with the row tensor $G(\mathbf{X}_i)$; after each transformation the dimensions are rotated (transposed) to bring the next dimension into line. This suggests the following general formulae.

(a) *Linear functions*: the elements of $\mathbf{X}\boldsymbol{\theta}$ (and similarly for $\mathbf{X}'\mathbf{W}_\delta\mathbf{z}$) are given by the d-dimensional array

$$\rho[\mathbf{X}_d, \ldots, \rho\{\mathbf{X}_2, \rho(\mathbf{X}_1, \boldsymbol{\Theta})\}\ldots]. \tag{3.3}$$

(b) *Inner products*: the elements of the inner product $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ are given by the d-dimensional array

$$\rho(G(\mathbf{X}_d)', \ldots, \rho[G(\mathbf{X}_2)', \rho\{G(\mathbf{X}_1)', \mathbf{W}\}]\ldots). \tag{3.4}$$

This $c_1^2 \times \ldots \times c_d^2$ array must now be reorganized into $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$, a square matrix of size $c_1c_2\ldots c_d$; efficient software to achieve this reorganization is given in Section 5.

We give the formula for the diagonal elements of $\text{var}(\mathbf{X}\hat{\boldsymbol{\theta}})$ in its general form. Let $\mathbf{S}_m$ be a square matrix of size $c_1c_2\ldots c_d$ and let $\mathbf{S}$, $c_1^2 \times \ldots \times c_d^2$, be the d-dimensional array of reorganized elements of $\mathbf{S}_m$; see Section 5 for details of the reorganization. Then we have the following.

(c) *Diagonal function*: the diagonal elements of $\mathbf{X}\mathbf{S}_m\mathbf{X}'$ are given by the d-dimensional array

$$\rho(G(\mathbf{X}_d), \ldots, \rho[G(\mathbf{X}_2), \rho\{G(\mathbf{X}_1), \mathbf{S}\}]\ldots). \tag{3.5}$$

It is worth emphasizing that expression (3.5) holds for any square matrix $\mathbf{S}_m$, not necessarily symmetric. The diagonal elements of $\text{var}(\mathbf{X}\hat{\boldsymbol{\theta}})$ are obtained by setting $\mathbf{S}_m$ equal to $(\mathbf{X}'\hat{\mathbf{W}}_\delta\mathbf{X})^{-1}$.

The array arithmetic that is defined in expression (3.3) unifies the calculation of linear functions, weighted inner products and diagonal values. An alternative way of describing expressions (3.4) and (3.5) is to say that both $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ and $\text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$ can be written as a Kronecker

Table 2. Number of multiplications that are required to compute $\mathbf{X}\boldsymbol{\theta}$, $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ and $\text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$ with $\mathbf{X}_i$, $n \times c$, $c < n < c^2$, $\phi_1 = n/c$, $\phi_2 = c^2/n$ and $\phi_i^* = 1 - (1/\phi_i)^d \approx 1$

Function	*GLM*	*GLAM*	*Ratio: $n = 50$, $c = 10$, $d = 3$*
$\mathbf{X}\boldsymbol{\theta}$	$n^d c^d$	$n^{d+1}\phi_1^*/(\phi_1 - 1) \approx n^{d+1}/(\phi_1 - 1)$	$81 \approx 80$
$\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$	$n^d c^{2d}$	$c^{2d+2}\phi_2^*/(\phi_2 - 1) \approx c^{2d+2}/(\phi_2 - 1)$	$1429 \approx 1250$
$\text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$	$n^d c^{2d}$	$c^{2d+2}\phi_2^*/(\phi_2 - 1) \approx c^{2d+2}/(\phi_2 - 1)$	$1429 \approx 1250$

product times a vector, and hence expression (3.3) may be applied to both these functions. We make the following definitions.

Definition 5. Let $\mathbf{X}_i$, $n_i \times c_i$, $i = 1, \ldots, d$, be matrices and $\mathbf{X} = \mathbf{X}_d \otimes \ldots \otimes \mathbf{X}_1$; let $\mathbf{A}$, $c_1 \times \ldots \times c_d$, be an array and $\mathbf{a} = \text{vec}(\mathbf{A})$. Then we say

(a) the vector $\mathbf{Xa}$ is in Kronecker product form or *K-form*,
(b) the array $\rho[\mathbf{X}_d, \ldots, \rho\{\mathbf{X}_2, \rho(\mathbf{X}_1, \mathbf{A})\}\ldots]$ is in array form or *A-form*.

With this terminology we can say that $\mathbf{X}\boldsymbol{\theta}$, $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ and $\text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$ have both K-form and A-form representations. The interface between $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ and its A-form, and between $\text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$ and its A-form, is described in Section 5. However, the outputs from the A-form calculations of expressions (3.3) and (3.5) are d-dimensional arrays whose format matches that of the data array $\mathbf{Y}$, and hence give fitted values, confidence intervals and graphical summaries directly.

Our method uses a nested approach with each dimension processed in turn and this explains the speed of our algorithms. Table 2 compares the number of multiplications that are required in a d-dimensional GLAM by expressions (3.3), (3.4) and (3.5) with the corresponding matrix evaluation. In constructing Table 2 we have

(a) assumed that each marginal matrix $\mathbf{X}_i$ is $n \times c$, $c < n < c^2$,
(b) ignored the multiplication by $\mathbf{W}_\delta$ in the matrix evaluation of the inner product and
(c) computed $\text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$ by equation (2.4), the formulae for a one-dimensional GLAM.

Table 2 also uses the ratio of the number of multiplications to compare GLM and GLAM computation; the efficiency of the GLAM is evident. As an aside, we note that the number of multiplications for each of $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ and $\text{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$ is exactly the same.

Our method has an overhead of rearrangement and redimensioning, but these are both very efficient operations. In Section 7 we smooth count data that are arranged in a $105 \times 40 \times 12$ array. The most demanding component of algorithm (1.2) is the repeated calculation of $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ which requires the multiplication of two large matrices. Table 3 gives some comparative figures

Table 3. User central processor unit times to calculate $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$

Coefficient array	*Number of coefficients*	*Times (s) for the GLM*	*Times (s) for the GLAM*	*Ratio*
$6 \times 6 \times 6$	216	8.5	0.0165	500:1
$7 \times 7 \times 7$	343	20.9	0.0305	700:1
$8 \times 8 \times 8$	512	46.5	0.0717	650:1
$9 \times 9 \times 9$	729	91.7	0.1403	650:1

on timing for various sizes of the coefficient array for this data set. The GLAM is so quick that it was necessary to use a loop to obtain accurate timings, a device which paradoxically will have degraded its performance. The results reinforce the conclusions of Table 2.

We can now make some connections between our work and earlier work. Yates's (1937) well-known algorithm computes the treatment effects in a 2^k factorial experiment; this algorithm extends to more general factorial experiments, as described in Gower (1982). In a d-dimensional factorial experiment the design matrix $\mathbf{X}$ can be written as the Kronecker product of d orthogonal matrices. It is easy to check that $\mathbf{X}'\mathbf{X} = \mathbf{I}_c$ where c is the number of parameters, so there is no need to evaluate the inner products: the treatment effects are estimated directly from $\mathbf{X}'\mathbf{y}$, which may be calculated by expression (3.3). The intermediate output arrays from expression (3.3) correspond to the intermediate output vectors from Yates's algorithm. Furthermore,

$$\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' = \mathbf{I}$$

so variances of fitted values are immediately available. Thus, the special structure of the factorial experiment leads to a very simple analysis where only expression (3.3) is required.

In an important paper, de Boor (1979) gave an algorithm for the efficient solution of the linear equation $\mathbf{X}\boldsymbol{\theta} = \mathbf{y}$ where $\mathbf{X}$ is the Kronecker product of d invertible matrices. Since $\mathbf{X}^{-1} = \mathbf{X}_d^{-1} \otimes \ldots \otimes \mathbf{X}_1^{-1}$ we obtain from expression (3.3)

$$\hat{\boldsymbol{\Theta}} = \rho[\mathbf{X}_d^{-1}, \ldots, \rho\{\mathbf{X}_2^{-1}, \rho(\mathbf{X}_1^{-1}, \mathbf{Y})\}\ldots], \tag{3.6}$$

which is equivalent to de Boor's result. de Boor (1979) also discussed the evaluation of $\mathbf{Xy}$ for general marginal matrices, $\mathbf{X}_i$; expression (3.3) is equivalent to his result. From this perspective, our methods show how efficient computation can be extended to matrix functions other than $\mathbf{Xy}$. Of course, since both $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ and $\mathrm{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$ can be written in K-form any K-form computational scheme other than expression (3.3) could equally be applied to these two functions.

In the case of general regression with independent identically distributed normal errors we solve $\mathbf{X}'\mathbf{X}\boldsymbol{\theta} = \mathbf{X}'\mathbf{y}$ where $\mathbf{X}$ is the Kronecker product of marginal regression matrices $\mathbf{X}_i$, $n_i \times c_i$. Without the presence of the weight matrix $\mathbf{W}_\delta$, we have

$$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' = (\mathbf{X}_d'\mathbf{X}_d)^{-1}\mathbf{X}_d' \otimes \ldots \otimes (\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1' \tag{3.7}$$

so the normal equations can be solved efficiently with expression (3.3). Moreover, since

$$\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' = \mathbf{X}_d(\mathbf{X}_d'\mathbf{X}_d)^{-1}\mathbf{X}_d' \otimes \ldots \otimes \mathbf{X}_1(\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1' \tag{3.8}$$

the standard errors are computed directly as the Kronecker product of the diagonal elements of the $\mathbf{X}_i(\mathbf{X}_i'\mathbf{X}_i)^{-1}\mathbf{X}_i'$ (each of which may be computed with equation (2.4) if appropriate).

4. Generalized linear array model: algebraic details

We have developed an array arithmetic for the efficient evaluation of linear functions, inner products and variances; the formulae are given in expressions (3.3), (3.4) and (3.5). The following formal derivations of our results lead to the efficient software that is presented in the next section; the proofs also demonstrate the nested nature of our algorithms and thus explain the gains in speed that are achieved. The main idea is the use of array indexing to index vectors and matrices; thus in a d-dimensional problem a vector becomes a d-dimensional array and a variance matrix becomes a $2d$-dimensional array.

We prove first expression (3.3), the formula for the linear function $\mathbf{X}\boldsymbol{\theta}$. The elements of the coefficient array $\boldsymbol{\Theta}$ are $\theta_{(i_1,i_2,\ldots,i_d)}$ and we use this d-tuple index to index the elements of the coefficient vector $\boldsymbol{\theta}$. Similarly, we let $x_{(k_1,k_2,\ldots,k_d),(i_1,i_2,\ldots,i_d)}$ denote the $((k_1, k_2, \ldots, k_d), (i_1, i_2, \ldots, i_d))$

entry in $\mathbf{X}$, i.e.

$$x_{(k_1,k_2,\ldots,k_d),(i_1,i_2,\ldots,i_d)} = \xi(1)_{k_1,i_1}\,\xi(2)_{k_2,i_2}\ldots\xi(d)_{k_d,i_d} \tag{4.1}$$

where $\xi(t)_{k,i}$ is the (k,i) entry in $\mathbf{X}_t$. Hence, the $(k_1,k_2,\ldots,k_d)$ entry in $\mathbf{X}\boldsymbol{\theta}$ is

$$\sum_{i_d=1}^{c_d}\cdots\sum_{i_1=1}^{c_1} x_{(k_1,k_2,\ldots,k_d),(i_1,i_2,\ldots,i_d)}\theta_{(i_1,i_2,\ldots,i_d)} \tag{4.2}$$

$$=\sum_{i_d=1}^{c_d}\cdots\sum_{i_1=1}^{c_1}\xi(1)_{k_1,i_1}\,\xi(2)_{k_2,i_2}\ldots\xi(d)_{k_d,i_d}\theta_{(i_1,i_2,\ldots,i_d)}. \tag{4.3}$$

Now consider the array evaluation of $\mathbf{X}\boldsymbol{\theta}$ with expression (3.3). The $(k_1,\ldots,k_d)$ entry of $\rho[\mathbf{X}_d,\ldots,\rho\{\mathbf{X}_2,\rho(\mathbf{X}_1,\boldsymbol{\Theta})\}\ldots]$ is

$$\sum_{i_d=1}^{c_d}\xi(d)_{k_d,i_d}\left(\ldots\left[\sum_{i_2=1}^{c_2}\xi(2)_{k_2,i_2}\left\{\sum_{i_1=1}^{c_1}\xi(1)_{k_1,i_1}\theta_{(i_1,i_2,\ldots,i_d)}\right\}\right]\ldots\right) \tag{4.4}$$

which is expression (4.3). Expression (4.4) shows $\mathbf{X}\boldsymbol{\theta}$ 'as a nested set of sums of products' (Gower, 1982).

We next show that $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ can be evaluated with expression (3.4). The proof is similar to that given for expression (3.3). The elements of the array of weights $\mathbf{W}$ are $w_{(k_1,k_2,\ldots,k_d)}$ and we use this d-tuple index to describe the elements of the diagonal matrix $\mathbf{W}_\delta$. Hence the $((i_1,i_2,\ldots,i_d),(j_1,j_2,\ldots,j_d))$ entry in $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ is

$$\sum_{k_d=1}^{n_d}\cdots\sum_{k_1=1}^{n_1} x_{(k_1,k_2,\ldots,k_d),(i_1,i_2,\ldots,i_d)}w_{(k_1,k_2,\ldots,k_d)}x_{(k_1,k_2,\ldots,k_d),(j_1,j_2,\ldots,j_d)} \tag{4.5}$$

$$=\sum_{k_d=1}^{n_d}\cdots\sum_{k_1=1}^{n_1}\xi(1)_{k_1,i_1}\,\xi(2)_{k_2,i_2}\ldots\xi(d)_{k_d,i_d}w_{(k_1,k_2,\ldots,k_d)}\,\xi(1)_{k_1,j_1}\,\xi(2)_{k_2,j_2}\ldots\xi(d)_{k_d,j_d}. \tag{4.6}$$

Now consider the array evaluation of $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ with expression (3.4). Let $\mathbf{T}_i = G(\mathbf{X}_i)$ be the row tensor of $\mathbf{X}_i$ as in equation (2.2). Then $\mathbf{T}_i'$ is a $c_i^2\times n_i$ matrix whose $((j,k),l)$ entry is $\xi(i)_{lj}\,\xi(i)_{lk}$. Then $\rho[\mathbf{T}_d',\ldots,\rho\{\mathbf{T}_2',\rho(\mathbf{T}_1',\mathbf{W})\}\ldots]$ is a d-dimensional array of size $c_1^2\times c_2^2\times\ldots\times c_d^2$ whose $((i_1,j_1),(i_2,j_2),\ldots,(i_d,j_d))$ entry is

$$\sum_{k_d=1}^{n_d}\xi(d)_{k_d,i_d}\,\xi(d)_{k_d,j_d}\left(\ldots\left[\sum_{k_2=1}^{n_2}\xi(2)_{k_2,i_2}\,\xi(2)_{k_2,j_2}\left\{\sum_{k_1=1}^{n_1}\xi(1)_{k_1,i_1}\,\xi(1)_{k_1,j_1}w_{(k_1,k_2,\ldots,k_d)}\right\}\right]\ldots\right) \tag{4.7}$$

which is expression (4.6). Again, the nested nature of expression (4.7) is evident.

Finally we show that the diagonal elements of $\mathbf{X}\mathbf{S}_m\mathbf{X}'$ can be evaluated by expression (3.5). Let $\mathbf{S}_m$ be a $c_1c_2\ldots c_d\times c_1c_2\ldots c_d$ matrix with entries $s_{(i_1,i_2,\ldots,i_d),(j_1,j_2,\ldots,j_d)}$; as an aside, we note that there is no requirement that $\mathbf{S}_m$ be symmetric. The diagonal elements of $\mathbf{X}\mathbf{S}_m\mathbf{X}'$ are a vector of length $n_1n_2\ldots n_d$ whose entries are indexed by the d-tuple index $(k_1,k_2,\ldots,k_d)$. Hence the $(k_1,k_2,\ldots,k_d)$ entry is given by

$$\sum_{j_d=1}^{c_d}\sum_{i_d=1}^{c_d}\cdots\sum_{j_1=1}^{c_1}\sum_{i_1=1}^{c_1} x_{(k_1,k_2,\ldots,k_d),(i_1,i_2,\ldots,i_d)}s_{(i_1,i_2,\ldots,i_d),(j_1,j_2,\ldots,j_d)}x_{(k_1,k_2,\ldots,k_d),(j_1,j_2,\ldots,j_d)} \tag{4.8}$$

$$\begin{aligned}=\sum_{j_d=1}^{c_d}\sum_{i_d=1}^{c_d}\cdots\sum_{j_1=1}^{c_1}\sum_{i_1=1}^{c_1}&\xi(1)_{k_1,i_1}\,\xi(2)_{k_2,i_2}\ldots\xi(d)_{k_d,i_d}s_{(i_1,i_2,\ldots,i_d),(j_1,j_2,\ldots,j_d)}\\&\times\xi(1)_{k_1,j_1}\,\xi(2)_{k_2,j_2}\ldots\xi(d)_{k_d,j_d}.\end{aligned} \tag{4.9}$$

Now consider the array evaluation of the diagonal of $\mathbf{X}\mathbf{S}_m\mathbf{X}'$ with expression (3.5). Let $\mathbf{S}$ be the $c_1^2 \times c_2^2 \times \ldots \times c_d^2$ array with entries $s_{(i_1,j_1),(i_2,j_2),\ldots,(i_d,j_d)}$ corresponding to the matrix $\mathbf{S}_m$, i.e. the $((i_1, j_1), (i_2, j_2), \ldots, (i_d, j_d))$ entry in $\mathbf{S}$ is the $((i_1, i_2, \ldots, i_d), (j_1, j_2, \ldots, j_d))$ entry in $\mathbf{S}_m$. Then $\rho[\mathbf{T}_d, \ldots, \rho\{\mathbf{T}_2, \rho(\mathbf{T}_1, \mathbf{S})\}\ldots]$ is a d-dimensional array of size $n_1 \times n_2 \times \ldots \times n_d$ whose $(k_1, k_2, \ldots, k_d)$ entry is

$$\sum_{j_d=1}^{c_d}\sum_{i_d=1}^{c_d} \xi(d)_{k_d,j_d}\,\xi(d)_{k_d,i_d}\left[\cdots\left\{\sum_{j_1=1}^{c_1}\sum_{i_1=1}^{c_1} \xi(1)_{k_1,j_1}\,\xi(1)_{k_1,i_1}\,s_{(i_1,j_1),(i_2,j_2),\ldots,(i_d,j_d)}\right\}\cdots\right] \tag{4.10}$$

which is expression (4.9) and, once more, the nested form of expression (4.10) is clear.

These proofs establish the element-by-element equality between the three matrix functions $\mathbf{X}\boldsymbol{\theta}$, $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ and $\mathrm{diag}(\mathbf{X}\mathbf{S}_m\mathbf{X}')$ and their corresponding A-forms but it is important to realize that the gains in speed that are achieved by the A-form calculation come from the array form itself, i.e. the nesting in expressions (3.3)–(3.5) is at the array level, and not just at the element-by-element level.

5. Software considerations

We fit a GLAM by using expressions (3.3) and (3.4) to evaluate algorithm (1.1), the scoring algorithm. However, the outputs of these algorithms are d-dimensional arrays, so these must be redimensioned and rearranged to give the vector and matrix forms that are required in algorithm (1.1). The redimensioning and rearranging that are required are provided by the details of the proofs of the previous section but we feel that a much clearer picture is provided by presenting some illustrative software. We work in S-PLUS or R and MATLAB, but the code should be accessible to someone who is unfamiliar with these languages. We use the case $d = 3$ to illustrate the general method. Eilers *et al.* (2006) gives MATLAB code; the code below is in S-PLUS or R.

For completeness, we start with code for the row tensor function and the rotated H-transform. Below, an asterisk means element-by-element multiplication, '%*%' means matrix multiplication, `t()` means transpose and `aperm()` permutes the dimensions of an array.

```
# Row tensor of a matrix X
Rten = function(X){
    one = matrix(1, 1, ncol(X))
    kronecker(X, one) * kronecker(one, X)
}
# H-transform of an array A by a matrix X
H = function(X, A){
    d = dim(A)
    M = matrix(A, nrow = d[1])
    XM = X %*% M
    array(XM, c(nrow(XM), d[-1]))
}
# Rotation of an array A
Rotate = function(A){
    d = 1:length(dim(A))
    d1 = c(d[-1], d[1])
    aperm(A, d1)
}
# Rotated H-transform of an array A by a matrix X
RH = function(X, A) Rotate(H(X, A))
```

The output of expression (3.3) is an array (which is denoted here by `A`) and this is converted to a vector (denoted here by `a`) (and vice versa) simply by redimensioning, as in

```
a = as.vector(A)
A = array(a, c(c1, c2, c3))
```

where `c1`, `c2` and `c3` are the number of columns in the marginal regression matrices $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$ respectively. The inner product $\mathbf{X}'\mathbf{W}_\delta\mathbf{X}$ is found from expression (3.4). If `RT1`, `RT2` and `RT3` are the row tensors of $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$ then

```
XWX = RH(t(RT3), RH(t(RT2), RH(t(RT1), W)))
dim(XWX) = c(c1, c1, c2, c2, c3, c3)
XWX = matrix(aperm(XWX, c(1, 3, 5, 2, 4, 6)), nrow = c1 * c2 * c3)
```

Finally we calculate the diagonal of $\mathbf{X}\mathbf{S}_m\mathbf{X}'$ with expression (3.5) where $\mathbf{S}_m$ is any square matrix of size $c_1c_2c_3$.

```
S = array(S.m, c(c1, c2, c3, c1, c2, c3))
S = aperm(S, c(1, 4, 2, 5, 3, 6))
S = array(S, c(c1^2, c2^2, c3^2))
Diag = RH(RT3, RH(RT2, RH(RT1, S)))
```

an $n_1 \times n_2 \times n_3$ array which matches the data array $\mathbf{Y}$.

6. A mixed model representation

The link between mixed model methodology and smoothing is well known, at least in one dimension. Green (1985) referred to trend as 'incorporating fixed and random effects'. Verbyla *et al.* (1999) showed that the cubic smoothing spline could be expressed as the sum of a fixed linear effect and a random effect; Eilers (1999) gave a mixed model representation of *P*-splines with a *B*-spline basis in the discussion of Verbyla *et al.* (1999). Wand (2003) and Ruppert *et al.* (2003) gave thorough reviews of the mixed model approach and provided extensive bibliographies.

We begin with the mixed model with normal errors and make the extension to the generalized linear mixed model at the end of this section. In one dimension our model is

$$\mathbf{y} = \mathbf{B}\boldsymbol{\theta} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \tag{6.1}$$

where $\mathbf{B}$ is a regression matrix of *B*-splines; smoothness is obtained via a penalty matrix $\mathbf{P} = \lambda\mathbf{D}'\mathbf{D}$ where $\mathbf{D}$ is a difference matrix acting on the coefficients $\boldsymbol{\theta}$. The aim is to look for a new basis which allows the representation of model (6.1) with its associated penalty as a mixed model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \mathbf{G}),\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}), \tag{6.2}$$

where $\mathbf{G}$ is a diagonal matrix which depends on λ. Note the change of notation: here $\mathbf{X}\boldsymbol{\beta}$ represents the fixed component; the full model matrix is $\mathbf{B}$.

We seek a mixed model representation (6.2) in higher dimensions which will be computable using the low storage, high speed algorithms that were described earlier. Our method is perfectly general but for simplicity we consider two dimensions with cubic *B*-splines and a second-order penalty. We have $\mathbf{B} = \mathbf{B}_2 \otimes \mathbf{B}_1$ with penalty matrix $\mathbf{P}$, as in equation (2.10), given by

$$\mathbf{P} = \lambda_1 \underbrace{\mathbf{I}_{c_2} \otimes \mathbf{D}_1'\mathbf{D}_1}_{\mathbf{P}_1} + \lambda_2 \underbrace{\mathbf{D}_2'\mathbf{D}_2 \otimes \mathbf{I}_{c_1}}_{\mathbf{P}_2}. \tag{6.3}$$

The expression $\mathbf{P}_1 + \mathbf{P}_2$ (and similarly for $\mathbf{P}$ in higher dimensions, as in equation (7.4) later) is known as a Kronecker sum. The singular value decomposition of such sums allows the simultaneous diagonalization of $\mathbf{P}_1$ and $\mathbf{P}_2$ (Horn and Johnson, 1991) and enables us to obtain a mixed model which can be fitted by using the array methods that were laid out earlier. The representation results in a diagonal variance $\mathbf{G}$. This has two advantages: first, computation is efficient and, second, a diagonal penalty is interpretable in terms of shrinkage of the regression coefficients.

We define the quantities $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{G}$ in model (6.2). The idea is to use the singular value decomposition of $\mathbf{P}$ to partition the difference penalty into a null penalty (the fixed part) and a diagonal penalty (the random part). We first take the singular value decomposition of $\mathbf{D}_1'\mathbf{D}_1$ into $\mathbf{V}_1\boldsymbol{\Sigma}_1\mathbf{V}_1'$. Now $\mathbf{D}_1'\mathbf{D}_1$ has rank $c_1 - 2$ ($\mathbf{D}_1$ is $(c_1 - 2) \times c_1$) and so $\boldsymbol{\Sigma}_1 = \text{diag}(\tau_{11}, \ldots, \tau_{1c_1-2}, 0, 0)$, say, and the non-zero eigenvalues are arranged in descending order; similarly, we write

$$\mathbf{D}_2'\mathbf{D}_2 = \mathbf{V}_2\boldsymbol{\Sigma}_2\mathbf{V}_2'.$$

Let $\tilde{\mathbf{V}}_1$, $c_1 \times 2$, and $\tilde{\mathbf{V}}_2$, $c_2 \times 2$, be the matrices corresponding to the zero eigenvalues of $\mathbf{D}_1'\mathbf{D}_1$ and $\mathbf{D}_2'\mathbf{D}_2$ respectively. Then we may take the fixed part to be

$$\begin{aligned} \mathbf{X} &= \mathbf{B}(\tilde{\mathbf{V}}_2 \otimes \tilde{\mathbf{V}}_1) \\ &= \mathbf{B}_2\tilde{\mathbf{V}}_2 \otimes \mathbf{B}_1\tilde{\mathbf{V}}_1. \end{aligned} \tag{6.4}$$

We now define the random part. Let

$$\boldsymbol{\Sigma} = (\mathbf{I}_{c_2} \otimes \boldsymbol{\Sigma}_1) + (\boldsymbol{\Sigma}_2 \otimes \mathbf{I}_{c_1}),$$

a diagonal matrix with entries $\tau_{1i} + \tau_{2j}$, $i = 1, \ldots, c_1$, $j = 1, \ldots, c_2$. We remove the four 0 elements on the diagonal of $\boldsymbol{\Sigma}$ corresponding to $\mathbf{X}$ to leave $\tilde{\boldsymbol{\Sigma}}$; this can be achieved by writing $\tilde{\boldsymbol{\Sigma}} = \mathbf{U}'\boldsymbol{\Sigma}\mathbf{U}$ where $\mathbf{U}'\mathbf{U}$ is an identity matrix of dimension $c_1c_2 - 4$. With these definitions in place we define

$$\begin{aligned} \mathbf{Z} &= \mathbf{B}(\mathbf{V}_2 \otimes \mathbf{V}_1)\mathbf{U}\tilde{\boldsymbol{\Sigma}}^{-1/2} \\ &= (\mathbf{B}_2\mathbf{V}_2 \otimes \mathbf{B}_1\mathbf{V}_1)\mathbf{U}\tilde{\boldsymbol{\Sigma}}^{-1/2}. \end{aligned} \tag{6.5}$$

We note that both $\mathbf{X}$ and $\mathbf{Z}$ have a Kronecker product structure; it is this structure which allows the array algorithms to be used. Now let

$$\begin{aligned} \mathbf{T} &= (\tilde{\mathbf{V}}_2 \otimes \tilde{\mathbf{V}}_1 : (\mathbf{V}_2 \otimes \mathbf{V}_1)\mathbf{U}\tilde{\boldsymbol{\Sigma}}^{-1/2}) \\ \Rightarrow \mathbf{T}^{-1} &= (\tilde{\mathbf{V}}_2 \otimes \tilde{\mathbf{V}}_1 : (\mathbf{V}_2 \otimes \mathbf{V}_1)\mathbf{U}\tilde{\boldsymbol{\Sigma}}^{1/2})' \end{aligned} \tag{6.6}$$

and then $\mathbf{BT} = (\mathbf{X} : \mathbf{Z})$ so we may reparameterize as $\mathbf{B}\boldsymbol{\theta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\alpha}$ where

$$\begin{aligned} \boldsymbol{\beta} &= (\tilde{\mathbf{V}}_2 \otimes \tilde{\mathbf{V}}_1)'\boldsymbol{\theta}, \\ \boldsymbol{\alpha} &= ((\mathbf{V}_2 \otimes \mathbf{V}_1)\mathbf{U}\tilde{\boldsymbol{\Sigma}}^{1/2})'\boldsymbol{\theta}. \end{aligned} \tag{6.7}$$

Lastly, we consider the penalty. The penalty term $\boldsymbol{\theta}'\mathbf{P}\boldsymbol{\theta} = \boldsymbol{\omega}'\mathbf{T}'\mathbf{P}\mathbf{T}\boldsymbol{\omega}$ where $\boldsymbol{\omega}' = (\boldsymbol{\beta}', \boldsymbol{\alpha}')$. Now $\mathbf{P}(\tilde{\mathbf{V}}_2 \otimes \tilde{\mathbf{V}}_1) = \mathbf{0}$ so the penalty becomes $\boldsymbol{\alpha}'\mathbf{F}\boldsymbol{\alpha}$ for some $\mathbf{F}$. Further, since

$$(\mathbf{V}_2 \otimes \mathbf{V}_1)'\mathbf{P}(\mathbf{V}_2 \otimes \mathbf{V}_1) = \lambda_1\mathbf{I}_{c_2} \otimes \boldsymbol{\Sigma}_1 + \lambda_2\boldsymbol{\Sigma}_2 \otimes \mathbf{I}_{c_1} \tag{6.8}$$

it follows that $\mathbf{F}=\lambda_1\boldsymbol{\Psi}_1+\lambda_2\boldsymbol{\Psi}_2$ where $\boldsymbol{\Psi}_1=\tilde{\boldsymbol{\Sigma}}^{-1/2}\mathbf{U}'(\mathbf{I}_{c_2}\otimes\boldsymbol{\Sigma}_1)\mathbf{U}\tilde{\boldsymbol{\Sigma}}^{-1/2}=\mathrm{diag}\{\tau_{1i}/(\tau_{1i}+\tau_{2j})\}$ and $\boldsymbol{\Psi}_2=\mathbf{I}_{c_1c_2-4}-\boldsymbol{\Psi}_1$. So $\mathbf{F}$ is diagonal and depends on the smoothing parameters λ_1 and λ_2. We note that when $\lambda_1=\lambda_2=\lambda$, say, the isotropic case, then $\mathbf{F}=\lambda\mathbf{I}$.

With the new basis and new penalty we minimize the following penalized sum of squares:

$$S(\boldsymbol{\alpha},\boldsymbol{\beta};\mathbf{y},\lambda_1,\lambda_2)=(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\boldsymbol{\alpha})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\boldsymbol{\alpha})+\boldsymbol{\alpha}'\mathbf{F}\boldsymbol{\alpha} \tag{6.9}$$

from which it is easy to show that $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ satisfy

$$\begin{pmatrix}\mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z}\\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z}+\mathbf{F}\end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{\beta}}\\ \hat{\boldsymbol{\alpha}}\end{pmatrix}=\begin{pmatrix}\mathbf{X}'\\ \mathbf{Z}'\end{pmatrix}\mathbf{y}, \tag{6.10}$$

the mixed model equations corresponding to model (6.2) with $\mathbf{G}=\sigma^2\mathbf{F}^{-1}$; see Searle *et al.* (1992), page 276. Estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ now follow from standard mixed model theory:

$$\hat{\boldsymbol{\beta}}=(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}, \tag{6.11}$$

$$\hat{\boldsymbol{\alpha}}=\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}) \tag{6.12}$$

where $\mathbf{V}=\sigma^2\mathbf{I}+\mathbf{Z}\mathbf{G}\mathbf{Z}'$. Smoothing parameters may be selected by maximizing the residual log-likelihood

$$l(\sigma^2,\lambda_1,\lambda_2)=-\tfrac{1}{2}\log|\mathbf{V}|-\tfrac{1}{2}\log|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}|-\tfrac{1}{2}\mathbf{y}'(\mathbf{V}^{-1}-\mathbf{V}^{-1}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1})\mathbf{y}. \tag{6.13}$$

It remains to show that equations (6.11)–(6.13) can be computed with the array algorithms (3.3) and (3.4). The matrix $\mathbf{V}=\sigma^2\mathbf{I}+\mathbf{Z}\mathbf{G}\mathbf{Z}'$ is an example of a Schur complement and so its determinant and inverse can be written (Searle *et al.* (1992), page 453)

$$|\mathbf{V}|=\sigma^{2n}|\mathbf{G}||\mathbf{G}^{-1}+\frac{1}{\sigma^2}\mathbf{Z}'\mathbf{Z}|, \tag{6.14}$$

$$\mathbf{V}^{-1}=\frac{1}{\sigma^2}\{\mathbf{I}-\mathbf{Z}(\sigma^2\mathbf{G}^{-1}+\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\}. \tag{6.15}$$

The array computations are now readily obtained. For example, to compute $|\mathbf{V}|$ we need the array computation of $\mathbf{Z}'\mathbf{Z}$. From equation (6.5) it is sufficient to compute

$$(\mathbf{B}_2\mathbf{V}_2\otimes\mathbf{B}_1\mathbf{V}_1)'(\mathbf{B}_2\mathbf{V}_2\otimes\mathbf{B}_1\mathbf{V}_1)\equiv\rho[G(\mathbf{B}_2\mathbf{V}_2)',\rho\{G(\mathbf{B}_1\mathbf{V}_1)',\mathbf{W}\}] \tag{6.16}$$

where $\mathbf{W}=\mathbf{1}\mathbf{1}'$, $n_1\times n_2$, is the identity weight matrix. Of course, with normal errors, we may compute $\mathbf{Z}'\mathbf{Z}$ directly from $\mathbf{V}_2'\mathbf{B}_2'\mathbf{B}_2\mathbf{V}_2\otimes\mathbf{V}_1'\mathbf{B}_1'\mathbf{B}_1\mathbf{V}_1$. The remaining quantities, $\mathbf{X}'\mathbf{X}$, $\mathbf{X}'\mathbf{Z}$, $\mathbf{Z}'\mathbf{y}$ and $\mathbf{X}'\mathbf{y}$, are computed in a similar fashion; see Durban *et al.* (2005) for further details and examples.

In the generalized linear mixed model case we use the penalized quasi-likelihood of Breslow and Clayton (1993) and iterate between equations (6.11) and (6.12) on the one hand and equation (6.13) on the other; in a generalized linear mixed model, $\mathbf{V}=\mathbf{W}_\delta^{-1}+\mathbf{Z}\mathbf{G}\mathbf{Z}'$ where $\mathbf{W}_\delta$ reduces to $\mathrm{diag}\{\exp(\mathbf{X}\boldsymbol{\beta}+\mathbf{Z}\boldsymbol{\alpha})\}$ in the Poisson case; here, we need expression (6.16) to compute $\mathbf{Z}'\mathbf{W}_\delta\mathbf{Z}$ etc.

7. Further examples of smoothing with generalized linear array model

We have used our method in several areas. One important area is the construction of multi-dimensional histograms. Eilers *et al.* (2006) give an example in two dimensions with scattered data. We cover the area with a fine mesh of rectangular bins and compute two matrices: $\mathbf{W}$ holds the number of observations in each bin and $\mathbf{M}$ holds the average of the observed values

in each bin (or 0 if there are none). We then smooth $\mathbf{M}$ with weights $\mathbf{W}$. Eilers *et al.* (2006) also deal with the smoothing of two-dimensional scattered data with *P*-splines. However, the array approach in the present paper is not available in the general case; see Dierckx (1993) for a full discussion. We illustrate our method with two further examples: in the first example we use an additive model to analyse some microarray data where each component has a GLAM form, and in the second example we consider a large regression problem in three dimensions with data on the incidence of respiratory disease.

7.1. Example: an additive model with two generalized linear array model components

This example comes from microarray technology. Microarrays carry a grid of thousands of tiny spots, each specific for a unique complementary deoxyribonucleic acid (DNA) fragment. When a specially prepared fluid containing these fragments is put on the surface of a microarray for several hours, complementary DNA fragments hybridize (stick to) their 'own' spot. The amount that hybridizes is (approximately) proportional to the concentration in the fluid. The complementary DNA molecules have been labelled with a fluorescent dye so that, when the microarray is scanned with a laser, an image is obtained in which the brightness of each spot denotes the concentration of the corresponding complementary DNA.

Unfortunately, the surroundings of the spots are not completely dark. They also show fluorescence, which is known as the background. One way of correcting for the background is to measure it in the surroundings of a spot and to subtract it from the brightness of the spot itself (the foreground). Fig. 4(a) shows the background for a microarray with a 72×128 grid. We see three components: a smooth spatial trend, sudden changes at the boundaries of subgrids and noise. The subgrids stem from the way that a microarray is produced: the spots have been printed with a set of pins, arranged in two rows and four columns. Apparently each pin has a different influence on the background.

We could simply ignore the pin effects and use the GLAM to smooth the matrix of background values. However, the pin effects can also be modelled within the GLAM framework. We introduce a second Kronecker product basis using *B*-splines of zero degree (piecewise constant): $\mathbf{B}_0$, 72×2, for the first dimension (rows) and $\breve{\mathbf{B}}_0$, 128×4, for the second (columns). If $\mathbf{C}$, 2×4, is the matrix of pin coefficients, then the model becomes

$$E(\mathbf{Y}) = \mathbf{B}_3 \mathbf{A} \breve{\mathbf{B}}_3' + \mathbf{B}_0 \mathbf{C} \breve{\mathbf{B}}_0', \tag{7.1}$$

where now we indicate the degree of a *B*-spline basis by a subscript. The first term represents the cubic *B*-splines for the smooth trend; the second term covers the pin effect. We have the usual roughness penalties on (the rows and columns of) $\mathbf{A}$, but not on $\mathbf{C}$.

As it stands, this model is not identifiable, because adding an arbitrary constant to $\mathbf{A}$ and subtracting it from $\mathbf{C}$ will give the same values for $E(\mathbf{Y})$. Instead of trying to deflate one of the bases to correct for this, we use a small ridge penalty (the sum of the squares of the elements) on $\mathbf{A}$ and $\mathbf{C}$. This gives the unique minimum norm solution.

Equation (7.1) expresses the mean of $\mathbf{Y}$ as a sum, where each part is computed by using GLAM arithmetic. We also need to compute the matrix of inner products and for this we extend the definition of the row tensor $G(\mathbf{X})$ of a matrix $\mathbf{X}$ to the following.

Definition 6. The *row tensor* of the matrices $\mathbf{X}_1$, $n \times c_1$, and $\mathbf{X}_2$, $n \times c_2$, is defined as

$$G_2(\mathbf{X}_1, \mathbf{X}_2) = (\mathbf{X}_1 \otimes \mathbf{1}_{c_2}') * (\mathbf{1}_{c_1}' \otimes \mathbf{X}_2) \tag{7.2}$$

where $\mathbf{1}_{c_1}$ and $\mathbf{1}_{c_2}$ are vectors of 1s of lengths c_1 and c_2 respectively. We note that $G_2(\mathbf{X}, \mathbf{X}) = G(\mathbf{X})$.

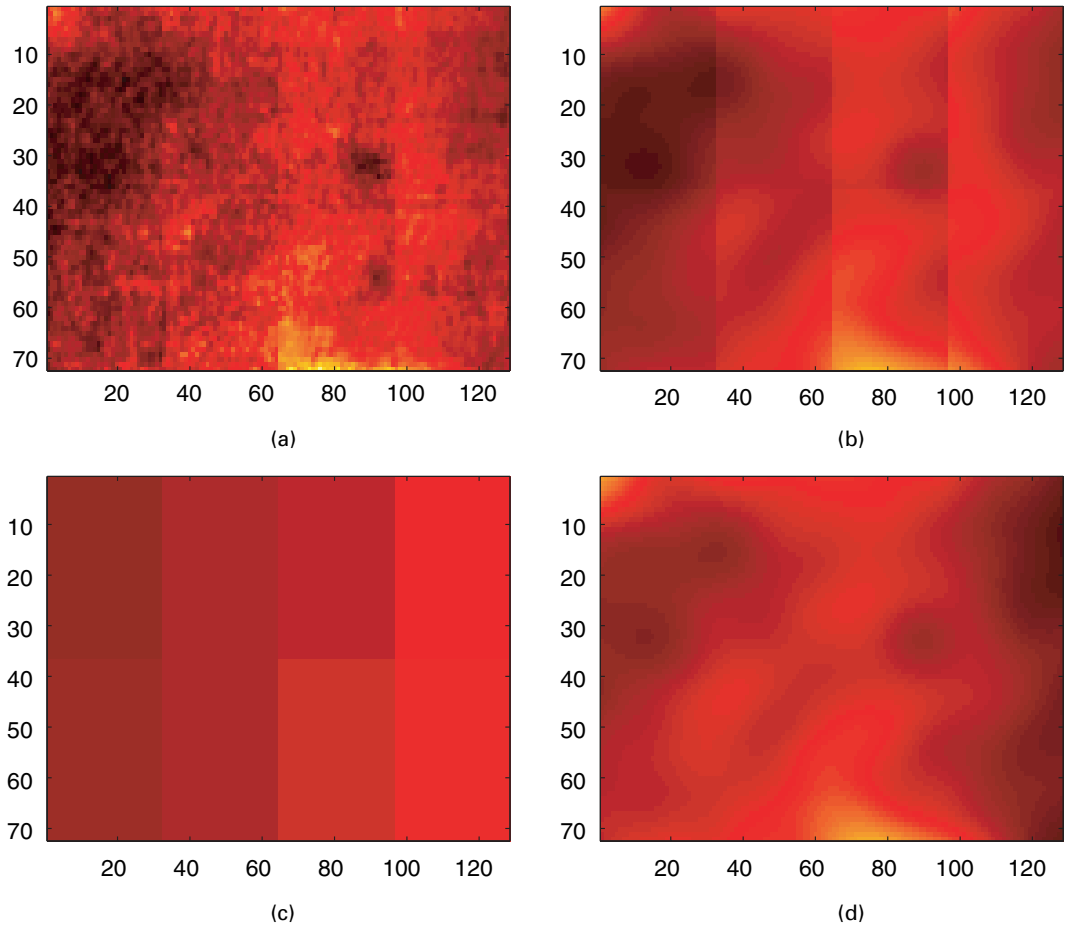

Fig. 4. Smoothing of background measurements on a microarray: (a) data; (b) fitted surface, consisting of a smooth trend and eight rectangles for pin effects; (c) pin effects; (d) smooth trend

The matrix of inner products for the penalized normal equations for the elements of $\mathbf{A}$ and $\mathbf{C}$ can be partitioned into four parts. On the diagonal we find the reshuffled elements of $G(\mathbf{B}_3)'\mathbf{W}\,G(\check{\mathbf{B}}_3)$ and $G(\mathbf{B}_0)'\mathbf{W}\,G(\check{\mathbf{B}}_0)$. The off-diagonal submatrices contain the reshuffled elements of $G_2(\mathbf{B}_3, \mathbf{B}_0)'\mathbf{W}\,G_2(\check{\mathbf{B}}_3, \check{\mathbf{B}}_0)$ and its transpose. Here $\mathbf{W}$ is a matrix of the size of $\mathbf{Y}$ containing 0s and 1s, indicating the absence or presence of a measured background. Thus, the GLAM algorithm also applies to additive models with multiple Kronecker product bases.

The four panels of Fig. 4 show the data, the fitted smooth surface with jumps between subgrids, the pin effects and the estimated spatial component.

As a final remark on this example we note that the pin effects are modelled with only eight parameters. An alternative approach to fitting model (7.1) is to use backfitting where we iterate between fitting the smooth component as a GLAM and the pin component as an ordinary regression. We do not recommend this approach in the present example but it could be used to fit non-rectangular block effects. One application is the study of the effect of influenza epidemics on the shape of the mortality surface, where theory indicates raised mortality in certain triangular regions, defined by cohorts, of the mortality surface (Oeppen, 2004). These regions may have been observed empirically (Richards *et al.*, 2005).

7.2. Example: a generalized linear array model in three dimensions

Our final example uses American data on the number of deaths from respiratory disease. The data array $\mathbf{Y} = Y[i, j, k]$ is indexed by age at death, $i = 1, \ldots, 105$, year of death, $j = 1, \ldots, 40$ (1959–1998), and month of death, $k = 1, \ldots, 12$. Thus $\mathbf{Y}$ has 50400 points arranged in a $105 \times 40 \times 12$ array. We model the number of deaths $Y[i, j, k]$ with a GLAM with Poisson error and log-link; the logarithm of the number of days in a month is used as an offset. The regression matrix $\mathbf{B}$ is defined via the marginal regression matrices of B-splines for age, $\mathbf{B}_a$, year, $\mathbf{B}_y$, and month, $\mathbf{B}_m$. We chose equally spaced knots as follows: at 1 and 105 with 11 internal knots for age, at 1 and 40 with six internal knots for year and at 1 and 12 with three internal knots for month. With cubic B-splines this gives $\mathbf{B}_a$, 105×15, $\mathbf{B}_y$, 40×10, and $\mathbf{B}_m$, 12×7. The full regression matrix $\mathbf{B} = \mathbf{B}_m \otimes \mathbf{B}_y \otimes \mathbf{B}_a$ has 1050 parameters arranged in a $15 \times 10 \times 7$ array, $\boldsymbol{\Theta}$. This is a large regression problem: the regression matrix $\mathbf{B}$ alone has over 5×10^7 elements. The GLAM form for the mean of $\mathbf{Y}$ is

$$E(\mathbf{Y}) = \mathbf{M} * \boldsymbol{\Gamma}, \quad \log(\boldsymbol{\Gamma}) = \rho[\mathbf{B}_m, \rho\{\mathbf{B}_y, \rho(\mathbf{B}_a, \boldsymbol{\Theta})\}] \tag{7.3}$$

where $\mathbf{M}$ is the $105 \times 40 \times 12$ array which holds the number of the days in the month.

It remains to define the penalty matrix $\mathbf{P}$. We penalize each dimension of $\boldsymbol{\Theta}$ in turn, i.e. we place penalties on the columns (dimension 1), rows (dimension 2), etc., of the array. We find

$$\mathbf{P} = \lambda_1 \mathbf{I}_{c_3} \otimes \mathbf{I}_{c_2} \otimes \mathbf{D}_1' \mathbf{D}_1 + \lambda_2 \mathbf{I}_{c_3} \otimes \mathbf{D}_2' \mathbf{D}_2 \otimes \mathbf{I}_{c_1} + \lambda_3 \mathbf{D}_3' \mathbf{D}_3 \otimes \mathbf{I}_{c_2} \otimes \mathbf{I}_{c_1} \tag{7.4}$$

where $\mathbf{D}_1$, $\mathbf{D}_2$ and $\mathbf{D}_3$ are difference matrices; this generalizes equation (2.10); see Currie *et al.* (2004) and Wood (2004) for further discussion of penalties with Kronecker product bases. Expression (7.4) indicates the general formula in d dimensions.

The parameters are estimated by using second-order penalties and the Bayesian information criterion. The fitted model has effective degrees of freedom of 305. Fig. 5 gives an idea of how the numbers of deaths vary with age, year and month. Fig. 5(b) suggests that the number of deaths in 1972 is an extreme outlier. We dealt with this by defining a diagonal weight matrix $\mathbf{V}_\delta$ with 0s for data corresponding to 1972 and 1s elsewhere and modifying the penalized scoring algorithm (1.2), as follows:

$$(\mathbf{B}' \mathbf{V}_\delta \tilde{\mathbf{W}}_\delta \mathbf{B} + \mathbf{P}) \hat{\boldsymbol{\theta}} = \mathbf{B}' \mathbf{V}_\delta \tilde{\mathbf{W}}_\delta \tilde{\mathbf{z}}. \tag{7.5}$$

The penalty function bridges the gap between 1971 and 1973. More generally, interpolation and extrapolation are achieved by defining the appropriate diagonal matrix $\mathbf{V}_\delta$ or its array equivalent $\mathbf{V}$. This device may also allow us to use a GLAM with d-dimensional data which do not lie in an array, provided that we can fill in the missing part of the array with dummy data values. For example, suppose that a number of scattered data values are missing from the $105 \times 40 \times 12$ data array. The data do not lie in an array but can be made to do so by defining dummy data values and the appropriate weight matrix or array.

The choice of initial values for the solution of algorithm (1.2) is extremely important. We use a two-stage approach. We use $\log\{(\mathbf{Y} + 0.5)/\mathbf{M}\}$ as an initial estimate of $\rho[\mathbf{B}_m, \rho\{\mathbf{B}_y, \rho(\mathbf{B}_a, \boldsymbol{\Theta})\}]$ and this in turn gives an initial estimate of $\boldsymbol{\Theta}$. We set $\lambda_a = \lambda_y = \lambda_m = 1$, say, and then iterate algorithm (1.2) until convergence. This gives a second and much improved initial estimate of $\boldsymbol{\Theta}$ which is used in the subsequent search for optimal values of λ_a, λ_y and λ_m; this simple device alone cuts computer time by around 75% in this example.

The order of the storage of the variables makes a difference to GLAM performance. The data should be stored with the largest variable (here age) varying fastest to smallest variable (here month) varying slowest; in our example, this ordering of the data implies that the regression matrix is $\mathbf{B}_m \otimes \mathbf{B}_y \otimes \mathbf{B}_a$. We conjecture that this order is optimal in the sense that the number

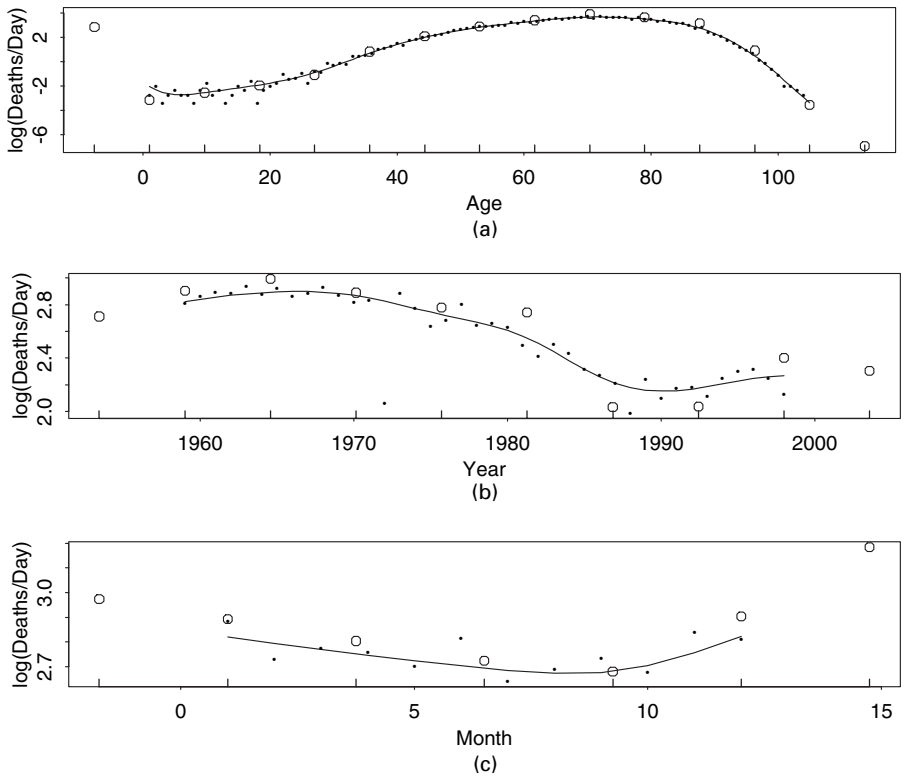

Fig. 5. Observed (·) and smoothed (———) numbers of log(deaths/day) by age, year and month (regression coefficients $\hat{\Theta}[i, j, k]$ plotted against knot position (∘)): (a) January 1959, $\hat{\Theta}[i, 2, 2]$, $i = 1, \ldots, 15$; (b) age 53 years, January, $\hat{\Theta}[8, j, 2]$, $j = 1, \ldots, 10$; (c) age 53 years, 1959, $\hat{\Theta}[8, 2, k]$, $k = 1, \ldots, 7$

of multiplications that are performed to calculate the elements of $\mathbf{B}'\mathbf{W}_\delta\mathbf{B}$ is minimized. The algorithm takes about 25% longer to run with the variables in the reverse order.

Performance may also be improved by noting that each row of the row tensor function of $\mathbf{X}$, $n \times c$, in equation (2.2) contains c^2 elements of which only $c(c+1)/2$ are distinct. It is possible to discard the duplicate elements in each row tensor function and then to apply expression (3.4). The number of multiplications is cut by a factor of about 2^d; an additional overhead is required to restore the missing output to give $\mathbf{B}'\mathbf{W}_\delta\mathbf{B}$. It will also be possible to take advantage of the sparse form of both the B-spline matrices $\mathbf{B}_i$ and their row tensor $G(\mathbf{B}_i)$; see de Boor (1979).

We have programmed our method in both S-PLUS or R and MATLAB and the simple form of expressions (3.3)–(3.5) means that this is straightforward. The method is also available for low level languages (C, Fortran, etc.) where the twin benefits of low storage and high speed apply equally.

8. Concluding remarks

In the example in Section 7.1 we used equation (7.1) to describe some microarray data. This model has one smooth (two-dimensional) term and one regression (two-factor) term where each component has the form of a GLAM and the components are combined additively. We also described a model for the effect of influenza on the mortality surface in terms of an additive model with one smooth two-dimensional GLAM term and one ordinary regression (factor) term. These examples illustrate how a GLAM can be part of a general model building process.

In more complex models the GLAM algorithms are used to fit components corresponding to general (high dimensional) smooth functions; these components are then combined with regression and/or low dimensional smooth terms. In its most general form we envisage an additive model with regression terms which can be in standard or GLAM form and smooth terms which may also be in standard or array form. The penalty matrix is block diagonal with a separate block for each additive smooth term; there is no penalty on ordinary regression terms. We give one simple example to indicate the possibilities.

The age–period–cohort model (Clayton and Schifflers, 1987) is an additive model for a mortality table. With the same notation as the example in Section 2.3 we write the linear predictor for the log-hazard as $\eta_{i,j} = \alpha_i + \beta_j + \gamma_k$ where i indexes age, j year and k cohort. A smooth version of this model can be obtained as follows. Let $\mathbf{B}_a$, $n_a \times c_a$, and $\mathbf{B}_y$, $n_y \times c_y$, be marginal B-spline bases for age and year, and $\mathbf{B}_c$, $n_a n_y \times c_c$, be a B-spline basis for cohort. Then

$$\boldsymbol{\eta} = (\mathbf{1}_{n_y} \otimes \mathbf{B}_a)\boldsymbol{\alpha} + (\mathbf{B}_y \otimes \mathbf{1}_{n_a})\boldsymbol{\beta} + \mathbf{B}_c\boldsymbol{\gamma} \tag{8.1}$$

is a smooth age–period–cohort model which can be fitted by using GLAM algorithms for the age and year terms and ordinary regression for the cohort term. This example illustrates a common feature of models for tables: the age and year terms are each univariate terms and so the array form has a 'degenerate' basis along the 'other' dimension. The penalty matrix is block diagonal with a separate block for each of age, period and cohort. If we replace $\mathbf{B}_y$ with $\mathbf{I}_{n_y}$ (which corresponds to B-splines of degree 0) and set the year penalty to 0 then we have a model with two GLAM terms, one smooth and one discrete, and one smooth regression term. The original discrete age–period–cohort model is a special case of model (8.1) with all the bases computed by using B-splines of degree 0 and no penalties.

The outputs of algorithms (3.3) and (3.5) have an attractive geometric interpretation: the smoothed values and their standard errors are all correctly positioned, as defined by the data array. Algorithm (3.4) has a similar geometric interpretation. In a univariate scheme the variance of a vector is a matrix. In our case we have a coefficient array $\mathbf{A}$ which sits in d dimensions so its variance 'matrix' should be an array in $2d$ dimensions; this is exactly what happens since the output of algorithm (3.4) is a d-dimensional array $c_1^2 \times \ldots \times c_d^2$ which is reshaped into a $2d$-dimensional array $c_1 \times c_1 \times \ldots \times c_d \times c_d$ in the software section. It seems that the GLAM sits naturally in d dimensions: data, means, variances and standard errors of fitted values.

The GLAM is designed for smoothing multidimensional arrays. It is often appropriate to use a Kronecker product basis to smooth a multidimensional array but a direct approach using standard regression algorithms can quickly become unusable when the number of data points is large, as will usually be the case. The GLAM is conceptually attractive since

(a) data, fitted values and standard errors sit naturally in the correct space and
(b) the coefficient array approximates the smoothed surface when the basis is computed as the Kronecker product of marginal B-spline bases, as in Fig. 2 and Fig. 5.

Computationally the method has two advantages: low storage and high speed. Complex additive models with high dimensional smooth terms can be computationally very challenging; GLAM algorithms will alleviate some of the computational burden and make such models more readily available.

Acknowledgements

We thank the Continuous Mortality Investigation Bureau for providing the data for the example in Section 2.3 and for providing financial support to all three authors. We are indebted to Profes-

sor Jim Howie of Heriot-Watt University who first suggested the use of the H-transform and to Dr Froilan Martinez-Dopico of Carlos III University who drew our attention to the properties of Kronecker sums. We are also grateful to Roland Rau of the Max Planck Institute of Demography who provided the respiratory data that were used in the example in Section 7.2. The work of Maria Durban was supported by Dirección General de Universidades e Investigación, Comunidad de Madrid, project HSE/0181/2004. Finally we are grateful to the referees for their most helpful comments.

References

Akaike, H. (1973) Maximum likelihood identification of Gaussian autoregressive moving average models. *Biometrika*, **60**, 255–265.
de Boor, C. (1979) Efficient computer manipulation of tensor products. *ACM Trans. Math. Softwr.*, **5**, 173–182.
Breslow, N. E. and Clayton, D. G. (1993) Approximate inference in generalized linear mixed models. *J. Am. Statist. Ass.*, **88**, 9–25.
Brewer, J. W. (1978) Kronecker products and matrix calculus in system theory. *IEEE Trans. Circ. Syst.*, **25**, 772–781.
Clayton, D. and Schifflers, E. (1987) Models for temporal variation in cancer rates: II, Age-period-cohort models. *Statist. Med.*, **6**, 469–481.
Craven, P. and Wahba, G. (1979) Smoothing noisy data with spline functions. *Numer. Math.*, **31**, 377–403.
Currie, I. D., Durban, M. and Eilers, P. H. C. (2004) Smoothing and forecasting mortality rates. *Statist. Modllng*, **4**, 279–298.
Dierckx, P. (1993) *Curve and Surface Fitting with Splines*. Oxford: Clarendon.
Durban, M., Currie, I. D. and Eilers, P. H. C. (2005) Multidimensional P-spline mixed models: an efficient method for estimation of multivariate densities. To be published.
Eilers, P. H. C. (1999) Discussion on 'The analysis of designed experiments and longitudinal data by using smoothing splines' (by A. P. Verbyla, B. R. Cullis, M. G. Kenward and S. J. Welham). *Appl. Statist.*, **48**, 307–308.
Eilers, P. H. C., Currie, I. D. and Durban, M. (2006) Fast and compact smoothing on large multidimensional grids. *Comput. Statist. Data Anal.*, **50**, 61–76.
Eilers, P. H. C. and Marx, B. D. (1996) Flexible smoothing with B-splines and penalties. *Statist. Sci.*, **11**, 89–121.
Gower, J. C. (1982) The Yates algorithm. *Util. Math.*, **21**, 99–115.
Green, P. J. (1985) Linear models for field trials, smoothing and cross-validation. *Biometrika*, **72**, 527–537.
Green, P. J. and Silverman, B. W. (1994) *Nonparametric Regression and Generalized Linear Models*. London: Chapman and Hall.
Horn, R. A. and Johnson, C. R. (1991) *Topics in Matrix Analysis*. Cambridge: Cambridge University Press.
Oeppen, J. (2004) Personal communication.
R Development Core Team (2004) *R: a Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing.
Richards, S. J., Kirkby, J. G. and Currie, I. D. (2005) The importance of year of birth in two-dimensional mortality data. To be published.
Ruppert, D., Wand, M. P. and Carroll, R. J. (2003) *Semiparametric Regression*. Cambridge: Cambridge University Press.
Schwarz, G. (1978) Estimating the dimension of a model. *Ann. Statist.*, **6**, 461–464.
Searle, S. R. (1982) *Matrix Algebra Useful for Statistics*. New York: Wiley.
Searle, S. R., Casella, G. and McCulloch, C. E. (1992) *Variance Components*. New York: Wiley.
Silverman, B. W. (1985) Some aspects of the spline smoothing approach to non-parametric regression curve fitting (with discussion). *J. R. Statist. Soc.* B, **47**, 1–52.
Van Loan, C. F. (2000) The ubiquitous Kronecker product. *J. Comput. Appl. Math.*, **123**, 85–100.
Verbyla, A. P., Cullis, B. R., Kenward, M. G. and Welham, S. J. (1999) The analysis of designed experiments and longitudinal data by using smoothing splines (with discussion). *Appl. Statist.*, **48**, 269–311.
Wahba, G. (1983) Bayesian "confidence intervals" for the cross-validated smoothing spline. *J. R. Statist. Soc.* B, **45**, 133–150.
Wand, M. P. (2003) Smoothing and mixed models. *Comput. Statist.*, **18**, 223–250.
Wood, S. N. (2000) Modelling and smoothing parameter estimation with multiple quadratic penalties. *J. R. Statist. Soc.* B, **62**, 413–428.
Wood, S. N. (2004) mgcv: GAMs with GCV smoothness estimation and GAMMs by REML/PQL. In *R Package Version 1.1-5*. Vienna: R Foundation for Statistical Computing.
Yates, F. (1937) The design and analysis of factorial experiments. *Technical Communication 35*, pp. 1–95. Commonwealth Bureau of Soils, Harpenden.