Array methods in statistics with applications to the modelling and forecasting of mortality

James Gavin Kirkby

Submitted for the degree of
Doctor of Philosophy

Heriot-Watt University

School of Mathematical and Computing Sciences

March 2009

Abstract

In this thesis we investigate the application of array methods for the smoothing of multi-dimensional arrays with particular reference to mortality data. A broad outline follows. We begin with an introduction to smoothing in one dimension, followed by a discussion of multi-dimensional smoothing methods. We then move on to review and develop the array methods of Currie et al. (2006), and show how these methods can be applied in additive models even when the data do not have a standard array structure. Finally we discuss the Lee-Carter model and show how we fulfilled the requirements of the CASE studentship.

Our main contributions are: firstly we extend the array methods of Currie et al. (2006) to cope with more general covariance structures; secondly we describe an additive model of mortality which decomposes the mortality surface into a smooth two-dimensional surface and a series of smooth age dependent shocks within years; thirdly we describe an additive model of mortality for data with a Lexis triangle structure.

To Susie - I love you.

Acknowledgements

The help and support of lots of people has enabled me to complete this thesis. I'd like to thank Rajeev Shah, Simon Spencer and Tony Leandro at the CMI for supporting the project and providing the data. Discussions and collaborations with the following people have helped me improve my understanding and helped me formulate new ideas: Giancarlo Camarda, Maria Durban, Paul Eilers, Jutta Gampe, Phillippe Lambert, Stephen Richards, and Richard Willets; special thanks to Richard for being such an understanding boss. I'd like to thank the following house-mates and office-mates for making my time in Edinburgh so much fun: Gudmund, Ellen, Bjorn, Breeda, Cian, Two-Dogs, and Katie. Thanks to Mum, Dad, Rich, Dan, and Mona for being so supportive and keeping me on track.

Most of all I'd like to thank my supervisor, Iain. Thank you for inspiring me with your beautiful ideas about arrays; and special thanks for all your kindness, patience, and support while I have been writing up.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this introduction we will first set out the motivation for modelling mortality. We will then give a brief over-view of the traditional actuarial methods used for the graduation of mortality tables. In the final section we will give a description of the data, focusing on the structure of the data, which is important in later chapters (particularly Chapter 4 on array methods).

1.1 Motivation for studying mortality

It may not be immediately obvious why we should be interested in aggregate mortality rates for a population; after all there is very little we can do to prevent deaths without looking at the circumstances of an individual. However, with a little thought we see that a country's government requires population forecasts so it can plan for increases (or decreases) in the demand for public services, and clearly mortality rates play a pivotal role in a country's population dynamics. In countries, like the UK, where there is provision of a state pension the government will also have an interest in mortality rates to establish the size of this liability. Many UK companies also have liabilities which carry some form of "longevity risk" as part of their final salary pension schemes. This longevity risk is fundamentally different from the mortality or longevity risk faced by an individual, which can be mitigated by the pooling of risks with others in an insurance company or mutual society, or indeed a company pension scheme. Pension schemes must bear the risk that their members live systematically longer

than anticipated, a risk that cannot be mitigated by pooling risk, as all schemes have nearly perfectly correlated risks. This type of longevity risk has been given a much higher profile after a number of UK companies have had to reduce the pensions benefits of their schemes because the assets backing the scheme do not meet the liabilities; this led to the creation of the Pension Protection Fund. One reason, often cited, for the under-funding of these schemes has been the improvement in the rates of mortality seen across the UK population. In reality, problems with the mortality assumptions were only part of the problem and expectations around investment returns which have not been realized since the beginning of the century have also contributed heavily to the unfavourable funding position of many UK pension schemes. Even for schemes currently in a better funding position, lower investment yields mean that more serious consideration has to be given to mortality assumptions because future cash-flows are not subject to such a heavy discount. More recently there has been a market developing for trading mortality and longevity risk in the form of financial derivatives. These markets may finally enable companies to trade away their longevity risk, but an understanding of mortality trends will be critical in evaluating the pricing in these markets and understanding what coverage is given by any hedging strategy developed using these instruments. Richards and Jones (2004) give a thorough description of the importance of longevity risk in assessing long term pensions and life assurance liabilities.

1.2 History of mortality models in the UK

In 1825, Gompertz noticed that the rate of mortality among adult members of a population was approximately linear with age on the log scale in many species. This gave rise to the famous formula

$$\mu_x = AB^x. \tag{1.1}$$

where μ_x is the force of mortality at age x. We will see in Chapter 2 how this formula is an example of a *Generalized Linear Model*. Ever since Gompertz made his discovery the idea of a mortality law, under which the age pattern of mortality is assumed to follow some parametric formula, has formed the basis of most actuarial models

used in the UK; see, for example, Forfar et al. (1988) for a discussion of traditional actuarial methods, and Richards (2008) for a good reference on mortality laws used by actuaries.

Since the early nineties the "market-leader" in mortality projection has been the model due to Lee and Carter (1992). It is a bi-linear model which extracts a single time component from the mortality surface (as shown in Fig. 1.2(a), for example) which is then extrapolated using a time series model. Lee and Carter applied their model to US population data, and since then it has been applied to data from countries around the world; see, for example, Tuljapurkar et al. (2000) who show the model applied to data from the G7 countries. The Lee-Carter model will be discussed in greater detail in Section 6.1.

1.3 Description of data

In this section, the general structure and format of the data are described. We go on to discuss the sources of the data and briefly outline our assumptions about the data and how they might be modelled.

1.3.1 General Description

In general we will be interested in modelling mortality tables. In the simplest case our data will take the form of two grids, one containing a set of death counts, the other the number exposed to risk (of dying).

A graphical representation of the data is shown in Figure 1.1. The number of deaths and the exposed to risk will be represented by the matrices $\boldsymbol{Y}$ and $\boldsymbol{E}$ respectively (using $\boldsymbol{Y}$ to maintain continuity with the statistical literature). The data are grouped into 1×1 blocks indexed by age last birthday and calendar year. For convenience later, we will define $\boldsymbol{x_a}$ to represent the values of the indices for age, and $\boldsymbol{x_y}$ as the corresponding index for year. Clearly $\boldsymbol{Y}$ and $\boldsymbol{E}$ must have the same dimension, and in general we will suppose them to have dimension $n_a \times n_y$, consequently $\boldsymbol{x_a}$ will have length n_a and $\boldsymbol{x_y}$ length n_y.

As with all data, the quantity and quality of the data depends greatly on the

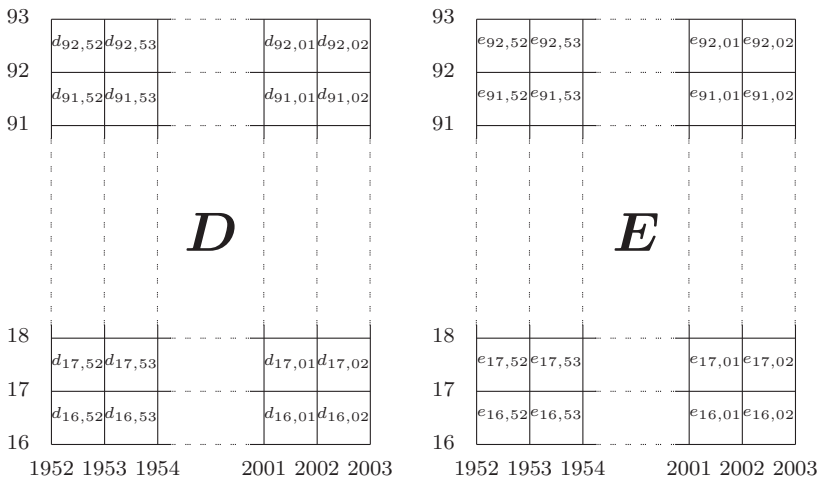


Figure 1.1: A graphical representation of the data set

source. We will be dealing with two types of data from two sources, UK insurance data from the *Continuous Mortality Investigation* (CMI), and population data from developed countries obtained from the *Human Mortality Database* (HMD). The main motivation for this research has been to find methods for the modelling and forecasting of the UK insurance data, but the methods can be applied to population data as it normally comes in the same format.

1.3.2 UK Insurance Data

In the UK, insurance data are collected through the CMI. This is a body supported by the UK life assurance industry to investigate patterns in mortality and morbidity among holders of life assurance contracts. The CMI performs two main functions; collating and cleaning the data submitted by member offices, and supervising and conducting research into methods for analysing the data.

Offices supporting the CMI submit data to the secretariat annually, where it is collated (and some cleaning is performed to remove obviously anomalous data). Historically, an in-depth analysis of the data has been carried out every ten years, normally culminating in the publication of a set of standard tables and a set of *reduction factors* which can be used to adjust the tables for different years; details of how reduction factors are applied are given in Section 6.2. As mentioned previously the favoured

methods of graduation have been based on the Gompertz model with adjustments to account for the non-linearity at the highest and lowest ages. Recently the CMI has backed away from the publication of tables of mortality reduction factors. Instead, they have favoured the publication of a suggested methodology for assessing mortality improvements for life offices to apply themselves. This change reflects just how difficult the problem of mortality forecasting is, and the amount of variability involved with any projection method. It is felt that the publication of standard tables would send out the wrong message to life offices, when responsible offices ought to be considering very carefully their exposure to mortality risk.

Until recently the offices only submitted aggregate data for the number of policies in force at the start of each year, and the number of claims (deaths) received over the year. Using the average of the number of policies in force at the start and end of the year gives a simple approximation to the central exposed to risk, and we then have the data in the format described in the previous section. The CMI has this data collated for member offices going back until 1947. When our work was started the CMI had published data only up until the end of 2002, since then more data has been released but to ensure the work remains consistent we will refer to data only up until the end of 2002. The ages in the data set run from 11 to 109 (with data for 110+ grouped), but the low exposures at the regions for the highest and lowest ages make the data unreliable, so we will restrict the data set to ages 16-92. The data are also known to be unreliable for the first few years, so we will use years 1951-2002. These data are summarised in Figure 1.1.

In 2006 the CMI started collecting data on an individual policy basis from self-administered pension schemes. In the future this will allow the study of effects such as changes in diet and smoking habits, as well as geographical and social-economic co-variates. Currently however, the data are not available in sufficient quantity for meaningful analysis.

1.3.3 Population Data

National population data are available from the HMD. Data are available from countries around the world, with particularly good data for Western Europe, North Amer-

ica, and Japan. The source of the data varies from country to country, particularly in the way that exposures are collected. For countries that keep a population register (e.g. Sweden and the Netherlands) the data are thought to be almost 100% accurate. For most countries however, the populations have to be estimated for inter-census years. Even when the exposures have to be estimated, we are compensated by the size of the exposed to risk compared to those observed in insurance data. For countries with population registers we can also obtain data on Lexis triangles, which gives us more detailed data but also requires we take a little more care when modelling; this will be discussed in Chapter 5.

1.3.4 Assumptions and Comments

With this type of data (from whichever source) we will assume that the number of deaths follow a Poisson Distribution,

$$d_{x,t} = \mathcal{P}(\tau_{x,t}\, e_{x,t}), \tag{1.2}$$

where $\tau_{x,t}$ represents the force of mortality at age x in year t.

We shall now consider how these assumptions hold in practise. The Poisson distribution is the obvious choice when faced with modelling counts data. However, the strict mean-variance relationship of the Poisson distribution is often violated in practical situations, causing over-dispersion (or under-dispersion). In the UK insurance data an obvious cause of over-dispersion is people holding multiple policies and, when these people die, we observe more than one claim in the data, and this leads to higher variance in the data. Mortality data also seem to suffer from over-dispersion in the form of random shocks to the underlying surface, most commonly by year. These tend to be physical events, normally cold winters (although there are instances of disease causing similar effects), which cause a shift across a large proportion of ages away from the underlying smooth surface. Methods for dealing with over-dispersion are discussed in depth in Section 5.3.2.

We will assume that τ is some smooth function of age and year, $\tau(x,t)$. Figure 1.2 shows a plot of the raw rates of mortality (i.e. the number of deaths divided by the exposure) on the log scale for the UK insurance data. By eye it looks as if a smooth

function could be fitted to the data, but the complexity of the function in the age direction suggest that parametric methods will not give a satisfactory explanation of the surface. More sophisticated smoothing methods have to be used in order to explain the dynamics of the mortality surface. In Chapter 2 we will give a more formal definition of what we mean by smooth functions and an introduction to smoothing methods.

We have also made the implicit assumption that our data arise from a homogeneous population. In fact, this is most unlikely to be true for the insurance data, as the data are made up from assured lives from various different policy types. This fact means that different selection effects are occurring at different parts of the age range, as certain products are more popular at some ages than at others. In the population data this is not such a problem, as long as we are careful not to combine data with big differences in the way it is collected. Obviously we must also avoid grouping data where big changes have been made to national borders, the obvious example being Germany before and after the fall of the Berlin Wall.

1.4 Plan of thesis

The remainder of the thesis will be divided into six chapters and broadly into three sections. Chapters 2 and 3 contain a review of the literature, Chapters 4 and 5 introduce new work, and Chapters 6 and 7 summarize some additional work, some conclusions, and suggestions for further work.

In Chapter 2 we give a broad description of smoothing models. The purpose of this chapter is to show the connection between the various smoothing techniques, and to show that, in terms of results, there is little to choose between the various methods in one dimension. Another theme of this chapter is the discussion of regression bases and penalized likelihood, particularly with reference to the P-splines of Eilers and Marx (1996). Topics such as model selection, and smoothing within the mixed model framework are also discussed here.

In Chapter 3 we focus on multi-dimensional smoothing. Three other smoothing techniques are discussed before we move on to multi-dimensional P-splines. The

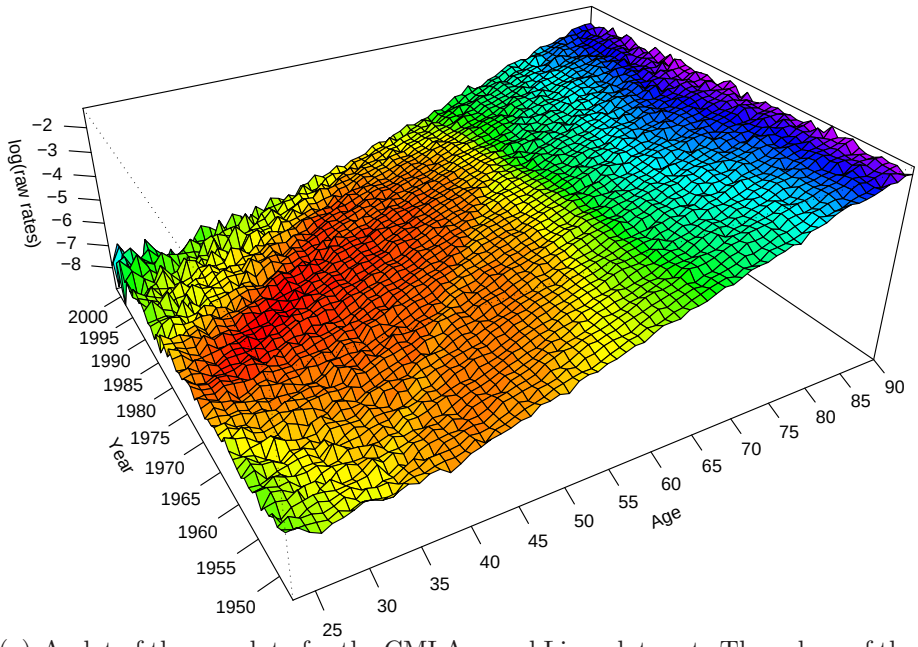
material in this chapter is illustrated with reference to the mortality data described in Section 1.3, and we draw an important distinction between the structure of the model when our data lie on a grid and when they do not.

In Chapter 4 we follow Currie et al. (2006), and show that when our data have a grid structure the use of multi-dimensional arrays can lead to large computational savings. We give a different perspective on how to interpret the array methods of Currie et al. (2006), and show how this can be used to extend their methods to allow the use of more general matrix structures in array models.

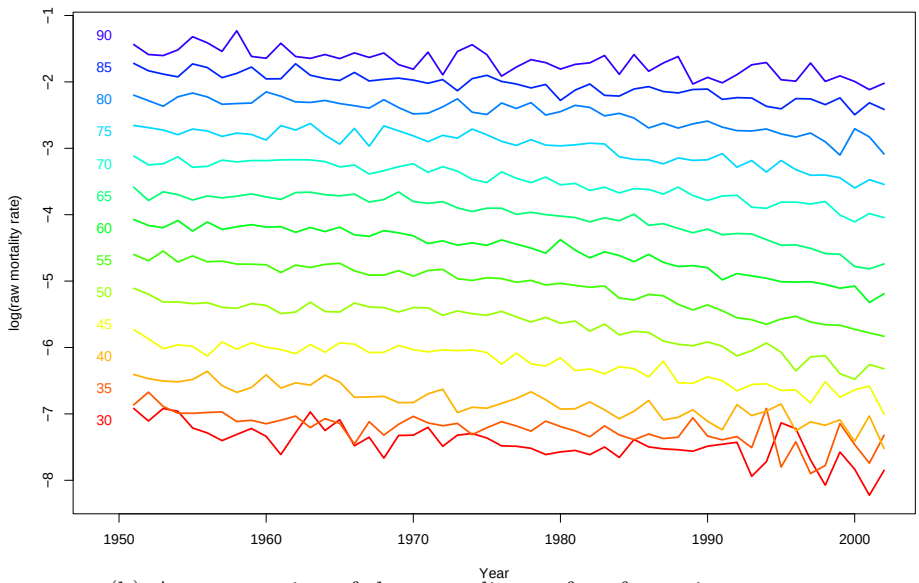
In Chapter 5 we show how the methods described in Chapter 4 can be used within additive models. Two examples are given in which a dis-continuous component is added to an under-lying two-dimensional surface to better explain the data; the first example uses an additive model to describe Lexis data, and the second uses an additive model to describe random shocks to the mortality surface (like those described in Section 1.3.4).

In Chapter 6 we describe the Lee-Carter model (which in its original form was not a smooth model). We show how the model can be expressed using the array methods from Chapter 4, and highlight the problems encountered when we try and introduce smoothness into an over-parameterized model such as the Lee-Carter. In this chapter we also document the work that was carried out in order to satisfy the requirements of our CASE studentship.

In the final chapter, Chapter 7, we draw our conclusions and suggest areas of further work.



(a) A plot of the raw data for the CMI Assured Lives data set. The colour of the facets correspond to the exposed to risk, hot colours represent high exposures.



(b) A cross section of the mortality surface for various ages.

Figure 1.2: The raw mortality surface.

Chapter 2

Smoothing One-dimensional Data

In this chapter smoothing in one dimension is introduced. In Section 2.1 we describe what is meant by smoothing, and describe general properties desirable for a good smoother. This is followed, in Section 2.2, by a discussion of three general methods of smoothing: local averaging, kernel smoothing, and smoothing splines. All these methods are examples of *full rank* smoothers. In Section 2.3 we focus on regression splines and penalized likelihood, with particular attention given to the use of B-splines as basis functions. In sections 2.4 and 2.5 model selection is considered, and we show the connection between smoothing and Mixed Models, and find a representation of Penalized B-splines as Mixed Models. In Section 2.6 we introduce the *Generalized Linear Model* (GLM), and show how smoothing can be used within the GLM framework to model data described in Section 1.3; we also show how to allow for random effects in a *Generalized Linear Mixed Model* (GLMM). In Section 2.7 we briefly discuss Bayesian smoothing, and note some advantages this has over classical models. We introduce one-dimensional forecasting for penalized B-splines in Section 2.8, and conclude with a general discussion in Section 2.9.

2.1 Smoothing

In this section we introduce the basic concepts of smoothing models. We use some data from a motorcycle crash simulation from Silverman (1985) as a motivating example. The data can be found in the `MASS` package of R (R Development Core Team, 2005).

We have 133 observations of the head acceleration (in $g \approx 9.81\, m/s^2$) at various times (measured in milliseconds) after a simulated motorcycle crash. We aim to model the head acceleration, $\boldsymbol{y}$, as a function of the time after the crash, $\boldsymbol{x}$. A scatter plot of these data is shown in Fig. 2.1.

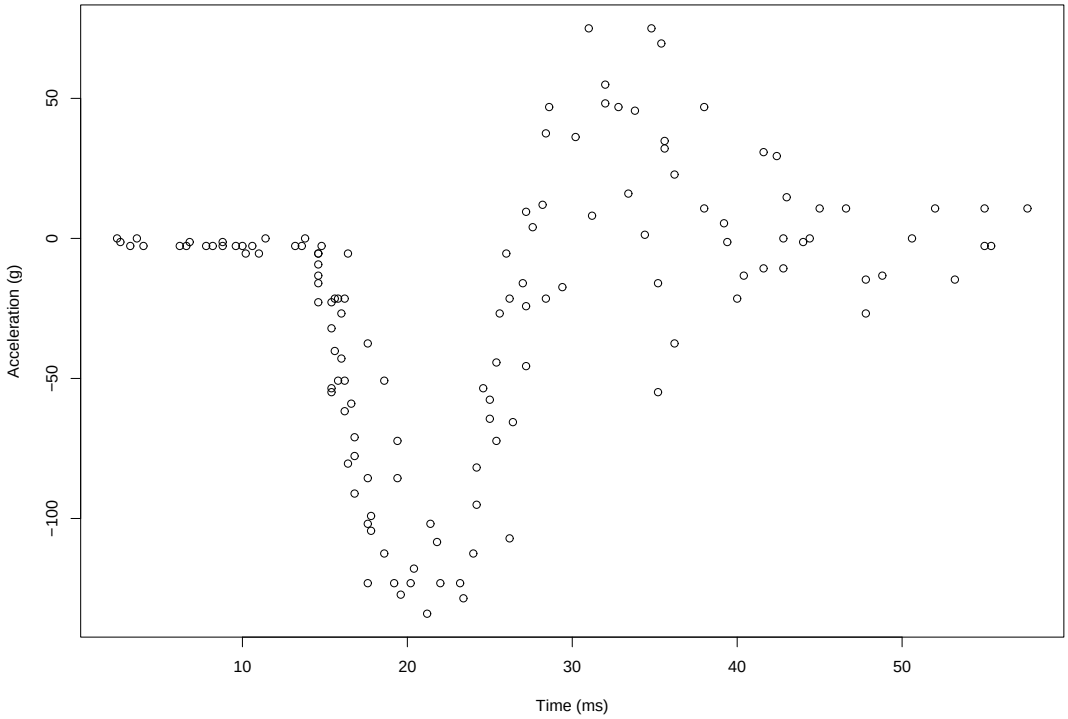


Figure 2.1: Scatterplot of the motorcycle data from Silverman (1985).

In the most general terms, scatterplot smoothing is a method by which we aim to estimate a smooth function from some data that is subject to random noise. Specifically, we assume that we have a set of observations on a response variable $\boldsymbol{y}' = (y_1, \ldots, y_n)$ and a corresponding set of observations on an explanatory variable $\boldsymbol{x}' = (x_1, \ldots, x_n)$, and that the observations are ordered such that $a \leq x_1 \leq \cdots \leq x_n \leq b$. We wish to find a function $S(\cdot)$ defined on $[a, b]$, such that

$$E(y_i) = \mu_i = S(x_i), \qquad i = 1, \ldots, n \tag{2.1}$$

where $S(\cdot)$ is a smooth function. For the rest of this section we will be dealing with the special case

$$y_i = S(x_i) + \epsilon_i \tag{2.2}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$.

The strictest and simplest smoothness requirement is to assume $S(\cdot)$ is constant for all values of x_i

$$y_i = S(x_i) = c \tag{2.3}$$

generally taking $c = \bar{y}$. This clearly gives a smooth function, but in most practical cases it would lead to an over-simplified model. Next, we could consider using polynomials to smooth our data

$$S(x_i) = p_k(x_i), \tag{2.4}$$

where $p_k(\cdot)$ is a polynomial of degree k, specifically

$$S(x_i) = \beta_0 + \beta_1 x_i + \ldots + \beta_k x_i^k = \boldsymbol{x}_i'\boldsymbol{\beta}, \tag{2.5}$$

where $\boldsymbol{x}_i' = (1, x_i, \ldots, x_i^k)$ and $\boldsymbol{\beta}' = (\beta_0, \beta_1, \ldots, \beta_k)$ is a set of unknown parameters. This can be expressed in matrix notation as

$$S(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\beta} \tag{2.6}$$

where $\boldsymbol{X}' = [\, \boldsymbol{x}_1 \vdots \cdots \vdots \boldsymbol{x}_n \,]$. We estimate the parameters using maximum likelihood, which in this case amounts to minimizing the residual sum of squares,

$$\mathrm{RSS}(\boldsymbol{\beta}) = \big(\boldsymbol{y} - S(\boldsymbol{x})\big)'\big(\boldsymbol{y} - S(\boldsymbol{x})\big), \tag{2.7}$$

leading to the well known estimate for $\boldsymbol{\beta}$

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}. \tag{2.8}$$

Setting $k = 1$ we get a linear function, which although smooth, may not have enough flexibility for some situations. By increasing k we increase the flexibility (until the data are interpolated when $k = n-1$) but at some point our function loses the smoothness property we desire. Figure 2.2 shows polynomials of varying degrees fitted to data from the motorcycle crash simulation. We see that for $k = 1$, 2, and 3, polynomials do not do a very good job of fitting to the data. We can improve the fit by using 5^{th} or even 10^{th} degree functions, but now the fitted functions exhibit features that do not seem to be driven by the data. We shall see in Section 2.3 that this is caused by the use of *global basis functions* in polynomial regression, and a key feature of better performing smoothers is the use of basis functions with *compact support*.

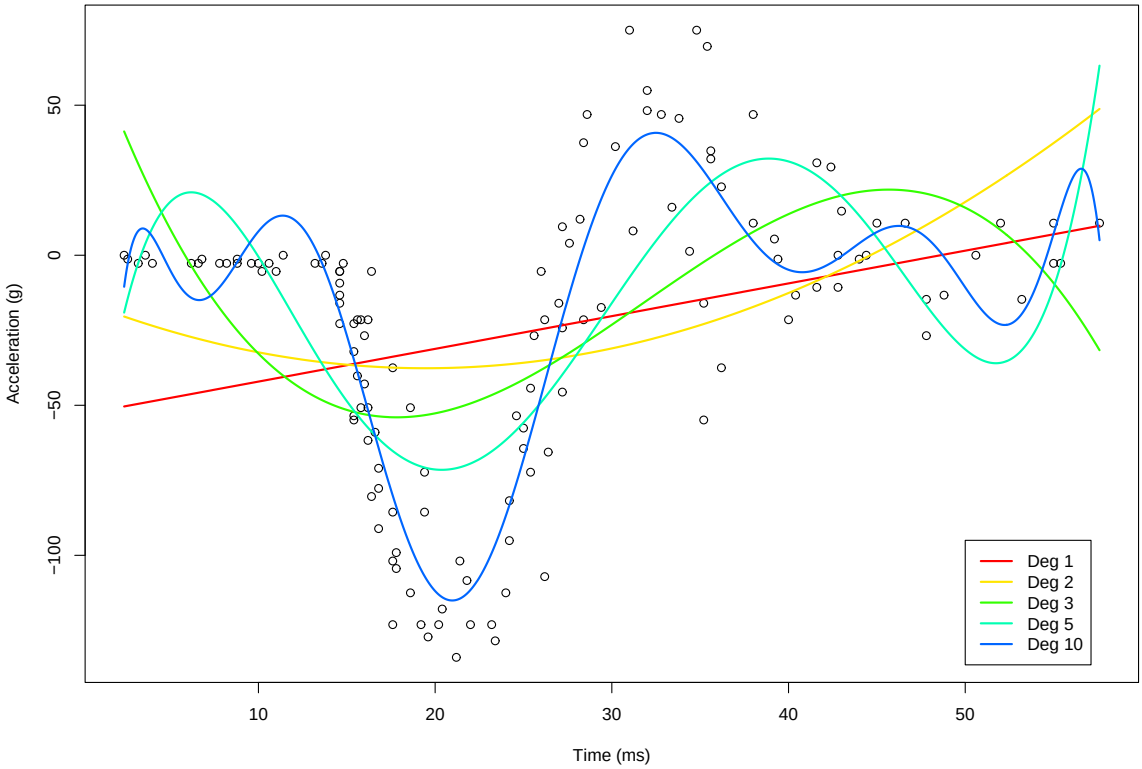


Figure 2.2: A plot of the motorcycle data with the best fitting 1st, 2nd, 3rd, 5th and 10th degree polynomials.

Having established that polynomials are not the solution to the smoothing problem, we must decide what properties we want our smooth function to exhibit. The inability of polynomials to target flexibility at those parts of the data where it is required is an unattractive feature. If we choose a low order function it may fit well to certain parts of the data, where the data appears quite smooth, but in areas where extra flexibility is required there is not enough freedom in the function. For higher order polynomials the converse is true. So a desirable property for our smooth would be to allow local flexibility while still enforcing smoothness elsewhere. We shall now describe several smoothing methods that have this property, and then look at the penalized spline method in Section 2.3.

2.2 Full Rank Smoothing Methods

In this section we will introduce three smoothing methods that can be categorized as *full rank* methods. Full rank methods are defined as methods which contain at

least as many parameters as data points. We shall see that for the example used in this chapter the full rank methods do a good job at smoothing the data. However, we should bear in mind that the computational requirements of these methods can become impractical when applied to the large multi-dimensional datasets described in Chapter 3.

2.2.1 Local Averaging

We start by looking at local averaging methods. These methods do not produce what we would normally consider smooth functions, but they are easily explained heuristically, and lead naturally into the kernel smoothing methods discussed in Section 2.2.2.

The simplest local averaging method is bin-smoothing. Here we simply split $[a, b]$ into bins and define $S(x)$ as the step function which is the average of the y in each bin. As mentioned above a dis-continuous function seems contrary to our notion of local smoothness, and gives a very crude solution to the smoothing problem.

The next step could be to introduce a running mean. Here we define the smooth as follows

$$S(x_i) = \frac{y_{i-k} + \cdots + y_i + \cdots + y_{i+k}}{2k+1}, \tag{2.9}$$

so our smooth at each data point x_i is simply the average of y taken over its k nearest neighbours on each side. On the edges of the data where there are no data on one side we simply average over the values available. The larger the value of k, the more values we will be averaging over, and so the smoother the values of $S(x_i)$ will be. This method offers an improvement over the bin-smoother in that there are no jumps between the data points. However the method is limited as we do not obtain a smooth function, only a set of smoothed values evaluated at the observed data points. Therefore there is no natural way to predict a value of the response at an unobserved data point, and some form of interpolation has to be used.

A natural extension of the running mean is the running line

$$S(x_i) = \beta_{0,i} + \beta_{1,i} x_i \tag{2.10}$$

where $\beta_{0,i}$ and $\beta_{1,i}$ are local least squares estimates. The values of $\beta_{0,i}$ and $\beta_{1,i}$ are

obtained by solving the least squares equations (2.8) for a local subset of the data,

$$\hat{\boldsymbol{\beta}}_i = (\boldsymbol{X}_i'\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i'\boldsymbol{y}_i \tag{2.11}$$

where $\boldsymbol{y}_i' = (y_{i-k}, \ldots, y_i, \ldots, y_{i+k})$ and $\boldsymbol{X}_i = [\,\mathbf{1} \vdots \boldsymbol{x}_i\,]$ with $\boldsymbol{x}_i' = (x_{i-k}, \ldots, x_i, \ldots, x_{i+k})$. Clearly the principles of the running line can be extended to running polynomials of higher degree, simply by adding columns of higher powers to the regression matrices $\boldsymbol{X}_i$; the running polynomial results. However, whichever order of polynomial is used, we do not solve the problem of finding a smooth function at unobserved data points.

2.2.2 Kernel Smoothing

Kernel smoothing follows on naturally from basic local averaging methods, but it also enables the evaluation of a smooth at unobserved data points. Instead of using the nearest neighbours at data points to determine the estimate at a particular point, all data points are considered but a weight is applied to each point which determines its influence on the estimate; this weight can be found even for unobserved values of x. A good reference for kernel smoothing is Bowman and Azzalini (1997); they propose a local linear estimator with (normal) kernel weights, as follows. In the general case, the estimate of the smooth at some point x is the intercept term of a local polynomial of order p fitted at x,

$$\hat{S}(x) = \hat{\beta}_{x,0}, \tag{2.12}$$

where $\hat{\boldsymbol{\beta}}_x' = (\hat{\beta}_{x,0}, \hat{\beta}_{x,1}, \ldots, \hat{\beta}_{x,p})$ is estimated by minimising

$$\sum_{i=1}^{n}(y_i - \beta_{x,0} - \beta_{x,1}(x - x_i) - \cdots - \beta_{x,p}(x - x_i)^p)^2 w_{x,i}, \tag{2.13}$$

where the weights, $w_{x,i}$ are known for any point x. From Generalized Least Squares theory the solution is

$$\hat{\boldsymbol{\beta}}_x = (\boldsymbol{X}_x'\boldsymbol{W}_x\boldsymbol{X}_x)^{-1}\boldsymbol{X}_x'\boldsymbol{W}_x\boldsymbol{y} \tag{2.14}$$

where

$$\boldsymbol{X}_x = \begin{bmatrix} 1 & x - x_1 & \ldots & (x - x_1)^p \\ \vdots & \vdots & & \vdots \\ 1 & x - x_n & \ldots & (x - x_n)^p \end{bmatrix}$$

and

$$\boldsymbol{W}_x = \text{diag}\{w_{x,1}, \ldots, w_{x,n}\}. \tag{2.15}$$

The weight of the i^{th} observation at the point x is obtained using a predetermined function, $K(\cdot)$, known as the *kernel*. Weights are obtained using

$$w_{x,i} = K_\lambda(x - x_i) = \lambda^{-1} K\big((x - x_i)/\lambda\big) \tag{2.16}$$

where $K(\cdot)$ is some kernel function and $\lambda > 0$ is known as the *bandwidth*. The kernel function is generally chosen to be a smooth unimodal function so that the weight attributed to each data point varies smoothly over $[a, b]$. Often a probability density function is selected, for example setting the kernel function equal to the density of a standard normal distribution, $K(x) = \phi(x)$, which means that $K_\lambda(\cdot)$ is the density for $N(0, \lambda^2)$. The bandwidth determines the influence of the data points close to the point of interest relative to those further away, with large values of λ increasing the influence of data points further away. An attractive property of local linear regression (setting $p = 1$) is that as λ becomes large the $\hat{S}(x)$ approach a linear function.

In practice the choice of kernel function is of secondary importance to the choice of bandwidth. Wand and Jones (1995) calculated asymptotic efficiency for several kernel functions and found relatively small differences between various symmetric unimodal kernels. In contrast the choice of bandwidth has a big influence on the smooth, balancing the requirement for a smooth function with the need to keep genuine features in the data. As we will see in the remainder of this chapter most smoothing methods include a parameter which performs a similar role to the bandwidth, and selection of this parameter and thus the amount of smoothing is an important problem. We will consider smoothing parameter selection in Section 2.4.

2.2.3 Smoothing Splines

In this section we introduce splines and briefly look at smoothing splines. We summarize the main results and give an example of smoothing splines using the motorcycle data. An excellent reference for smoothing splines is the book by Green and Silverman (1994).

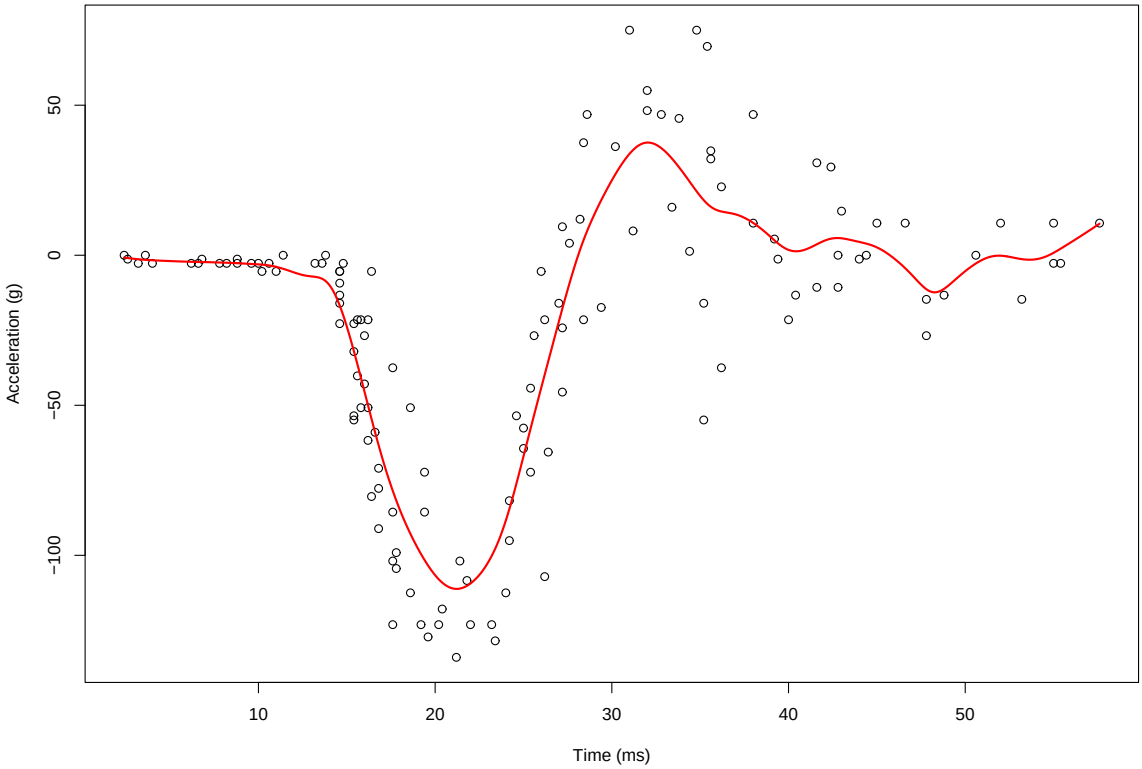


Figure 2.3: A smooth fitted to the motorcycle crash data using the `locpoly` function in `KernSmooth` library of `R`, with a normal kernel and a bandwidth = 1.45.

We begin by defining a way to measure the smoothness of a function. Looking at the polynomials in Fig. 2.2 the linear and quadratic functions seem smoother than the tenth degree polynomial, so it seems that our perception of smoothness of a function could be related to the number of turning points in the function. However, it is possible for a function to have inflexions which make it look wiggly without producing a turning point, so our preference would be for the function to be linear where possible. A key feature of linear functions is having a second derivative of zero, and this naturally leads to a measure of smoothness given by the *integrated second derivative*

$$\int_a^b \{S''(x)\}^2 \, \mathrm{d}x. \tag{2.17}$$

Strictly, the larger this measure, the rougher the function, so the measure is often referred to as a roughness penalty. Weighting our preference for smooth functions with the goodness of fit in expression (2.7) using a *smoothing parameter*, λ, we obtain

the penalized residual sum of squares

$$\text{PRSS}(S(\cdot)) = \sum_{i=1}^{n} \big(y_i - S(x_i)\big)^2 + \lambda \int_a^b \{S''(x)\}^2 \, \mathrm{d}x. \tag{2.18}$$

The smoothing parameter balances one's preference for goodness of fit with the smoothness requirement. For large values of λ the smoothness requirement becomes more important and to minimize the PRSS we would need to select a function approaching a linear function. As λ approaches zero, smoothness becomes less important and the PRSS will be minimized by a function that interpolates the data. We seek the function $\tilde{S}(\cdot)$ that minimizes (2.18) for a given value of λ. The choice of λ corresponds to the choice of the bandwidth in kernel smoothing.

At this point we take the opportunity to introduce splines, and in particular natural cubic splines before going on to fit a natural cubic spline to the motor cycle data. We begin with the definition of a spline (adapted from Green and Silverman (1994))

Definition 2.1 *Given a set of non-decreasing real numbers $t_1, \ldots, t_k \in [a, b]$, a function g defined on $[a, b]$ is a spline of degree j if g is a polynomial of degree j on each interval $(a, t_1), (t_1, t_2), \ldots, (t_{k-1}, t_k), (t_k, b)$ and the $(j-1)^{th}$ derivative of g is continuous at each of the points $t_1, \ldots, t_k$.*

The sequence $\boldsymbol{t}' = (t_1, \ldots, t_k)$ is known as a the *knot* sequence. The definition above is very general and includes several examples that we will use later on including B-splines and truncated power functions. An important special case in the context of smoothing splines are natural cubic splines

Definition 2.2 *A natural cubic spline is a spline of degree 3, with the added condition that $g''(a) = g'''(a) = g''(b) = g'''(b) = 0$.*

It can be shown that the unique minimizer of (2.18) is a natural cubic spline with a knot at each of the values $x_1, \ldots, x_n$. We will concentrate on low rank smoothing so we omit the proof of this important result which can be found in Green and Silverman (1994). A plot of a natural cubic spline fitted to the motorcycle data is shown in Fig. 2.4. The spline was fitted using the `smooth.spline` function in R, with the smoothing parameter, $\lambda = 0.6599$, selected using *Generalized Cross Validation* (see Section 2.4.2 for a discussion of smoothing parameter selection).

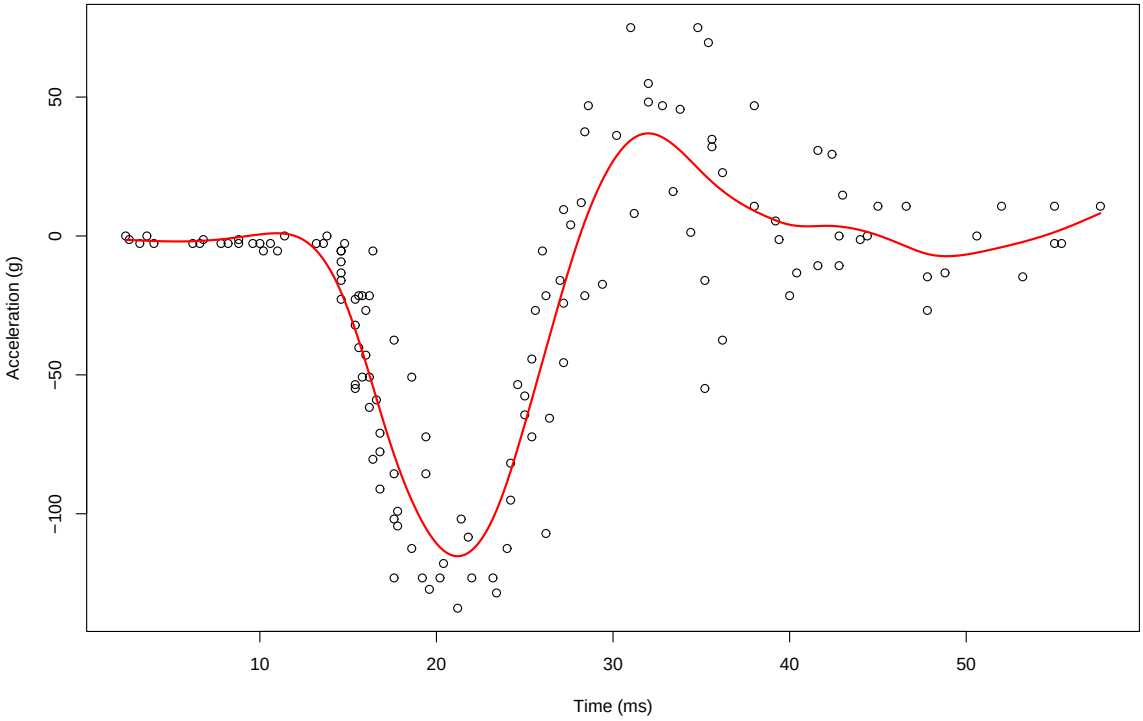


Figure 2.4: A smoothing spline fitted to the motorcycle data by GCV using the R function `smooth.spline`.

2.3 Penalized Splines

The main smoothing method we will use is P-splines. This name was first used by Eilers and Marx (1996) in reference to smoothing using a B-spline basis with a difference penalty on regression coefficients. Since then the meaning of the term P-splines has become rather ambiguous, and is often used for any regression spline setup where a penalty is used; see for example Ruppert et al. (2003) where the name P-splines is used in reference to *truncated power functions* (TPFs) with a *ridge penalty* on the regression coefficients. To avoid confusion we will use the name P-splines only in reference to the method described by Eilers and Marx (1996).

All of the methods described in this section are examples of *low rank* smoothing methods. Low rank methods are defined by the property that the model contains significantly fewer parameters than data points.

Truncated power functions (TPFs) offer an easy entrance point for semiparametric regression, as they follow on naturally from polynomial regression. In section 2.3.1 we introduce use TPFs as a tool to understanding different basis functions and penalized

likelihood. In section 2.3.2 we will concentrate on the more general P-spline regression, looking at the properties of P-spline smoothing, and the details of how a P-spline smoothing model is fitted in one dimension.

2.3.1 Truncated power functions

Truncated power functions as scatterplot smoothers were advocated by Ruppert et al. (2003). There have been doubts cast over the numerical practicalities of using TPFs; Wand (2002b), for example, states that truncated polynomials exhibit "sub-optimal numerical properties", and often transformation to a different basis is required. This was viewed as an implementational issue, which should not be considered in model formulation. Truncated polynomials do offer a nice introduction to smoothing for someone unfamiliar to the subject, due to their obvious similarity with polynomial regression. We shall view them purely as an introductory method, before moving on to P-splines.

In the introduction of Section 2.1 it was shown that polynomials were not suitable for smoothing in many cases, due to their instability when high-degree functions are used. Truncated power functions offer an alternative to increasing the degree of the polynomial by focusing flexibility where it is needed.

We take the motorcycle data as a motivating example. With reference to Fig. 2.1, we see that there are changes in the direction of the data at approximate times 13, 21, 33, and possibly 40. The basic premise of truncated polynomial curve fitting is to put hinge points or *knots* at these points. This is achieved by including truncated power functions in our smooth function which would have the form

$$S(x_i) = a_0 + a_1 x_i + \ldots + a_k x_i^k + \sum_{j=1}^{J} u_j (x_i - t_j)_+^k, \tag{2.19}$$

where $x_+ = \max(0, x)$. Writing (2.19) in matrix notation we obtain

$$S(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{a} + \boldsymbol{Z}\boldsymbol{u}, \tag{2.20}$$

Table 2.1: Knot sequences and GCV values for the linear, quadratic and cubic truncated power functions shown in Figure 2.5.

Degree	Knot sequence	GCV
Linear	14 20 33 43	623.4
Quadratic	13 14.5 17 24 36 43	563.1
Cubic	13.5 14.5 18 23 32 36	566.3

where $\boldsymbol{u}' = (u_1, \ldots, u_J)$ and

$$\boldsymbol{Z} = \begin{bmatrix} (x_1 - t_1)_+^k & \cdots & (x_1 - t_J)_+^k \\ \vdots & & \vdots \\ (x_N - t_1)_+^k & \cdots & (x_N - t_J)_+^k \end{bmatrix}. \tag{2.21}$$

The $\boldsymbol{t}' = (t_1, \ldots, t_J)$ are known as the knot positions, and they allow the targeting of flexibility at the parts of the data where it is required. With this method a smooth with as many continuous derivatives as desired can be constructed.

Figure 2.5 shows some smooths constructed from first, second and third degree truncated power functions fitted to the motorcycle data. Fitting by eye we found the knot sequences given in Table 2.1 gave reasonable results. Using *Generalized Cross Validation* (GCV) (Craven and Wahba, 1979) we see that there is little to choose between quadratic and cubic fits.

The ability to target flexibility is the first advantage of TPFs; another advantage is that they are much less volatile. Figure 2.6 shows smooths fitted to the motorcycle data. The red line in panel (a) is a seventh degree polynomial fitted to the full data set. This has the same problems as the higher degree smooths in Fig. 2.2, where artifacts of the function are being imposed on the data (for example the bumps at time 9 and time 52). Another worrying feature of polynomials is their sensitivity to specific data points. If we remove the first five data points and re-fit the polynomial (shown by the blue line), we see the smooth changes dramatically, not only at the points where the data has been removed, but over the whole domain of the function. This effect is shown more dramatically if we remove only the last point from the

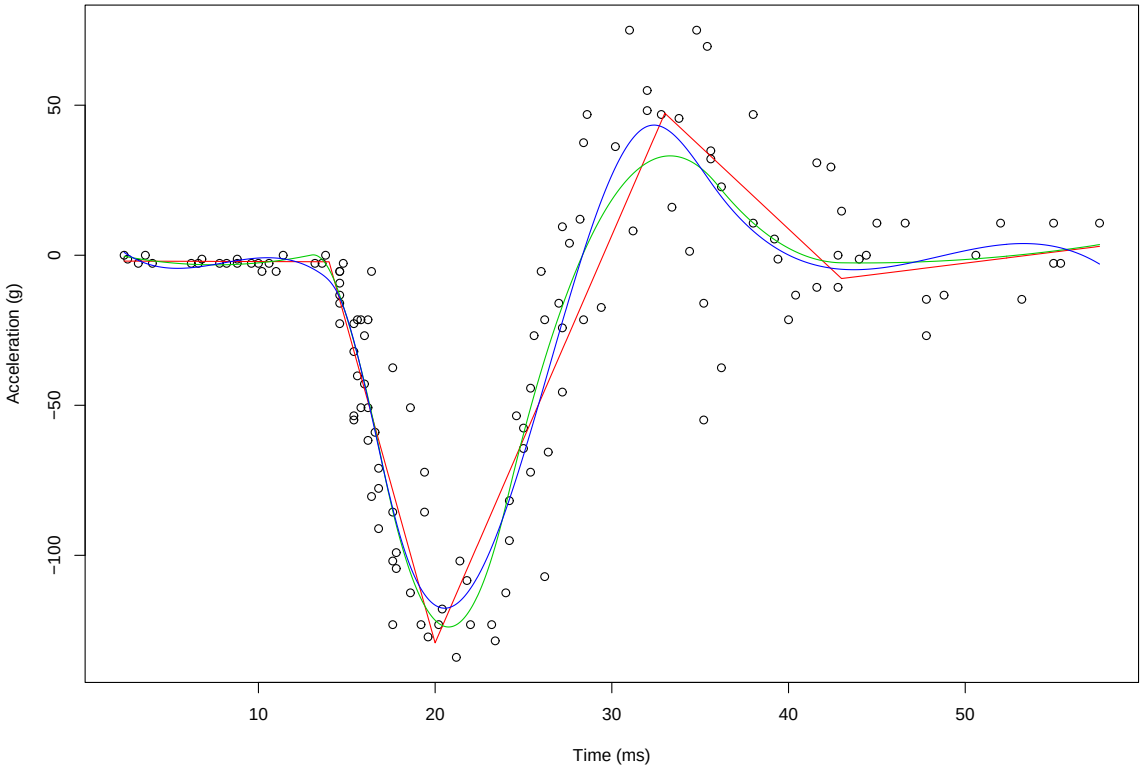
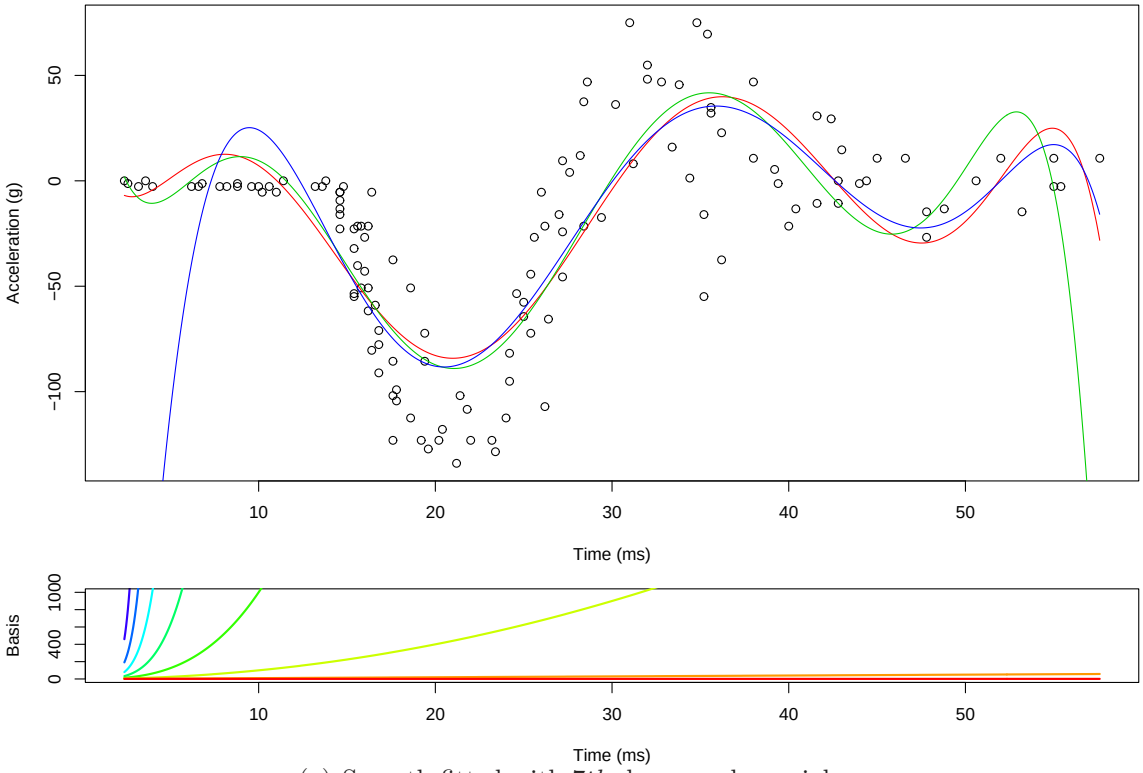


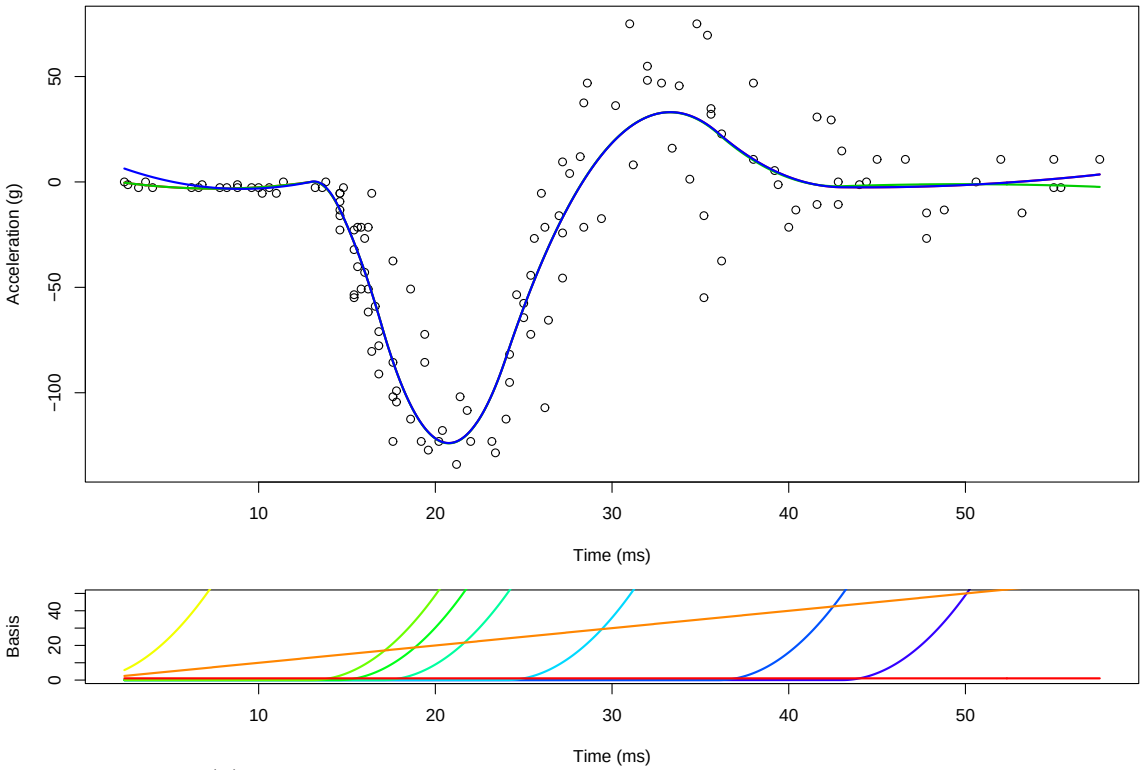
Figure 2.5: Smooth fitted to motorcycle data using truncated power functions of degree 1 (red), 2 (green) and 3 (blue).

data; we obtain the smooth shown by the green line. This sensitivity to the data is a symptom of using global basis functions. Underneath the data plot is a graph of the basis functions for the model; we see that in a polynomial regression the basis functions have support on the whole range of time. This means that a change in any of the parameters affects the fit over the whole range of the time values. The same exercise can be performed using TPFs; panel (b) shows the results from fitting second degree TPFs. We see that the green and blue lines diverge from the original fit where the data has been removed, but elsewhere the fits are almost identical; in fact the fits are so close that the red line cannot be seen behind the other two. The plot of the basis functions below shows that most of the basis functions do not offer support over the whole range of the time variable. This means if some of the coefficients are heavily influenced by a particular part of the data, other coefficients can be adjusted to maintain smoothness elsewhere in the data.

Although TPFs seem stable under changes to data, and do not impose features on the data (as with polynomials), they introduce a new problem: where to place the



(a) Smooth fitted with $7th$ degree polynomial.



(b) Smooth fitted with $2nd$ degree truncated power functions.

Figure 2.6: Smooths fitted to the motorcycle data with polynomials and truncated power functions. In each plot the red line shows the smooth fitted to the full data set, the blue line with the first five data points removed, and the green line with the last data point removed. The panel underneath each plot shows the basis for the smooth.

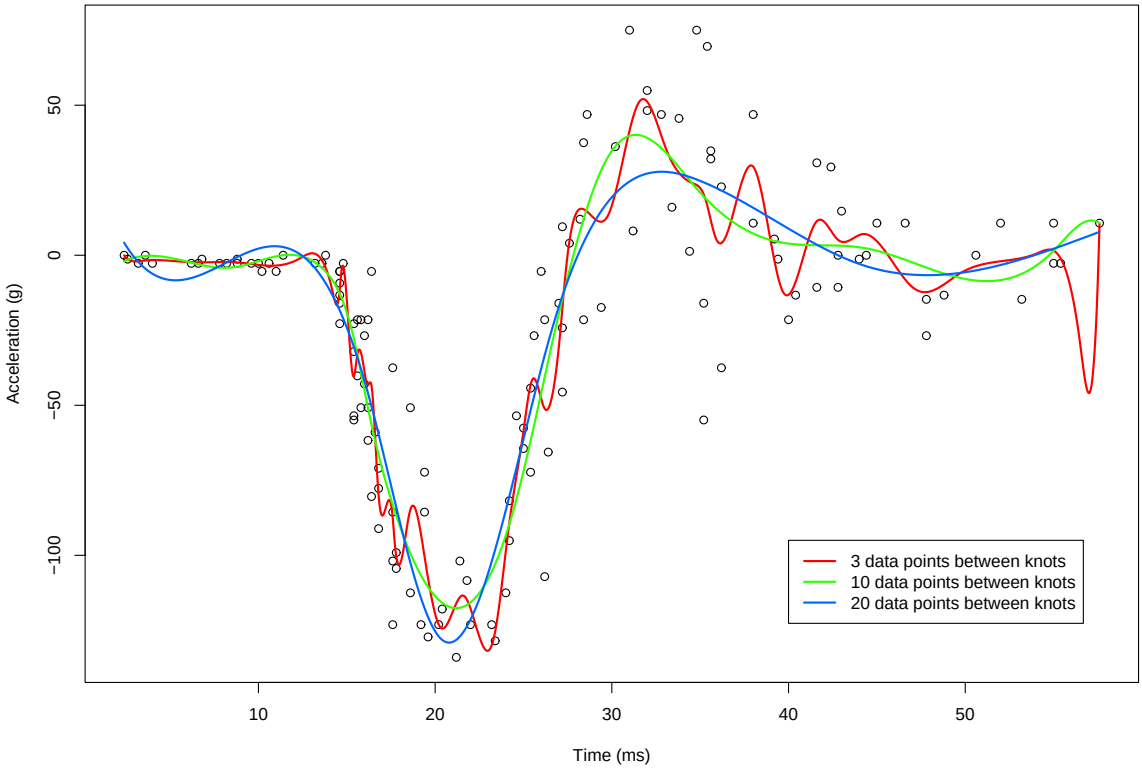


Figure 2.7: Smooths fitted to motorcycle data using penalized truncated power functions with different numbers of knots.

knots? Varying the number and position of the knots can have a dramatic effect on the fitted smooth; Fig. 2.7 shows more fits of the motorcycle data for several knot arrangements. We see that as we increase the number of knots we get gradually closer to interpolating the data, but again at some point we lose the smoothness property we desire. There are methods for automatically selecting knot positions but we will not be using such methods with our data; for a review see (Wand, 2000). Instead, we turn to penalization to enforce smoothness. The penalty is subtracted from the log likelihood to give the penalized log likelihood

$$\ell_p(\boldsymbol{a}, \boldsymbol{u}; \boldsymbol{y}) = \ell(\boldsymbol{a}, \boldsymbol{u}; \boldsymbol{y}) - \lambda \|\boldsymbol{u}\|^2. \tag{2.22}$$

This is similar to the method used for smoothing splines, but instead of penalizing the smoothness of the function directly, a ridge penalty is applied to the coefficients of the TPFs. For the normal distribution maximizing the penalized likelihood is equivalent to minimizing the penalized residual sum of squares

$$\text{PRSS}(\boldsymbol{a}) = (\boldsymbol{y} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{a}})'(\boldsymbol{y} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{a}}) + \lambda \|\boldsymbol{u}\|^2, \tag{2.23}$$

where $\tilde{\boldsymbol{X}} = [\,\boldsymbol{X} \vdots \boldsymbol{Z}\,]$, $\tilde{\boldsymbol{a}}' = [\,\boldsymbol{a}' \vdots \boldsymbol{u}'\,]$ and λ is the smoothing parameter. For a given value of λ minimization of (2.23) leads to the penalized least squares solution

$$(\tilde{\boldsymbol{X}}'\tilde{\boldsymbol{X}} + \boldsymbol{P})\tilde{\boldsymbol{a}} = \tilde{\boldsymbol{X}}'\boldsymbol{y}, \tag{2.24}$$

where $\boldsymbol{P}$ is the penalty matrix

$$\boldsymbol{P} = \lambda \begin{bmatrix} \mathbf{0}_K & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_J \end{bmatrix}, \tag{2.25}$$

where $\mathbf{0}_K$ is a $K \times K$ matrix of zeros and $\boldsymbol{I}_J$ is a $J \times J$ identity matrix. With a penalty the placement of the knots becomes less important provided that sufficient knots are used, as any excess flexibility in the fitted smooth is removed by the penalty. We will follow the simple knot placement strategy suggested by (Eilers and Marx, 1996) and space the knots evenly across the x_i. Figure 2.8 shows a smooth of penalized truncated power functions fitted to the motorcycle data. We see that the penalization successfully reintroduces smoothness into the model even with a large number of knots. As before the smoothing parameter was chosen by minimizing the GCV criterion.

2.3.2 P-splines

In the previous section we discussed smoothing using penalized likelihood with TPFs as the regression basis and a ridge penalty on the regression coefficients. In this section we describe the method of Eilers and Marx (1996) who used a B-spline basis and a difference penalty on adjacent coefficients. With P-splines, as this combination was named, the basis functions and penalty give the regression coefficients a natural and simple interpretation and it can be easily understood how the penalty enforces smoothness. This together with a relatively simple fitting procedure, and other attractive properties make P-splines an effective and transparent scatterplot smoother.

We begin by describing the B-spline basis in detail. A B-spline is a piecewise function constructed from polynomials, and is completely specified by a sequence of knots $\boldsymbol{t}$, the degree, k, of the polynomials used, and its position in the basis j. The definition is given by de Boor (2001). B-splines are best explained graphically,

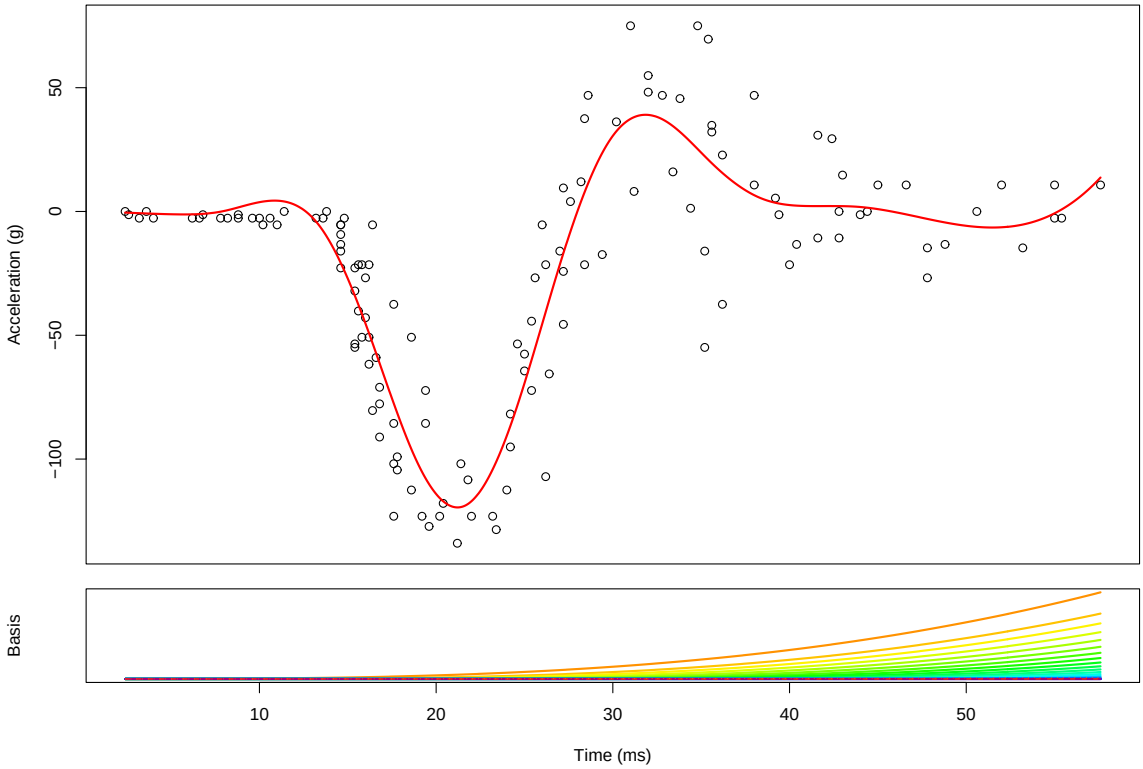


Figure 2.8: Smooth fitted to motorcycle data using penalized truncated power functions, with twenty-three evenly spaced knots.

and Fig. 2.9(a) shows examples of zero, first, second and third degree B-splines on an evenly spaced knot sequence. The zero degree spline is a discontinuous function which takes the value 1 between two knots (in this case t_1 and t_2) and 0 elsewhere. The first degree spline is a continuous function which takes positive values over a range of three consecutive knots (in this case between t_3 and t_5) and 0 elsewhere; it is constructed from two linear pieces between the knots, which gives rise to a discontinuity in its first derivative at each knot. The second degree spline is a piecewise quadratic function which is positive over a range spanned by four consecutive knots (in this case between t_6 and t_9) and zero elsewhere. It has a continuous first derivative, but has discontinuities in its second derivative at the knot positions. The third degree spline is also shown in Fig. 2.9(a), but from the descriptions so far we shall assume the properties of a k^{th} degree B-spline can be obtain by induction, and are as follows (from Eilers and Marx, 1996)

- it consists of $k+1$ polynomial pieces, each of degree k;

- the polynomial pieces join at k inner knots;
- at the joining points, derivatives up to order $k-1$ are continuous;
- the B-spline is positive on the domain spanned by $k+2$ knots; everywhere else it is zero;
- except at the boundaries, it overlaps with $2k$ polynomial pieces of its neighbours;
- at a given x, $k+1$ B-splines are nonzero.

The last two properties refer to a set or basis of B-splines and can be verified by looking at the plots in Fig. 2.9(b). B-splines can be evaluated using a recursive algorithm, which relates a B-spline of degree k to the difference of two B-splines of degree $k-1$, see de Boor (2001). Eilers and Marx (1996) recommend the use of evenly spaced knots, and provide a simplified algorithm for the evaluation of B-splines on an evenly spaced knot sequence. Once the B-splines have been evaluated, they can be used as a basis for regression in exactly the same way as polynomials or TPFs. Without a penalty we would be back to the familiar problem of how to choose the number and position of the knots. This problem is solved by using a rich basis of B-splines, that would lead to under smoothing of the data. Smoothness is then imposed by placing a difference penalty on the coefficients of adjacent B-splines which enters though the log-likelihood in the same way as (2.22), giving the penalized log-likelihood

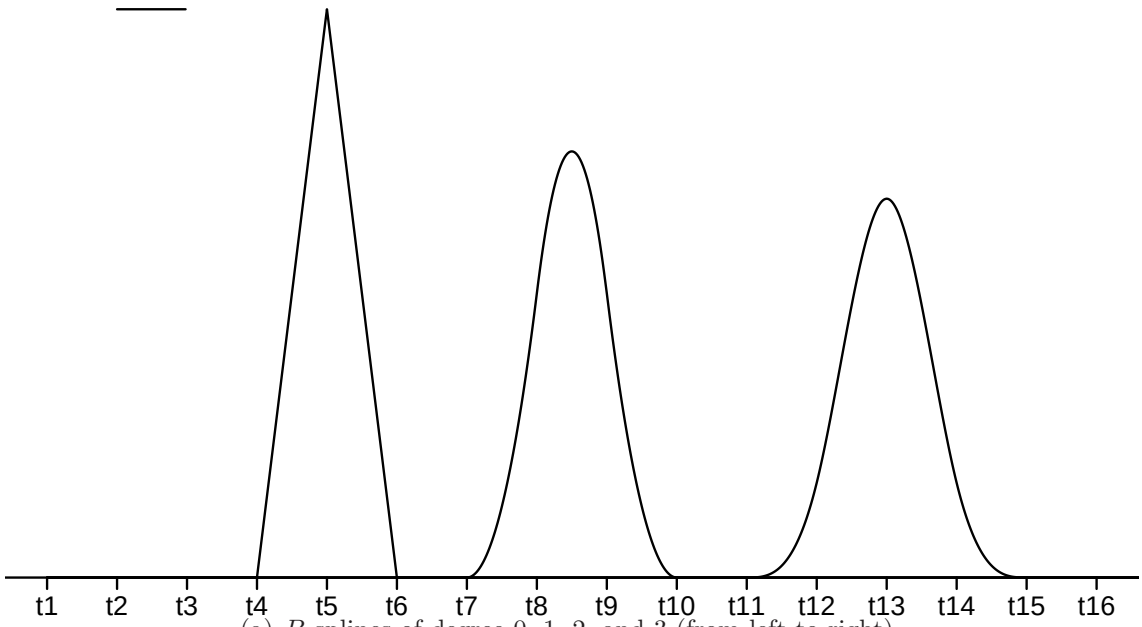
$$\ell_p(\boldsymbol{\theta};\boldsymbol{y}) = \ell(\boldsymbol{\theta};\boldsymbol{y}) - \lambda \sum_{i=d+1}^{c} (\Delta^d \theta_i)^2, \tag{2.26}$$

for the difference operator Δ where $\Delta\theta_i = \theta_i - \theta_{i-1}$, and $\Delta^{d+1}\theta_i = \Delta(\Delta^d\theta_i)$. As before, in the Normal case maximizing the penalized log likelihood yields the penalized least squares solutions

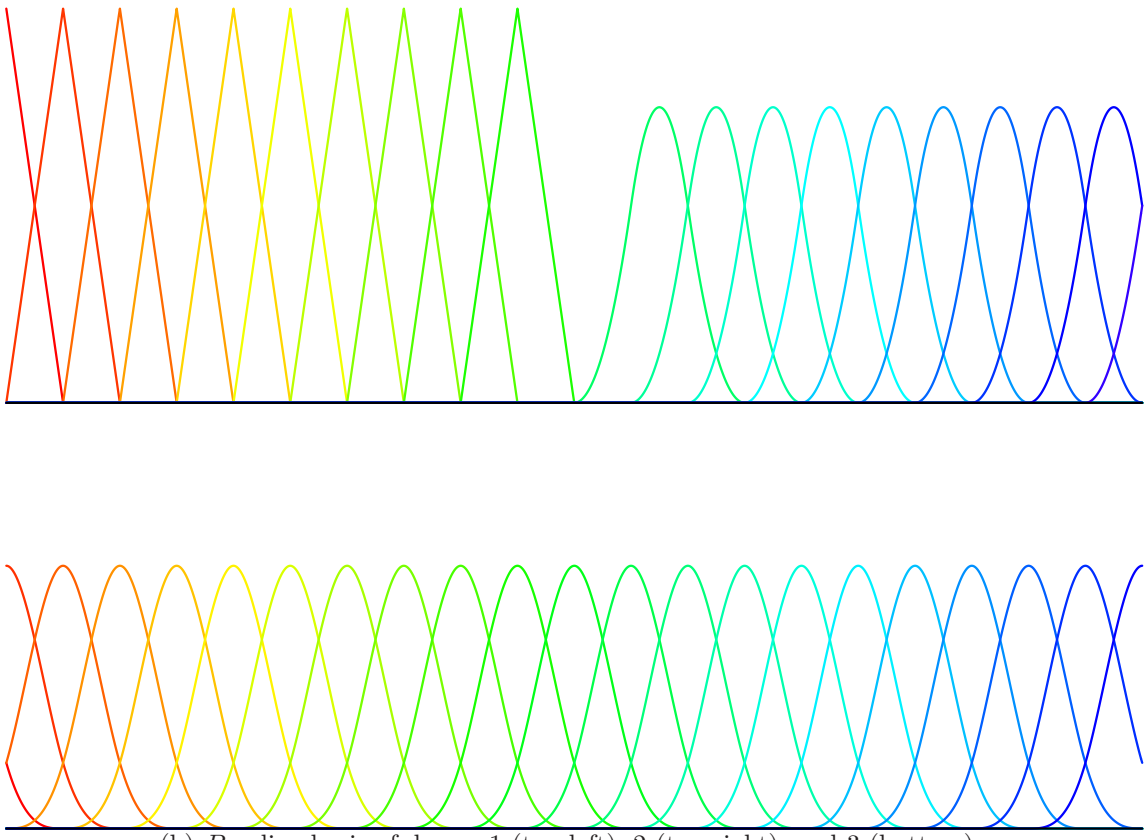
$$(\boldsymbol{B}'\boldsymbol{B} + \boldsymbol{P})\boldsymbol{\theta} = \boldsymbol{B}'\boldsymbol{y}, \tag{2.27}$$

where

$$\boldsymbol{B} = \begin{bmatrix} b_1(x_1) & \dots & b_k(x_1) \\ \vdots & & \vdots \\ b_1(x_n) & \dots & b_k(x_n) \end{bmatrix} \tag{2.28}$$



(a) B-splines of degree 0, 1, 2, and 3 (from left to right)



(b) B-spline basis of degree 1 (top left), 2 (top right), and 3 (bottom)

Figure 2.9: Graphs of B-splines with regularly spaced knots.

where $b_j(x_i)$ is the $j\,th$ B-spline in the basis evaluated at x_i. In this case the penalty matrix, $\boldsymbol{P} = \lambda \boldsymbol{D}_d' \boldsymbol{D}_d$, where $\boldsymbol{D}_d$ is a difference matrix of order d, for example

$$\boldsymbol{D}_2 = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \end{bmatrix}. \tag{2.29}$$

In some ways the P-spline method has complicated the choices of settings for the model. As well as the position of the knots, and the degree of the splines, we also have to consider the order of the penalty (the order of the differences used in the penalty). As already mentioned, Eilers and Marx (1996) recommend using an evenly spaced sequence of knots obtained by dividing $[a, b]$ into m equal intervals, giving $m+1$ knots. In order to specify the full basis $m+2k+1$ knots are required, resulting in $c = m+k$ B-splines. Instead, we prefer to choose directly the position of one knot

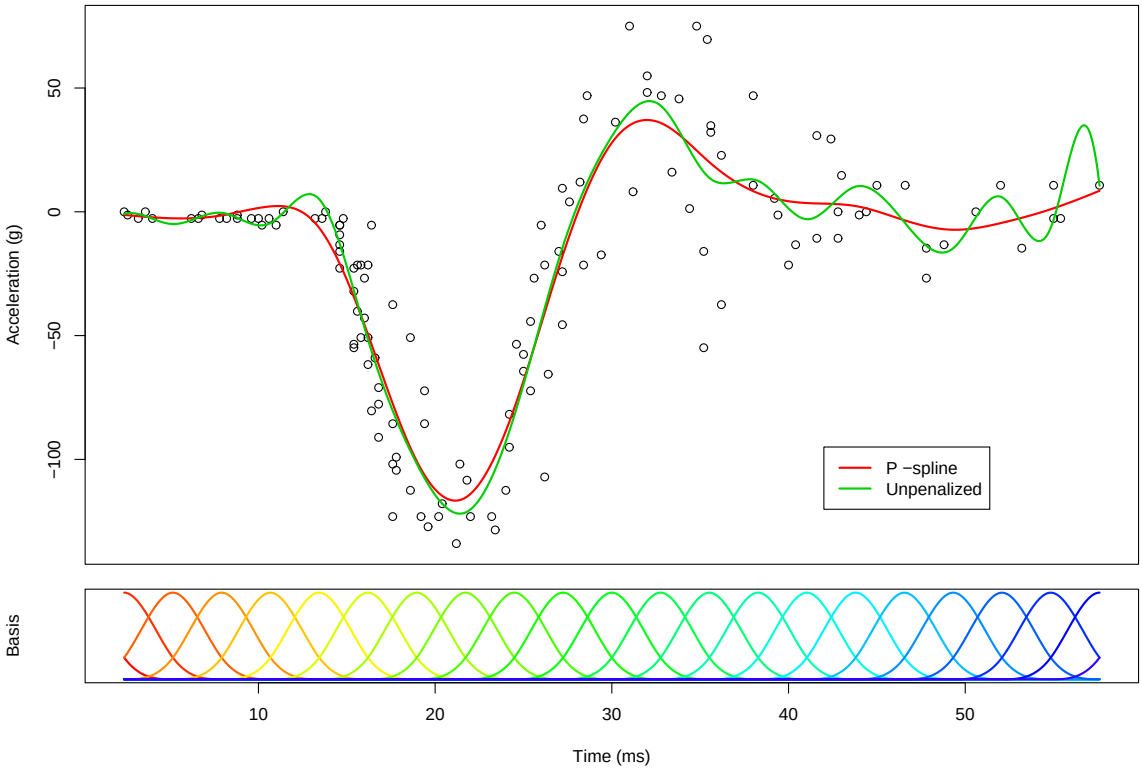


Figure 2.10: Smooth fitted to motorcycle data using P-splines. The red line shows the P-spline fit with the smoothing parameter selected by GCV and the green line shows an unpenalized model fitted with the same basis.

and the distance between knots; this makes it easier to compare different models if, for example, we obtain more data points for the data set under discussion. We will tend to position a knot at $x_{\max}$ or $x_{\min}$ depending on where the data set might be expanded; for example, with mortality data we would expect to obtain more data as time goes by, so it makes sense to position a knot at the most recent data point. The distance between the knots should not matter provided it is sufficiently small, as any excess flexibility will be removed by the penalty. The order of the penalty becomes important for large values of the smoothing parameter, when it determines the polynomial limit of the smooth

$$\lim_{\lambda \to \infty} S(x) = p_{d-1}(x), \; d \geq k - 1, \tag{2.30}$$

as shown by Eilers and Marx (1996). A P-spline model fitted to the motorcycle data is shown in Fig 2.10, using twenty-three evenly spaced B-splines with the smoothing parameter selected by GCV; the plot also shows the unpenalized B-spline model using the same basis.

In the one dimensional case there seems very little to choose between smoothing methods. Low rank smoothers such as P-splines and TPFs offer some computational advantages, but this is of little practical importance when using modern computers. There is little to choose between the methods in terms of complexity; it is argued that TPFs follow on naturally from polynomial regression but the inclusion of the penalty requires an extra level understanding. However once an understanding of penalties and bases is obtained, P-splines offer a more intuitive method. Given a scatterplot and a set of knot positions, one could guess with relative accuracy what the values of the P-spline coefficients would be; it is an attractive property that the coefficients have a "physical" interpretation. One very important advantage that P-splines offer are their natural application in higher dimensional situations; in Chapter 3 we will see how the principles described in one dimension are almost identical in higher dimensions. This combined with good numerical properties and some attractive computational advantages that come from the construction of the basis in multiple dimensions mean that P-splines should be given careful consideration as a scatterplot smoother.

2.4 Smoothing parameter selection

All of the smoothing methods in sections 2.2 and 2.3 have involved the use of a smoothing parameter. The smoothing parameter is normally considered as a hyper-parameter, with the other parameters chosen by a penalized likelihood conditional on this parameter, so changing the smoothing parameter gives a different penalized likelihood. This is a similar problem to that of whether or not to included a particular parameter in a regular parametric model, say, or whether or not to included a cubic term in a polynomial regression. The only difference is that in a parametric model we are selecting from a discrete set of models, whereas the selection of a smoothing parameter amounts to selecting from a continuous set of models. In Section 2.4.1 we discuss the effective dimension of a model, and follow this by a discussion of model selection criteria in section 2.4.2.

2.4.1 Effective Dimension of a Model

We will begin the discussion of model selection with a discussion of model dimension. Determining the dimension of a model plays an important role in model selection: a number of model selection criteria depend directly on its calculation, and it can be used to compare the models selected under different criteria. A full discussion of the effective dimension of a model is given by Hastie and Tibshirani (1990, chap. 3).

Finding the dimension, or *degrees of freedom*, of the model is trivial in simple regression models; provided the basis for the regression has full column rank, then the dimension of the model is simply the number of columns in the regression basis matrix. So a polynomial regression of degree p would have dimension $p+1$, which coincides with the the trace of the matrix $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$, since

$$\begin{aligned} \mathrm{tr}(\boldsymbol{H}) &= \mathrm{tr}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}') \\ &= \mathrm{tr}((\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{X}) \\ &= \mathrm{tr}(\boldsymbol{I}_{p+1}) \\ &= p+1. \end{aligned} \tag{2.31}$$

Let $\hat{\boldsymbol{y}}$ denote the fitted values of $\boldsymbol{y}$. Then the matrix $\boldsymbol{H}$ is often referred to as the "Hat" matrix since

$$\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}. \tag{2.32}$$

Conveniently, using this method to determine the dimension of the model allows us to generalize to any model for which the fitted values can be written in the form of (2.32). For smoothing models we will use the term *effective dimensions*, d_e, for the effective number of free parameters in the model. For example, in the P-spline model the coefficients of the B-splines are restricted by the penalty, so simply using the number of columns of the regression matrix as the dimension of the model would be misleading. However, from (2.27) we see that the Hat matrix for the P-spline model is given by

$$\boldsymbol{H} = \boldsymbol{B}(\boldsymbol{B}'\boldsymbol{B} + \boldsymbol{P})^{-1}\boldsymbol{B}', \tag{2.33}$$

and so the effective dimension of a P-spline model can be calculated efficiently by

$$\begin{aligned} d_e &= \operatorname{tr}\left\{(\boldsymbol{B}'\boldsymbol{B} + \boldsymbol{P})^{-1}\boldsymbol{B}'\boldsymbol{B}\right\} \\ &= c - \operatorname{tr}\left\{(\boldsymbol{B}'\boldsymbol{B} + \boldsymbol{P})^{-1}\boldsymbol{P}\right\}. \end{aligned} \tag{2.34}$$

where c is the number of columns in $\boldsymbol{B}$. This second form of the effective dimension shows how the penalty reduces the degrees of freedom available from the regression matrix $\boldsymbol{B}$.

2.4.2 Model Selection Criteria

In parametric models one is often faced with the problem of choosing which parameters should be included in the final model. Commonly, this will require the choice of a particular combination of explanatory variables and their interactions from a finite set of potential explanatory variables. The model is often chosen by starting from the null model, with just a mean term, and sequentially adding parameters, or starting from the saturated model, with all the parameters, and sequentially removing parameters. At each stage the significance of the added (or removed) parameters can be tested using an F-test or a likelihood ratio test, and parameters are added (or removed) until none of the remaining parameters under consideration is found to be significant. In

this sequential procedure, at each stage the models under consideration are *nested*, in that the parameters in one of the models are a subset of the parameters in the other model.

However, it may turn out that we are interested in comparing models which are not nested. The F-test and likelihood ratio tests are no longer available because we are no longer simply testing for the significance of an extra parameter. One possible solution is to use a goodness of fit test; for the Normal models discussed so far this would lead to the selection of the model that minimized the residual sum of squares. However, goodness of fit can always be improved by adding extra parameters, so using goodness of fit as a model selection criteria naturally favours larger models. To compensate for this, model selection criteria often take the form

$$-2\ell(\boldsymbol{y};\boldsymbol{\theta}) + w\,p, \tag{2.35}$$

where p is the number of parameters in the model, the preferred model being the one that minimizes (2.35). The first component measures the goodness of fit (this turns out to be the RSS in the normal case), the second component penalizes model complexity, and w weights our preference for parsimony over goodness of fit. Given a model selection criterion of the form (2.35), model selection then turns on the choice of the weight w. Table 2.2 gives two of the best known values of w: for the Akaike information criterion (AIC) (Akaike, 1971) $w = 2$, and for the Bayesian information criterion (BIC) (Schwarz, 1978) $w = \log n$. The AIC can be derived by minimizing the Kullback-Leibler distance, which measures the expected difference between the true model and some other model. The AIC has been found to be biased in certain situations, and various corrections have been made to deal with its short-comings, including the modified AIC of Hurvich and Tsai (1989) which is designed to correct for bias in complex models with small samples. Using a Bayesian argument BIC amounts to choosing the model that is *a posteriori* most probable under some general assumptions. An important point is that the BIC will have a stronger preference for simple models compared to those chosen by AIC for any sample size greater than seven.

In the context of smoothing we are not interested in selecting particular explanatory variables, but in the overall level of smoothing. In the previous section we showed

in (2.34) how to calculate the effective dimension for a linear smooth, so for smoothing models we evaluate the selection criterion by replacing p in (2.35) with d_e. The level of smoothing is then determined by minimizing the model selection criterion with respect to the smoothing parameter, and so model selection amounts to a constrained optimization problem (constrained by the requirement that the smoothing parameter is positive), which can be solved using a quasi-newton method with finite approximation to the derivatives. As we will see in chapter 3, in higher dimensions we may need to optimize over several smoothing parameters, which can become quite time consuming for large models; a potential solution to this problem (Wood, 2007) is to differentiate the scoring algorithm to get a better informed optimization procedure.

Two other model selection criteria that are used in smoothing are *cross validation* (CV) and *generalized cross validation* (GCV); see Hastie and Tibshirani (1990, pp 42–52). These criteria aim to minimize prediction error for unobserved data points, and employ a "leave one out" strategy to achieve this. More precisely we minimize the average predictive squared error

$$\text{PSE} = \frac{1}{n}\sum_{i=1}^{n} E\left\{y_i^* - \hat{S}_\lambda(x_i)\right\}^2 \tag{2.36}$$

over λ, where y_i^* is a fresh observation at x_i and $\hat{S}_\lambda(x_i)$ is the predicted smooth value at x_i.

In general there is no selection criterion that will universally out-perform the rest. In most cases the practitioner needs to make a judgement based on the results and how the results will be used. If time allows, simulation studies could be used determine the best performing criterion for a given situation. Our experience is that for the mortality data in the one-dimensional case the AIC will perform better, but in the two-dimensional case AIC tends to under-smooth the surface, and use of BIC is a better choice. In two dimensions we will fit models with several hundred parameters so the tendency for AIC to under-smooth is a real danger; further, with a sample size of several thousand, $\log n \gg 2$, so BIC will produce stiffer fits than AIC.

Table 2.2: Weights used for two model selection criteria of the form (2.35)

Model selection criteria	Weight, w
AIC	2
BIC	$\log n$

2.5 Mixed Models

2.5.1 Introduction to Mixed Models

The mixed model is an extension of the basic linear model, in which random effects are added to the linear predictor. The most obvious use for these type of models is in longitudinal studies in which we observe repeated measures on the same subject. A good example is the pig-weight data given on page 92 of Ruppert et al. (2003), in which forty-nine pigs have their weights measured weekly for nine weeks, and we are interested in how the weight of the pigs changes over time. We need to take into account the effect of each individual, so one might consider the following model

$$y_{ij} = \beta_0 + x_j\beta_1 + u_i + \epsilon_{ij}, \tag{2.37}$$

where y_{ij} is the weight of the i th pig in week j, β_0 is the population mean, β_1 is the coefficient of the time variable, and x_j is the week index, u_i is fixed effect which alters the intercept for each pig, and ϵ_{ij} is random noise on each measurement. Clearly this contains as a special case the simple linear regression model, if we set all the $u_i = 0$. The full model has two shortcomings. Firstly, it is not identifiable, so a location constraint would be required on $\boldsymbol{u}$. Secondly, even with the location constraint on the u_i, a parameter for each pig would place too much importance on the specific sample under analysis. Alternatively, we could assume that the pigs in the sample come from a population in which the individual pig effects, u_i, come from a normal distribution; in the notation of Ruppert et al. (2003) we have

$$\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \sigma_u^2\boldsymbol{I}). \tag{2.38}$$

The main parameters of interest are $\boldsymbol{\beta}$ and σ_u^2, but we may also be interested in the values $\boldsymbol{u}$. Under this model the u_i are no longer free, as they are constrained by the distributional assumption. So with a normal distribution large values of u_i would not be expected as:

$$P(|u_i| > 1.96\sigma_u) = 0.05. \tag{2.39}$$

As stated by Ruppert et al. (2003), "(the random effect) takes into account the randomness due to other samples of pigs". As we will see later the constraints on the parameters are similar to those imposed under penalized likelihood, and using this connection we will see that smoothing models can be fitted in the mixed model framework.

Equations (2.37) and (2.38) form a simple example of a mixed model. In general we have a model of the form

$$\boldsymbol{y} = \boldsymbol{X\beta} + \sum_{i=1}^{r} \boldsymbol{Z}_i \boldsymbol{u}_i + \boldsymbol{\epsilon}, \tag{2.40}$$

where, using the terminology of Searle et al. (1992), $\boldsymbol{\beta}$ is a fixed effects vector, and $\boldsymbol{u}_i$ is a random vector representing all the levels of the i^{th} of r factors. The random effects are generally assumed to be multivariate normal, which leads to the joint distribution for $\boldsymbol{y}$ and $\boldsymbol{u}$

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{u} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{X\beta} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{G} & \boldsymbol{ZR} \\ \boldsymbol{RZ}' & \boldsymbol{R} \end{bmatrix} \right), \tag{2.41}$$

where $\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{Z}_1 & \boldsymbol{Z}_2 & \dots & \boldsymbol{Z}_r \end{bmatrix}$. The covariance matrices $\boldsymbol{G} = \boldsymbol{G}_\gamma$ and $\boldsymbol{R} = \boldsymbol{R}_\delta$ are functions of the unknown variance parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$. For the pig weight example $r = 1$ and $\boldsymbol{u}_1$ is a vector of length 49 (one level for each pig), with $\boldsymbol{G} = \sigma_\epsilon^2 \boldsymbol{I}$ and $\boldsymbol{R} = \sigma_u^2 \boldsymbol{I}$, so $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ are the scalars σ_ϵ^2 and σ_u^2 respectively.

We estimate $\boldsymbol{\beta}$ and predict $\boldsymbol{u}$ using their *best linear unbiased predictors* (BLUPs) (see Robinson (1991)). The BLUPs are defined to be the estimates which are linear in the data, $\boldsymbol{y}$, and minimize the mean squared error:

$$E(\|(\boldsymbol{X}\tilde{\boldsymbol{\beta}} + \boldsymbol{Z}\tilde{\boldsymbol{u}}) - (\boldsymbol{X\beta} + \boldsymbol{Zu})\|^2), \tag{2.42}$$

subject to the unbiasedness condition

$$E(\boldsymbol{u}) = E(\tilde{\boldsymbol{u}}). \tag{2.43}$$

As remarked by Robinson (1991), it should be noted that (2.43) is different from the usual definition of unbiasedness

$$E(\tilde{\boldsymbol{u}}|\boldsymbol{u}) = \boldsymbol{u}. \tag{2.44}$$

It can be shown that the BLUP estimates satisfy the following set of equations, known as the mixed model equations (see Searle et al. (1992))

$$\begin{bmatrix} \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{X} & \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{Z} \\ \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{X} & \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{Z} + \boldsymbol{G}^{-1} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\beta}} \\ \tilde{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{y} \\ \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{y} \end{bmatrix}, \tag{2.45}$$

which yield the solutions (see Robinson (1991))

$$\begin{aligned} \tilde{\boldsymbol{\beta}} &= (\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{y} \\ \tilde{\boldsymbol{u}} &= \boldsymbol{G}\boldsymbol{Z}'\boldsymbol{V}^{-1}(\boldsymbol{y} - \boldsymbol{X}\tilde{\boldsymbol{\beta}}) \end{aligned} \tag{2.46}$$

where $\boldsymbol{V} = \boldsymbol{R} + \boldsymbol{Z}\boldsymbol{G}\boldsymbol{Z}'$. The variance parameters are estimated by maximizing either the log likelihood (ML) or the residual log likelihood (REML)

$$\ell_{ML}(\boldsymbol{\gamma}, \boldsymbol{\delta}) = -\frac{1}{2}\log|\boldsymbol{V}| - \frac{1}{2}\boldsymbol{y}'(\boldsymbol{V}^{-1} - \boldsymbol{V}^{-1}\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{V}^{-1})\boldsymbol{y} \tag{2.47}$$

$$\ell_{REML}(\boldsymbol{\gamma}, \boldsymbol{\delta}) = \ell_{ML}(\boldsymbol{\gamma}, \boldsymbol{\delta}) - \frac{1}{2}\log|\boldsymbol{X}'\boldsymbol{V}^{-1}\boldsymbol{X}|. \tag{2.48}$$

Searle et al. (1992) give examples of some models where closed form solutions for the ML and REML estimates can be found. Wand (2002a) shows how to find the information matrix for the general variance components model where

$$\boldsymbol{V}_{\boldsymbol{\theta}} = \sum_{i=1}^{m} \theta_i \boldsymbol{K}_i \tag{2.49}$$

for known $\boldsymbol{K}_i$, which allows the use of the Newton-Raphson algorithm to find solutions numerically. For more complicated variance specifications more general optimization techniques are required. Experience has shown that the "L-BGFS-B" or "Nelder-Mead" algorithms implemented in the `optim` function in R are normally suitable.

2.5.2 Equivalent Bases

In order to establish the connection between mixed models and smoothing it is helpful to be familiar with changes in parametrisation. In a linear model a change of param-

eterization can be obtained by finding another matrix which spans the column space of the regression matrix. This is easily achieved using the following theorem,

Theorem 2.1 *For any $m \times n$ matrix, $\boldsymbol{A}$, of full column rank, and any $n \times n$ matrix, $\boldsymbol{L}$, of full rank, the matrix $\boldsymbol{AL}$ has full column rank, and $\boldsymbol{AL}$ and $\boldsymbol{A}$ span the same column space, $\mathcal{C}(\boldsymbol{A}) = \mathcal{C}(\boldsymbol{AL})$.*

The proof of this can be deduced from more general results given in Harville (1997). This theorem can be used to re-parameterize some of the models discussed so far. Suppose we have a linear regression model with a basis given by the regression matrix $\boldsymbol{X}$ with a corresponding set of coefficients $\boldsymbol{\alpha}$, then a re-parameterization can be found forming the products $\breve{\boldsymbol{X}} = \boldsymbol{XL}$ and $\breve{\boldsymbol{\alpha}} = \boldsymbol{L}^{-1}\boldsymbol{\alpha}$ with some appropriately dimensioned non singular matrix $\boldsymbol{L}$. Clearly the two parameterizations are equivalent as $\breve{\boldsymbol{X}}\breve{\boldsymbol{\alpha}} = \boldsymbol{XLL}^{-1}\boldsymbol{\alpha} = \boldsymbol{X\alpha}$. Within the framework of penalised regression we would also need to make an appropriate adjustment to the penalty, so taking $\breve{\boldsymbol{P}} = \boldsymbol{L}'\boldsymbol{PL}$ we see the penalised residual sum of squares remains unchanged

$$\begin{aligned}
\text{PRSS}(\breve{\boldsymbol{\alpha}}) &= (\boldsymbol{y} - \breve{\boldsymbol{X}}\breve{\boldsymbol{\alpha}})'(\boldsymbol{y} - \breve{\boldsymbol{X}}\breve{\boldsymbol{\alpha}}) + \breve{\boldsymbol{\alpha}}'\breve{\boldsymbol{P}}\breve{\boldsymbol{\alpha}} \\
&= (\boldsymbol{y} - \boldsymbol{XLL}^{-1}\boldsymbol{\alpha})'(\boldsymbol{y} - \boldsymbol{XLL}^{-1}\boldsymbol{\alpha}) + \boldsymbol{\alpha}'(\boldsymbol{L}')^{-1}\boldsymbol{L}'\boldsymbol{PLL}^{-1}\boldsymbol{\alpha} \qquad (2.50) \\
&= (\boldsymbol{y} - \boldsymbol{X\alpha})'(\boldsymbol{y} - \boldsymbol{X\alpha}) + \boldsymbol{\alpha}'\boldsymbol{P\alpha}.
\end{aligned}$$

If two regression bases $\boldsymbol{X}$ and $\breve{\boldsymbol{X}}$ are equivalent, then there exists $\boldsymbol{L}$, and in this case $\boldsymbol{L} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\breve{\boldsymbol{X}}$.

2.5.3 Smoothers as Mixed Models

The connection between mixed models and smoothing models has long been established. Green (1985) is an early reference to the idea of splitting a trend into fixed and random effects. The book by Ruppert et al. (2003) gives a thorough exploration of the mixed model approach and an extensive bibliography. Re-expressing (2.24) in terms of its partitions we obtain

$$\begin{bmatrix} \boldsymbol{X}'\boldsymbol{X} & \boldsymbol{X}'\boldsymbol{Z} \\ \boldsymbol{Z}'\boldsymbol{X} & \boldsymbol{Z}'\boldsymbol{Z} + \lambda\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}'\boldsymbol{y} \\ \boldsymbol{Z}'\boldsymbol{y} \end{bmatrix}. \qquad (2.51)$$

Setting $\lambda = \sigma_\epsilon^2/\sigma_u^2$ and rearranging we obtain

$$\begin{bmatrix} \boldsymbol{X}'(\sigma_u^2\boldsymbol{I})^{-1}\boldsymbol{X} & \boldsymbol{X}'(\sigma_u^2\boldsymbol{I})^{-1}\boldsymbol{Z} \\ \boldsymbol{Z}'(\sigma_u^2\boldsymbol{I})^{-1}\boldsymbol{X} & \boldsymbol{Z}'(\sigma_u^2\boldsymbol{I})^{-1}\boldsymbol{Z} + (\sigma_\epsilon^2\boldsymbol{I})^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}'(\sigma_u^2\boldsymbol{I})^{-1}\boldsymbol{y} \\ \boldsymbol{Z}'(\sigma_u^2\boldsymbol{I})^{-1}\boldsymbol{y} \end{bmatrix}, \tag{2.52}$$

which is exactly the form of (2.45) with $\boldsymbol{R} = \sigma_u^2\boldsymbol{I}$ and $\boldsymbol{G} = \sigma_\epsilon^2\boldsymbol{I}$.

The connection between smoothing and mixed models has a simple explanation. It is clear that in smoothing models we wish to reduce the freedom of the parameters to ensure the smoothness of our fitted function. Random effects are also restricted in mixed models by the distributional assumption, as can be seen readily in equation (2.39). With this connection established, it is a small step to consider a smoothing model within the mixed model framework. Taking the model in (2.19) for the motorcycle data, and then assuming

$$\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \sigma_u^2\boldsymbol{I}) \tag{2.53}$$

we have a smooth model expressed as a mixed model. With this formulation we can estimate the variance components by ML or REML, with the smoothing parameter given by $\lambda = \sigma_\epsilon^2/\sigma_u^2$.

The formulation as a mixed model from a TPF smoothing model is simple, and it is dealt with in detail by Ruppert et al. (2003). With a P-spline model the formulation of a mixed model is not so obvious, as the distinction between fixed and random effects is not clear. Currie and Durban (2002) showed how a change of basis could be used to give a mixed model representation of P-splines; this was simplified by Currie et al. (2006), and Wood (2004 (unpublished) gives a general method for converting penalized likelihood models into mixed models.

Taking the P-spline model in Section 2.3.2 we could assume that all the parameters were random effects, using a pure random effects model

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_\theta, \sigma_\theta^2\boldsymbol{D}'\boldsymbol{D}). \tag{2.54}$$

Among other problems the variance matrix for this model would be singular. As suggested by Wood (2004 (unpublished) we can work in the eigenspace of the penalty matrix, and transform to a different basis. Details are given by Currie et al. (2006),

but the basic procedure is to take the singular value decomposition of the penalty matrix

$$\boldsymbol{P} = \boldsymbol{U}\boldsymbol{\Delta}\boldsymbol{U}' \tag{2.55}$$

where $\boldsymbol{\Delta}$ is a diagonal matrix of eigenvalues (assumed to be in ascending order), with $\boldsymbol{U}$ containing the corresponding eigenvectors. The matrix of eigenvectors is then partitioned $\boldsymbol{U} = [\,\boldsymbol{U}_N \vdots \boldsymbol{U}_S\,]$, so that $\boldsymbol{U}_N$ correspond to the 0 eigenvalues, or the null space of the penalty matrix, and consequently $\boldsymbol{U}_S$ spans the eigenspace of the penalty matrix. These matrices are used to split the basis into fixed and random parts:

$$\boldsymbol{X} = \boldsymbol{B}\boldsymbol{U}_N \qquad \text{and} \qquad \boldsymbol{Z} = \boldsymbol{B}\boldsymbol{U}_S \tag{2.56}$$

with the corresponding coefficients

$$\boldsymbol{\beta} = \boldsymbol{U}'_N\boldsymbol{\theta} \qquad \text{and} \qquad \boldsymbol{u} = \boldsymbol{U}'_S\boldsymbol{\theta}. \tag{2.57}$$

With the change of basis the non-zero eigenvalues are used in the variance matrix, so we can formulate the mixed model with $\boldsymbol{X}$ and $\boldsymbol{Z}$ given in (2.56) with the variance matrix of random effects given by

$$\boldsymbol{R} = \sigma_u^2\boldsymbol{\Delta}_S \tag{2.58}$$

where $\boldsymbol{\Delta}_S$ is a diagonal matrix of the non-zero eigenvalues in ascending order.

2.6 Generalized Linear Models

In Section 1.3 it was assumed that the mortality counts data follow a Poisson distribution. We now show how a parametric GLM can be fitted to Poisson data like the mortality data we are interested in. We introduce GLMs as a generalization of the basic linear model described in Section 2.1.

One limitation of the basic linear model is the assumption made in (2.2), that the response variable follows a normal distribution. Nelder and Wedderburn (1972) showed how regression models could be extended to a broader family of distributions. *Generalized Linear Models* (GLMs) encompass many special cases, including the basic linear regression model, the logit and probit models, and the log-linear model (which

we will see corresponds to the famous Gompertz Model for mortality). The generalization allows the response variable to follow any distribution from the exponential family, that is any distribution whose density can be written in the form

$$f(y;\theta,\phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right). \tag{2.59}$$

The parameter θ is known as the natural parameter, and ϕ as the scale or nuisance parameter. It is easy to show that any distribution of the form (2.59) has mean $\mu = b'(\theta)$; see for example Dobson (2002).

The GLM also extends the linear model by relating the mean to the linear predictor through a link function,

$$g(\boldsymbol{\mu}) = \boldsymbol{\eta} = \boldsymbol{X\beta}. \tag{2.60}$$

The link function can be any monotonic differentiable function, but often a convenient choice is the *canonical link* obtained by setting $\boldsymbol{\theta} = \boldsymbol{\eta}$, which leads to

$$g^{-1}(\cdot) = b'(\cdot). \tag{2.61}$$

Clearly, taking the identity as the link function, $g(\boldsymbol{\mu}) = \boldsymbol{\mu}$, we obtain (2.6) for the linear regression model.

To summarize, in the Generalized Linear Model data consist of n observations on a response variable y_i (assumed to be realizations from an exponential family distribution), and a corresponding vector of observations on a set of explanatory variables $\boldsymbol{x}_i$. The mean of y_i is related to the *linear predictor* $\eta_i = \boldsymbol{x}_i'\boldsymbol{\beta}$ through a link function $g(\cdot)$, which can be written in matrix form

$$\boldsymbol{\mu} = g^{-1}(\boldsymbol{X\beta}), \tag{2.62}$$

which becomes

$$\boldsymbol{\mu} = b'(\boldsymbol{X\beta}), \tag{2.63}$$

when using a canonical link.

Estimates of $\boldsymbol{\beta}$ are obtain by maximum likelihood, with the log-likelihood function (in the case of the canonical link) given by

$$\mathcal{L}(\boldsymbol{\beta}) = \frac{\boldsymbol{y'X\beta} - \mathbf{1}'b(\boldsymbol{X\beta})}{a(\phi)} + \mathbf{1}'c(\boldsymbol{y},\phi). \tag{2.64}$$

Closed form solutions for $\hat{\boldsymbol{\beta}}$ cannot be found, except in the case of the normal distribution (where the estimates coincide with the least squares estimates), and therefore numerical methods have to be used in order maximise (2.64). McCullagh and Nelder (1990) show how a Newton-Raphson scheme can be use to find estimates, or by replacing the Hessian by its expectation we can use the *Fisher Scoring* algorithm. Both methods amount to using an *iterative weighted least squares* (IWLS) algorithm

$$\boldsymbol{X}'\tilde{\boldsymbol{W}}\boldsymbol{X}\tilde{\boldsymbol{\beta}} = \boldsymbol{X}'\tilde{\boldsymbol{W}}\tilde{\boldsymbol{z}}, \tag{2.65}$$

where $\tilde{\boldsymbol{W}}$ is the diagonal matrix of weights with

$$\tilde{w}_{ii}^{-1} = g'(\tilde{\mu}_i)^2 \mathrm{var}(y_i), \tag{2.66}$$

and $\tilde{\boldsymbol{z}}$ is the *working vector* whose ith element is

$$\tilde{z}_i = (y_i - \tilde{\mu}_i)g'(\tilde{\mu}_i) + \boldsymbol{x}_i'\tilde{\boldsymbol{\beta}}, \tag{2.67}$$

see McCullagh and Nelder (1990, pp. 33). For the Normal distribution we have $\theta = \mu$, $b(\theta) = \frac{1}{2}\theta^2$, and the canonical link gives the identity link function. Substituting the identity function in (2.66) and (2.67) leaves $\boldsymbol{z} = \boldsymbol{y}$ and $\boldsymbol{W} = \sigma^2\boldsymbol{I}$, and we obtain the least squares solution given in (2.8). In the Poisson case with the canonical or log link we have from (2.66) $w_{ii} = \mu_i = \exp(\eta_i)$.

Within the GLM framework we can also obtain confidence intervals for our fitted model. The Fisher Information for the estimates obtained from (2.65) is

$$I(\hat{\boldsymbol{\beta}}) = \boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X} \tag{2.68}$$

which gives an estimated variance for our estimates as

$$\mathrm{Var}(\hat{\boldsymbol{\beta}}) = I(\hat{\boldsymbol{\beta}})^{-1} = (\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1} \tag{2.69}$$

and

$$\mathrm{Var}(\boldsymbol{X}\hat{\boldsymbol{\beta}}) = \boldsymbol{X}(\boldsymbol{X}'\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}\boldsymbol{X}'. \tag{2.70}$$

In the normal case with constant variance, (2.69) gives the familiar estimate $\mathrm{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\boldsymbol{X}'\boldsymbol{X})^{-1}$. The square root of the diagonal elements of $\mathrm{Var}(\boldsymbol{X}\hat{\boldsymbol{\beta}})$ give the standard errors of the fitted linear predictor and can be used to construct pointwise $(1-\alpha)$ confidence intervals for the fitted curve.

2.6.1 Modelling Mortality Data with a GLM

We shall now consider the situation of modelling the mortality data described in Section 1.3 for a single age or year. Writing the Poisson distribution function in exponential family form

$$f(y;\theta,\phi) = \exp\big\{(y\log\mu - \mu) - \log(y!)\big\} \tag{2.71}$$

we see $\theta = \log\mu$, $b(\theta) = e^{\theta}$, and $a(\phi) = 1$.

For a one dimensional mortality table we observe the number of deaths at the ith age y_i, and the corresponding exposed to risk e_i. Assuming the number of deaths follows the Poisson distribution we have

$$\boldsymbol{y} \sim \mathcal{P}(\boldsymbol{\tau} * \boldsymbol{e}), \tag{2.72}$$

where, here and below, $*$ indicates element-by-element multiplication. Using a canonical link, and treating the $\boldsymbol{e}$ as a known offset (see McCullagh and Nelder (1990) page 138), gives the mean number of deaths as

$$\mu_i = \exp\big(\boldsymbol{x}_i'\boldsymbol{\beta} + \log e_i\big). \tag{2.73}$$

Taking $\boldsymbol{x}_i' = (1, \text{age}_i)$ and re-expressing (2.73) as

$$\log\tau_i = \boldsymbol{x}_i'\boldsymbol{\beta} \tag{2.74}$$

we obtain the formula given in (1.1), with $A = e^{\beta_0}$ and $B = e^{\beta_1}$. Within the GLM framework it is straightforward to extend the Gompertz model by using higher order polynomials; we add columns to the regression matrix, e.g. $\boldsymbol{x}_i' = (1, \text{age}_i, \text{age}_i^2)$ gives a quadratic function.

However the discussion in Section 2.1 concluded that polynomials were not suitable for many situations in the normal case. Therefore we will use the mortality data as a motivating example to develop the penalized spline method within the GLM framework.

Then, as with the linear model, a B-spline basis can be used in a GLM simply by setting $\boldsymbol{X} = \boldsymbol{B}$. As with the normal case we place a penalty on the coefficients

of adjacent B-splines, and estimate the coefficients by maximizing the penalized log-likelihood. Maximizing (2.26) using the log-likelihood in (2.64) leads to the *penalized scoring algorithm*

$$(\boldsymbol{B}'\tilde{\boldsymbol{W}}\boldsymbol{B} + \boldsymbol{P})\hat{\boldsymbol{\theta}} = \boldsymbol{B}'\tilde{\boldsymbol{W}}\tilde{\boldsymbol{z}} \tag{2.75}$$

with $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{z}}$ defined as the weight matrix and working vector used in (2.65). By analogy with (2.69) we estimate the variance of our estimates by

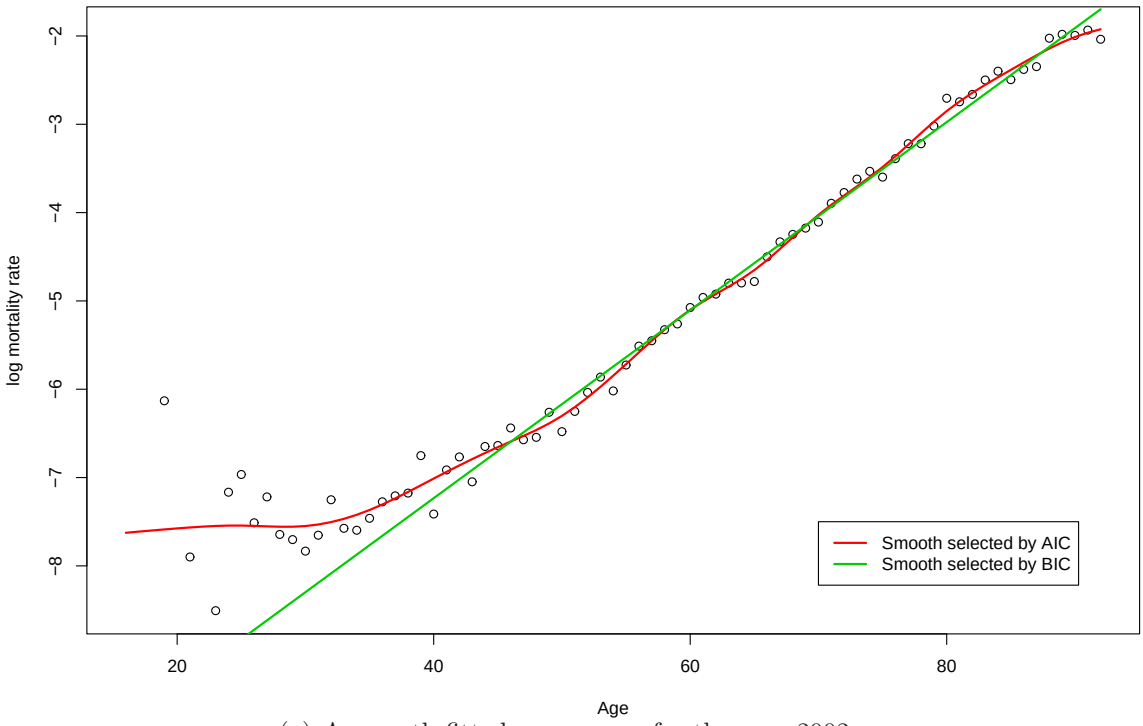
$$\mathrm{Var}(\boldsymbol{B}\hat{\boldsymbol{\theta}}) = \boldsymbol{B}(\boldsymbol{B}'\hat{\boldsymbol{W}}\boldsymbol{B} + \boldsymbol{P})^{-1}\boldsymbol{B}'. \tag{2.76}$$

In the case $\lambda = 0$ (2.76) reduces to (2.70), and in practice as $\lambda \to \infty$ (2.76) converges to the estimate obtained for the polynomial limit of the penalty in (2.30).

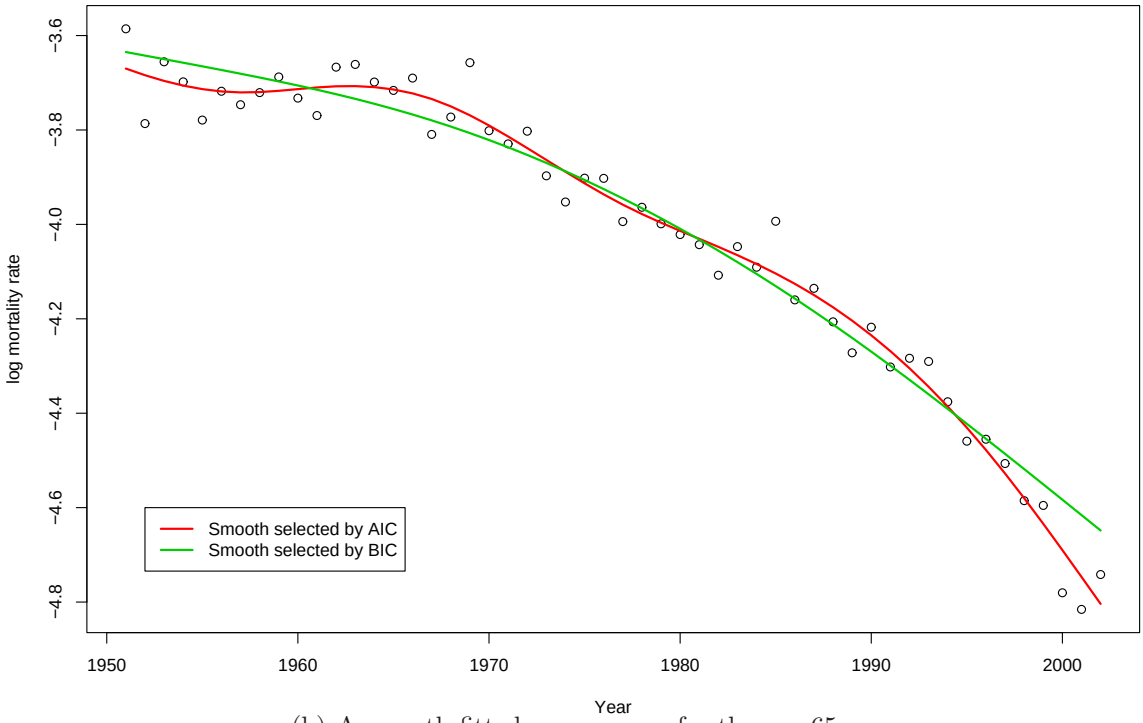
We shall now look at two examples of P-splines fitted to cross-sections of the assured lives mortality data. Figure 2.11(a) shows a smooth fitted across ages for the year 2002; in this case we take $\boldsymbol{x}' = \boldsymbol{x_a}' = (16, \ldots, 92)$ so we have $n = n_a = 77$. Using cubic B-splines and placing a knot at age sixty-five with five years between knots, we need the knot vector $\boldsymbol{t}' = (0, 5, \ldots, 105, 110)$ to cover $\boldsymbol{x}$. This results in a B-spline basis matrix, $\boldsymbol{B}$, with dimension 77×19. Figure 2.11(b) shows the corresponding plot taking a cross-section for age sixty-five, with $\boldsymbol{x}' = \boldsymbol{x_y}' = (1951, \ldots, 2002)$. This time we place a knot at the final year, 2002; with five years between knots, we need the knot vector $\boldsymbol{t}' = (1932, 1937, \ldots, 2012, 2017)$ to cover $\boldsymbol{x}$ which leads to a B-spline basis matrix, $\boldsymbol{B}$ with dimension 52×14. To distinguish the two B-spline basis matrices, we will refer to the basis matrix for age as $\boldsymbol{B}_a$ and the basis matrix for year as $\boldsymbol{B}_y$. These matrices will play an important role when we consider two-dimensional modelling; it is convenient to define

- Marginal basis for age: $\boldsymbol{B}_a$, and
- Marginal basis for year: $\boldsymbol{B}_y$.

With AIC we find smoothing parameters $\lambda_a = 4.3$ and $\lambda_y = 123.1$. We see that more smoothing is required in the year model than in the age model; this point will be discussed in more detail in Chapter 3.



(a) A smooth fitted across ages for the year 2002.



(b) A smooth fitted across years for the age 65.

Figure 2.11: Some P-splines fitted to the assured lives data.

2.6.2 Generalized Linear Mixed Models

We have discussed both mixed models and GLMs and our next step is to combine the two, to give a *generalized linear mixed model* or GLMM. This is an ongoing area of research, and at present only approximate estimates are available for GLMMs.

A GLMM is defined as follows. We observe a vector of responses $\boldsymbol{y}$; for each element of $\boldsymbol{y}$ we observe a set of explanatory variables $\boldsymbol{x}_i'$ with $\boldsymbol{X}$ corresponding to the vector $\boldsymbol{y}$. We also observe a matrix $\boldsymbol{Z}$ similar to that used in (2.41) which indicates the levels of each random effect. We now assume that the following relationship holds

$$E(\boldsymbol{y}|\boldsymbol{u}) = g^{-1}(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u}), \tag{2.77}$$

where $\boldsymbol{\beta}$ is a fixed unknown we wish to estimate, and the $\boldsymbol{u}$ are realizations of the random effects. The conditional distribution of $\boldsymbol{y}$ is assumed to come from the exponential family, so with a canonical link we have

$$f_{y|u}(\boldsymbol{y}|\boldsymbol{u}) = \exp\left\{\frac{\boldsymbol{y}'(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u}) - \mathbf{1}'b(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u})}{a(\phi)} + \mathbf{1}'c(\boldsymbol{y}, \phi)\right\}. \tag{2.78}$$

The random effects are generally assumed to follow a multivariate normal distribution

$$\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_\delta). \tag{2.79}$$

There are examples where we may wish to relax this assumption; see for instance page 464 of Pawitan (2001). However, as Pawitan remarks, this generalization tends to complicate an already difficult problem of estimation.

In order to obtain a likelihood we need to obtain the unconditional density of $\boldsymbol{y}$, but this requires the evaluation of the multidimensional integral

$$\begin{aligned} \ell(\boldsymbol{\beta}, \boldsymbol{\theta}) &= f_y(\boldsymbol{y}) \\ &= \int f_{y,u}(\boldsymbol{y}, \boldsymbol{u})\, d\boldsymbol{u} \\ &= \int f_{y|u}(\boldsymbol{y}|\boldsymbol{u}) f_u(\boldsymbol{u})\, d\boldsymbol{u} \end{aligned} \tag{2.80}$$

which is generally intractable. Evaluation or approximation of the integral (2.80) has been the stumbling block, and has been the area of most focused research in the area of

GLMMs. Currently there are two widely used approaches to the problem. One could specify the GLMM as a hierarchical Bayesian model, which enables us to use MCMC to sample from the posterior distribution of $\boldsymbol{\theta}$ without evaluating (2.80); this will be discussed in more detail in Section 2.7. The other approach is to follow the method of Breslow and Clayton (1993) and use the Laplace approximation to the integral, which leads to estimating (predicting) $\boldsymbol{\beta}$ and $\boldsymbol{u}$ by maximizing the *penalized quasi-likelihood* (PQL). Ruppert et al. (2003) derive the PQL for the exponential family with normal random effects

$$\ell_{PQL}(\boldsymbol{\beta}, \boldsymbol{u}; y) = \log\left(f(\boldsymbol{y}|\boldsymbol{u})\right) - \boldsymbol{u}'\boldsymbol{G}^{-1}\boldsymbol{u}. \tag{2.81}$$

The PQL is so named because the likelihood has the same form as a penalized likelihood; see for example (2.22) and (2.26). For given values of the variance parameters, estimation of $\boldsymbol{\beta}$ and $\boldsymbol{u}$ proceeds by maximizing (2.81); the variance parameters are estimated either by ML or REML. The (restricted) likelihood for the variance parameters is the same as that given in (2.47) (or (2.48)) but with $\boldsymbol{R}$ replaced by a weight matrix which is determined by the conditional distribution of $\boldsymbol{y}$. As shown by Ruppert et al. (2003, pp. 205), for the Poisson GLMM the weight matrix is the diagonal matrix of conditional expectations given in (2.77), this leads to the iterative equation for $\boldsymbol{\beta}$ and $\boldsymbol{u}$

$$\begin{bmatrix} \boldsymbol{X}'\tilde{\boldsymbol{W}}\boldsymbol{X} & \boldsymbol{X}'\tilde{\boldsymbol{W}}\boldsymbol{Z} \\ \boldsymbol{Z}'\tilde{\boldsymbol{W}}\boldsymbol{X} & \boldsymbol{Z}'\tilde{\boldsymbol{W}}\boldsymbol{Z} + \boldsymbol{G}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\beta} \\ \hat{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}'\tilde{\boldsymbol{W}}\tilde{\boldsymbol{z}} \\ \boldsymbol{Z}'\tilde{\boldsymbol{W}}\tilde{\boldsymbol{z}} \end{bmatrix}, \tag{2.82}$$

where $\tilde{\boldsymbol{z}}$ and $\tilde{\boldsymbol{W}}$ are the working vector and weight matrix used in a GLM, based on the current values of $\boldsymbol{\beta}$ and $\boldsymbol{u}$. For the purpose of calculating ML or REML we use $\boldsymbol{V} = \boldsymbol{W}^{-1} + \boldsymbol{Z}\boldsymbol{G}\boldsymbol{Z}'$ in $\ell_{ML}(\boldsymbol{\gamma}|\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{u}})$ or $\ell_{REML}(\boldsymbol{\gamma}|\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{u}})$ in (2.47) or (2.48) respectively.

Smoothing within the GLMM follows directly from the explanation given for the linear mixed model. We must first split the basis into the fixed and random part (for a P-spline model we can use the decomposition described in Section 2.5.3). The same arguments that were mentioned in Section 2.5.3 can be used in the case of smooth generalized linear models to formulate them as GLMMs, as shown in Wood (2004 (unpublished) and Currie et al. (2006).

2.7 Bayesian Smoothing

As outlined in Section 2.4 smoothing parameter selection is one of the key problems in fitting smooth models. The main difficulty is that the smoothing parameter is a hyper-parameter that sits outside the likelihood, because the penalized log-likelihood is conditional on the smoothing parameter. Such hyper-parameters are common in hierarchical Bayesian models, where there are several layers of parameters, with each layer conditional on the one above. In a sense the mixed model representation of penalized splines in section 2.5.3 is a compromise between the Bayesian and classical approaches, as we are attaching a normal probability distribution to our uncertainty about the shape of the smooth above the linear trend (for a second order penalty), which gives us a likelihood for the smoothing parameter. However, one could argue that the use of other selection criteria is equivalent to maximizing the likelihood for some other, unknown "prior" for the "random effects".

To fit a full Bayesian smoothing model we could extend the mixed model representation by also specifying a prior for the fixed components $\boldsymbol{\beta}$, and the variance parameters $\boldsymbol{\theta}$. Ruppert et al. (2003) suggest the use of an improper uniform prior for $\boldsymbol{\beta}$ and a sequence of inverse gamma distributions for the components of $\boldsymbol{\theta}$. The hyper-parameters are selected to give non-informative priors for the variance components. These priors are conjugate to the normal likelihood and using the hierarchical structure of the model we can implement the Gibbs sampler to draw from the posterior distribution. However this scheme requires the inversion of a $c \times c$ matrix, where c is the number of parameters in the model, at every iteration so can be very heavy computationally. In the case of the GLMM we cannot easily sample from the conditional distribution of the fixed or the random effects so the Gibbs sampler is not available. Ruppert et al. (2003) give details of how to use the Metropolis-Hastings algorithm to sample from the posterior distribution, but again this requires the inversion of a $c \times c$ matrix at each iteration. A different approach is used by Lambert and Eilers (2006), who put a prior directly on the differences of the B-spline coefficients. They show how a Langevin-Hastings algorithm can be used to explore the posterior without having to compute or invert large matrices, and show how to draw directly from

the marginal posterior of the B-spline coefficients without having to sample from the penalty parameters.

2.8 Forecasting

In this section we will describe the method of forecasting in one dimension, taking a single age from the assured live mortality data as an example.

As suggested by Currie et al. (2004) the forecasting is treated as a missing data problem, and the penalty is used to "fill in" these missing data. They suggested that the forecast could be obtained by augmenting the data with dummy values for the forecast region, and then weighting the dummy data out of the fitting procedure with an indicator matrix. The adjusted penalized scoring algorithm is

$$(\boldsymbol{B}'\boldsymbol{V}\tilde{\boldsymbol{W}}\boldsymbol{B}+\boldsymbol{P})\hat{\boldsymbol{\theta}}=\boldsymbol{B}'\boldsymbol{V}\tilde{\boldsymbol{W}}\tilde{\boldsymbol{z}} \tag{2.83}$$

where $\boldsymbol{V}=\text{diag}([\mathbf{1}'\ \mathbf{0}'])$. This can be streamlined since all that is required to produce a forecast are the extra coefficients for the forecast region. This can be achieved by appending the required number of columns of zeros to the basis matrix. This is most easily explained using an example. Taking the data from the assured lives data used in Fig. 2.11(b) we have $n=52$ and $\boldsymbol{x}=\boldsymbol{x_y}$. To forecast for twenty years we append the extra years to the explanatory variable $\boldsymbol{x}$

$$\boldsymbol{x}^*=[\,\boldsymbol{x}'\vdots\boldsymbol{x}_p'\,]' \tag{2.84}$$

where $\boldsymbol{x}_p'=(2003,\ldots,2022)$. The basis is then evaluated over the new index which gives a larger B-spline basis matrix, $\boldsymbol{B}^*$. Using the same knot positions as previously we obtain a B-spline matrix of dimension 72×18. However, in order to fit the model we remove the extra rows added for the forecast, this gives us a new regression matrix

$$\boldsymbol{B}_m^*=[\,\boldsymbol{B}\vdots\mathbf{0}\,], \tag{2.85}$$

where $\boldsymbol{B}$ is the same 52×14 matrix used for Fig. 2.11(b), and in this case $\mathbf{0}$ has dimension 52×4. Clearly in an unpenalized regression this would result in a singular system, but as mentioned at the beginning of the section the penalty enables the

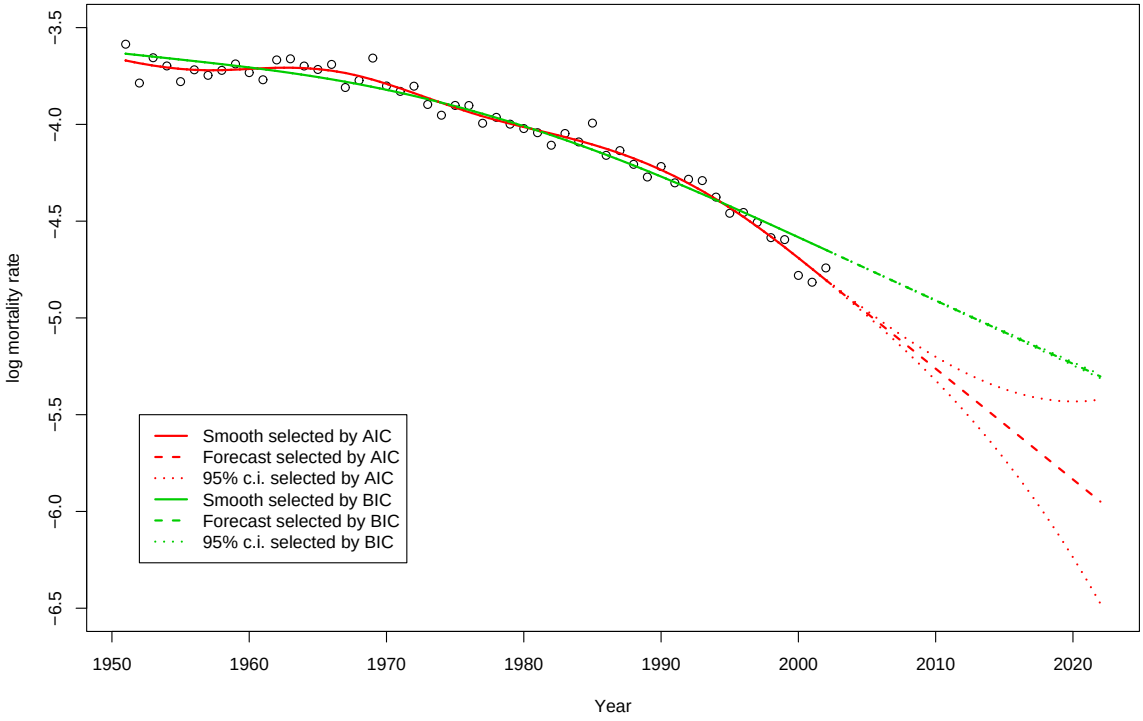


Figure 2.12: A twenty year forecast for age sixty-five for the assured lives data.

estimation of the coefficients in the forecast region. Using (2.75) we obtain a set of forecast coefficients, $\boldsymbol{\theta}^*$, which can be used with the full forecast basis to evaluate the smooth in the forecast region.

The forecasted log rates at age sixty-five are shown in Fig. 2.12, with confidence intervals added using the diagonal elements of (2.76). We note the difference in the width of the confidence intervals (and in particular the extremely narrow confidence intervals for the model selected by BIC); this will be discussed further in Section 2.8.1. We see that in the forecast region the trend appears linear. Equation (2.30) gave a polynomial limit for P-splines, but the rate of convergence to the limit depends on the amount of data available. In the forecast region there is no data, so we obtain the limiting polynomial for the model. As we have been using cubic B-splines with a second order difference, by (2.30) we get a linear limit and thus a linear forecast. With a second order penalty the forecast is a linear extrapolation of the last two coefficients, and it would be possible to fit the model using the data and then extrapolate to obtain the forecast. In section 3.3.6 we look at forecasting a two-dimensional surface where obtaining the forecast is more complicated and we have to use the method explained here.

2.8.1 Model Uncertainty

The two lines shown in Fig. 2.12 highlight a problem that occurs in many regression models but is particularly visible in the context of smoothing: model uncertainty. In this section we will discuss model uncertainty with reference to forecasting in semi-parametric models, before suggesting some ways of taking it into account.

Figure 2.12 shows two P-spline models fitted to the age sixty-five cross-section of the assured lives mortality data, with the smoothing parameter for the green line selected using BIC and the smoothing parameter for the red line fitted using AIC. The dotted lines surrounding each line represent the 95 percent confidence interval for the estimated log rate of mortality. We note that the confidence interval in forecast region gets wider as we move away from the data. We also notice that within the region of the data neither of the confidence intervals generally contains the mean trend of the other model, even though both models seem reasonably plausible given the data. The reason why these two seemingly reasonable models are not within each others' confidence intervals is because the confidence intervals are conditional on the model itself. This can be clearly seen in expression (2.76), where the smoothing parameter, through the penalty matrix $\boldsymbol{P}$, is used in the calculation of the variance of the fitted values. The conditioning on the model can be seen very clearly in the forecast region, where the confidence intervals are much narrower for the model selected using BIC because the model is less flexible and places a heavier penalty on movements away from a linear trend.

This example highlights a problem that is ignored in many areas of statistics, and was summed up very nicely by Hjort and Claeskens (2003), "The traditional use of model selection methods in practise is to proceed as if the final selected model had been chosen in advance, without acknowledging the additional uncertainty introduced by model selection". Interesting opinions on the matter can also be found in Chatfield (1995).

Seemingly the only way to remove this problem in the example would be to unwind the conditioning on the selected model when calculating the confidence interval. Hjort and Claeskens (2003) suggest a model averaging approach within the frequentist framework although this is only applied to situations where a relatively small number

of models are under consideration. To use this in the context of smoothing, we would have to average over a predetermined set of possible smoothing parameters, which does not sit well with the use of a model selection criterion to optimize the smoothing parameter.

Another possible solution is to use a Bayesian approach described in Section 2.7. With the *Markov chain Monte Carlo* (MCMC) algorithm we obtain simulations from the posterior distribution of the fitted line which take into account the uncertainty surrounding the smoothing parameter; these simulations could then be used to find an empirical estimate of the confidence interval for the smooth which takes into account an element of the uncertainty surrounding the smoothing parameter. However, this puts a greater importance on the parameterization of the prior for the smoothing parameter, about which often we have little or no information.

2.9 Discussion

We now sum up the main conclusions from the work of this chapter and highlight the main ideas that will be used in the remainder of this thesis.

Comparing all the smoothing models described in this chapter we see there is relatively little to choose between them in terms of results. With reference to the plots in Fig. 2.13 we can see that applying each method to the motorcycle crash data it is difficult to tell the results apart. Generally, the choice of model selection criterion, and thus the amount of smoothing, is of much greater importance than the smoothing method itself.

However, as we move into multi-dimensional smoothing, computational efficiency becomes a greater consideration. The multi-dimensional analogues of the full rank smoothers described in this chapter: kernel smoothing and natural cubic splines, run into computational difficulties when they are applied to large multi-dimensional data sets.

Although, based on the results in one dimension the methods are difficult to separate, conceptually and computationally, we feel that regression splines, and P-splines in particular, offer several advantages:

- A straightforward extension to multiple dimensions: Regression splines fall within the penalized likelihood framework; this simplifies the extension to multi-dimensional modelling. To extend the model, we need only specify a multi-dimensional basis and penalty; the fitting procedure can then be carried out in the same way, as will be shown in the next chapter. This is in marked contrast to thin plate splines, for example, which are significantly more complicated to fit than natural cubic splines.

- Further computational advantages: For multi-dimensional data with a grid or array structure, using P-splines we can make further improvements in computational efficiency, by taking advantage of the structure of the basis. This is shown in Chapter 4.

- Simple additive models: If we keep all the components within the penalized likelihood framework, fitting additive models can be simplified: we do not require the back-fitting algorithm. This is shown in Chapter 5.

- Interpretation of the parameters: The B-spline coefficients in a P-spline model are essentially local averages of the data over the non-zero domain of the B-spline (see Eilers and Marx, 2004). This makes it easier to adapt the penalty in specific instances when a different structure is required in the model.

- Good numerical properties: The model can be fitted directly as described in this chapter without the need to change basis for fear of numerical difficulties. There is a general impression that P-splines and penalized TPFs are equivalent, subject to a change of basis like that described in Section 2.5.2. The change of basis is used when fitting a penalized TPF model: in practise one converts to a B-spline, or some other numerically suitable basis, for the computation. However, as noted by Welham (2005), P-splines are more general that penalized TPFs because there are some combinations of a B-spline basis and difference penalty which cannot be represented as a penalized TPF; for example, one cannot find a simple TPF equivalent of a cubic B-spline basis with a second order difference penalty.

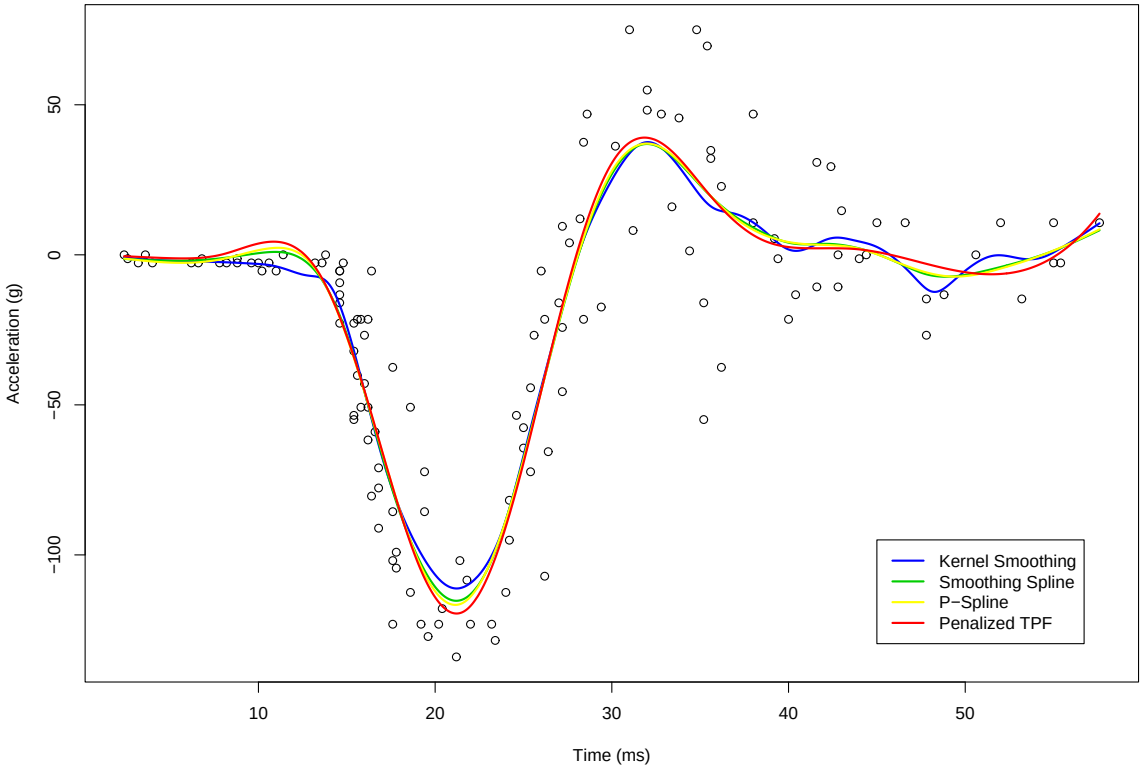


Figure 2.13: A comparison of the smoothing methods discussed in this chapter applied to the motorcycle data.

Chapter 3

Multi-dimensional smoothing

In this chapter we look at multi-dimensional smoothing methods. As in the last chapter we introduce several different approaches before describing the P-spline method in more detail.

In Section 3.1 we describe the general multi-dimensional smoothing problem, starting with two-dimensional smoothing; we show how the data described in Section 1.3 give rise to an example of the general two-dimensional smoothing problem, but with a particular structure. In Section 3.2 we describe some full-rank multi-dimensional smoothing methods, before describing the multi-dimensional P-spline model in Section 3.3.

3.1 Data

In this section we will describe the general multi-dimensional smoothing problem, and show how the mortality data described in Section 1.3 give rise to a special case of the multi-dimensional smoothing problem. We will first specify the two-dimensional case, and then describe the problem in d-dimensions.

In a two-dimensional smoothing problem our data consist of n observations of the pairs $(y_i, \boldsymbol{x}_i)$ where $y_i \in \mathbb{R}$ and $\boldsymbol{x}_i \in \mathbb{R}^2$. We assume that each y_i is a realization of a random variable with

$$E(y) = \mu = S(\boldsymbol{x}) \tag{3.1}$$

for some smooth function $S : \mathbb{R}^2 \to \mathbb{R}$, which we wish to estimate by the function

$\hat{S}$. Often it is easier to think in terms of each dimension of the data, and imagine our data stored as a response vector $\boldsymbol{y}$ and explanatory vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, where $\boldsymbol{y}, \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$. We can then specify the model in matrix notation

$$E(\boldsymbol{y}) = S(\boldsymbol{x}_1, \boldsymbol{x}_2) = [S(x_{1,1}, x_{2,1}), \ldots, S(x_{1,n}, x_{2,n})]'. \tag{3.2}$$

The mortality data in Section 1.3 has additional structure to that described above: the data lie on a grid. In order to get the data into the format described above we must re-arrange it. Firstly, we take the matrix of deaths, $\boldsymbol{Y}$, $n_a \times n_y$, and vectorize it; similarly, we vectorize the matrix of exposures, $\boldsymbol{E}$, $n_a \times n_y$. Applying the vec operator, we obtain

$$\boldsymbol{y} = \text{vec}(\boldsymbol{Y}) \qquad \text{and} \qquad \boldsymbol{e} = \text{vec}(\boldsymbol{E}); \tag{3.3}$$

i.e., the vec operator stacks the columns of $\boldsymbol{Y}$ and $\boldsymbol{E}$ on top of each other to give the vectors $\boldsymbol{y}$ and $\boldsymbol{e}$, both $n_a n_y \times 1$. Thus for the assured lives data set we have

$$\begin{aligned} \boldsymbol{y}' &= [y_{16,1951}, \ldots, y_{92,1951}, \ldots\ldots, y_{16,2002}, \ldots, y_{92,2002}] \\ \boldsymbol{e}' &= [e_{16,1951}, \ldots, e_{92,1951}, \ldots\ldots, e_{16,2002}, \ldots, e_{92,2002}]. \end{aligned} \tag{3.4}$$

We aim to model the mortality as a smooth function of age and year, so we need to define the vector $\boldsymbol{x}_1 = \boldsymbol{x_A}$ for age and $\boldsymbol{x}_2 = \boldsymbol{x_Y}$ for year. Following the same process as with $\boldsymbol{Y}$ and $\boldsymbol{E}$ we set

$$\begin{aligned} \boldsymbol{x_A} &= \text{vec}(\boldsymbol{X_A}) \\ \boldsymbol{x_Y} &= \text{vec}(\boldsymbol{X_Y}) \end{aligned} \tag{3.5}$$

where $\boldsymbol{X_A}$ and $\boldsymbol{X_Y}$ are matrices giving the age index and year index corresponding to the deaths and exposures; so for the assured lives data

$$\boldsymbol{X_A} = [\boldsymbol{x_a}, \ldots, \boldsymbol{x_a}] = \begin{bmatrix} 16 & 16 & \cdots & 16 & 16 \\ 17 & 17 & \cdots & 17 & 17 \\ \vdots & \vdots & & \vdots & \vdots \\ 91 & 91 & \cdots & 91 & 91 \\ 92 & 92 & \cdots & 92 & 92 \end{bmatrix}, \tag{3.6}$$

and

$$\boldsymbol{X_Y} = [\boldsymbol{x_y}, \ldots, \boldsymbol{x_y}]' = \begin{bmatrix} 1951 & 1952 & \cdots & 2001 & 2002 \\ 1951 & 1952 & \cdots & 2001 & 2002 \\ \vdots & \vdots & & \vdots & \vdots \\ 1951 & 1952 & \cdots & 2001 & 2002 \\ 1951 & 1952 & \cdots & 2001 & 2002 \end{bmatrix}. \tag{3.7}$$

We note that (3.6) and (3.7) define the unique age and year index vectors

$$\begin{aligned} \boldsymbol{x_a}' &= (16, 17, \ldots, 91, 92), \\ \boldsymbol{x_y}' &= (1951, 1952, \ldots, 2001, 2002). \end{aligned} \tag{3.8}$$

The process of vectorizing seems rather artificial as the natural format for these data is in a matrix. We will see in Chapter 4 that it is not necessary to vectorize the data. The matrix methods described there are conceptually more attractive and computationally more efficient; furthermore these methods extend to general array methods in higher dimensions.

The extension from a two-dimensional to a d-dimensional smoothing problem is straight-forward. In the d-dimensional case the response variable is indexed by d explanatory variables, so our data set consists of a vector of responses $\boldsymbol{y}$, and the d vectors of corresponding explanatory variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$; we then seek a smooth function such that

$$E(\boldsymbol{y}) = S(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d). \tag{3.9}$$

Fitting a smooth model in more than two dimensions in practice is probably not the best option for three reasons. Firstly, spreading the data about in four- or higher-dimensional space is likely to leave quite big areas where no information about the behaviour of the smooth is available, so the result will be heavily dependent on the interpolation properties of the smoothing method. Secondly, the curse of dimensionality means that smoothing models beyond two dimensions quickly become large and unmanageable simply because of the number of parameters needed to describe the smooth. Thirdly, beyond two dimensions we lose the ability to visualise the function so we are reliant on cross-sections and contours, which hampers both model diagnostics and inference from the model. As recommended by Hastie and Tibshirani (1990),

it is often better to use an additive model when dealing with data indexed by more than two explanatory variables.

3.2 Full Rank Smoothers

In this section we will briefly describe some full rank smoothing methods for multi-dimensional problems. These methods are only practical when the number of data points is relatively small, as we are required to solve a system of n equations in order to fit using these methods. The methods described in this section are implemented in the R package `fields`.

3.2.1 Thin Plate Splines

Thin plate splines are the multi-dimensional analogue of natural cubic splines; see Green and Silverman (1994, Chap. 7) and Wood (2006, pp. 154–160). Matters are more complicated in the multi-dimensional case because there is no natural order to the data points. However we can still describe the smoothness of a function; for example, in the two-dimensional case we use the integrated sum of squared second order partial derivatives

$$\int\int \left\{\frac{\mathrm{d}^2 S}{\mathrm{d}x_1^2}\right\}^2 + 2\left\{\frac{\mathrm{d}^2 S}{\mathrm{d}x_1 \mathrm{d}x_2}\right\}^2 + \left\{\frac{\mathrm{d}^2 S}{\mathrm{d}x_2^2}\right\}^2 \mathrm{d}x_1 \mathrm{d}x_2. \tag{3.10}$$

With this measure of smoothness of a two-dimensional function we can proceed as in the one-dimensional case and seek the minimizer of the functional

$$E(S) = \sum_{i=1}^{n} [y_i - S(\boldsymbol{x}_i)]^2 + \lambda \int\int \left\{\frac{\mathrm{d}^2 S}{\mathrm{d}x_1^2}\right\}^2 + 2\left\{\frac{\mathrm{d}^2 S}{\mathrm{d}x_1 \mathrm{d}x_2}\right\}^2 + \left\{\frac{\mathrm{d}^2 S}{\mathrm{d}x_2^2}\right\}^2 \mathrm{d}x_1 \mathrm{d}x_2. \tag{3.11}$$

As shown by Green and Silverman (1994, Chap. 7), for given λ, (3.11) has a unique minimizer which is a thin plate spline. Selection of the smoothing parameter can then proceed by optimization of one of the model selection criteria described in Section 2.4. We present the results of a thin plate spline smooth applied to the assured lives mortality data in Section 3.2.4.

3.2.2 Kriging

In this section we will give a brief description of isotropic Kriging. Kriging is not widely used in one-dimensional problems, and in its multi-dimensional form is used mainly in geostatistics, where it was originally developed. As observed by Hastie and Tibshirani (1990) the Kriging approach is "interesting because it illustrates the stochastic function approach to smoothing"; in this respect its starting point is fundamentally different from the imposition of smoothing by either penalization or the use of weighted averages, for example. Kriging assumes that the observations are realizations from a stochastic process such that

$$y_i = \mu + S(\boldsymbol{x}_i) + \epsilon_i \tag{3.12}$$

where $S(\boldsymbol{x})$ is a zero mean stationary stochastic process. We seek an estimate of $S()$, and in particular $\hat{S}(\boldsymbol{x}_0)$ for the unobserved location $\boldsymbol{x}_0$. The standard assumption is that the covariance of the stochastic process at two locations only depends on how far the two locations are apart. This is the standard isotropic assumption:

$$\text{Cov}(S(\boldsymbol{x}), S(\boldsymbol{x}+\boldsymbol{h})) \text{ depends only on } \mid \boldsymbol{h} \mid . \tag{3.13}$$

We follow Ruppert et al. (2003, Sec. 13.3). The best linear predictor of $S(\boldsymbol{x}_0)$ at an unobserved location $\boldsymbol{x}_0$ is given by

$$\hat{S}(\boldsymbol{x}_0) = \boldsymbol{c}_0'(\boldsymbol{C} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \mu\boldsymbol{1}); \tag{3.14}$$

here

$$\boldsymbol{c}_0 = [\text{Cov}(\hat{S}(\boldsymbol{x}_0), \hat{S}(\boldsymbol{x}_1)), \ldots, \text{Cov}(\hat{S}(\boldsymbol{x}_0), \hat{S}(\boldsymbol{x}_n))]$$

and

$$\begin{aligned} \boldsymbol{C} &= \text{Cov}([\hat{S}(\boldsymbol{x}_1), \ldots, \hat{S}(\boldsymbol{x}_n)]') \\ &= [C(||\boldsymbol{x}_i - \boldsymbol{x}_j||)], \quad 1 \le i, j \le n, \end{aligned} \tag{3.15}$$

by the isotropic assumption in (3.13). Now let

$$C(r) = \sigma_s^2 C_0(r)$$

where $\sigma_s^2 = \text{Var}(S(\boldsymbol{x}))$, the variance of the stochastic process for $S(\boldsymbol{x})$; one common assumption is to set $C_0(r) = e^{-|r|}$, the exponential correlation function. A potential weakness of Kriging is the lack of consensus on how to select the function C_0 and its parameters; the classical approach is to select the covariance function using *variogram* analysis; for more details see Ruppert et al. (2003) who give an extensive list of references for Kriging.

We remark that the isotropic assumption is unlikely to work well with mortality data since there is no reason to believe that the amount of smoothing in the age direction should be the same as that in the year direction. The results of performing Kriging on the assured lives data set are given in the next section.

3.2.3 Radial Bases

In this section we will describe the multi-dimensional "equivalent" of truncated power functions. As mentioned in section 2.3.1 truncated power functions suffer from poor numerical properties, and these poor properties are exacerbated by moving into multiple dimensions. One solution to this problem, described by Ruppert et al. (2003), is to switch to a radial basis.

In one dimension switching to a radial basis from a basis of truncated power functions amounts to switching from the basis defined by the matrix $\boldsymbol{Z}$ in (2.21) to

$$\boldsymbol{Z}_R = \begin{bmatrix} (|x_1 - t_1|)^k & \cdots & (|x_1 - t_J|)^k \\ \vdots & & \vdots \\ (|x_n - t_1|)^k & \cdots & (|x_n - t_J|)^k \end{bmatrix}. \tag{3.16}$$

Using the change of basis technique described in Section 2.5.2 we can find the penalty matrix so the models are equivalent.

For multi-dimensional smoothing with radial bases, we define the columns of the basis matrix $\boldsymbol{Z}_R$ as follows

$$\boldsymbol{Z}_R = \begin{bmatrix} (||\boldsymbol{x}_1 - \boldsymbol{t}_1||)^k & \cdots & (||\boldsymbol{x}_1 - \boldsymbol{t}_J||)^k \\ \vdots & & \vdots \\ (||\boldsymbol{x}_n - \boldsymbol{t}_1||)^k & \cdots & (||\boldsymbol{x}_n - \boldsymbol{t}_J||)^k \end{bmatrix}, \tag{3.17}$$

where $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_J \in \mathbb{R}^d$ are the multi-dimensional knot positions. The specification of the penalty is not straightforward using this approach as we have no intuitive

understanding of how the penalty behaves in one dimension. Ruppert et al. (2003) suggest using a radial basis in a mixed model then separately specifying an isotropic covariance matrix; they show connections between this method and both Kriging and thin plate splines.

3.2.4 Smoothing the assured lives data with the `fields` package

In this section we will show how the full rank methods described in this chapter can be applied to data. We will use the methods implemented in the `fields` package by Nychka (2007) to smooth the two-dimensional assured lives data. Currently the package can only be used with normal responses, so in this section we will be smoothing the log of the raw mortality rates as a normal response rather than modelling the number of deaths as a Poisson response. Modelling the raw rates directly in this way means that we cannot deal with data points where zero deaths are observed, so in order to make the modelling simpler we reduce the data by only modelling ages $25, \ldots, 80$. The code below shows how a thin plate spline and a Kriging model can be fitted to the raw rates. We begin by calculating the log rates and storing them in a matrix `R`, this is then vectorized and stored as `y`, we then need to calculate the matrices $\boldsymbol{X}_A$ and $\boldsymbol{X}_Y$ from (3.6) and (3.7) which are stored as `XA` and `XY` respectively. The three matrices are then vectorized and joined in a two-column matrix so they can be entered as a parameter into the smoothing functions `Krig` and `Tps` as the locations for the vector of responses `y`. For the Kriging function we must also specify any additional parameters of the covariance function. In this example we use the default covariance function, the exponential function, so $C_0(r) = e^{-|r|}$ in this case; the parameter r becomes the parameter `theta` in the implementation. There is no implementation of variograms in the `fields` library and no discernible guidance on how to select `theta`; in the example we have chosen $r =$ `theta` $= 100$ by trial and error to give a pleasing look to the smooth surface shown in 3.2. The `R` code used to produce Fig. 3.1 and Fig. 3.2 is displayed below.

```
# Load data and library
load("Assured.RData")
library("fields")

# Cut data down for ages 25-80, and years 1951-2002
Xa <- Xa[15:70]
Xy <- Xy[-(1:4)]
Y <- Y[15:70,-(1:4)]
E <- E[15:70,-(1:4)]

# Get dimensions of Xa and Xy
na <- length(Xa)
ny <- length(Xy)

# Calculate log rates
R <- log(D/E)

# Calculate location matrices corresponding to R
Xa <- matrix(Xa, na, ny)
Xy <- matrix(Xy, na, ny, TRUE)

# Vectorize rates, and vectorize and join XA and XY
y <- c(R)
x <- cbind(c(XA),c(XY))

# Fit the thin plate spline
Mort.Tps <- Tps(x, y)

# Fit the Kriging model
Mort.Krig <- Krig(x, y, theta = 100)
```

3.3 Multi-Dimensional P-splines

In this section we extend the model described in section 2.3.2 to the multi-dimensional case using the method proposed by Currie et al. (2004). As we shall see the principles are exactly the same as the one-dimensional case, and extending the model basically amounts to some dimensional book-keeping.

Importantly we see that the bases for multi-dimensional P-splines are constructed from the one-dimensional marginal bases, giving us a low-rank multi-dimensional smoothing method suitable for use on large data sets such as the CMI mortality data.

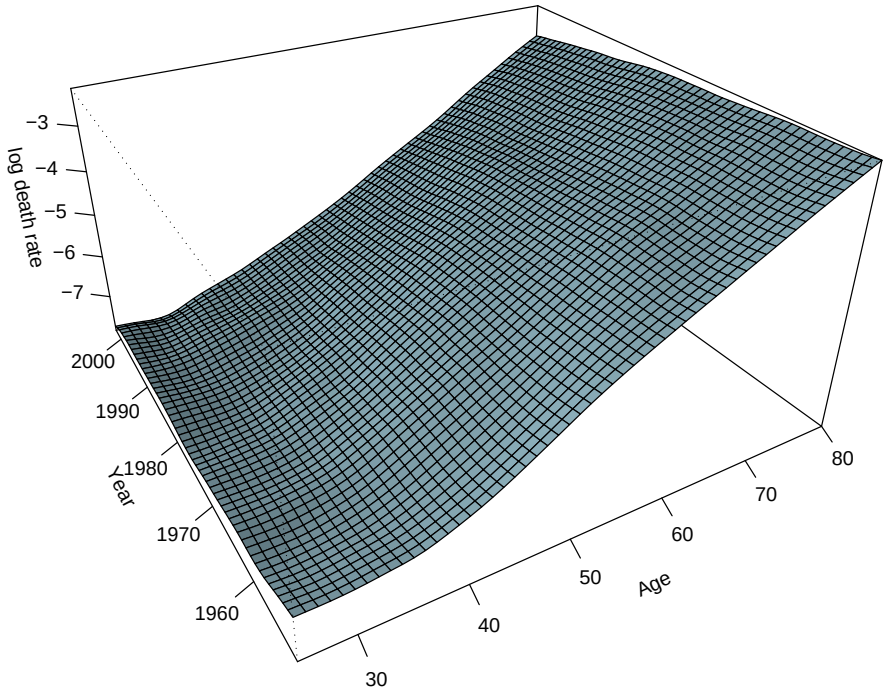


Figure 3.1: A thin plate spline fitted to the assured lives mortality data for ages 25-80, and the years 1951-2002.

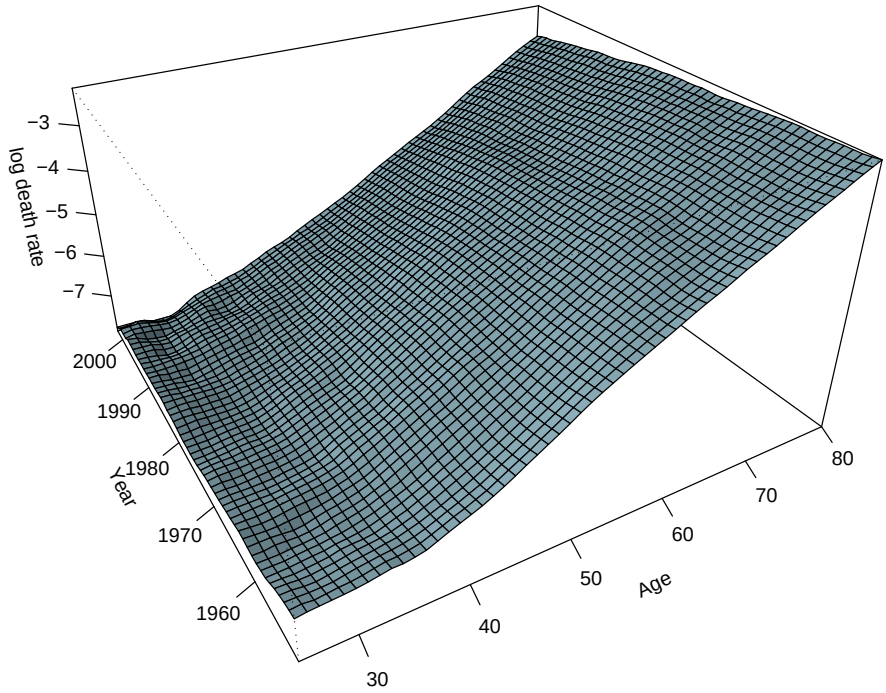


Figure 3.2: A Kriging model fitted to the assured lives mortality data for ages 25-80, and the years 1951-2002.

3.3.1 Some Linear Algebra

The multi-dimensional P-spline model uses some non-standard matrix operations which we will briefly describe in this section. The *Kronecker product* of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ denoted $\boldsymbol{A} \otimes \boldsymbol{B}$ is a matrix containing all scalar products of the elements of $\boldsymbol{A}$ and $\boldsymbol{B}$; specifically if $\boldsymbol{A}$ and $\boldsymbol{B}$ have dimensions $m \times n$ and $p \times q$ respectively then

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{1,1}\boldsymbol{B} & \dots & a_{1,n}\boldsymbol{B} \\ \vdots & & \vdots \\ a_{m,1}\boldsymbol{B} & \dots & a_{m,n}\boldsymbol{B} \end{bmatrix}, \tag{3.18}$$

has dimension $mp \times nq$.

The *row tensor* of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, denoted $\boldsymbol{A} \square \boldsymbol{B}$, is a matrix containing the element-by-element multiplication of each column of $\boldsymbol{A}$ with each column of $\boldsymbol{B}$. Clearly the products of the columns are only defined if the matrices have the same number of rows, hence $\boldsymbol{A} \square \boldsymbol{B}$ is defined if and only if $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same number of rows, in which case

$$\boldsymbol{A} \square \boldsymbol{B} = (\boldsymbol{A} \otimes \mathbf{1}_q') * (\mathbf{1}_p' \otimes \boldsymbol{B}), \tag{3.19}$$

is of dimension $n \times pq$, for $\boldsymbol{A}$ and $\boldsymbol{B}$ of dimensions $n \times p$ and $n \times q$ respectively. Details of some properties of Kronecker products and row tensors are given in Appendix B.

3.3.2 The Two-Dimensional Case

In this section we will show how to fit a two-dimensional P-spline model with reference to the mortality data described in Section 1.3.

By vectorizing the data we obtain the data in the conventional format for a two-dimensional smooth model. We have the set of triples $(y_i, x_{A,i}, x_{Y,i})$, where $x_{A,i}$ and $x_{Y,i}$ are the ith elements of $\boldsymbol{x_A}$ and $\boldsymbol{x_Y}$ in (3.5); thus each count of deaths is indexed by an age and a year.

In order to fit the model we must define knot sequences, $\boldsymbol{t}_a$ and $\boldsymbol{t}_y$, for age and year respectively. As in the one-dimensional models described in Section 2.6.1, we will use a knot every five years for both age and year, and place a knot at age 65, and a knot at the last year of data, 2002; we obtain the knot sequences $\boldsymbol{t}_a = (0, 5, \dots, 105, 110)$

and $\boldsymbol{t}_y = (1932, 1937, \ldots, 2012, 2017)$. We then define two sets of cubic B-spline basis functions over $\boldsymbol{t}_a$ and $\boldsymbol{t}_y$ for age and year respectively. We obtain the bases $\{B_{a,1}(), \ldots, B_{a,c_a}()\}$ for age, and $\{B_{y,1}(), \ldots, B_{y,c_y}()\}$ for year, where in this example $c_a = 19$ and $c_y = 14$. We then define the B-spline regression matrices

$$\boldsymbol{B}_A = B_a(\boldsymbol{x}_A) \qquad \text{and} \qquad \boldsymbol{B}_Y = B_y(\boldsymbol{x}_Y) \tag{3.20}$$

where $(\boldsymbol{B}_A)_{i,j} = B_{a,j}(x_{A,i})$ and $(\boldsymbol{B}_Y)_{i,j} = B_{y,j}(x_{Y,i})$. Generally $\boldsymbol{B}_A$ has dimensions $n_a n_y \times c_a$ and $\boldsymbol{B}_Y$ has dimensions $n_a n_y \times c_y$, and in this example $\boldsymbol{B}_A$ is 4004×19 and $\boldsymbol{B}_Y$ is 4004×14. Taking the B-spline basis for each dimension, we form the two dimensional basis by taking the row tensor of the two bases, which will be denoted as

$$\boldsymbol{B} = \boldsymbol{B}_Y \square \boldsymbol{B}_A. \tag{3.21}$$

This operation forms a rectangular grid of basis functions over the space covered by the observations. For the assured lives data this results in a 4004×266 matrix, with the central knots of the basis functions spread evenly over a rectangle whose bottom left point is at $(1942, 10)$ and top right point is at $(2007, 100)$. This is shown graphically in Fig. 3.3. As in the one-dimensional case the parameters in the model can be positioned at the central knot positions of their corresponding basis function, so the parameters also form a grid over the data. In order to enforce smoothness we place a penalty on coefficients in the same row and same column, one penalty in the age direction and one in the year direction. With reference to Fig. 3.3 we can imagine taking the first column of the coefficients and penalizing them as in the one dimensional case. Repeating this we obtain a set of differencing operations to be applied to the coefficients in each column. For computational purposes the coefficients are put into a vector in which the columns are stacked on top of each other. With the coefficients in this format the penalty in the age direction is given by

$$\boldsymbol{P}_A = \boldsymbol{I}_{c_y} \otimes (\boldsymbol{D}_a' \boldsymbol{D}_a), \tag{3.22}$$

where $\boldsymbol{D}_a$ is the $(c_a - d) \times c_a$ difference matrix, of order d, for age. For a penalty in the year direction we simply penalize coefficients in the same row as shown in Fig. 3.3, which leads to

$$\boldsymbol{P}_Y = (\boldsymbol{D}_y' \boldsymbol{D}_y) \otimes \boldsymbol{I}_{c_a}, \tag{3.23}$$

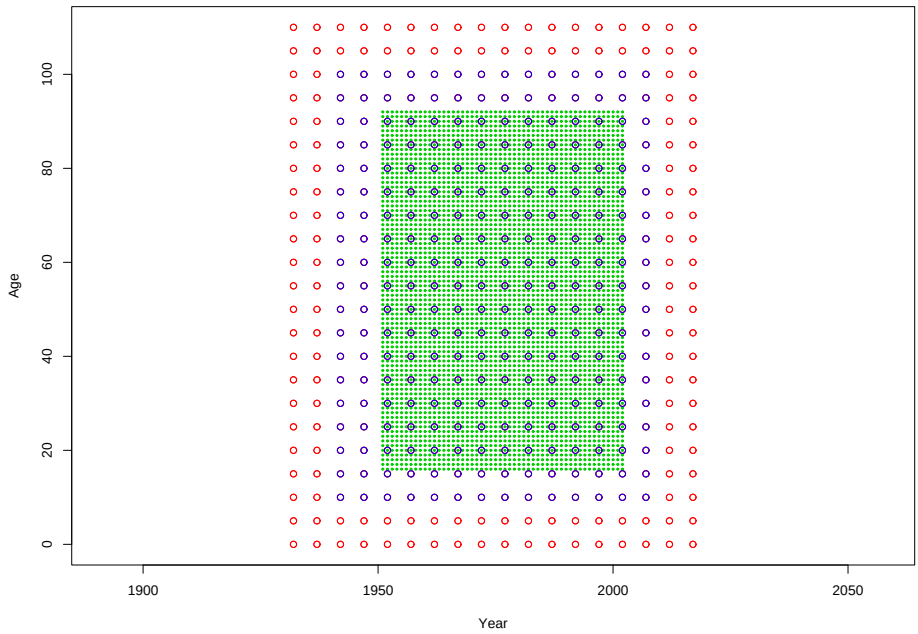


Figure 3.3: A graphical representation of the CMI mortality data. The green dots are the data points, the circles are the knot positions: the blue ones are the central knots which have a parameter attached, while the red knots are only used to define the splines on the edge of the data.

where $\boldsymbol{D}_y$ is the $(c_y - d) \times c_y$ difference matrix, of order d, for year. Weighting each of these penalties with a smoothing parameter we obtain the overall penalty

$$\boldsymbol{P} = \lambda_a \boldsymbol{P}_A + \lambda_y \boldsymbol{P}_Y. \tag{3.24}$$

With the basis and penalty in place, for given values of λ_a and λ_y we can fit our model using the penalized scoring algorithm in equation (2.75). The smoothing parameters can be selected in exactly the same way by optimizing some model selection criterion.

Using separate smoothing parameters increases the time required to fit the model, but this is necessary as in general there is no reason to suppose that the smoothness of the function should be isotropic. If there is evidence *a priori* that a single smoothing parameter is suitable then setting $\lambda = \lambda_a = \lambda_y$ will significantly reduce the computational time to fit the model.

An important special feature of the mortality data is the grid structure. When the data do not lie on a grid we are forced to form the basis using (3.20). The basis $\boldsymbol{B}_A$ is used to smooth the full two-dimensional grid by age only, and similarly the basis $\boldsymbol{B}_Y$

is used to smooth the grid by year. A single basis function for each of age and year is shown in Fig. 3.4. As we can see, the two-dimensional age spline is simply a spline from a one-dimensional spline basis for age repeated across all years (and vice-versa for the two-dimensional year spline). A basis function from the full two-dimensional basis is then a product of a two-dimensional age spline with a two-dimensional year spline, as shown in the top panel of Fig. 3.5. The row tensor of the two bases $\boldsymbol{B}_A$ and $\boldsymbol{B}_Y$ is a systematic way of performing all the cross-multiplications required to obtain the full two-dimensional basis in the lower panel of Fig. 3.5.

The method of forming a two-dimensional basis as the row tensor of the bases $\boldsymbol{B}_A$ and $\boldsymbol{B}_Y$ works in general for any configuration of the data and, in particular, does not require the data to lie on a grid. However, when the data lie on a grid the full two-dimensional basis can be obtained directly from the *marginal* one-dimensional bases, $\boldsymbol{B}_a$ and $\boldsymbol{B}_y$ as defined in Section 2.6.1 (page 45). First, we note that the indices of the data can be written as Kronecker products

$$\boldsymbol{x_A} = \mathbf{1}_{n_y} \otimes \boldsymbol{x_a} \qquad \text{and} \qquad \boldsymbol{x_Y} = \boldsymbol{x_y} \otimes \mathbf{1}_{n_a}, \tag{3.25}$$

and from this it is straightforward to see that

$$\boldsymbol{B_A} = \mathbf{1}_{n_y} \otimes \boldsymbol{B_a} \qquad \text{and} \qquad \boldsymbol{B_Y} = \boldsymbol{B_y} \otimes \mathbf{1}_{n_a}. \tag{3.26}$$

Using the definition of the row tensor in (3.19) we have

$$\begin{aligned} \boldsymbol{B_Y} \square \boldsymbol{B_A} &= (\boldsymbol{B_y} \otimes \mathbf{1}_{n_a}) \square (\mathbf{1}_{n_y} \otimes \boldsymbol{B_a}) \\ &= (\boldsymbol{B_y} \otimes \mathbf{1}_{n_a} \otimes \mathbf{1}'_{c_a}) * (\mathbf{1}'_{c_a} \otimes \mathbf{1}_{n_y} \otimes \boldsymbol{B_a}) \\ &= (\boldsymbol{B_y} \otimes \boldsymbol{J}_{n_a,c_a}) * (\boldsymbol{J}_{n_y,c_y} \otimes \boldsymbol{B_a}) \\ &= \boldsymbol{B_y} \otimes \boldsymbol{B_a}. \end{aligned} \tag{3.27}$$

Thus, when the data lie on a grid, the general row tensor method of forming the two-dimensional basis can be replaced by the more direct Kronecker product of the marginal bases.

In summary, we have data $\boldsymbol{y} = \text{vec}(\boldsymbol{Y})$, $\boldsymbol{e} = \text{vec}(\boldsymbol{E})$, regression matrix $\boldsymbol{B} = \boldsymbol{B}_y \otimes \boldsymbol{B}_a$, and penalty

$$\boldsymbol{P} = \lambda_a(\boldsymbol{I}_{n_y} \otimes \boldsymbol{D}'_{\boldsymbol{a}}\boldsymbol{D_a}) + \lambda_y(\boldsymbol{D}'_{\boldsymbol{y}}\boldsymbol{D_y} \otimes \boldsymbol{I}_{n_a}). \tag{3.28}$$

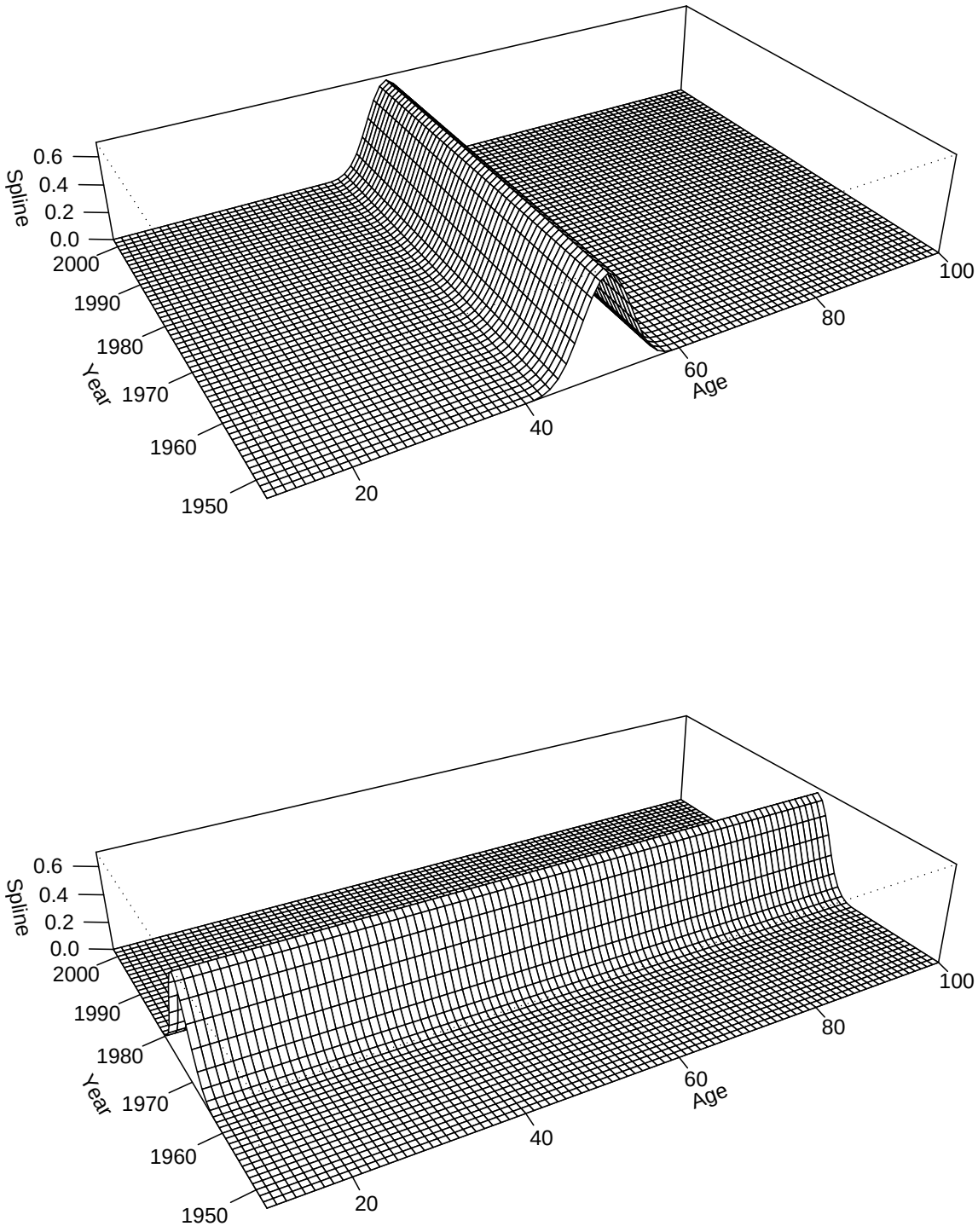


Figure 3.4: Coverage of a single age spline, $B_{a,i}(\cdot)$, on a two-dimensional data set (top), and for a single year spline, $B_{y,j}(\cdot)$ (bottom).

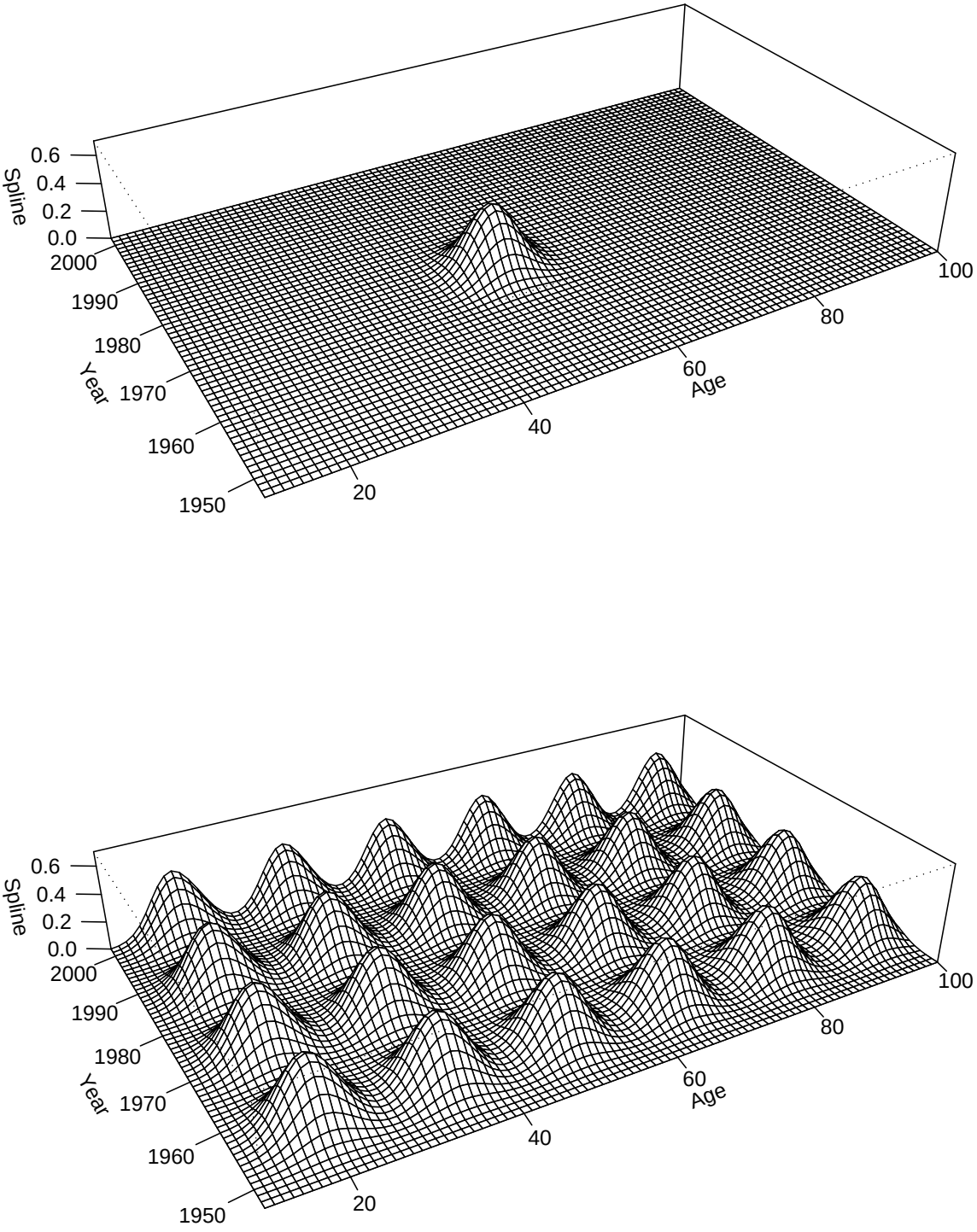


Figure 3.5: Top panel shows the coverage of a single two-dimensional spline, $B_{y,j}(\cdot) \otimes B_{a,i}(\cdot)$ (the result of multiplying the two splines in Fig. 3.4). Bottom panel shows a subset of a two-dimensional basis, $\boldsymbol{B_y} \otimes \boldsymbol{B_a}$.

With the data in matrix-vector format, and the basis and penalty specified, we can proceed to fit the model using the penalized scoring algorithm (2.75). The smooth fitted to the assured lives mortality data is shown in Fig. 3.6. The smoothing parameters λ_a and λ_y are chosen by minimizing BIC. We find $\lambda_a = 80.8$ and $\lambda_y = 623.5$; strong evidence that an isotropic smooth is not appropriate in this example.

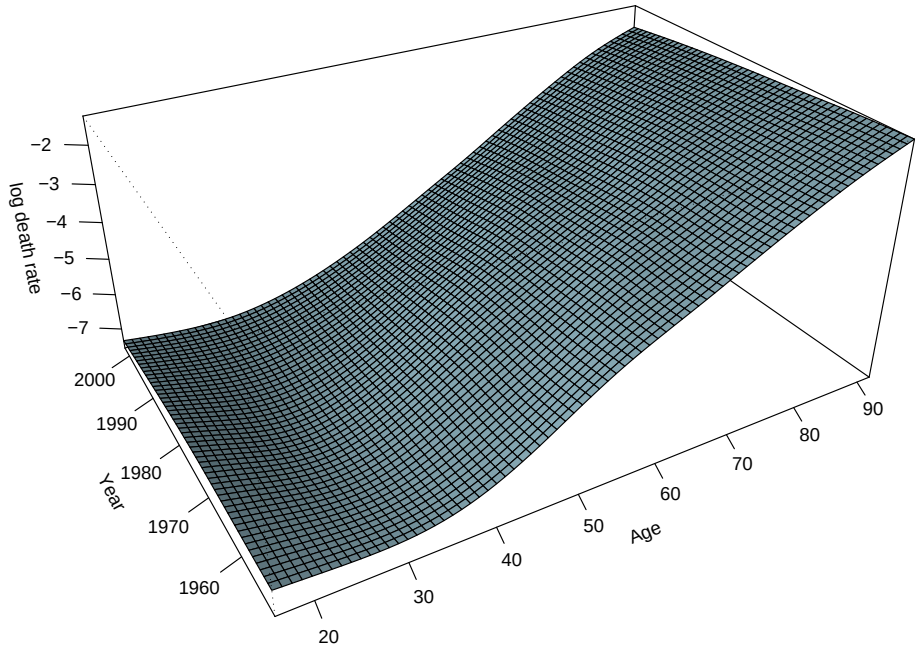


Figure 3.6: The mortality surface shown in Fig. 1.2(a) smoothed by a two-dimensional P-spline model.

3.3.3 Mixed Model Representation

Multi-dimensional P-splines also have a mixed model representation. Currie et al. (2006) showed that we can find an equivalent basis for a mixed model by working in the eigenspace of the penalty matrices.

In Section 2.5 we showed that in one dimension we can find a transformation from the original B-spline basis to an equivalent basis which leads to a mixed model representation. In more than one dimension we can use the singular value decomposition of the marginal penalty matrices $\boldsymbol{D}_a'\boldsymbol{D}_a$ and $\boldsymbol{D}_y'\boldsymbol{D}_y$ in (3.24) to form the fixed and random parts of the new basis and new penalty matrix. In one dimension it is straightforward to remove the zero eigenvalues to give the random part of the

basis, as shown in Section 2.5.3. However, in more than one dimension the Kronecker product structure of the penalty matrices complicates the transformation because the zero eigenvalues are mixed up in the transformed penalty matrix, and special steps need to be taken to remove them; see Currie et al. (2006) for details of this in two dimensions.

Table 3.1 shows a comparison of numerical output of various models fitted to the assured lives data. The models were fitted by varying the distance between knots (dx), while keeping a knot at the point $(65, 2002)$; for each value of dx three models were fitted: selecting the smoothing parameters using AIC, BIC, and REML (using the two-dimensional mixed model formulation). As expected, selecting the smoothing parameter using BIC imposes much heavier smoothing than if we use AIC; we also find that using REML also results in lighter smoothing than if we use BIC. Another point worth noting is that in this example the penalty does not seem to enforce the same amount of smoothing as we increase the number of basis functions (reduce dx). We see that for smaller values of dx the effective dimensions of the model ($\text{tr}(\boldsymbol{H})$) are higher than for larger dx, though there appears to be more consistency if we select the model using BIC than with either AIC or REML.

3.3.4 Model selection

Once the basis and penalty have been chosen for the model we must choose a criterion for selecting our model. The premise of P-splines is that the number of splines in our basis should not dramatically effect the fit of the model. Table 3.1 shows the output from a P-spline model fitted to the assured lives data with the smoothing parameter selected by AIC, BIC and REML. The most striking feature of these data is the dramatic difference in the degrees of freedom of the fitted model depending on which criterion is used. As expected the AIC prefers more flexible models to the BIC due to the lower weight given to model complexity in (2.35). We also note that the REML criterion also tends to select more flexible models than BIC; this is perhaps evidence to support Kauermann (2005) who used an asymptotic argument to show that REML has a tendency to under smooth in large sample problems. It is also worth noting that the degrees of freedom of the selected model is quite variable, tending to

increase as we introduce more basis functions. Over all BIC appears to be the best model selection criterion for this problem; in particular we note that the effective dimensions of the model appears least effected by the size of the basis when we select by BIC.

3.3.5 The general multidimensional P-spline model

The method used for the examples in the previous section can be extended to give a method for data of any dimension. In the general case, our data will be points in $\mathbb{R}^{d+1}$, the first d dimensions establishing the position of the data point, the $(d+1)$ th dimension contain the values we wish to smooth. Our aim is to fit a hyper-surface through the points, which varies smoothly over the first d dimensions.

In the general d-dimensional case we will have a vector of observations on the response variable $\boldsymbol{y}$, and a set of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$ containing the position of each data point. First we form the basis for the model as

$$\boldsymbol{B} = \boldsymbol{B}_d \square \boldsymbol{B}_{d-1} \square \cdots \square \boldsymbol{B}_1, \tag{3.29}$$

where

$$\boldsymbol{B}_i = B_i(\boldsymbol{x}_i) \qquad \text{for } i = 1, \ldots, d. \tag{3.30}$$

Then, as in the two-dimensional case, there is one penalty for each dimension, with the overall penalty defined as:

$$P = \sum_{i=1}^{d} \lambda_i \left(\bigotimes_{j=i+1}^{d} \boldsymbol{I}_{c_j} \right) \otimes \boldsymbol{D}_i' \boldsymbol{D}_i \otimes \left(\bigotimes_{j=1}^{i-1} \boldsymbol{I}_{c_j} \right). \tag{3.31}$$

A special case in d-dimensions occurs when the data lie in an array. Precisely, the data would lie in a d-dimensional array, with each dimension indexed by a response variable, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$. The mortality data is a two-dimensional example of this structure, and the d-dimensional fitting procedure is a generalisation of this example. We form what would be the one-dimensional marginal basis for each dimension

$$\boldsymbol{B}_i = B_i(\boldsymbol{x}_i), \quad i = 1, \ldots, d, \tag{3.32}$$

the full basis for the model is then the Kronecker product of these marginal bases:

$$\boldsymbol{B} = \bigotimes_{i=1}^{d} \boldsymbol{B}_i. \tag{3.33}$$

The penalty for this model is the same as in the scattered case, given in expression (3.31).

The models described in this chapter are not practical to implement beyond three dimensions. If we imagine a four-dimensional example with only ten basis functions used for the marginal bases of each dimension, then the full four-dimensional basis still has ten thousand columns. Using the penalized scoring algorithm in this case requires solving a system of ten thousand equations, beyond the capability of most computers currently available. Even in the three-dimensional case we are at the limit of most machines if we try and solve with one thousand parameters. Solving the equations is not the only problem: simply storing the regression matrix can be quite a restriction on the size of models that can be fitted. Taking the two-dimensional mortality data as an example and using one knot every five years, we end up with a regression matrix that will be 4004×266. Fortunately, as we will see in the next chapter, there are methods that can avoid the calculation and storage of this matrix.

3.3.6 Forecasting

Following the discussion of multi-dimensional smoothing in the previous section and the discussion of one-dimensional forecasting in Section 2.8 we now present two-dimensional forecasting using P-splines with reference to the assured lives data. We seek a forecast of the mortality schedule across ages for, say, twenty years into the future. As in Section 2.8 we follow the method proposed by Currie et al. (2004) with the modification described in Section 2.8 to improve computational efficiency.

As in the one-dimensional case forecasting is treated as a missing data problem. We first extend the year index to cover the years included in the forecast. The B-spline basis is then evaluated over the extended index to give the extended regression matrix $\boldsymbol{B}_y^*$; we then remove the extra rows from $\boldsymbol{B}_y^*$ to give

$$\boldsymbol{B}_y^+ = [\, \boldsymbol{B}_y \vdots \mathbf{0} \,]; \tag{3.34}$$

this is exactly the same as the one-dimensional case described in Section 2.8. The two-dimensional basis is then obtained using the Kronecker product, $\boldsymbol{B}^+ = \boldsymbol{B}_y^+ \otimes \boldsymbol{B}_{\boldsymbol{a}}$. The penalty is again applied to coefficients in the same row and column using the

Table 3.1: Numerical output from the two-dimensional P-spline model, showing the values of the smoothing parameters, the effective dimensions, and the deviance for varying dx and with different model selection criteria

dx	λ_a	λ_a	tr($\boldsymbol{H}$)	Deviance	AIC	BIC	REML
10	0.000	0.000	104.094	6830.045	**7038.233**	7692.778	6283.114
	5.732	22.347	46.382	7028.953	7121.717	**7413.368**	6723.236
	0.770	5.239	56.052	6972.281	7084.386	7436.844	**6765.587**
9	0.206	0.315	78.917	6816.493	**6974.327**	7470.563	6767.044
	9.680	25.400	51.373	6948.020	7050.766	**7373.800**	6745.888
	1.056	4.479	65.278	6860.360	6990.916	7401.385	**6798.962**
8	0.000	4.804	85.718	6787.319	**6958.756**	7497.755	6672.973
	13.162	68.034	53.695	6923.858	7031.247	**7368.881**	6748.397
	1.570	11.196	71.495	6825.068	6968.058	7417.621	**6812.571**
7	0.033	12.932	92.204	6725.279	**6909.687**	7489.469	6794.877
	28.572	107.510	57.709	6883.627	6999.045	**7361.922**	6737.290
	2.470	18.132	81.798	6757.690	6921.285	7435.632	**6831.559**
6	1.438	24.289	101.638	6654.757	**6858.032**	7497.132	6849.916
	36.252	239.439	62.224	6834.612	6959.060	**7350.330**	6754.482
	3.504	33.870	94.521	6671.563	6860.604	7454.954	**6855.130**
5	5.211	49.778	115.217	6628.373	**6858.807**	7583.297	6853.599
	74.892	627.143	63.153	6841.629	6967.935	**7365.043**	6721.439
	6.851	76.969	106.798	6646.565	6860.160	7531.707	**6855.904**
4	10.471	0.653	212.352	6327.391	**6752.096**	8087.374	6778.813
	144.089	1363.977	67.462	6805.713	6940.637	**7364.841**	6713.590
	12.497	126.742	130.762	6530.646	6792.170	7614.409	**6878.780**
3	18.370	2.715	292.246	6040.743	**6625.235**	8462.889	6819.492
	290.978	3640.963	71.371	6786.123	6928.866	**7377.651**	6698.222
	23.817	249.840	162.661	6404.315	6729.638	7752.457	**6897.654**

penalty in (3.24). For efficiency $\boldsymbol{B}^+$ is used in the scoring algorithm, and, once the smoothing parameters and regression coefficients have been selected, the full model matrix, $\boldsymbol{B}^* = \boldsymbol{B}^*_y \otimes \boldsymbol{B}_{\boldsymbol{a}}$, can be used to obtain the forecasted values.

We use the assured lives data to illustrate the forecasting method for a twenty year forecast. In Section 2.8 we defined $\boldsymbol{B}^*_y$ and $\boldsymbol{B}^+_y$, which are 72×18 and 52×18 respectively and from the previous section we have $\boldsymbol{B}_{\boldsymbol{a}}$, 77×19, giving $\boldsymbol{B}^+$ dimension 4004×342. Figure 3.7 shows the smoothed and forecasted log mortality rates with the smoothing parameter selected by BIC (as mentioned in Section 2.4 AIC tends to under smooth with large data sets). Figure 3.8 shows the cross sections of the age profile changing over time. Figure 3.9 shows individual age cross sections in more detail, plotted with the raw data points and with a 95% confidence interval calculated as the diagonal elements of (2.76). It is not easy to see from this plot, but the forecast is no longer linear for each age. In the two-dimensional model there are two penalties acting on the coefficients in the forecast region with each attempting to enforce linearity in that direction. For this particular data set the forecasting procedure appears relatively successful; this is because a larger smoothing parameter was selected for year than for age, so the penalty is more inclined to force linearity in the year direction.

Sometimes, however, the combination of second order penalties in each direction can lead to strange results when forecasting. For example, if we take population mortality data from England and Wales (ages $20-89$, and years $1962-2002$) and perform a forecast using the standard penalty we obtain the forecast shown in Fig. 3.10 and Fig. 3.11. We see that as the forecast moves further into the future the limiting function in the age direction takes over and linearity is enforced in the age direction. The dominance of the age penalty in the forecast region is caused by a larger smoothing parameter for age than year being selected to smooth the surface where we have data; but when the data runs out the penalties take over and result in this strange forecast. One could solve this problem by tampering with smoothing parameters until a satisfactory forecast was obtained. Changing the smoothing parameters to alter the forecast is rather subjective and we would prefer a methodology that automatically produced a sensible forecast. A possible solution to this problem is to change the

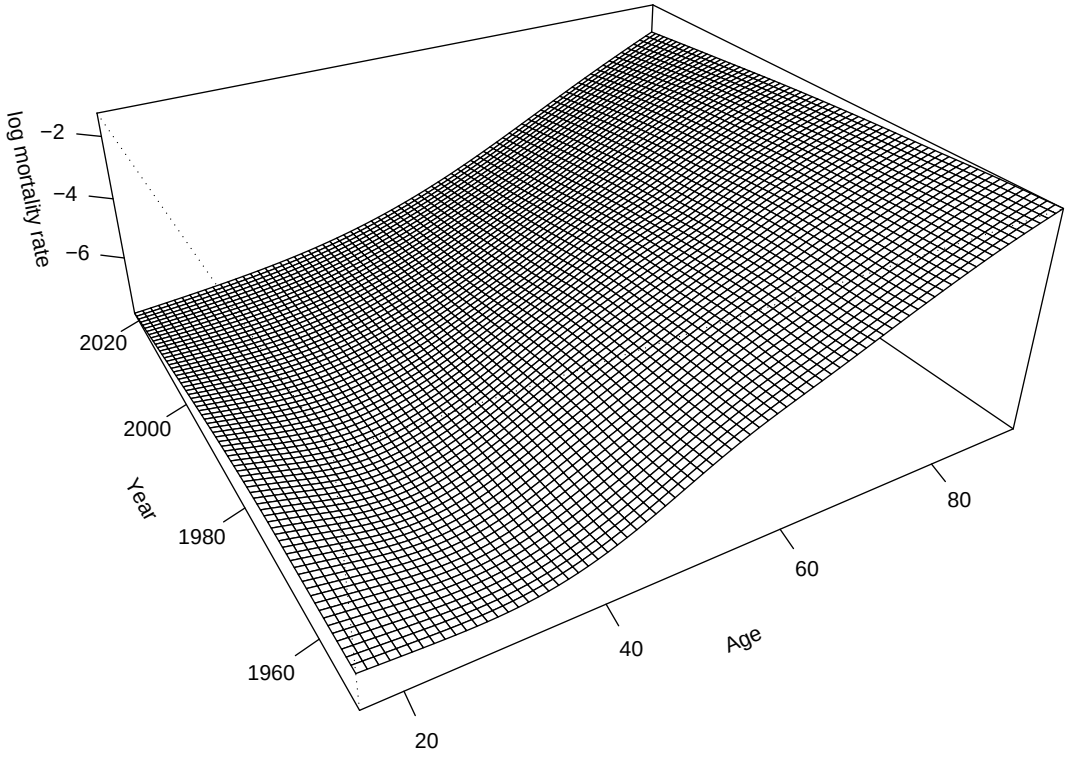


Figure 3.7: Perspective plot of the forecasted mortality surface.

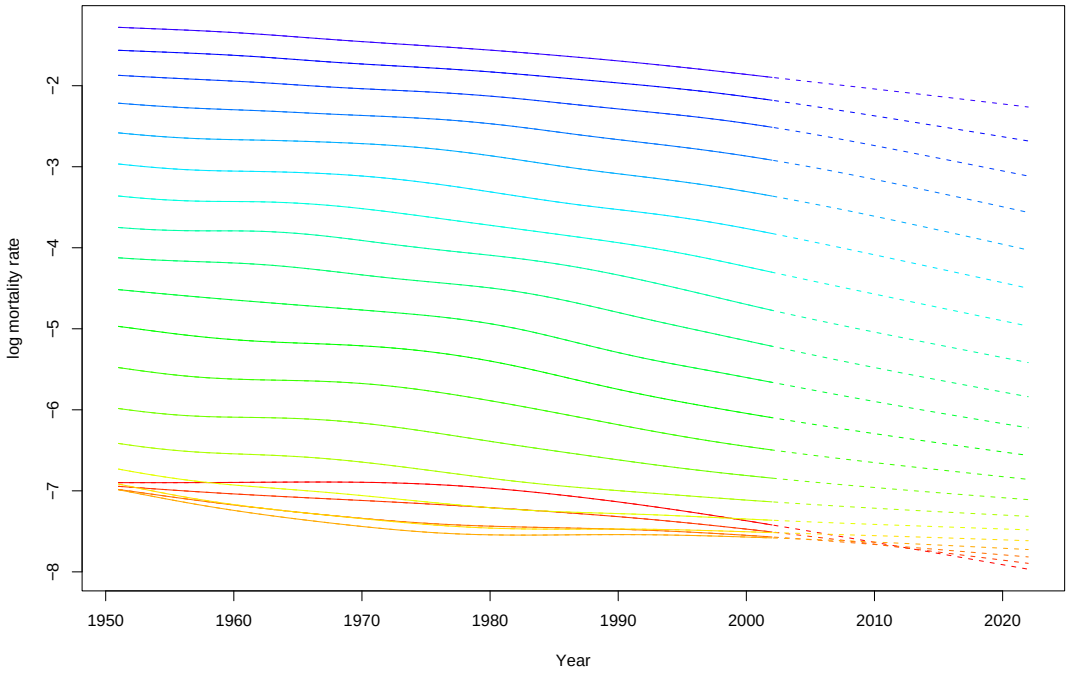


Figure 3.8: Age cross-section of the forecasted mortality surface.

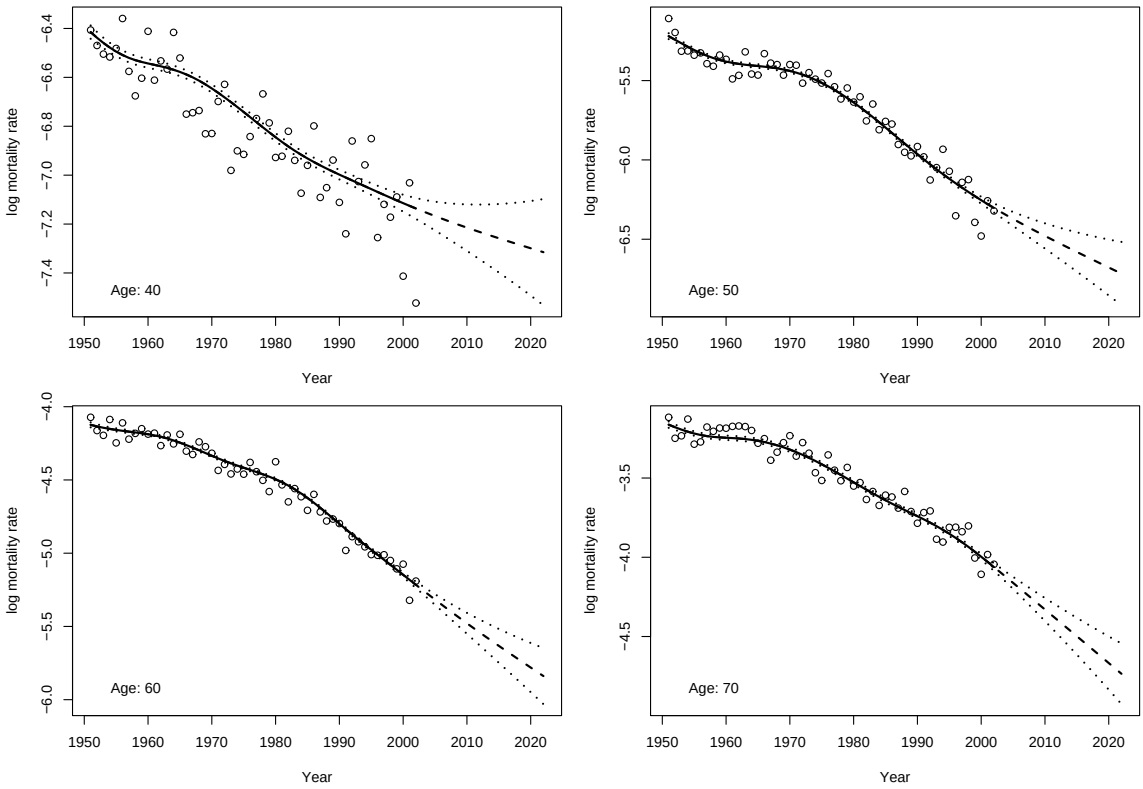


Figure 3.9: Plots of the age cross-sections with 95% confidence intervals.

structure of the penalties, and thus the limiting functions obtained in the forecast. Using P-splines in this problem, we benefit from the understanding we have of how the basis functions behave in conjunction with the penalty: we can design a penalty to improve the forecast. We could start by increasing the order of the penalty in the age direction; a third or fourth order penalty would result in a quadratic or cubic function in the age direction in the forecast region, but we are then faced with the same problems with polynomials that were described in Section 2.1. An alternative solution is to replace the age penalty with a cross-penalty term of the form

$$\boldsymbol{P}_c = \lambda_c \boldsymbol{D}_y' \boldsymbol{D}_y \otimes \boldsymbol{D}_a' \boldsymbol{D}_a. \tag{3.35}$$

This can be interpreted as a penalty that maintains differences in differences. Figure 3.12 and Fig. 3.13 show the results of forecasting using the England and Wales population data using the penalty

$$\boldsymbol{P} = \lambda_c (\boldsymbol{D}_{y,1}' \boldsymbol{D}_{y,1} \otimes \boldsymbol{D}_{a,1}' \boldsymbol{D}_{a,1}) + \lambda_y (\boldsymbol{D}_{y,2}' \boldsymbol{D}_{y,2} \otimes \boldsymbol{I}_{c_a}), \tag{3.36}$$

where $\boldsymbol{D}_{y,1}$ and $\boldsymbol{D}_{a,1}$ are first order difference matrices for age and year respectively, and $\boldsymbol{D}_{y,2}$ is a second order difference matrix for year. Using this combination of cross-

penalty with a second order year penalty the forecast seems to follow the trend in the data while retaining the familiar age structure in the forecast region, as shown in Fig. 3.13.

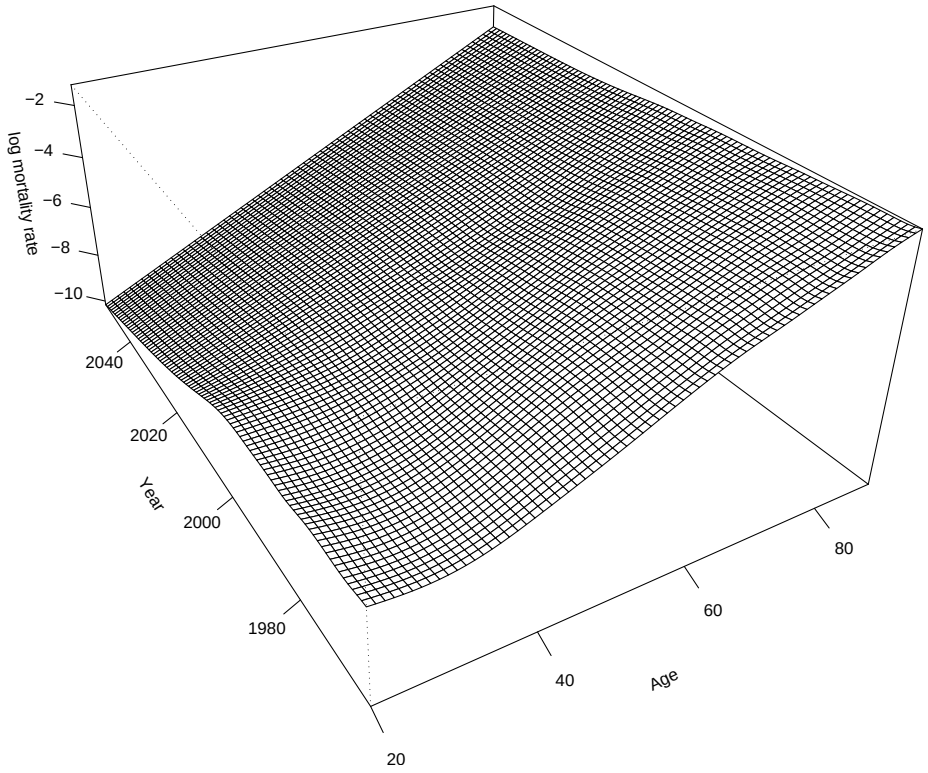


Figure 3.10: Perspective plot of the two-dimensional mortality surface for England and Wales population data using the standard penalty in (3.28).

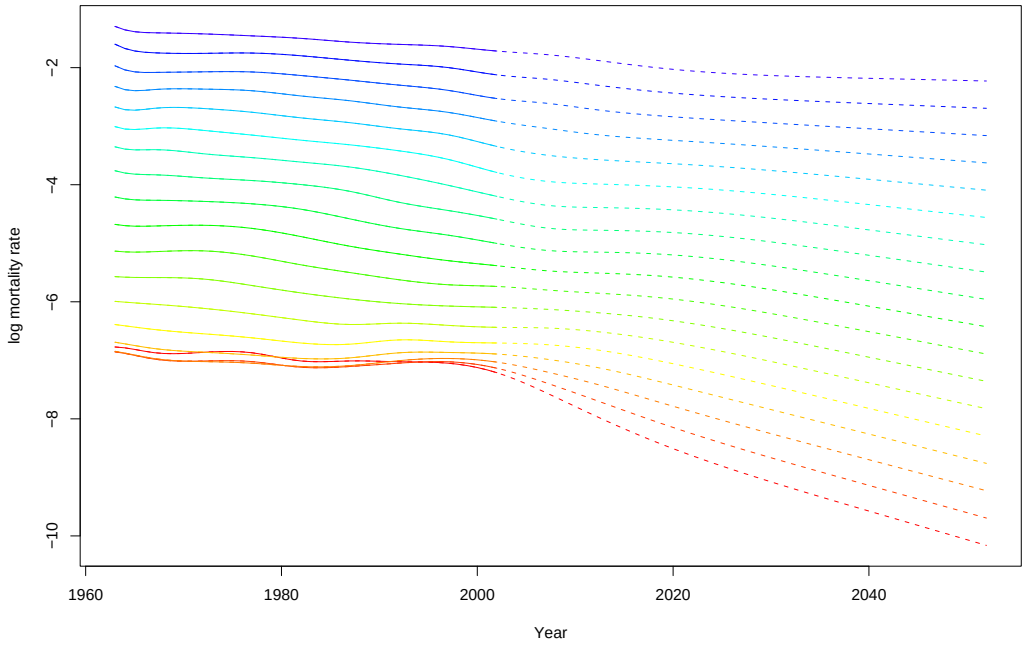


Figure 3.11: Cross-section plot of the two-dimensional mortality surface for England and Wales population data using the standard penalty in (3.28).

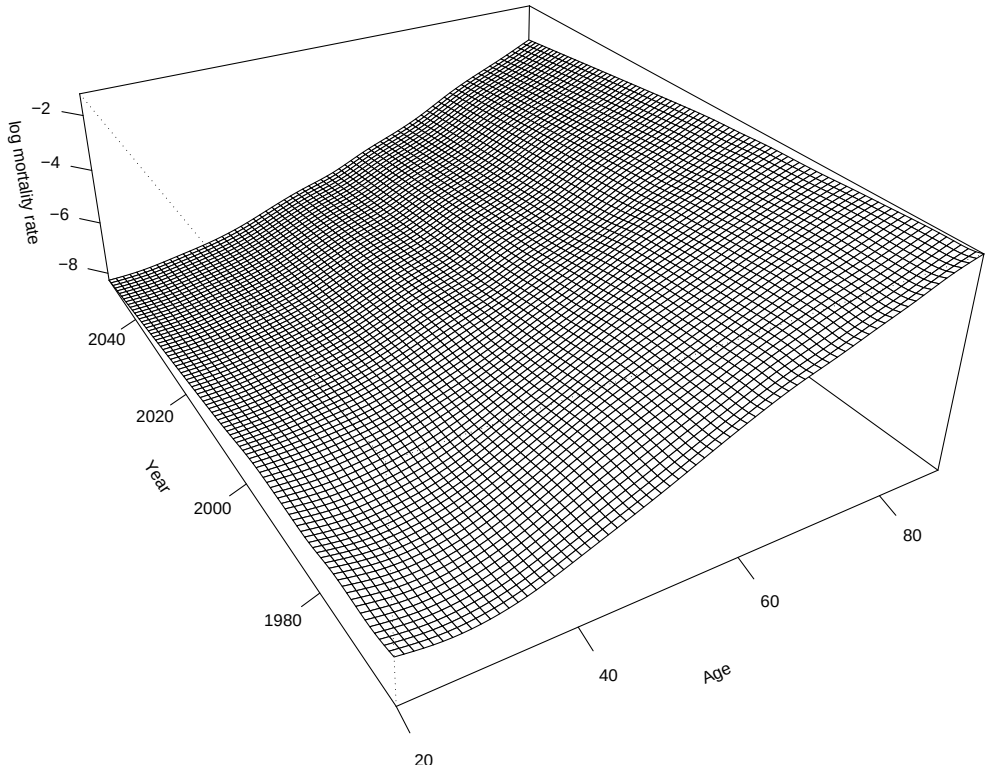


Figure 3.12: Perspective plot of the two-dimensional mortality surface for England and Wales population data using the cross penalty in (3.36).

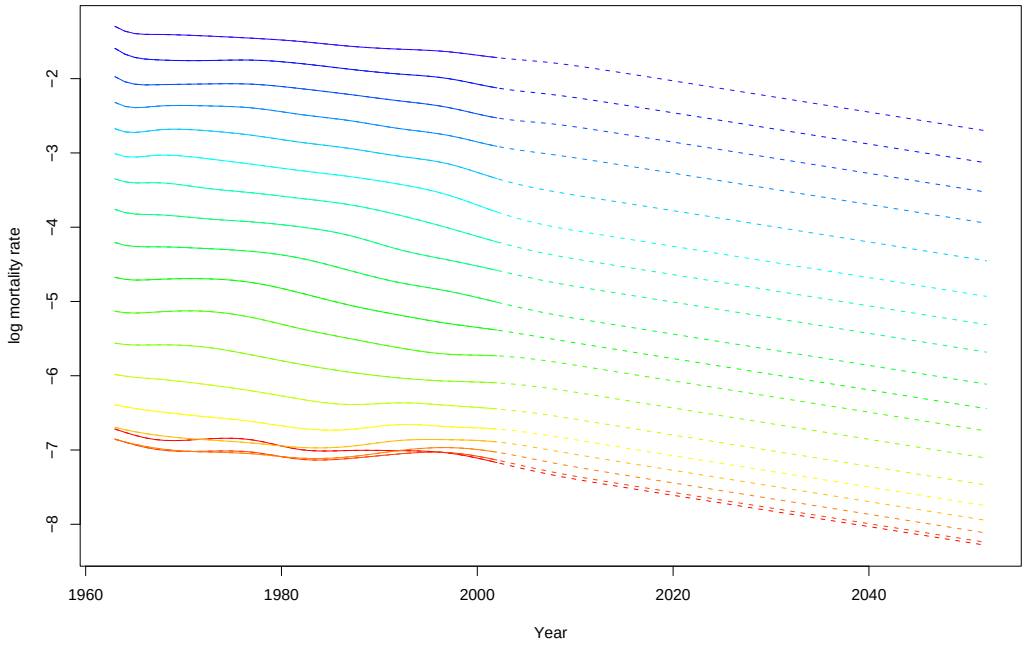


Figure 3.13: Perspective plot of the two-dimensional mortality surface for England and Wales population data using the cross penalty in (3.36).

Chapter 4

Array Methods

In the previous chapter we looked at multidimensional smoothing, where it was noted that as the number of dimensions increases we can run into computational problems. In this chapter we look at some methods that can be used to reduce the computational overhead for models with a grid structure on the observations. Although these methods cannot help in high dimensional problems where the sheer number of parameters and data points becomes a problem, they will significantly reduce the storage as well as the computational requirements in two- and three-dimensional problems. These methods contain, as a special case, a version of *Yates's Algorithm* (Yates, 1937). Some of the identities described had been noted by de Boor (1979), however it was first Eilers et al. (2006) and then Currie et al. (2006) who showed how these methods could be extended and exploited to fit models with a Kronecker product structure without multiplying up the full basis, and in the process make storage and computational savings.

This chapter describes array methods in general terms; in the next chapter we will apply these methods to various models of mortality.

4.1 Introduction

The content of this chapter follows from a well known identity that can be found in most books on linear algebra (see for example Harville (1997))

$$(\mathbf{B} \otimes \mathbf{A})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{A}\mathbf{X}\mathbf{B}') \tag{4.1}$$

for any conformable matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{X}$. What is not widely recognised is that the form on the right is more efficient in terms of computation and storage. The savings in storage are obvious, as on the left hand side we have to multiply up and store the matrix $\mathbf{B} \otimes \mathbf{A}$, whereas on the right hand side the multiplication is done separately with $\mathbf{A}$ and $\mathbf{B}$. We also get a surprise computational saving, as the number of multiplications required on the right hand side is an order of magnitude less than required on the left.

We begin by looking at Yates's algorithm in Section 4.2, which has been used in factorial design problems. We show that this can be viewed as an alternative form of (4.1). However, we go on to show that the form in (4.1) lends itself more easily to higher dimensional generalizations, and also allows us to generalize to matrix operations other than the matrix-vector operation.

In Section 4.3 we will see how this type of matrix-vector multiplication can be extended to higher dimensions, and go on to show in Section 4.4 that similar methods are available for other frequently used matrix operations. In Section 4.5 we give a comparison of computational results for the array methods and traditional methods. Finally, in Section 4.6 we show how an array structure can sometimes be obtained even if initially the data do not seem to fit such a structure.

4.2 Yates's Algorithm

The original motivation for Yates's Algorithm was in the analysis of factorial design experiments. The algorithm is best explained with reference to an example, so we will describe how it is used in a 2^3 factorial experiment. In a 2^3 factorial experiment we wish to analyze the effects of three factors (each with two levels) and their interactions on a response variable. To analyze the effect of each factor we use the contrast matrix

$$\boldsymbol{X}_i = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{4.2}$$

for $i = 1, 2, 3$. In order to analyze the factors together with all interactions we use the Kronecker product of the individual contrast matrices

$$\boldsymbol{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix}. \tag{4.3}$$

Essentially Yates's algorithm is based on the observation that $\boldsymbol{X}$ has a cube root, $\breve{\boldsymbol{X}}^3 = \boldsymbol{X}$, (Gower, 1982) with the very simple form

$$\breve{\boldsymbol{X}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}. \tag{4.4}$$

We can then evaluate $\boldsymbol{X}\boldsymbol{y}$ recursively using $\breve{\boldsymbol{X}}(\breve{\boldsymbol{X}}(\breve{\boldsymbol{X}}\boldsymbol{y}))$ with the equivalent of sparse matrix methods to avoid onerous multiplications by zero. With this description of the Yates's algorithm it is easy to see it is a special case of an algorithm due to Good (1958), who showed that for the square matrices $\boldsymbol{A}$ $(m \times m)$ and $\boldsymbol{B}$ $(n \times n)$ that

$$\boldsymbol{B} \otimes \boldsymbol{A} = \tilde{\boldsymbol{B}}\tilde{\boldsymbol{A}} \tag{4.5}$$

where

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{a}_{1\cdot} \otimes \boldsymbol{I}_n \\ \vdots \\ \boldsymbol{a}_{m\cdot} \otimes \boldsymbol{I}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_{1\cdot} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{a}_{1\cdot} & \dots & \boldsymbol{0} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{a}_{1\cdot} \\ \boldsymbol{a}_{2\cdot} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{a}_{2\cdot} & \dots & \boldsymbol{0} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{a}_{m\cdot} \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{b}_{1\cdot} \otimes \boldsymbol{I}_m \\ \vdots \\ \boldsymbol{b}_{n\cdot} \otimes \boldsymbol{I}_m \end{bmatrix} \tag{4.6}$$

and $\boldsymbol{a}_{i\cdot}$ and $\boldsymbol{b}_{i\cdot}$ are the i th rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively.

We can use (4.5) to efficiently calculate the matrix-vector product

$$(\boldsymbol{B} \otimes \boldsymbol{A})\boldsymbol{x} = \tilde{\boldsymbol{B}}(\tilde{\boldsymbol{A}}\boldsymbol{x}), \tag{4.7}$$

where sparse matrix methods are used to efficiently multiply by $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$. Comparing (4.1) with (4.7) we can see that the methods basically amount to the same thing: splitting the multiplication of the Kronecker product into its components and multiplying them separately. The only difference is that in (4.1) we ensure the components conform by re-shaping the vector, and in (4.7) we reposition the components of the Kronecker product. However, the method in (4.1) has two advantages. Firstly, it can be applied when $\boldsymbol{A}$ and $\boldsymbol{B}$ are not square, and secondly, it has a natural extension to multiple Kronecker products, as shown in the next section.

4.3 Higher Dimensions

In this section we will show how we can extend the identity in (4.1) to higher dimensions. For example, how could we efficiently perform the multiplication

$$(\boldsymbol{C} \otimes \boldsymbol{B} \otimes \boldsymbol{A})\boldsymbol{x}, \tag{4.8}$$

using a method similar to (4.1)? To solve this problem we first note that the right hand side of (4.1) can be written as

$$\operatorname{vec}\Big(\big(\boldsymbol{B}(\boldsymbol{A}\boldsymbol{X})'\big)'\Big). \tag{4.9}$$

We see that the operation is now performed by taking $\mathbf{A}$ and $\mathbf{B}$ in turn, and multiplying them onto the correct dimension of $\mathbf{X}$ (using the transpose). Taking (4.9) as a base case, one might expect to be able to write something like

$$(\boldsymbol{C} \otimes \boldsymbol{B} \otimes \boldsymbol{A})\text{vec}(\boldsymbol{X}) = \text{vec}\Big(\Big(\boldsymbol{C}\big(\boldsymbol{B}(\boldsymbol{A}\boldsymbol{X})'\big)'\Big)'\Big). \tag{4.10}$$

However, looking at this more closely we see that unless the number of rows in $\mathbf{A}$ is equal to the number of columns in $\mathbf{C}$ then the right hand side does not conform, because the matrix $\boldsymbol{C}$ is being multiplied on to the same dimension as $\boldsymbol{A}$. In order for (4.10) to behave in the same way as (4.1) we require that $\boldsymbol{X}$ has three dimensions. However, the normal rules of linear algebra do not apply to three-dimensional arrays, so in order to compute (4.10) we need to define how a matrix is multiplied onto an array with more than two dimensions, and also define the transpose of an array with three or more dimensions. A matrix-array multiplication is a natural extension of matrix multiplication.

Definition 4.1 *For the $n_1 \times c_1$ matrix $\boldsymbol{A}$ and the $c_1 \times c_2 \times \cdots \times c_d$ array $\boldsymbol{X}$, the product $\boldsymbol{A}\boldsymbol{X}$ is the $n_1 \times c_2 \times \cdots \times c_d$ array whose $i_1 i_2 \ldots i_d^{th}$ element is*

$$\sum_{s=1}^{c_1} a_{i_1,s} x_{s,i_2,\ldots,i_d}. \tag{4.11}$$

Clearly for $d = 2$, $\boldsymbol{X}$ is a matrix and (7.1) defines the usual matrix product. This definition can be interpreted as a regular matrix product by contracting the second to d^{th} dimensions of $\boldsymbol{X}$ into one, giving a $c_1 \times (c_2 \ldots c_d)$ matrix and then multiplying $\boldsymbol{A}\boldsymbol{X}$ in the usual way. However, this interpretation does little to improve our understanding of (4.10), and we prefer to interpret $\boldsymbol{X}$ as an array.

The transpose of a d-dimensional array is achieved simply by permuting its dimensions.

Definition 4.2 *The transpose, $\boldsymbol{X}'$, of the $c_1 \times c_2 \times \cdots \times c_d$ array $\boldsymbol{X}$, is the $c_2 \times \cdots \times c_d \times c_1$ array such that*

$$x'_{i_2,\ldots,i_d,i_1} = x_{i_1,i_2,\ldots,i_d}. \tag{4.12}$$

Once again for $d = 2$ we obtain the usual definition of a matrix transpose.

With these definitions it is possible to efficiently multiply any Kronecker product with a vector. For the product

$$(\boldsymbol{A}_d \otimes \cdots \otimes \boldsymbol{A}_1)\boldsymbol{x}, \tag{4.13}$$

where $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_d$ are $n_1 \times c_1, \ldots, n_d \times c_d$ dimensional matrices respectively and $\boldsymbol{x}$ is a vector of dimension $c_1 \ldots c_d$, the algorithm is as follows

- Re-dimension $\boldsymbol{x}$ into a d-dimensional array $\boldsymbol{X}$, $c_1 \times \cdots \times c_d$, such that $\boldsymbol{x} = \text{vec}(\boldsymbol{X})$
- Perform the following multiplications and transpositions

$$\boldsymbol{AX} = \Big(\boldsymbol{A}_d \ldots \big(\boldsymbol{A}_2(\boldsymbol{A}_1\boldsymbol{X})'\big)'\Big)'. \tag{4.14}$$

- Re-dimension $\boldsymbol{AX}$ to give $\boldsymbol{Ax} = \text{vec}(\boldsymbol{AX})$.

A proof of (4.14) is given in Currie et al. (2006). A strong heuristic justification for the algorithm can be seen by comparing (4.9) with (4.14). In the two-dimensional case the two components $\boldsymbol{B}_1$, $n_1 \times c_1$, and $\boldsymbol{B}_2$, $n_2 \times c_2$, of the Kronecker product, $\boldsymbol{B}_2 \otimes \boldsymbol{B}_1$, are taken in turn and multiplied onto the appropriate dimension of the matrix $\boldsymbol{X}$, $c_1 \times c_2$. This method is then inducted into d dimensions in (4.14), where the d components of the Kronecker product are taken individually and multiplied onto the appropriate dimension of the array, $\boldsymbol{X}$, $c_1 \times \ldots \times c_d$. In fact to obtain the final answer, it does not matter in which order we multiply the components provided we multiply them onto the correct dimension. This can be easily seen in the two-dimensional case upon observing that

$$\big(\boldsymbol{B}(\boldsymbol{AX})'\big)' = \boldsymbol{AXB}' = (\boldsymbol{BX}'\boldsymbol{A}')' = \big(\boldsymbol{A}(\boldsymbol{BX}')'\big)'. \tag{4.15}$$

The algorithm (4.14) is just a convenient and systematic way of performing the multiplications.

To illustrate this interpretation Appendix C.1 shows a graphical representation of the method being applied in a three-dimensional case. As we see each matrix comes in turn to be multiplied onto the array, which is rotated after each multiplication so the next matrix is matched with the corresponding dimension of the array. An alternative method would be to multiply each matrix directly onto a specified dimension without rotation of the array, the graphical representation of this would show the matrices coming from different directions without the rotation of the array.

4.4 General Array Methods

4.4.1 Array Notation

In subsequent sections we will frequently be referring to multidimensional arrays. These arrays will often correspond to re-arrangements of matrices, so in order to make the connections between objects clear, we will describe our array notation in this section.

Providing a comprehensive array notation is quite challenging, and can be quite long-winded. Here we will simply specify a notation for our purpose; for a more complete solution see Harshman (2001). The main requirements of our notation will be to distinguish between a matrix and another array arrangement of its elements; but we will also need to be able to specify sub-arrays of an array. For a matrix $\boldsymbol{X}$ if we wish to specify some array arrangement of its elements we will represent this as $\overset{a}{\boldsymbol{X}}$, and for clarity we will also refer to $\boldsymbol{X}$ as $\overset{m}{\boldsymbol{X}}$. In order to select sub-arrays we will use a similar notation to our matrix notation, using the $\bullet$ symbol to represent an unspecified index. For a four-dimensional array $\overset{a}{\boldsymbol{X}}$, we would use $\overset{a}{\boldsymbol{X}}_{2,\bullet,4,\bullet}$ to represent the two-dimensional sub-array (matrix) of all the elements of $\overset{a}{\boldsymbol{X}}$ in the second position of the first dimension and fourth position of the third dimension. The precise dimensions of an array will be made clear in the situation. In particular, the vector $\boldsymbol{x}$, $n_1 \dots n_d \times 1$, has the array form $\overset{a}{\boldsymbol{X}}$, $n_1 \times \ldots \times n_d$, such that $\text{vec}(\overset{a}{\boldsymbol{X}}) = \boldsymbol{x}$.

4.4.2 Other Array Operations

The method described in (4.14) to efficiently multiply a Kronecker product matrix by a vector is interesting and is useful in other applications, but we can make the method more widely applicable by describing efficient methods for other matrix operations.

It was Currie et al. (2006) who first applied other Kronecker product methods to the penalised scoring algorithm

$$(\boldsymbol{B}'\tilde{\boldsymbol{W}}_\delta\boldsymbol{B} + \boldsymbol{P})\boldsymbol{\theta} = \boldsymbol{B}'\tilde{\boldsymbol{W}}_\delta\tilde{\boldsymbol{z}}, \tag{4.16}$$

where $\boldsymbol{B}$ is a Kronecker product. We introduce their methods in this context before describing a slightly different interpretation, and some generalizations. We will usually

write the diagonal weight matrix as $\boldsymbol{W}_\delta$ but occasionally we will write it as $\overset{m}{\boldsymbol{W}}_\delta$ to emphasize its matrix form.

The penalized scoring algorithm can be broken down into smaller components. The first requirement is to calculate the diagonal weight matrix, $\boldsymbol{W}_\delta$. For a penalized Poisson GLM with canonical link this is a diagonal matrix containing the current values of the mean

$$\boldsymbol{W}_\delta = \text{diag}(\boldsymbol{\mu}) = \text{diag}(g^{-1}(\boldsymbol{\eta})) = \text{diag}(\boldsymbol{e} * \exp(\boldsymbol{B\theta})), \tag{4.17}$$

where $\boldsymbol{e} = \text{vec}(\boldsymbol{E})$ is the vector of exposures, and $*$ indicates element-by-element multiplication. The non-zero elements of this matrix can be computed efficiently by using the algorithm (4.14) in the previous section to calculate $\boldsymbol{B\theta}$ as

$$\boldsymbol{B\theta} = \text{vec}\Big\{\Big(\boldsymbol{B}_d \ldots \big(\boldsymbol{B}_2(\boldsymbol{B}_1\boldsymbol{\Theta})'\big)'\Big)'\Big\}, \tag{4.18}$$

where $\boldsymbol{\Theta}$ is a d-dimensional array such that $\text{vec}(\boldsymbol{\Theta}) = \boldsymbol{\theta}$. In the two-dimensional case this is simply

$$\text{vec}(\boldsymbol{B}_1\boldsymbol{\Theta}\boldsymbol{B}_2'). \tag{4.19}$$

Next we have to calculate the working vector, $\boldsymbol{z}$. From (2.67) we have

$$\boldsymbol{z} = \boldsymbol{B\theta} + \boldsymbol{W}_\delta^{-1}(\boldsymbol{y} - \boldsymbol{\mu}). \tag{4.20}$$

We see that this only requires the efficient calculation of the linear predictor shown in (4.18). We can also use the same algorithm to evaluate the right hand side of (4.16). In matrix notation the product $\overset{m}{\boldsymbol{W}}_\delta\boldsymbol{z} = \text{diag}(\overset{m}{\boldsymbol{W}}_\delta) * \boldsymbol{z}$ is a vector of length $n_1 n_2 \ldots n_d$; this vector can be put into a $n_1 \times n_2 \times \ldots \times n_d$ dimensional array. Thus the product $\boldsymbol{B}'\overset{m}{\boldsymbol{W}}_\delta\boldsymbol{z}$ can be computed as

$$\text{vec}\Big\{\Big(\boldsymbol{B}_d' \ldots \big(\boldsymbol{B}_2'(\boldsymbol{B}_1'(\overset{a}{\boldsymbol{W}}_\delta * \overset{a}{\boldsymbol{Z}}))'\big)'\Big)'\Big\}, \tag{4.21}$$

where $\overset{a}{\boldsymbol{W}}_\delta$ is the $n_1 \times \ldots \times n_d$ dimensional array such that

$$\text{vec}(\overset{a}{\boldsymbol{W}}_\delta) = \text{diag}(\overset{m}{\boldsymbol{W}}_\delta) = \text{diag}(\boldsymbol{W}_\delta). \tag{4.22}$$

To compute the left hand side of (4.16) we require a new operation to deal with the inner product, $\boldsymbol{B}'\overset{m}{\boldsymbol{W}}_\delta\boldsymbol{B}$. In matrix form $\overset{m}{\boldsymbol{W}}_\delta$ is a diagonal matrix of dimension

$n_1 n_2 \ldots n_d \times n_1 n_2 \ldots n_d$, but we can efficiently calculate its diagonal elements using (4.18). Currie et al. (2006) show that if we keep the diagonal elements in array form, then the inner product will contain the same elements as the array product

$$\left(\boldsymbol{G}_d' \ldots \left(\boldsymbol{G}_2'(\boldsymbol{G}_1' \overset{a}{\boldsymbol{W}}_\delta)'\right)'\right)' \tag{4.23}$$

where $\boldsymbol{G}_i = \boldsymbol{B}_i \square \boldsymbol{B}_i$, where the row tensor, $\square$, is defined in (3.19). In order to match the elements of $\boldsymbol{B}' \overset{m}{\boldsymbol{W}}_\delta \boldsymbol{B}$ with (4.23) we can use the following algorithm:

- Take the output of (4.23), and factor out each dimension to give a $c_1 \times c_1 \times c_2 \times c_2 \times \cdots \times c_d \times c_d$ array.
- Permute the indices to give an $c_1 \times \cdots \times c_d \times c_1 \times \cdots \times c_d$ array
- Contract the first d dimensions and the last d dimensions to form a $c_1 \ldots c_d \times c_1 \ldots c_d$ dimensional matrix. This will equal $\boldsymbol{B}' \overset{m}{\boldsymbol{W}}_\delta \boldsymbol{B}$.

We will not reproduce the proof of this algorithm, which can be found in Currie et al. (2006), but instead give a more general explanation of why it works.

The algorithm works by taking advantage of two aspects of the structure of the matrix $\boldsymbol{B}' \overset{m}{\boldsymbol{W}}_\delta \boldsymbol{B}$. First the array structure of $\overset{a}{\boldsymbol{W}}_\delta$ allows us to sequentially multiply onto each dimension, so we achieve similar savings in storage and computation to those obtained in (4.18). Secondly the use of the row tensor, allows us to ignore the zero elements in $\overset{m}{\boldsymbol{W}}_\delta$. Even in ordinary linear algebra if we were to calculate an inner product with a diagonal matrix we would not store it as a full matrix but rather as a vector which is then multiplied directly onto one of the matrices in the inner product before multiplying them together. The array $\overset{a}{\boldsymbol{W}}_\delta$ should naturally be a $2d$-dimensional array, but we prefer to work with the d-dimensional sub-array which consists of its non-zero values. If $\overset{a}{\boldsymbol{W}}_\delta$ were in its $2d$-dimensional form we would first multiply $\boldsymbol{B}$ on to its "front" d dimensions, forming the $\boldsymbol{B}' \overset{m}{\boldsymbol{W}}_\delta$ part, and then multiply $\boldsymbol{B}$ onto its "back" d dimensions which gives the full product $\boldsymbol{B}' \overset{m}{\boldsymbol{W}}_\delta \boldsymbol{B}$ in array form. With $\overset{a}{\boldsymbol{W}}_\delta$ stored in its d-dimensional form there are simply not enough dimensions to pre- and post-multiply; however using the row tensor allows us to perform both multiplications at the same time. As a consequence of performing the multiplication in this way we obtain a strange d-dimensional array with its "front" and "back" dimensions mixed

up, and we have to factor and sort the dimensions in order to recover the familiar matrix arrangement of the elements.

The two algorithms above are required in order to fit the penalized scoring algorithm, but there are other algorithms for Kronecker products. For instance, in a penalized GLM the variances of the fitted values are given by

$$\text{diag}(\text{var}(\hat{\boldsymbol{\eta}})) = \text{diag}\Big(\boldsymbol{B}(\boldsymbol{B}'\boldsymbol{W}_\delta\boldsymbol{B} + \boldsymbol{P})^{-1}\boldsymbol{B}'\Big). \tag{4.24}$$

Remarkably, Currie et al. (2006) also show that the diagonal elements of (4.24) can be found efficiently using a now familiar expression

$$\Big(\boldsymbol{G}_d \ldots \big(\boldsymbol{G}_2(\boldsymbol{G}_1 \overset{a}{\boldsymbol{S}})'\big)'\Big)'; \tag{4.25}$$

here $\overset{a}{\boldsymbol{S}}$, $c_1^2 \times \ldots \times c_d^2$ is the array form of the matrix $\overset{m}{\boldsymbol{S}} = (\boldsymbol{B}'\boldsymbol{W}_\delta\boldsymbol{B} + \boldsymbol{P})^{-1}$, $c_1 \ldots c_d \times c_1 \ldots c_d$, although the identity holds for any matrix $\overset{m}{\boldsymbol{S}}$.

We now turn our attention to additive models. With two additive terms we consider a model of the form

$$\boldsymbol{A}\boldsymbol{\theta}_A + \boldsymbol{B}\boldsymbol{\theta}_B \tag{4.26}$$

where $\boldsymbol{A} = \boldsymbol{A}_d \otimes \cdots \otimes \boldsymbol{A}_1$ and $\boldsymbol{B} = \boldsymbol{B}_d \otimes \cdots \otimes \boldsymbol{B}_1$; here $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ are $n_i \times c_{a_i}$ and $n_i \times c_{b_i}$ dimensional matrices respectively. We require the evaluation of a partition matrix of the form

$$\begin{bmatrix} \boldsymbol{A}'\boldsymbol{W}_\delta\boldsymbol{A} & \boldsymbol{A}'\boldsymbol{W}_\delta\boldsymbol{B} \\ \boldsymbol{B}'\boldsymbol{W}_\delta\boldsymbol{A} & \boldsymbol{B}'\boldsymbol{W}_\delta\boldsymbol{B} \end{bmatrix} \tag{4.27}$$

The two diagonal blocks can be evaluated using (4.23), but the off-diagonal blocks are of the form:

$$(\boldsymbol{A}_d \otimes \cdots \otimes \boldsymbol{A}_1)'\boldsymbol{W}_\delta(\boldsymbol{B}_d \otimes \cdots \otimes \boldsymbol{B}_1) \tag{4.28}$$

It can be shown that these off-diagonal products can be evaluated using (4.23), by setting $\boldsymbol{G}_i = \boldsymbol{B}_i \square \boldsymbol{A}_i$ (see Example 7.1 of Currie et al., 2006). This form of column partitioned matrix of Kronecker products also occurs in the multi-dimensional version of the mixed model representation of P-splines described in Section 3.3.3.

We now describe a key contribution of this thesis, and show how we can can generalize the methods described by Currie et al. (2006) to other matrix products. A good starting point is the inner product with a non-diagonal matrix. To clarify,

we seek an efficient method to evaluate the product $\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B}$ for a general matrix $\boldsymbol{W}$, where $\boldsymbol{B} = \boldsymbol{B}_d \otimes \cdots \otimes \boldsymbol{B}_1$. When $\boldsymbol{W}$ is a diagonal weight matrix it is efficient to store the non-zero elements in a d-dimensional array, but if $\boldsymbol{W}$ does not have this special structure we need to arrange the elements into a $2d$-dimensional array. To be precise we would re-arrange the elements into a $n_1 \times n_1 \times \ldots \times n_d \times n_d$ array such that

$$\overset{a}{\boldsymbol{W}}_{i_1,j_1,\ldots,i_d,j_d} = \overset{m}{\boldsymbol{W}}_{\delta(i_1,\ldots,i_d,n_1,\ldots,n_d),\delta(j_1,\ldots,j_d,n_1,\ldots,n_d)}, \tag{4.29}$$

where we emphasize that in (4.29) the suffix δ indicates the function

$$\delta(i_1,\ldots,i_d,n_1,\ldots,n_d) = i_1 + \sum_{s=2}^{d}\left((i_s-1)\prod_{t=1}^{s-1} n_t\right). \tag{4.30}$$

It is worth noting that this clumsy notation is easily implemented in R using the `aperm()` function; for example, in three dimensions we start with a matrix, `Wm`, of dimension $n_1n_2n_3 \times n_1n_2n_3$ and applying

```
aperm(array(Wm,c(n1,n2,n3,n1,n2,n3)), c(1,4,2,5,3,6))
```

we obtain an array of dimension $n_1 \times n_1 \times n_2 \times n_2 \times n_3 \times n_3$. This arrangement of the array $\overset{a}{\boldsymbol{W}}$ is such that d-dimensional sub-arrays obtained by holding the indices of even dimensions fixed contain the same elements as the columns of the matrix arrangement; for example

$$\mathrm{vec}(\overset{a}{\boldsymbol{W}}_{\bullet,1,\ldots,\bullet,1}) = \overset{m}{\boldsymbol{W}}_{\bullet,1}, \tag{4.31}$$

with the corresponding relationship for rows

$$\mathrm{vec}(\overset{a}{\boldsymbol{W}}_{1,\bullet,\ldots,1,\bullet}) = (\overset{m}{\boldsymbol{W}}_{1,\bullet})'. \tag{4.32}$$

With $\overset{a}{\boldsymbol{W}}$ in this arrangement we can obtain the elements of the matrix product $\boldsymbol{B}'\overset{m}{\boldsymbol{W}}\boldsymbol{B}$ with the recursive array product

$$\left(\boldsymbol{B}_d'\Big(\boldsymbol{B}_d'\ldots\big(\boldsymbol{B}_1'(\boldsymbol{B}_1'\overset{a}{\boldsymbol{W}})'\big)'\Big)'\right)', \tag{4.33}$$

which leaves a $c_1 \times c_1 \times \ldots c_d \times c_d$ dimensional array. We then retrieve the matrix arrangement $\boldsymbol{B}'\overset{m}{\boldsymbol{W}}\boldsymbol{B}$ by permuting the indices of (4.33) and grouping the first d and last d dimensions.

We note some similarities and some differences between (4.23) and (4.33). Firstly both formulae are designed to evaluate the weighted inner product $\boldsymbol{B}'\overset{m}{\boldsymbol{W}}\boldsymbol{B}$. In the former we take advantage of the diagonal form of $\overset{m}{\boldsymbol{W}}$ by multiplying each $\boldsymbol{B}_i$ onto the front and back of $\overset{a}{\boldsymbol{W}}$ simultaneously through the use of the row tensor function. In contrast, in (4.33) we sequentially multiply each matrix $\boldsymbol{B}_i$ onto the front and then onto the back of $\overset{a}{\boldsymbol{W}}$.

The algorithms in (4.23) and (4.33) are the extremes, and in some cases we may have a weight matrix somewhere in between the two. In a mixed model for example, we may have a two-way model with a correlation structure on the first factor, which would result in a block diagonal structure to the covariance matrix. Alternatively, a correlation structure on the second factor would result in a diagonal band matrix. These structures of the weight matrix can also be computed efficiently by "mixing and matching" the methods for diagonal and full weight matrices. In a d-dimensional problem we may have h dimensions of the inner product which are filled out with interactions (i.e. h dimensions which require two dimensions in the weight matrix) and $d-h$ dimensions without interactions. We would re-arrange our weight matrix, $\overset{m}{\boldsymbol{W}}$, into an array, $\overset{a}{\boldsymbol{W}}$, with $d+h$ dimensions. To form the inner product we use the appropriate method depending on whether each dimension has an interaction or not. For those dimensions which are represented by two dimensions in the weight matrix we multiply the corresponding $\boldsymbol{B}_i$ matrix onto both dimensions, and for those with only one dimension we multiply the row tensor of the corresponding matrix $\boldsymbol{G}_i$ onto those dimensions. Once we have finished the multiplications, those dimensions that were multiplied by a row tensor have to be factored out, then we must perform the shuffle of the dimensions, and finally re-arrange the elements into the appropriately dimensioned matrix. The method is most easily demonstrated with reference to an example.

Suppose we have the marginal matrices

$$\boldsymbol{B}_1 = \begin{bmatrix} 2 & 3 \\ 3 & 4 \\ 1 & 5 \\ 4 & 2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{B}_2 = \begin{bmatrix} 3 & 2 \\ 1 & 4 \\ 2 & 1 \end{bmatrix} \tag{4.34}$$

and a block-diagonal weight matrix

$$\overset{m}{\boldsymbol{W}} = \begin{bmatrix} 2 & 2 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 3 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 3 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 2 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 1 & 1 & 3 \end{bmatrix}. \tag{4.35}$$

Then

$$\boldsymbol{B} = \boldsymbol{B}_2 \otimes \boldsymbol{B}_1 = \begin{bmatrix} 6 & 9 & 4 & 6 \\ 9 & 12 & 6 & 8 \\ 3 & 15 & 2 & 10 \\ 12 & 6 & 8 & 4 \\ 2 & 3 & 8 & 12 \\ 3 & 4 & 12 & 16 \\ 1 & 5 & 4 & 20 \\ 4 & 2 & 16 & 8 \\ 4 & 6 & 2 & 3 \\ 6 & 8 & 3 & 4 \\ 2 & 10 & 1 & 5 \\ 8 & 4 & 4 & 2 \end{bmatrix}, \tag{4.36}$$

and

$$\boldsymbol{B}'\overset{m}{\boldsymbol{W}}\boldsymbol{B} = \begin{bmatrix} 3086 & 4111 & 2468 & 3314 \\ 4881 & 6076 & 3758 & 4706 \\ 2468 & 3314 & 3869 & 5156 \\ 3758 & 4706 & 5324 & 6685 \end{bmatrix}. \tag{4.37}$$

We will now show how to use array methods to produce the same result, and we will illustrate the array methods with some R code. We can rearrange non-zero elements of $\overset{m}{\boldsymbol{W}}$ into a $4 \times 4 \times 3$ array, $\overset{a}{\boldsymbol{W}}$, such that

$$\overset{a}{\boldsymbol{W}}_{\bullet,\bullet,1} = \begin{bmatrix} 2 & 2 & 0 & 5 \\ 0 & 3 & 3 & 2 \\ 1 & 4 & 3 & 4 \\ 0 & 4 & 3 & 0 \end{bmatrix} \quad \overset{a}{\boldsymbol{W}}_{\bullet,\bullet,2} = \begin{bmatrix} 2 & 1 & 2 & 1 \\ 1 & 3 & 1 & 1 \\ 1 & 0 & 1 & 3 \\ 4 & 3 & 0 & 1 \end{bmatrix} \quad \overset{a}{\boldsymbol{W}}_{\bullet,\bullet,3} = \begin{bmatrix} 2 & 3 & 1 & 5 \\ 3 & 2 & 1 & 2 \\ 4 & 0 & 2 & 3 \\ 4 & 1 & 1 & 3 \end{bmatrix}. \quad (4.38)$$

We first "pre"-multiply $\boldsymbol{B}_1$ onto the first dimension and perform one rotation on the result,

```
X1 <- rotate.array(mat.array(t(B1), W))
```

and we obtain the array, $\boldsymbol{X}_1$, $4 \times 3 \times 2$, where

$$(\boldsymbol{X}_1)_{\bullet,\bullet,1} = \begin{bmatrix} 5 & 24 & 33 \\ 33 & 23 & 16 \\ 24 & 8 & 11 \\ 20 & 12 & 31 \end{bmatrix} \quad (\boldsymbol{X}_1)_{\bullet,\bullet,2} = \begin{bmatrix} 11 & 23 & 46 \\ 46 & 21 & 19 \\ 33 & 15 & 19 \\ 43 & 24 & 44 \end{bmatrix}. \quad (4.39)$$

Next, we "post"-multiply $\boldsymbol{B}_1$ onto the second dimension and rotate,

```
X2 <- rotate.array(mat.array(t(B1), X1))
```

which leaves the array, $\boldsymbol{X}_2$, $3 \times 2 \times 2$, where

$$(\boldsymbol{X}_2)_{\bullet,\bullet,1} = \begin{bmatrix} 213 & 365 \\ 173 & 220 \\ 249 & 344 \end{bmatrix} \quad (\boldsymbol{X}_2)_{\bullet,\bullet,2} = \begin{bmatrix} 307 & 468 \\ 228 & 276 \\ 280 & 397 \end{bmatrix}. \quad (4.40)$$

Finally, since there is no covariance structure on the second variable, we multiply the row tensor of $\boldsymbol{B}_2$ onto the third dimension of the array and rotate again,

```
X3 <- rotate.array(mat.array(t(G2), X2))
```

and we obtain the array, $\boldsymbol{X}_3$, $2 \times 2 \times 4$, where

$$(\boldsymbol{X}_3)_{\bullet,\bullet,1} = \begin{bmatrix} 3086 & 4111 \\ 4811 & 6076 \end{bmatrix} \quad (\boldsymbol{X}_3)_{\bullet,\bullet,2} = \begin{bmatrix} 2468 & 3314 \\ 3758 & 4706 \end{bmatrix}$$
$$(\boldsymbol{X}_3)_{\bullet,\bullet,3} = \begin{bmatrix} 2468 & 3314 \\ 3758 & 4706 \end{bmatrix} \quad (\boldsymbol{X}_3)_{\bullet,\bullet,4} = \begin{bmatrix} 3869 & 5156 \\ 5324 & 6685 \end{bmatrix}. \tag{4.41}$$

We can see that this contains the same elements as (4.37). We need to factor out the third dimension to give a four-dimensional array, permute the indices, and then convert back to a matrix arrangement; this can be achieved using the R code:

```
BWB2 <- matrix(aperm(array(X3, c(c.1, c.1, c.2, c.2)),
                     c(1,3,2,4)), c.1*c.2, c.1*c.2)
```

and we recover (4.37).

4.4.3 Interpretation and Implementation

In the preceding sections of this chapter we have viewed the array methods purely as a computational tool for performing operations involving Kronecker products efficiently. However, often the array interpretation gives a much clearer picture of the situation, as illustrated by the following examples.

First, we note that when multiplying out a Kronecker product it is just as easy to form a higher dimensional array as a matrix. For example, two marginal B-spline bases $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ can be used to form the two-dimensional basis:

$$\boldsymbol{B} = \boldsymbol{B}_2 \otimes \boldsymbol{B}_1. \tag{4.42}$$

We normally consider $\boldsymbol{B}$ to be a $n_1 n_2 \times c_1 c_2$ matrix, but it is just as easy to represent $\boldsymbol{B}$ as a $n_1 \times c_1 \times n_2 \times c_2$ array. The array representation is more closely in keeping with the definition of tensors used in physics and engineering, and can make the object $\boldsymbol{B}$ easier to deal with. The statistical package, R, has good in-built functionality for dealing with arrays, and the code below can be used to plot the two-dimensional B-spline basis as shown in Fig. 3.5. The code illustrates how treating $\boldsymbol{B}$ as an array is both easier and importantly more robust than treating it as a matrix. The first

section of code shows how to produce the plot if we treat $\boldsymbol{B}$ as an array and the second section if we treat $\boldsymbol{B}$ as a matrix.

```
#
# Array Method
#
B <- outer(B2, B1)

B.sub <- matrix(0, nrow(B1), nrow(B2))
for(i in seq(4, ncol(B1), 3)){
  for(j in seq(4, ncol(B2), 3)){
    B.sub <- B.sub + B[,j,,i]
  }
}

persp(A, Y, B.sub, theta = -30, phi = 30, zlim = c(0,0.7),
      scale = F, expand = 20, ticktype = "detailed",
      xlab = "Age", ylab = "Year", zlab = "b()")
#
# Matrix-vector method
#
B <- kronecker(B2, B1)

B.sub <- rep(0, nrow(B1) * nrow(B2))
for(i in seq(4, ncol(B1), 3)){
  for(j in seq(4, ncol(B2), 3)){
    B.sub <- B.sub + B[,i + ((j - 1) * ncol(B1))]
  }
}

B.sub <- matrix(B.sub, nrow(B1), nrow(B2))
persp(A, Y, B.sub, theta = -30, phi = 30, zlim = c(0,0.7),
      scale = F, expand = 20, ticktype = "detailed",
      xlab = "Age", ylab = "Year", zlab = "b()")
```

With the array method both the data and the indices of the basis functions are in two-dimensional arrays. The outer function forms a four-dimensional array from the marginal bases, from which we can easily pick the age and year index for the basis function we want to plot. Holding the two indices of the basis functions constant leaves us with a two-dimensional array containing the basis function evaluated over the age and year grid which can be used directly as the z argument in the three-dimensional plotting function. In contrast, with the matrix-vector method picking out the basis functions is complicated by the fact that the index has been vectorised,

and once the basis function has been identified we have to put the vectorised data into an array in order to plot the the basis function.

Another area where using arrays can be clearer is in the penalty. In the two-dimensional model we use the penalty from (3.24)

$$\boldsymbol{P} = \lambda_1 \boldsymbol{\theta}'(\boldsymbol{I}_{c_2} \otimes \boldsymbol{D}_1'\boldsymbol{D}_1)\boldsymbol{\theta} + \lambda_2 \boldsymbol{\theta}'(\boldsymbol{D}_2'\boldsymbol{D}_2 \otimes \boldsymbol{I}_{c_1})\boldsymbol{\theta}. \tag{4.43}$$

From this definition it is not clear how the penalty is acting on the coefficients, but if we re-write in array form we obtain the penalty

$$\boldsymbol{P} = \lambda_1 \big(\mathrm{vec}(\boldsymbol{D}_1\boldsymbol{\Theta})\big)'\mathrm{vec}(\boldsymbol{D}_1\boldsymbol{\Theta}) + \lambda_2 \big(\mathrm{vec}(\boldsymbol{\Theta}\boldsymbol{D}_2')\big)'\mathrm{vec}(\boldsymbol{\Theta}\boldsymbol{D}_2'). \tag{4.44}$$

This form makes it clear that the difference matrix $\boldsymbol{D}_1$ is working on the rows of the coefficient array, and $\boldsymbol{D}_2$ is working on the columns.

Using arrays can also be useful when it comes to computing. We benefit from the fact that each component of the regression matrix is explicitly attached to a dimension of the parameter and data array; this insures an extra level of conformity. For example, in matrix-vector form, a programming slip might evaluate the Kronecker product basis in the wrong order, this would mean that the basis no longer matched the data, but the data vector would still conform with this incorrect basis. This bug might be difficult to track down. Using arrays, trying to multiply the component onto the wrong dimension would throw an error that the multiplication does not conform, and we find the problem immediately.

The methods described in Sections 4.3 and 4.4.2 have been implemented to fit the scoring algorithm (4.16); Currie et al. (2006) and Eilers et al. (2006) give code in R and Matlab respectively.

In practise implementing these methods in R is reasonably straight forward as there is already good support for manipulating arrays, in particular all machinery for re-dimensioning arrays is in place. The only need to revert to matrices is in order to find the solutions to the linear system, but wrapper functions can be written which bury the reorganisation of indices, and means the user only need deal with arrays. Once these functions have been written, implementation of the penalized scoring algorithm is just as easy with arrays as with matrices and vectors. Although we can make

do by rotating and re-dimensioning the arrays, ideally we would prefer to implement the methods in a lower level programming language. One would write a class which stored a Kronecker product as a list of its components, with an overloaded function for matrix-vector multiplication and calculation of the inner product, combined with an overloaded method to solve a system of linear equations which accepted "square" arrays. Implementing the methods in this way would free the practitioner from having to convert between arrays and matrices. Efficiency could also be improved by multiplying the components of the Kronecker product directly onto the correct dimension of the array without having to perform the rotations.

We now turn to the practical limitations of the array methods. Arguably the most important aspect of the array methods is the ability to fit a model without having to multiply and store the full basis, as storing the basis is often the limiting factor in the size of model that one can work with. With this obstacle removed the limiting factor becomes manipulation and storage of data and the covariance matrix. Using a modern computer we can fairly comfortably cope with arrays with up to one million elements, this puts quite large three-dimensional problems (up to 100 levels in each dimension) and moderately sized four-dimensional problems (about 30 levels for each dimension) within our grasp, and would also allow some large two-dimensional models with a covariance structure on one of the dimensions.

4.5 Computational Results

So far we have only described the methods and algorithms for dealing with Kronecker products, without giving details of the computational savings obtained by using them. In this section we will take a two-dimensional example to give an idea of the saving that can be made. We shall compare the array methods to regular matrix methods, first by calculating the number of scalar multiplications in each product, and then showing how this translates into computer time.

In the examples we shall perform comparisons for the following four operations:

- MV - a matrix-vector product
- IPD - an inner-product with a diagonal weight matrix

- IPB - an inner-product with a block diagonal weight matrix
- IPF - an inner-product with a full weight matrix

For both methods there is some initial calculation required, multiplying out the Kronecker product for the matrix method, and forming the row tensors for the array method; these will not be included in the comparison as they would only need to be performed once in an iterative algorithm such as (4.16). Tables 4.1 and 4.2 give the number of scalar multiplications needed to perform each of the operations using array and regular matrix operations. In this form it is difficult to tell which is the quicker method; although, for instance, one can see that for the MV operation the number of multiplications required is the product of $2d$ numbers, whereas the array operation is the sum of d products of $d+1$ numbers, so provided d was small compared with any of the n_i or c_i the array method will be quicker. Table 4.3 shows the number of multiplications required for a two-dimensional example with $\boldsymbol{B}_1$ of dimension 90×25 and $\boldsymbol{B}_2$ of dimension 60×20; here it is easy to see that the array methods perform significantly better than regular matrix methods, in particular the IPB operation requires 460 times fewer multiplications.

Simply comparing the number of multiplications is not an entirely fair test of the methods because there is some additional computer overhead required for the rotation of the arrays and repositioning of the elements. An alternative comparison can be obtained by comparing the time taken to perform the operations on a computer. Table 4.4 shows the time taken for a computer to complete the calculations for examples used in Table 4.3. Based on this comparison the two methods are closer, but with array methods still significantly quicker. The times were calculated using the statistical package R, but as mentioned in Section 4.4.3 this is not the ideal environment for testing and comparing these methods. For larger problems the rotation of arrays can become quite cumbersome and time consuming for the computer, but if one was to write a library from scratch to implement the array methods, the sequential multiplications could be performed without rotating the array; this would improve performance. No matter how we multiply matrices onto the array, we reduce the storage requirements required to fit the model because we do not have to multiply

Table 4.1: Number of scalar multiplications for various matrix operations using array methods.

Operation	Number of multiplications
MV	$\sum_{i=1}^{d}(\prod_{j=1}^{i} n_j)(\prod_{k=i}^{d} c_k)$
IPD	$\sum_{i=1}^{d}(\prod_{j=1}^{i} c_j^2)(\prod_{k=i}^{d} n_k)$
IPF	$\{\sum_{i=1}^{d} c_i(\prod_{j=1}^{i-1} c_j^2)(\prod_{k=i}^{d} n_k^2)\} + \{\sum_{i=1}^{d} n_i(\prod_{j=1}^{i} c_j^2)(\prod_{k=i+1}^{d} n_k^2)\}$

Table 4.2: Number of scalar multiplications for various matrix operations using matrix methods.

Operation	Number of multiplications
MV	$\prod_{i=1}^{d} n_i c_i$
IPD	$\prod_{i=1}^{d} n_i c_i + \prod_{i=1}^{d} n_i c_i^2$
IPF	$\prod_{i=1}^{d} n_i^2 c_i + \prod_{i=1}^{d} n_i c_i^2$

Table 4.3: Comparison of the number of multiplications for regular matrix methods and array methods: an example, $\boldsymbol{B}_1$, 90×25, and $\boldsymbol{B}_2$, 60×20.

Operation	Matrix Methods	Array Methods	Ratio
MV	2.4e6	1.36e5	17:1
IPD	1.2e9	8.68e6	130:1
IPB	1.27e10	2.76e7	460:1
IPF	1.27e10	8.16e8	15:1

Table 4.4: Comparison of the timings (seconds) for regular matrix methods and array methods: an example, $\boldsymbol{B}_1$, 90×25, and $\boldsymbol{B}_2$, 60×20.

Operation	Matrix Methods	Array Methods	Ratio
MV	0.044	< 0.001	44 : 1
IPD	2.060	0.040	51 : 1
IPB	22.794	0.064	356 : 1
IPF	23.101	3.292	6 : 1

and store the Kronecker product.

4.6 Putting Data on an Array

The array methods discussed so far in this chapter clearly apply to the multi-dimensional P-spline model described in Section 3.3, but can they be applied in other models? In this section we show when the array methods can be applied in multi-dimensional smoothing models, and also show how they can be used in models for longitudinal data.

For the mortality data in Section 1.3, the data lie on a grid indexed by age and year. Using the two-dimensional P-spline model we then place a coarser grid of coefficients over this grid; this is shown graphically in Fig. 4.1. Having the data and coefficients lie on a grid allows the basis to be formed as a Kronecker product, and the converse is true: if the basis for a model can be written as a Kronecker product it means that the data and coefficients both have an interpretation as an array. Often however the array interpretation is not obvious and we have to re-arrange the data to make use of the array methods. For example if we wish to smooth the mortality data by age and year of birth rather than by age and year of observation we no longer obtain a

Kronecker product basis, because the cohort indicator,

$$\boldsymbol{x}_C = \text{vec}(\boldsymbol{X}_C) = \text{vec}\left(\begin{bmatrix} 1935 & 1936 & \cdots & 1986 & 1987 \\ 1934 & 1935 & \cdots & 1985 & 1986 \\ \vdots & \vdots & & \vdots & \vdots \\ 1860 & 1861 & \cdots & 1912 & 1912 \\ 1859 & 1860 & \cdots & 1910 & 1911 \end{bmatrix}\right) \tag{4.45}$$

cannot be arranged as a repeat of a smaller vector as the equivalent indicators for age and year can in (3.25). Without the Kronecker product we have to revert to the row tensor to form the basis and we are unable to take advantage of the array methods. Figure 4.2 shows the equivalent plot for the age-cohort model as the plot shown in Fig. 4.1 for age and year. The basis functions have support on a parallelogram in age and year co-ordinates; if however, we change co-ordinates to age and cohort, we see that the basis functions return to having support on a square and the coefficients form a grid. The change of co-ordinates means that the data no longer form a grid, as shown by the blue points which are missing from the grid. However, using the forecast method of Currie et al. (2004) we can create dummy data for the points missing from the grid and use a weight matrix to remove the dummy data from influencing the fitting procedure. Once the data and coefficients are arranged on a grid we can proceed with a Kronecker product basis and take advantage of the fast computational methods.

An area where arrays crop up naturally is longitudinal studies. If we observe repeated measures on the same subjects, and if the measurements are taken at the same time for each member of the study then the data form an array. The pig weight example discussed in Section 2.5.1 is a typical example of a longitudinal study with an array interpretation. The convention is to store the observations as a double indexed vector, but if we store the observations as a matrix, with each row corresponding to a pig, and each column corresponding to a week of the study, we have the array structure and we can then bring to bear the fast array methods. In this particular example the relatively small number of observations and parameters mean that the savings are relatively modest, but in larger studies the computational burden could be cut dramatically.

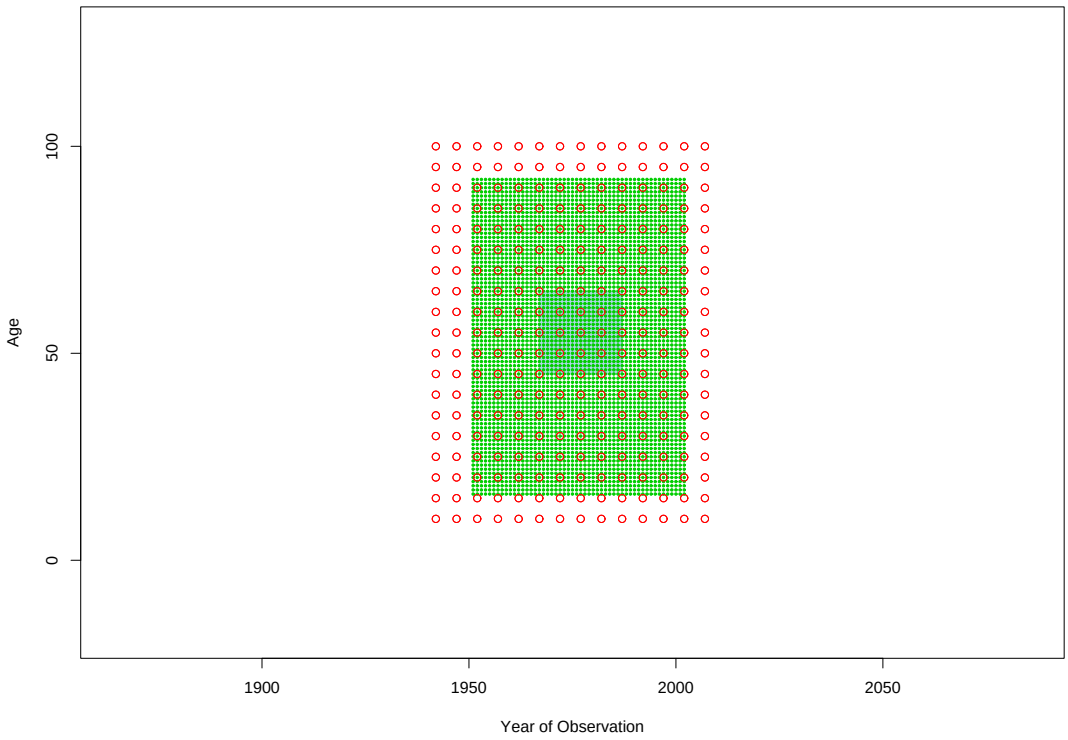


Figure 4.1: A graphical representation of the Age-Period model, green dots - data points, red rings - central knot position, light blue background - support for a selected single B-spline

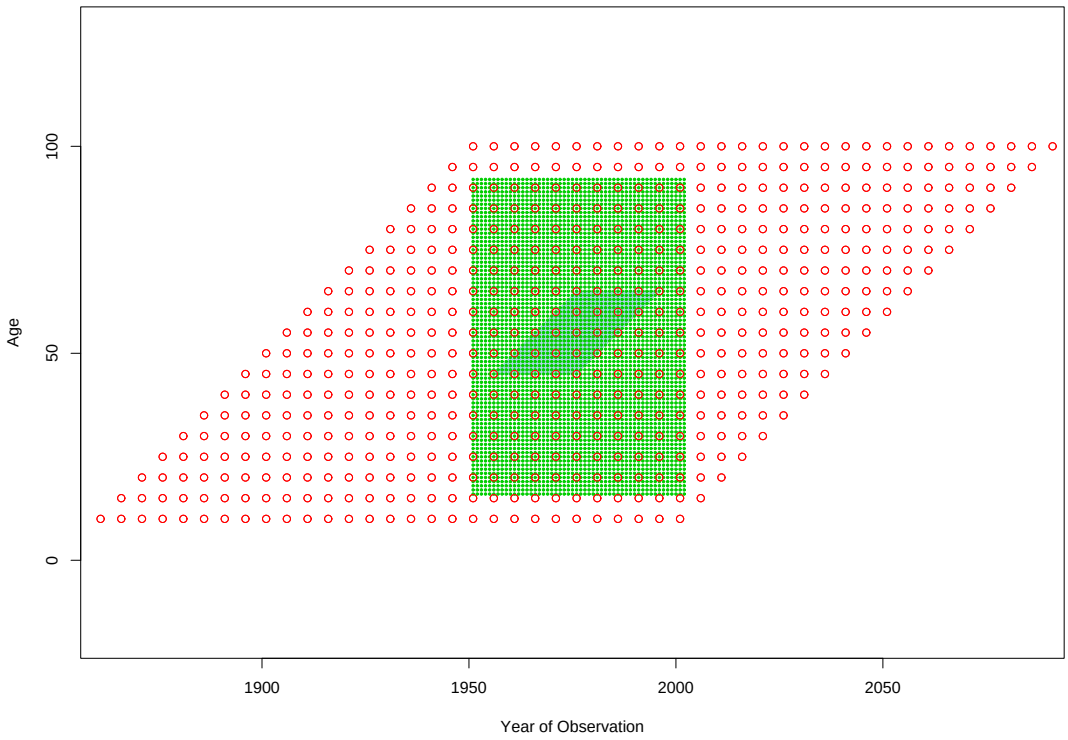


Figure 4.2: A graphical representation of the Age-Cohort model in regular format, green dots - data points, red rings - central knot position, light blue background - support for a selected single B-spline

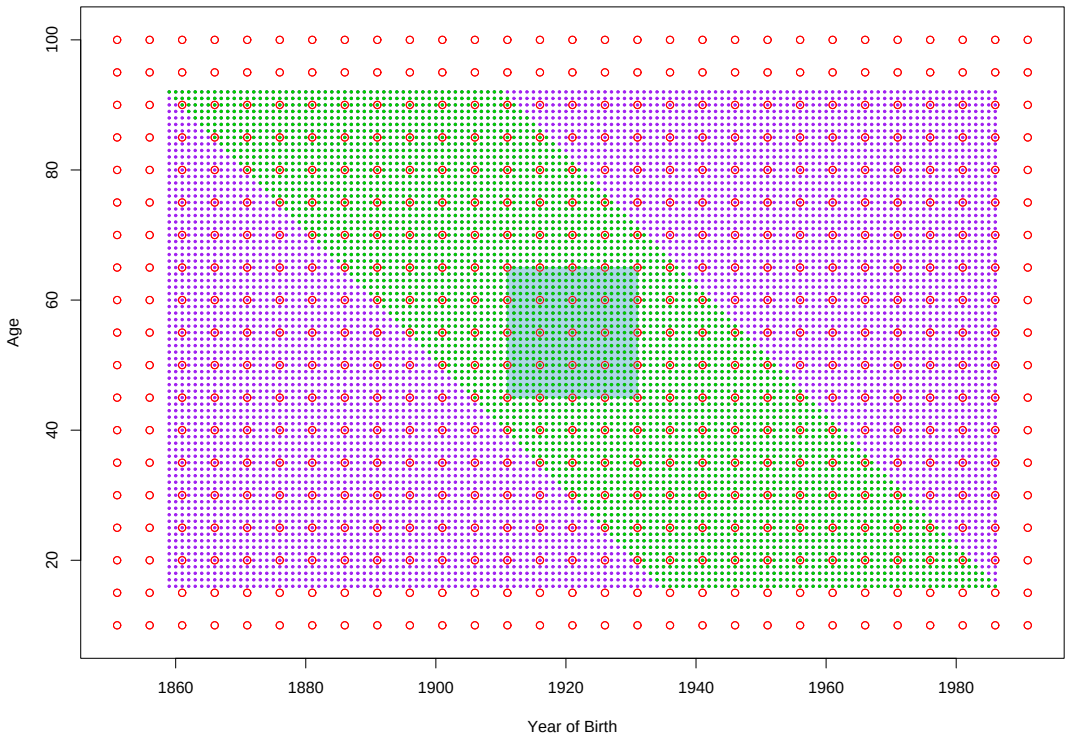


Figure 4.3: A graphical representation of the Age-Cohort model in an array format, green dots - data points, purple dots - dummy data points, red rings - central knot position, light blue background - support for a selected single B-spline

Chapter 5

Generalized Additive Array Models

Often a single smooth line or surface is not sufficient to model a particular set of data. In this case we may require an additive model to explain the processes that gave rise to the data. In a generalized additive model (GAM) the response variable is modelled by the sum of p smooth functions

$$\boldsymbol{\eta} = \sum_{i=1}^{p} S_i(\boldsymbol{x}). \tag{5.1}$$

GAMs are described in detail by Hastie and Tibshirani (1990), who give many examples of using GAMs, and introduce the back-fitting algorithm as a tool to fit GAMs in the general case.

In Sections 5.1 and 5.2 we analyse two sets of mortality data where the basic smooth model described in Chapter 3 is not successful in modelling the data. In both cases we propose additive models with two components which give a better explanation of the data, and give a smoother model for the under-lying mortality trend. For both models the components fall within the penalized likelihood framework, and each has at least one component which has the array structure described in Chapter 4. Two advantages result: firstly, a penalty can be used to maintain identifiability in the models and regulate the effect of each component, and secondly we can avoid the use of the back-fitting algorithm, and fit by adjusting the penalized scoring algorithm in (2.75). In each case we also describe implementation of the fast array methods described in Chapter 4. In Section 5.3, we discuss other possible applications for the models, possible improvements, and further work.

5.1 Modelling Lexis Data

In this Section we describe a model for mortality data on the Lexis diagram with reference to German mortality data (Statistiches Bundesamt, 2006). In Section 5.1.1 we describe the Lexis data, and our assumptions for the model. The extra complexity of the data means we have to adjust the smooth model from Chapter 3 in order to fit to data in Lexis format. In Section 5.1.2, we show how a basic smooth model can be fitted using two offset data arrays with corresponding Kronecker product bases. For the German data in the example we find that a smooth model does not adequately describe the data, so in Section 5.1.3 we propose an additive model which copes better with specific features of the data.

5.1.1 Description of the data

In this section we introduce a change from the data described in Section 1.3. The deaths are indexed by age, year of observation, and year of birth, giving rise to a format of the data shown in Fig. 5.1.

We assume that we observe census data on a $n_a \times n_y$ grid. These data consist of population counts on the 1st January, grouped by age last birthday for n_a ages, over n_y years. We represent these counts by the matrix $\boldsymbol{E}$, and as in Section 1.3 we represent the age and year indices by the vectors $\boldsymbol{x}_a$ and $\boldsymbol{x}_y$ respectively. We also observe counts of the number of deaths corresponding to each of the cells in $\boldsymbol{E}$, but the deaths are also split by age last birthday at the time of death. Referring to Fig. 5.1, we see that the exact age at death for the lives in a particular cell in $\boldsymbol{E}$ span two years, and their age last birthday (at death) will take one of two values. We will refer to those deaths where the age last birthday at death is the same as the age last birthday on the preceding 1st January as Type-A deaths, and those for which age last birthday at death is one year higher as Type-B deaths. Clearly, for any of the population counts in $\boldsymbol{E}$ it is possible to observe some Type-A and some Type-B deaths and so we split our death counts into two grids; $\boldsymbol{D}_A$ is a $n_a \times n_y$ matrix which gives the number of Type-A deaths corresponding to each element of $\boldsymbol{E}$, and $\boldsymbol{D}_B$ gives the corresponding number of Type-B deaths.

We wish to model the mortality rate within each Lexis triangle, for which we need an exposed to risk for each triangle. Assuming a uniform age distribution among the initial population, we can split the exposure time of those who survive equally between the Type-A and Type-B triangles. Calculating the exposed to risk for the lives that die is more complicated. We start by assuming a uniform distribution of deaths in each triangle. To simplify the calculations: for a given age, x, and year, t, we define u as the time in years since t and v as the age of an individual minus x. We can then calculate the expected exposure time for those that die as follows; with reference to the left panel in Fig. 5.2, we can see that lives that die in Type-A triangles contribute no exposure time in the corresponding Type-B triangle. Therefore, v at time of death takes a value between 0 and 1, and full value of u at the time of death is assigned to the exposure in the Type-A triangle. Assuming the uniform distribution of death over the Type-A triangles means that the joint probability distribution of age and time at death takes a uniform value of 2 over u and v. The expected exposure time for a life dying in a Type-A triangle is therefore given by

$$\begin{aligned}
\boldsymbol{E}_{A|\boldsymbol{D}_A} &= \int_{v=0}^{v=1} \int_{u=0}^{u=v} (2u)\, du\, dv \\
&= \int_{v=0}^{v=1} [u^2]_0^v \, dv \\
&= \int_{v=0}^{v=1} (v^2)\, dv \\
&= [\tfrac{1}{3}v^3]_0^1 \\
&= \tfrac{1}{3}.
\end{aligned} \tag{5.2}$$

The right panel of Fig. 5.2 shows that those that die in Type-B triangles are expected to contribute some exposure in the Type-A triangles and some in the Type-B triangles, with v at time of death taking a value between 1 and 0. Those that die in Type-B triangles contribute $u - v + 1$ to the exposure in the Type-A triangles and $v - 1$ to the Type-B triangles. Assuming a uniform distribution of death, we note the joint probability distribution of age and time at death also takes a value of 2 over u and v for Type-B triangles. The exposure time in the Type-A triangle for a life dying in a

Type-B triangle is therefore given by

$$\begin{aligned}\boldsymbol{E}_{A|\boldsymbol{D}_B} &= 2\int_0^1\int_1^{u+1}(u-v+1)\,dv\,du\\ &= 2\int_0^1 [uv-\tfrac{1}{2}v^2+v]_1^{u+1}\,du\\ &= \int_0^1 \big((u+1)^2-2(u+1)+1\big)\,du\\ &= \int_0^1 u^2\,du\\ &= \tfrac{1}{3},\end{aligned} \tag{5.3}$$

and the exposure time in the Type-B triangle for a life dying in a Type-B triangle is given by

$$\begin{aligned}\boldsymbol{E}_{B|\boldsymbol{D}_B} &= 2\int_0^1\int_1^{u+1}(v-1)\,dv\,du\\ &= 2\int_0^1 [\tfrac{1}{2}v^2-v]_1^{u+1}\,du\\ &= \int_0^1 (u+1)^2-2(u+1)+1\,du\\ &= \tfrac{1}{3}.\end{aligned} \tag{5.4}$$

Using (5.2), (5.3), and (5.4) we see that the total exposed to risk for a Type-A triangle is

$$\begin{aligned}\boldsymbol{E}_A &= \tfrac{1}{2}(\boldsymbol{E}-\boldsymbol{D}_A-\boldsymbol{D}_B)+\tfrac{1}{3}\boldsymbol{D}_A+\tfrac{1}{3}\boldsymbol{D}_B\\ &= \tfrac{1}{2}\boldsymbol{E}-\tfrac{1}{6}\boldsymbol{D}_A-\tfrac{1}{6}\boldsymbol{D}_B;\end{aligned} \tag{5.5}$$

similarly, we can show for type-B triangles

$$\boldsymbol{E}_B = \tfrac{1}{2}\boldsymbol{E}-\tfrac{1}{2}\boldsymbol{D}_A-\tfrac{1}{6}\boldsymbol{D}_B. \tag{5.6}$$

As an aside we note that summing (5.5) and (5.6) we find the total exposure is $\boldsymbol{E}-\frac{2}{3}\boldsymbol{D}_A-\frac{1}{3}\boldsymbol{D}_B$, which is a generalization of the actuarial estimate of initial exposed to risk.

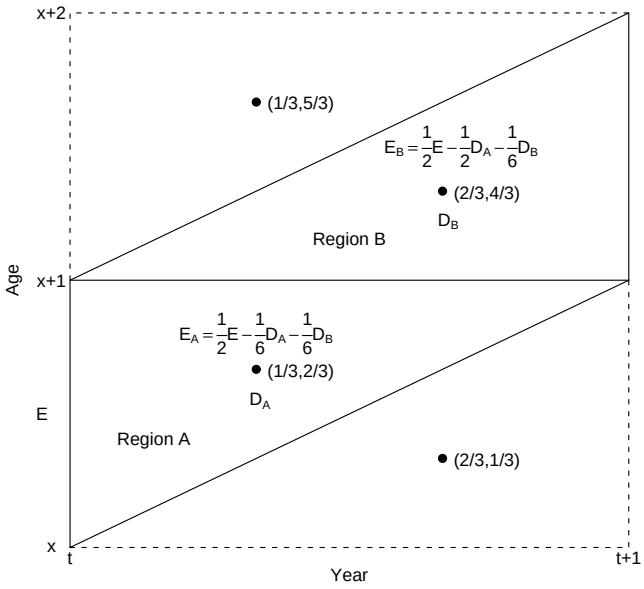


Figure 5.1: Lexis diagram: E lives between age x and $x+1$ at year t, D_A deaths and E_A years lived in triangle A, D_B deaths and E_B years lived in triangle B.

We can also calculate the expected age and time of death in each triangle. For a life that dies in A the expected time of death is $t + \frac{1}{3}$ and the expected age at death is $x + \frac{2}{3}$; similarly, for a life that dies in B the expected time of death is $t + \frac{2}{3}$ and the expected age at death is $x + \frac{4}{3}$. The situation is summarised in Fig. 5.1. We now model the number of deaths as follows

$$d_{A,x,t} \sim \mathcal{P}(e_{A,x,t}\, \mu_{x+2/3,t+1/3}) \tag{5.7}$$

$$d_{B,x,t} \sim \mathcal{P}(e_{B,x,t}\, \mu_{x+4/3,t+2/3}) \tag{5.8}$$

where $\mu_{x,t}$ is the hazard rate at age x and time t. In this model the deaths and exposures in triangles of Type-A are located on a rectangular grid. The same remark applies to the data in triangles of Type-B so the whole data set consists of two interleaved grids. We aim to fit a smooth surface through the data, but the added complexity in the data structure means that without care we would have to abandon

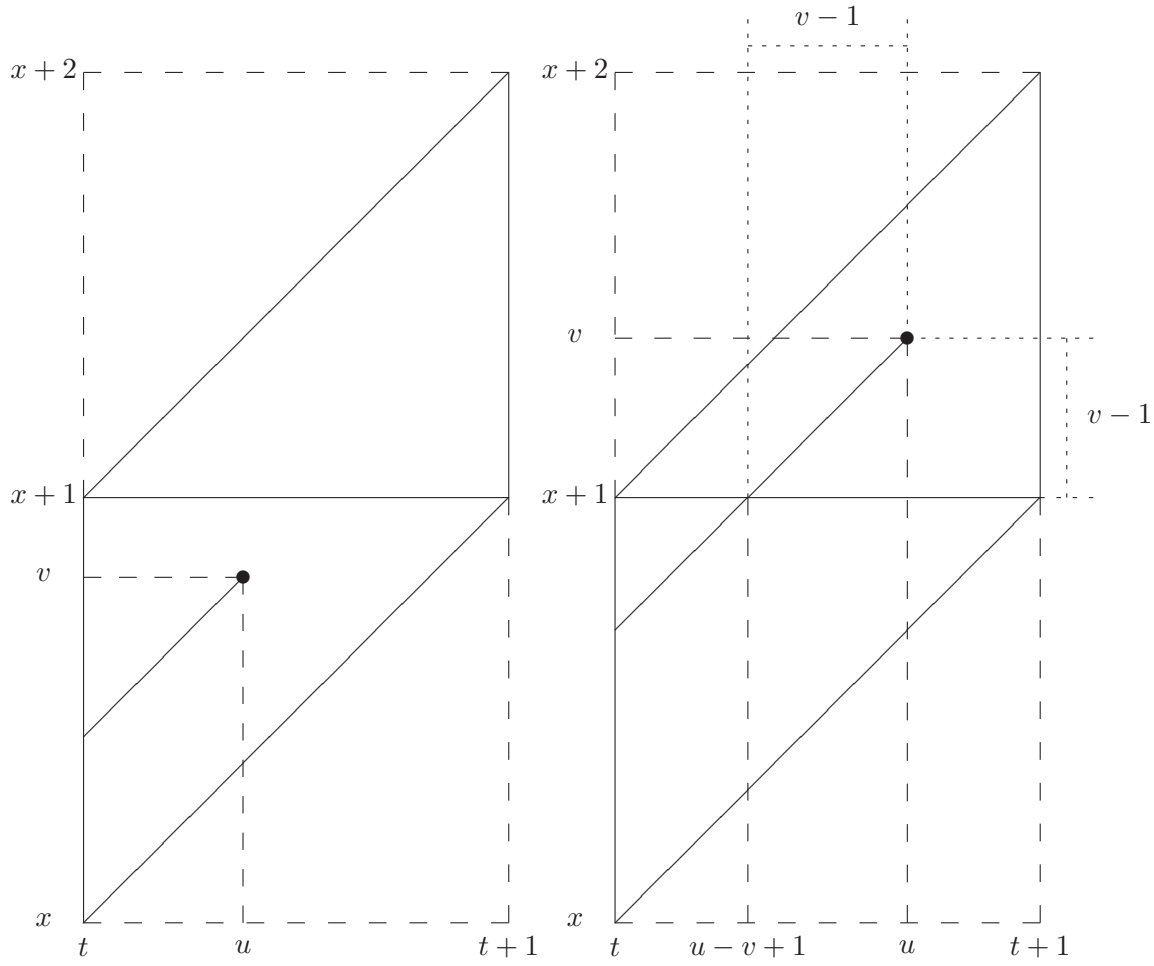


Figure 5.2: Life lines: The left panel shows a life dying in a Type-A triangle, and that this life only contributes exposure to the Type-A triangle. The right panel shows a life dying in a Type-B triangle, and how much exposure this life contributes to both the Type-A and Type-B triangles.

the array methods described in Chapter 4.

In the next section we will show how the two-dimensional surface can be fitted using the array methods. To illustrate the methods, we will be using the German mortality data (Statistiches Bundesamt, 2006) for which we have population counts for 13 years and 70 ages, with $\boldsymbol{x_a}' = (20, \ldots, 80)$ and $\boldsymbol{x_y}' = (1990, \ldots, 2002)$. For this particular example using array methods provides a small computational saving because the data set is so small, but the methods could be applied to a larger data set with the same structure, and again the use of array methods also helps to clarify

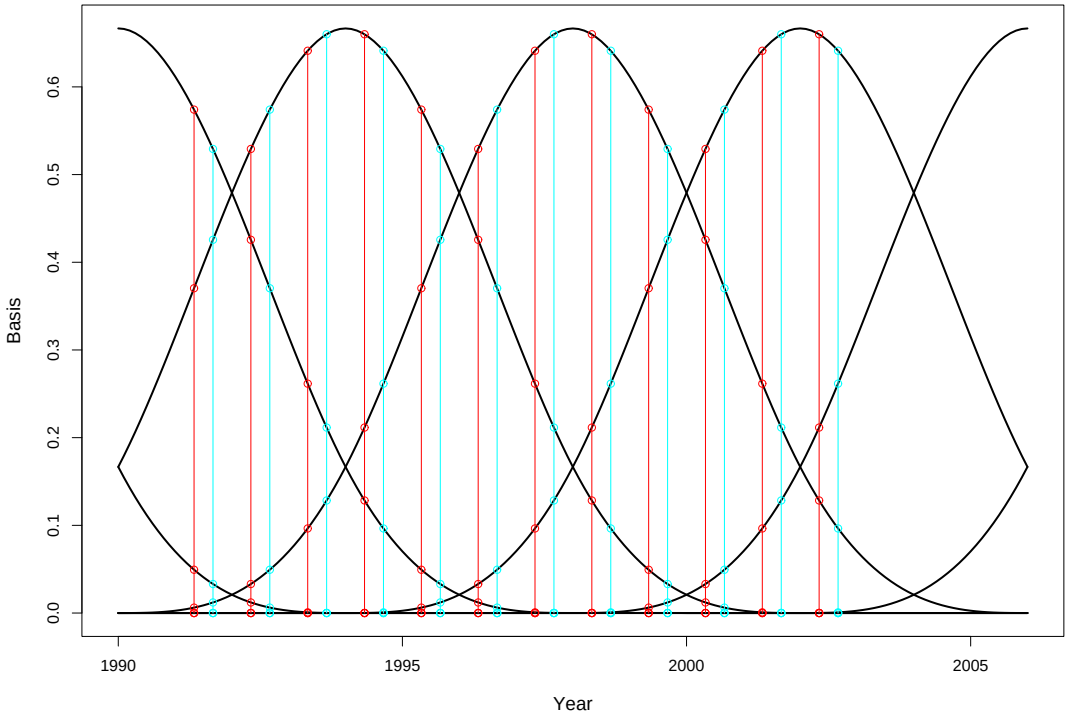


Figure 5.3: The marginal basis for year for the German mortality data showing the two inter-leaved bases in red and blue.

the connection between the basis and the parameters in the model.

5.1.2 A two-dimensional smooth surface

As in Chapter 3 we start by defining our basis for the model. For the German data the data points are on two grids: the type-A data points are centred at ages $(20\frac{2}{3}, \ldots, 80\frac{2}{3})$ and years $(1990\frac{1}{3}, \ldots, 2002\frac{1}{3})$, and the type-$B$ data points are at ages $(21\frac{1}{3}, \ldots, 81\frac{1}{3})$ and years $(1990\frac{2}{3}, \ldots, 2002\frac{2}{3})$. In order to fit a single smooth surface to both grids we must use a single basis for both grids. Referring to Fig. 5.3 we can see how the marginal basis for year is obtained: a B-spline basis is placed over a knot sequence which covers the range $(1990\frac{1}{3}, 2002\frac{2}{3})$, we then evaluate the splines separately for the Type-A and Type-B data points giving rise to marginal regression matrices $\boldsymbol{B}_{A_y}$ and $\boldsymbol{B}_{B_y}$. Similarly we obtain the regression matrices $\boldsymbol{B}_{A_a}$ and $\boldsymbol{B}_{B_a}$ for age, by evaluating an age basis covering the range $(20\frac{2}{3}, 81\frac{1}{3})$ for the Type-A and Type-B triangles. The two-dimensional regression matrices for both types can then be formed using the Kronecker products, $\boldsymbol{B}_A = \boldsymbol{B}_{A_y} \otimes \boldsymbol{B}_{A_a}$ and $\boldsymbol{B}_B = \boldsymbol{B}_{B_y} \otimes \boldsymbol{B}_{B_a}$. The main point here is that the same coefficients are applied to both regression matrices, because

both regression matrices are derived from the same underlying basis but evaluated at different positions. Consequently, our linear predictors for the two types of triangle are

$$\log(\boldsymbol{\tau}_A) = \boldsymbol{B}_A\boldsymbol{\theta}, \qquad \log(\boldsymbol{\tau}_B) = \boldsymbol{B}_B\boldsymbol{\theta} \tag{5.9}$$

or

$$\begin{bmatrix}\log(\boldsymbol{\tau}_A)\\ \log(\boldsymbol{\tau}_B)\end{bmatrix} = \begin{bmatrix}\boldsymbol{B}_A\\ \boldsymbol{B}_B\end{bmatrix}\boldsymbol{\theta} \tag{5.10}$$

or writing (5.9) in array form we have

$$\log(\boldsymbol{T}_A) = \boldsymbol{B}_{A_a}\boldsymbol{\Theta}\boldsymbol{B}'_{A_y} \qquad \text{and} \qquad \log(\boldsymbol{T}_B) = \boldsymbol{B}_{B_a}\boldsymbol{\Theta}\boldsymbol{B}'_{B_y}. \tag{5.11}$$

As usual a penalty is applied to adjacent coefficients in rows and columns of the parameter array $\boldsymbol{\Theta}$, and we use the penalty from (3.24)

$$\boldsymbol{P} = \lambda_a(\boldsymbol{I}_y \otimes \boldsymbol{D}'_{\boldsymbol{a}}\boldsymbol{D}_{\boldsymbol{a}}) + \lambda_y(\boldsymbol{D}'_{\boldsymbol{y}}\boldsymbol{D}_{\boldsymbol{y}} \otimes \boldsymbol{I}_a). \tag{5.12}$$

In order to fit a GLM we also need the matrices $\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B}$ and $\boldsymbol{B}'\boldsymbol{W}\boldsymbol{z}$, and so reverting to regular linear algebra we can show that

$$\begin{aligned}\boldsymbol{B}'\boldsymbol{W}\boldsymbol{B} &= \begin{bmatrix}\boldsymbol{B}'_A & \boldsymbol{B}'_B\end{bmatrix}\begin{bmatrix}\boldsymbol{W}_A & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{W}_B\end{bmatrix}\begin{bmatrix}\boldsymbol{B}_A\\ \boldsymbol{B}_B\end{bmatrix}\\ &= \boldsymbol{B}'_A\boldsymbol{W}_A\boldsymbol{B}_A + \boldsymbol{B}'_B\boldsymbol{W}_B\boldsymbol{B}_B\end{aligned} \tag{5.13}$$

and similarly

$$\boldsymbol{B}'\boldsymbol{W}\boldsymbol{z} = \boldsymbol{B}'_A\boldsymbol{W}_A\boldsymbol{z}_A + \boldsymbol{B}'_B\boldsymbol{W}_B\boldsymbol{z}_B. \tag{5.14}$$

Clearly all the components in (5.13) and (5.14) have an array form and so we can use the methods from Chapter 4 to compute them efficiently.

Continuing with the example of the German mortality data, Fig. 5.4 shows a comparison of the fitted smooth with the unsmoothed raw data for the Type-A triangles. In the raw data we can see diagonal crests and troughs in the data; these are cohort effects. The most noticeable of these coincide with the cohort that would have been aged about 20 at the start of the second world war, so it seems likely that the war is the contributing factor. We can see that the smooth surface in Fig. 5.4 is heavily

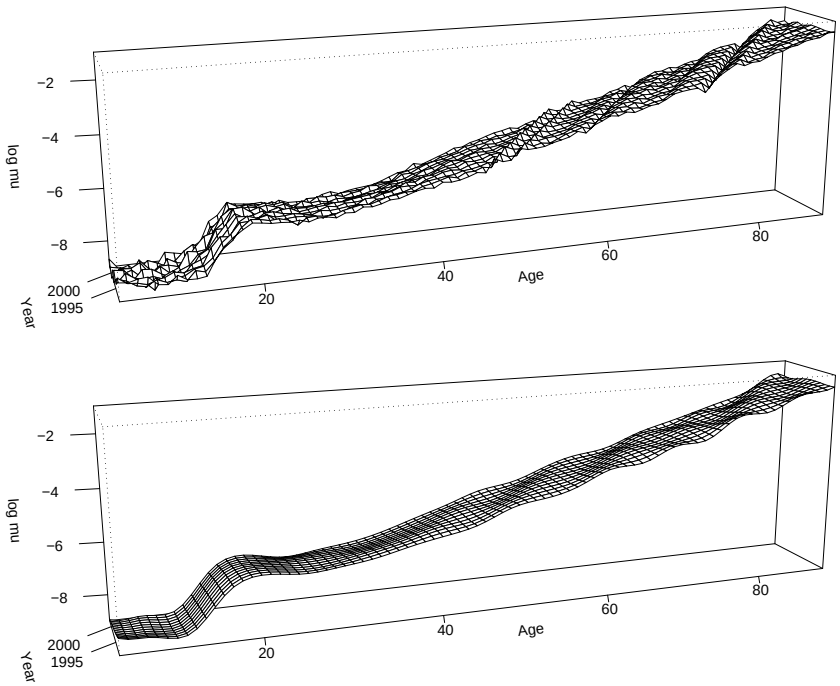


Figure 5.4: Cohort effects: observed mortality surface (top) for triangles of type A, and smooth surface for model in Section 5.1.2 plotted at data points of type A.

influenced by the cohort effects as the surface attempts to flex to allow for the peaks and troughs. However there is not enough flexibility in the surface to make a good job of adjusting for the shocks, but in attempting to do so weaker smoothing parameters are selected, meaning that the data is under-smoothed in the areas that the cohort effects do not occur. Therefore the model does not do a satisfactory job of modelling either the shocks or the underlying trend, so in Section 5.1.3 we introduce an additive model to fully explain these data.

Although the basic smooth model is not appropriate for the German data, the example illustrates how array methods can be applied even if initially the data appears not to lie on a grid. Provided the data can be grouped into subsets which have a grid structure then the array methods can be applied.

5.1.3 Adding cohort effects

Clearly a completely smooth model is not appropriate for these data. The cohort effects create discrete movements away from the underlying trend which cannot be

accounted for in a single smooth surface. We can extend the basic smooth model to include cohort effects by adjusting the linear predictor as follows

$$\boldsymbol{\eta} = \boldsymbol{B\theta} + \boldsymbol{C\gamma} \tag{5.15}$$

where $\boldsymbol{C}$ is the design matrix for the individual cohort effects. A ridge penalty with smoothing parameter κ is applied to $\boldsymbol{\gamma}$. The ridge penalty maintains identifiability and the size of κ can be tuned to the observed cohort effects. The ridge penalty ensures that the smooth features of the data are described by the B-spline surface and only the additional variation caused by the cohort effects is absorbed by $\boldsymbol{\gamma}$.

For given values of the smoothing parameters, estimates of $\boldsymbol{a}$ and $\boldsymbol{\gamma}$ are obtained by solving the scoring algorithm

$$\begin{bmatrix} \boldsymbol{B}'\tilde{\boldsymbol{W}}\boldsymbol{B} + \boldsymbol{P}_1 & \boldsymbol{B}'\tilde{\boldsymbol{W}}\boldsymbol{C} \\ \boldsymbol{C}'\tilde{\boldsymbol{W}}\boldsymbol{B} & \boldsymbol{C}'\tilde{\boldsymbol{W}}\boldsymbol{C} + \boldsymbol{P}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}'\tilde{\boldsymbol{W}}\tilde{\boldsymbol{z}} \\ \boldsymbol{C}'\tilde{\boldsymbol{W}}\tilde{\boldsymbol{z}} \end{bmatrix} \tag{5.16}$$

where $\tilde{\boldsymbol{z}}$ is the usual working vector, $\boldsymbol{P}_1$ is the difference matrix from (3.24) and $\boldsymbol{P}_2 = \kappa\boldsymbol{I}$ is a ridge penalty matrix. The values of the smoothing parameters are chosen by BIC.

Type-A and Type-B triangles in the same cohort in Fig. 5.1 are modelled by a single parameter in model (5.15). However, lives in triangle A are born predominantly in the second half of the year while those in triangle B are born predominantly in the first half, and the data in triangles A and B may be subject to different cohort effects. Thus we can split the cohort effect into two parts. We extend model (5.15) and use

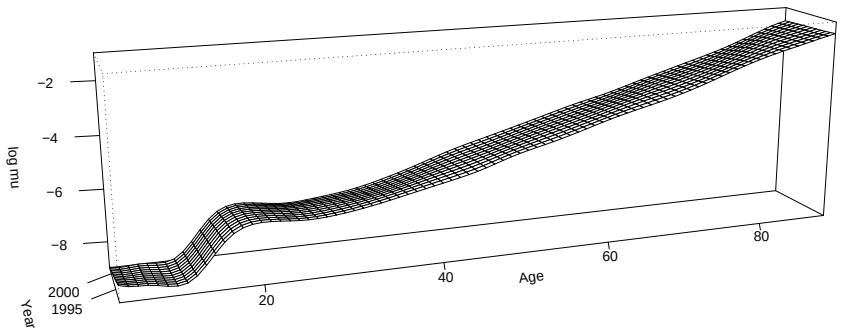


Figure 5.5: Smooth Surface: The smoothed surface obtained from the model with the linear predictor given in (5.17) plotted at data points of type A.

Table 5.1: Some numerical output for the models fitted to the Lexis mortality data described in this section, with the linear predictors given in (5.10), (5.15), and (5.17)

Linear Predictor	λ_a	λ_a	κ	tr($\boldsymbol{H}$)	Deviance	BIC
(5.10)	0.0435	6.1233	∞	104	76086	76806
(5.15)	1.9765	0.0024	0.0038	182	15059	16370
(5.17)	0.0692	146.4196	639.0567	223	4475	6020

separate parameters, one for each of the two types of triangle within a year of birth; the linear predictor becomes

$$\boldsymbol{\eta} = \boldsymbol{B\theta} + \boldsymbol{C}_A\boldsymbol{\gamma}_A + \boldsymbol{C}_B\boldsymbol{\gamma}_B. \tag{5.17}$$

Table 5.1 shows how the three models compare for the German mortality data. We see that the additive models compare favourably with the basic smooth model, and by adding relatively few degrees of freedom we get a marked improvement in the deviance. We can also see that the smoothing parameters for the smooth component have been strengthened by the inclusion of the additive term. This is supported by Fig. 5.5 - the smooth is now less flexible in general, but exhibits a more consistent amount of flexibility across the surface. Figure 5.6 shows the cohort effects that have been extracted from the data, for recent cohorts the effects are quite small and it seems that a single parameter might have been sufficient for the Type-A and Type-B triangles in these cohorts. However the older cohorts show bigger movements away from the surface and there seems less consistency between the Type-A and Type-B triangles.

The work in this section is described in Kirkby and Currie (2006).

5.2 Smooth Period Shocks

The basic smooth model described in Chapter 3 assumes that the data are a smooth function of the explanatory variables subject to some stochastic variation. However,

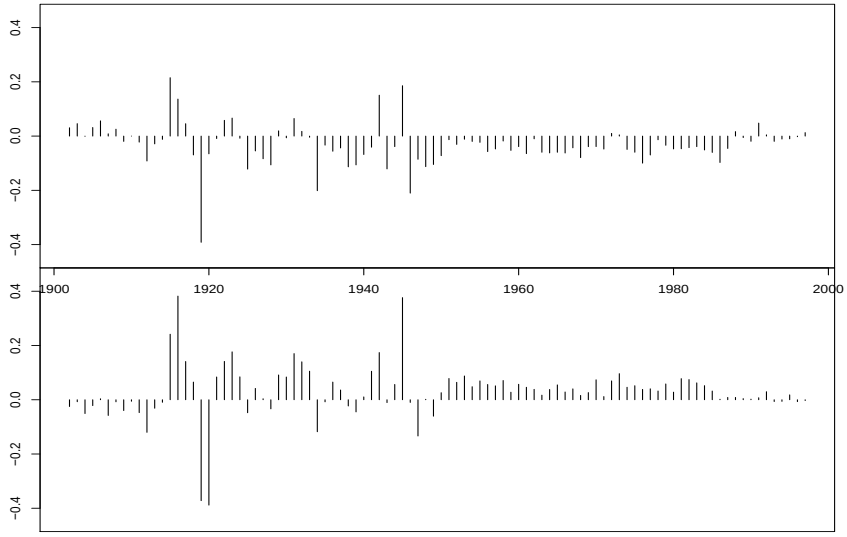


Figure 5.6: Cohort Shocks: Shocks estimated with (5.17) $\boldsymbol{\gamma}_A$ (top), $\boldsymbol{\gamma}_B$ (bottom).

observed mortality can be subject to events that cause systematic deviations from the underlying smooth. Specific examples include cold winters or disease epidemics which can have dramatic effects on the mortality experience in certain years. The Spanish flu pandemic of 1918-19 is an extreme example of this kind of effect. Such an effect can often have a different effect on different ages; for example, the Spanish flu pandemic principally affected those aged between 20 and 50, and had little or no effect on those over the age of 50. In this section we propose a model for dealing with such "shocks" to the surface, and illustrate the model with reference to some Swedish mortality data (Human Mortality Database).

The Swedish mortality data are in the format described in Section 1.3, with $\boldsymbol{x_a} = [10, \ldots, 90]'$ and $\boldsymbol{x_y} = [1900, \ldots, 2003]'$. Figure 5.7 shows a plot of the log mortality surface that we wish to model; a striking feature of these data is the Spanish flu epidemic which causes the ridge running along the lower ages in 1918. Fitting the model described in Section 3.3.2 to these data results in the smooth log mortality surface shown in Fig. 5.8. There is a clear bump around 1918 as the surface attempts to adjust for the higher mortality due to the flu epidemic. However, these bumps in the surface do not model the mortality successfully for either 1918 or the surrounding years. The influence of 1918 also has a dramatic effect on the selection of the smoothing parameters, for which lower values are selected to allow for the bumps around

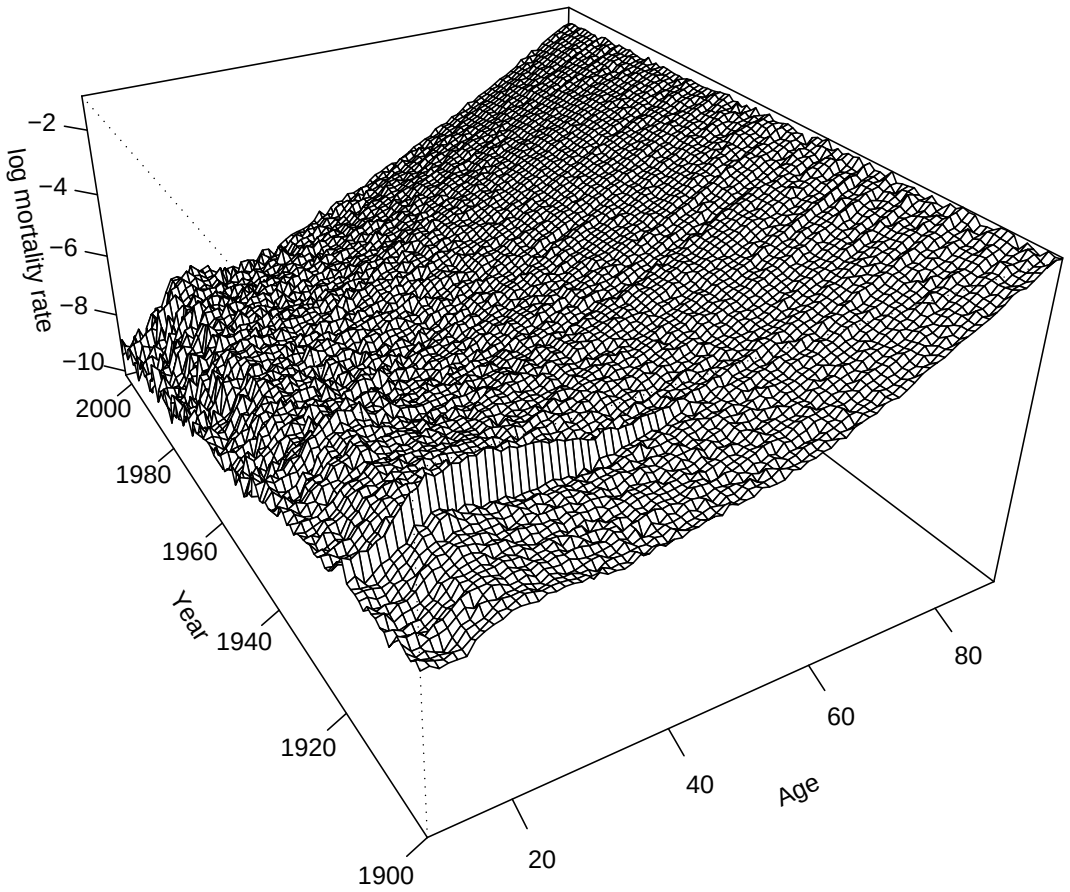


Figure 5.7: The observed log mortality rates for the Swedish data

1918, but this results in under-smoothing of the data in other parts of the surface.

The basic smooth model is not appropriate for modelling mortality data with features like the 1918 flu; instead we require a model which allows for large differences in mortality in neighbouring years. We propose an additive model with two components: the first is a smooth two-dimensional surface and the second describes the period shocks. The underlying smooth can be modelled using the two-dimensional model described in Section 3.3.2. Initially we might try to model the shocks with a single parameter for each year which describes a mean adjustment to the mortality surface across all ages. However, looking at Fig. 5.7 we see that the effect of the 1918 flu was not the same across all ages. The upper right panel of Figure 5.9 shows in greater detail that the mortality of those aged between twenty and forty was most dramatically effected, but almost no effect is observed for those over sixty; further, this effect appears to follow a smooth trend over age. We therefore model each period shock with an individual spline across ages. We define a second regression matrix by

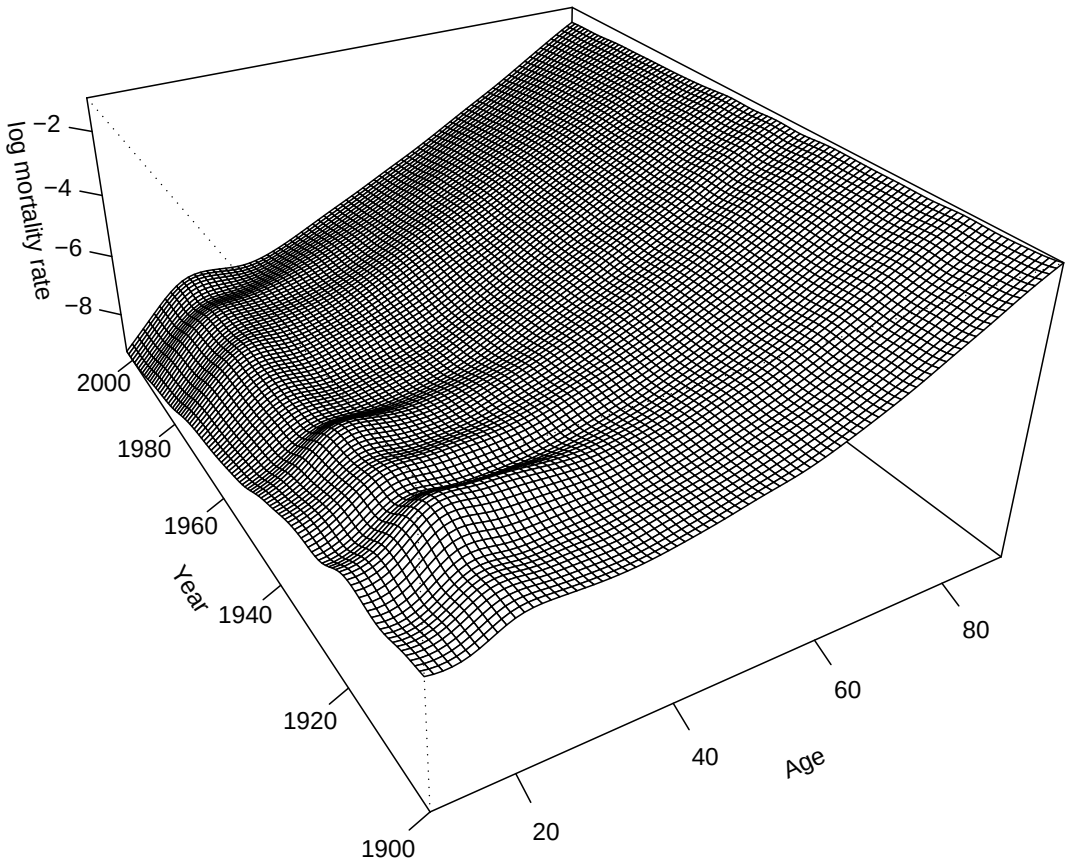


Figure 5.8: The smooth rates from the basic two-dimensional model described in Chapter 3.

$\boldsymbol{I}_{n_y} \otimes \breve{\boldsymbol{B}}_a$ where $\breve{\boldsymbol{B}}_a$ is a marginal regression matrix of B-splines of dimension $n_a \times c$, and model the log rate of mortality as

$$\log \boldsymbol{\tau} = (\boldsymbol{B_y} \otimes \boldsymbol{B_a})\boldsymbol{\theta} + (\boldsymbol{I}_{n_y} \otimes \breve{\boldsymbol{B}}_a)\breve{\boldsymbol{\theta}}. \tag{5.18}$$

In this form the model is not identifiable: we apply a ridge penalty to $\breve{\boldsymbol{\theta}}$ to maintain identifiability, and we have a penalty matrix of the form

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{0} \\ \boldsymbol{0} & \breve{\boldsymbol{P}} \end{bmatrix}, \tag{5.19}$$

where $\breve{\boldsymbol{P}} = \lambda_r \boldsymbol{I}_{n_y} \otimes \boldsymbol{I}_c$ and $\boldsymbol{P}$ is given in (3.24). A further effect of the ridge penalty is to shrink the coefficients $\breve{\boldsymbol{\theta}}$ towards zero in the same fashion as the random effects in a mixed model are shrunk. This is in keeping with the idea that such shocks to the surface could be considered as random effects. This model now includes $c_a c_y + c n_y$ parameters, and even taking c to be small so that the modelling of the within year

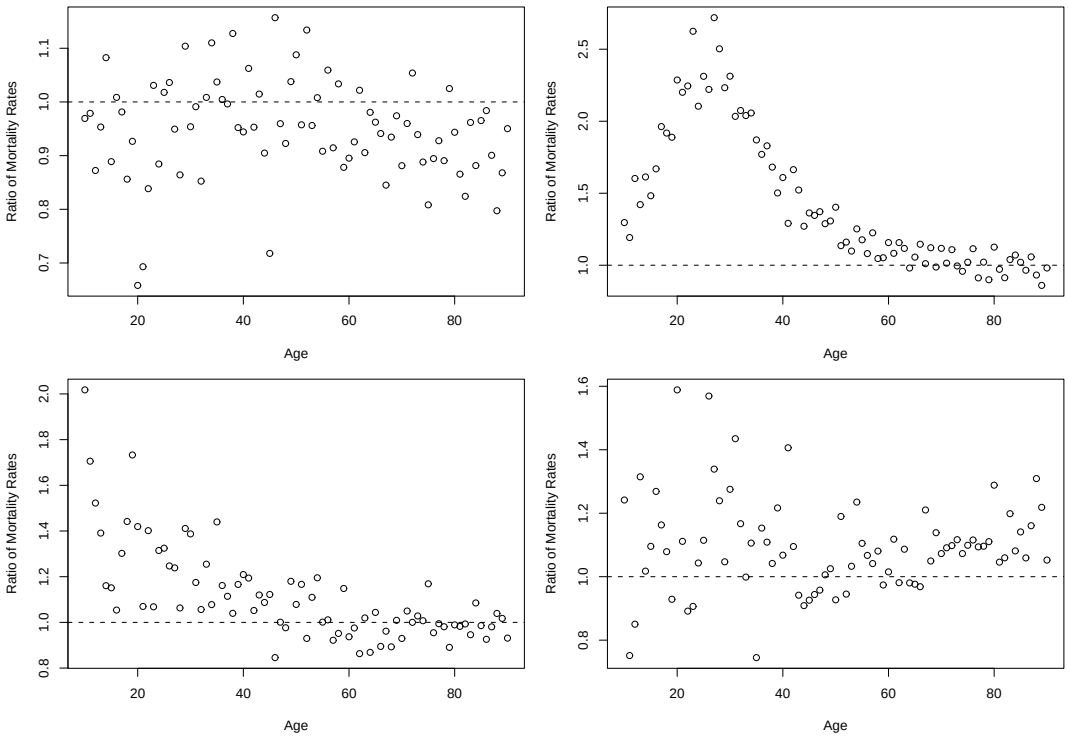


Figure 5.9: Ratios of year on year observed mortality rates for 1914/15 (top left), 1918/19 (top right), 1945/46 (bottom left), 1947/48 (bottom right).

effects is not precise, it is still a computationally demanding problem and we have to take advantage of the fast array methods described in Chapter 4.

The model can be fitted using the penalized scoring algorithm. Clearly, the linear predictor in (5.18) can be computed efficiently by twice applying (4.14), and written in array form, we have

$$\log \boldsymbol{T} = \boldsymbol{B_a}\boldsymbol{\Theta}\boldsymbol{B_y}' + \breve{\boldsymbol{B}}_a\breve{\boldsymbol{\Theta}}. \tag{5.20}$$

We also require evaluation of the inner-product matrix

$$\begin{bmatrix} (\boldsymbol{B_y} \otimes \boldsymbol{B_a})'\boldsymbol{W}_\delta(\boldsymbol{B_y} \otimes \boldsymbol{B_a}) & (\boldsymbol{B_y} \otimes \boldsymbol{B_a})'\boldsymbol{W}_\delta(\boldsymbol{I}_{n_y} \otimes \breve{\boldsymbol{B}}_a) \\ (\boldsymbol{I}_{n_y} \otimes \breve{\boldsymbol{B}}_a)'\boldsymbol{W}_\delta(\boldsymbol{B_y} \otimes \boldsymbol{B_a}) & (\boldsymbol{I}_{n_y} \otimes \breve{\boldsymbol{B}}_a)'\boldsymbol{W}_\delta(\boldsymbol{I}_{n_y} \otimes \breve{\boldsymbol{B}}_a) \end{bmatrix}. \tag{5.21}$$

The inner product is in the partition form given in (4.27). However, for the scoring algorithm we need the inverse of the inner product, which can be calculated more efficiently from its components. Splitting the weight matrix into partitions for each

year we see that the bottom right partition of (5.21) is the block diagonal matrix

$$(\boldsymbol{I}_{n_y} \otimes \breve{\boldsymbol{B}}_a)'\boldsymbol{W}_\delta(\boldsymbol{I}_{n_y} \otimes \breve{\boldsymbol{B}}_a) = \begin{bmatrix} \breve{\boldsymbol{B}}_a'\boldsymbol{W}_{\delta 1}\breve{\boldsymbol{B}}_a & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \breve{\boldsymbol{B}}_a'\boldsymbol{W}_{\delta 2}\breve{\boldsymbol{B}}_a & & \\ \vdots & & \ddots & \\ \boldsymbol{0} & & & \breve{\boldsymbol{B}}_a'\boldsymbol{W}_{\delta n_y}\breve{\boldsymbol{B}}_a \end{bmatrix}, \quad (5.22)$$

where $\boldsymbol{W}_{\delta j}$ is the j th $n_a \times n_a$ diagonal block in $\boldsymbol{W}_\delta$. The structure of (5.22) allows us to benefit from two computational advantages. Firstly, we can use the one-dimensional array method to efficiently compute the components. Normally the one-dimensional array computation is less efficient than regular matrix computation; however in this case, the repeated appearance of the matrix, $\breve{\boldsymbol{B}}_a$, in the inner product allows for a small computational gain in forming the inner products using

$$\text{vec}(\breve{\boldsymbol{B}}_a'\boldsymbol{W}_{\delta j}\breve{\boldsymbol{B}}_a) = (\boldsymbol{G}(\breve{\boldsymbol{B}}_a))'\text{diag}(\boldsymbol{W}_{\delta j}), \quad j = 1, \ldots, n_y, \quad (5.23)$$

where $\text{diag}(\boldsymbol{W}_{\delta j})$ is the vector containing the diagonal entries from $\boldsymbol{W}_\delta$ corresponding to the jth year. Thus, all the elements of the matrix on the left hand side of (5.22) can be efficiently calculated using the one-dimensional GLAM formula

$$(\boldsymbol{G}(\breve{\boldsymbol{B}}_a))'\boldsymbol{W}, \quad (5.24)$$

where $\boldsymbol{W} = \overset{a}{\boldsymbol{W}}_\delta$, $n_a \times n_y$. The second computational advantage comes from the block diagonal structure. We can find the inverse of (5.22) efficiently by separately inverting the n_y components each of which is a $c \times c$ matrix. With an efficient way of inverting (5.22) we can use the following theorem to invert the full matrix in (5.21).

Theorem 5.1 *A partition matrix of the form*

$$\begin{bmatrix} \boldsymbol{W} & \boldsymbol{V} \\ \boldsymbol{U} & \boldsymbol{T} \end{bmatrix}, \quad (5.25)$$

is non-singular if and only if $\boldsymbol{T}$ *is non-singular, and the matrix*

$$\boldsymbol{Q} = \boldsymbol{W} - \boldsymbol{V}\boldsymbol{T}^{-1}\boldsymbol{U} \quad (5.26)$$

is non-singular, in which case

$$\begin{bmatrix} \boldsymbol{W} & \boldsymbol{V} \\ \boldsymbol{U} & \boldsymbol{T} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{Q}^{-1} & : & -\boldsymbol{Q}^{-1}\boldsymbol{V}\boldsymbol{T}^{-1} \\ -\boldsymbol{T}^{-1}\boldsymbol{U}\boldsymbol{Q}^{-1} & : & \boldsymbol{T}^{-1} + \boldsymbol{T}^{-1}\boldsymbol{U}\boldsymbol{Q}^{-1}\boldsymbol{V}\boldsymbol{T}^{-1} \end{bmatrix}. \quad (5.27)$$

The matrix $\boldsymbol{Q}$ *is known as the Schur complement.*

Table 5.2: Time taken to find the inverse of the inner product matrix (5.21) using R (R Development Core Team, 2005) on an Intel Core Duo 1.73 GHz Linux machine. These figures are given for the Swedish data with $n_a = 81$, $n_y = 104$, $c_a = 19$, $c_y = 24$, and $c = 9$.

Method	Time (in seconds)
Direct calculation	38.830
Array methods only, (5.24)	9.749
Array methods with Schur complement, (5.24) and (5.27)	3.211

A proof of this is given by Harville (1997). For the matrix in (5.21) the Schur complement has dimension $c_a c_y \times c_a c_y$, and inverting a matrix of this size would be required for the scoring algorithm anyway. Therefore, the additional computational overhead required for inverting (5.21) is reduced to the inversion of (5.22) plus a little extra for constructing the components of (5.27). Timings for the construction of the inverse of (5.21) for the Swedish Male example are given in Table 5.2. We see that using array methods together with the Schur complement reduces the time required to calculate the inverse of (5.21) by a factor of over ten.

The two components of the additive model fitted to the Swedish male data are shown in Fig. 5.11 and Fig. 5.12, and the full fitted mortality surface is shown in Fig. 5.13. We can see that the fitted mortality surface is no longer smooth across years, in particular a substantial fin shaped spline is added to the surface to model the excess mortality caused by the Spanish flu in 1918. There also appears to be a noticeable shock for the lower ages in 1944. At the oldest ages the size of the shocks are not as large, but small adjustments to the surface appear fairly regularly, perhaps explained by particularly cold or mild winters.

We summarize the results for the following three models:

- Basic: The basic two-dimensional smooth model with linear predictor in array form

$$\boldsymbol{B_a \Theta B_y}'. \tag{5.28}$$

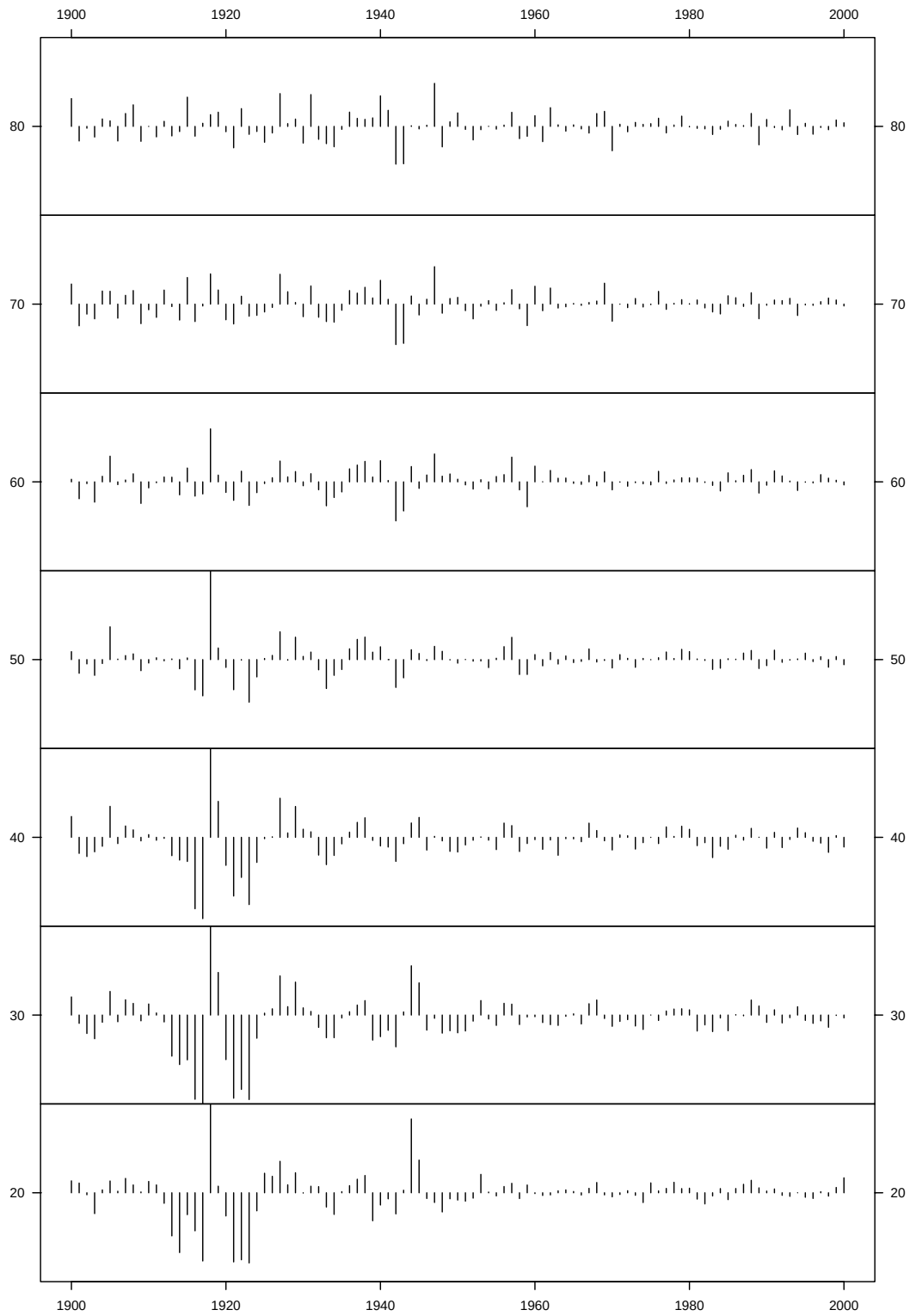


Figure 5.10: Cross-sections of the spline shocks in each year for selected ages.

- Mean Shock: The two-dimensional smooth model with a separate mean shock for each year with linear predictor in array form

$$\boldsymbol{B_a}\boldsymbol{\Theta}\boldsymbol{B_y}' + \mathbf{1}_{n_a}\breve{\boldsymbol{a}}'. \tag{5.29}$$

- Smooth Shock: The two-dimensional smooth model with a separate smooth spline shock for each year with linear predictor in array form

$$\boldsymbol{B_a}\boldsymbol{\Theta}\boldsymbol{B_y}' + \breve{\boldsymbol{B}}_a\breve{\boldsymbol{\Theta}}. \tag{5.30}$$

The mean shock model in equation (5.29) adds a single shock (also subject to a ridge penalty) across all ages in each year. We include the mean shock model for comparison with the smooth shock model to illustrate the age dependence of some of the shocks.

The addition of the within-year shocks has had a noticeable effect on the underlying smooth component of the model. Figure 5.11 appears smoother to the eye than Fig. 5.8; this is consistent with the values of the smoothing parameters selected for the models. Table 5.3 shows the smoothing parameters selected by BIC for the three models discussed in this section. For the basic smooth model a relatively small value is selected for both smoothing parameters, but the smoothing parameter for age is bigger than that for year. For the model with a mean shock for each year we see an increase in the year smoothing parameter but a decrease in that for age. This pattern continues when we introduce a within-year spline by further strengthening the year penalty. The strengthening of the year penalty makes sense for this model, as the effects that are modelled by the within-year splines would have to be modelled by the smooth surface in the basic model, giving rise to the bumps in the mortality surface in Fig. 5.8. Figure 5.12 shows the within-year shocks that have been separated from the underlying trend; 1918 dominates the plot, but we can see a series of consecutive shocks at younger ages in the mid 1940s; this can be seen more clearly in Fig. 5.10, where the shock only appears most prominent at younger ages in 1944. Combining the two components shown in Fig. 5.11 and Fig. 5.12 we obtain the additive two-dimensional function shown in Fig. 5.13. The discrete shocks to the surface by year seem to do a better job of explaining the data shown in Fig. 5.7, this is supported by a marked reduction in the BIC shown in Table 5.3. Separating the within-year

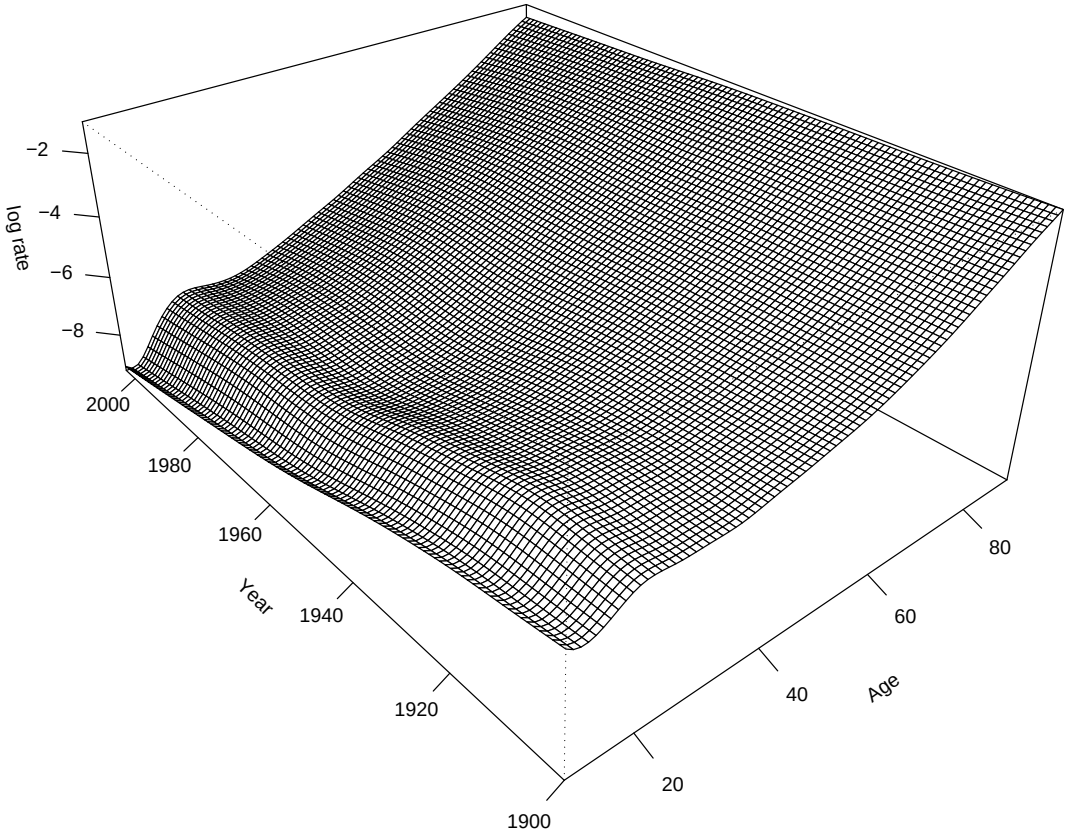


Figure 5.11: The smooth component of the model in (5.20) applied to the Swedish male data.

features from the smooth trend over time should also enable improved forecasting, since, as shown in Section 3.3.6, stronger year penalties generally lead to more stable forecasts of the mortality surface.

The work in this section is described in Kirkby and Currie (2007).

5.3 Discussion

In Sections 5.1 and 5.2 we have applied P-spline components within additive models to specific mortality examples. In both cases we were able to fit a model with a large number of parameters, and in the second case we were able to make large computational savings based on methods in Chapter 4. In Section 5.3.1 we discuss how the models might be successfully applied to other mortality data sets, and to other slightly different situations. In Section 5.3.2 we discuss how these type of additive models are connected to models for over-dispersion, and in Section 5.3.3

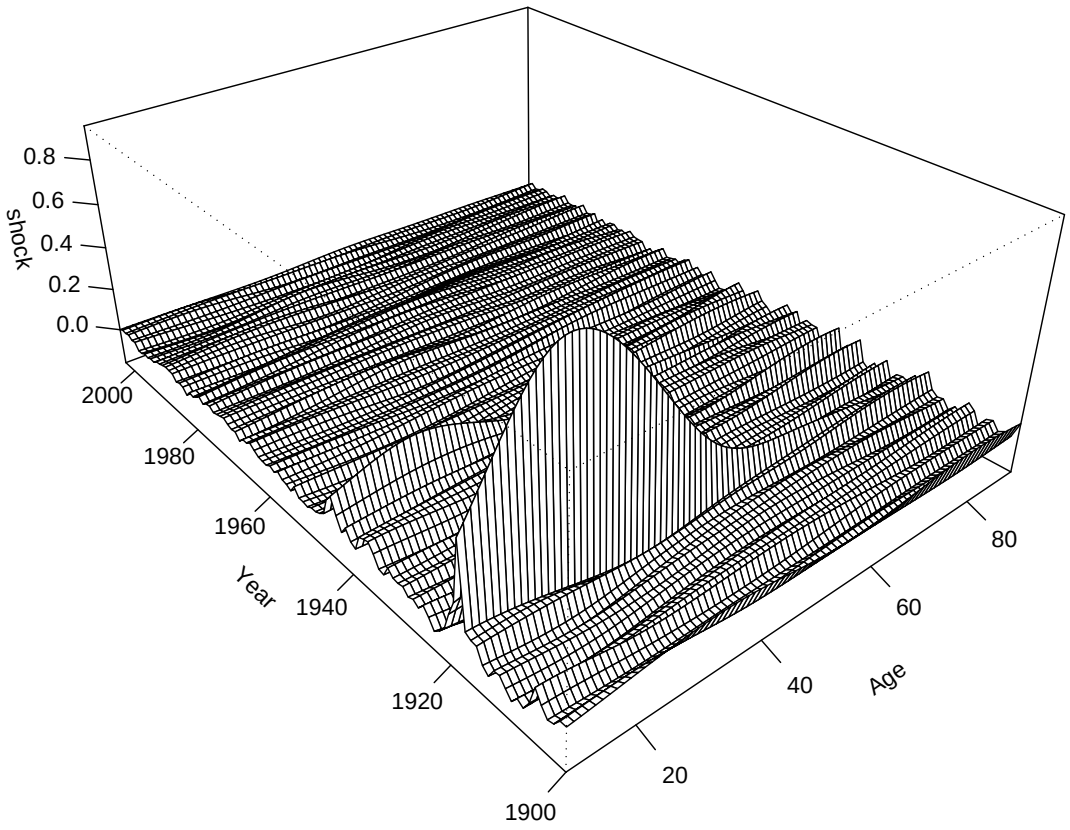


Figure 5.12: The shock spline components of the model in (5.20) applied to the Swedish male data.

we look at possible improvements to the model and how these improvements could be implemented.

5.3.1 Other Applications

The Spanish flu epidemic visible in the data for the Swedish males examined in Section 5.2 is an extreme example of the kind of period shocks that can occur in mortality data, and does not appear in many data sets. Nevertheless, the model should be an improvement for most data sets because the smaller adjustments fitted at the highest ages would be appropriate for most data sets. For example, Fig. 5.14 shows a strip of the mortality surface for the highest ages of the CMI assured lives data, we can clearly see that ridges are visible in 1964, 1977, and 1993. These sorts of features are common in many mortality data sets as the severity of the winter in a particular year tends to effect geriatric mortality in that year.

The model also highlights a period shock in 1944, although this is not as extreme

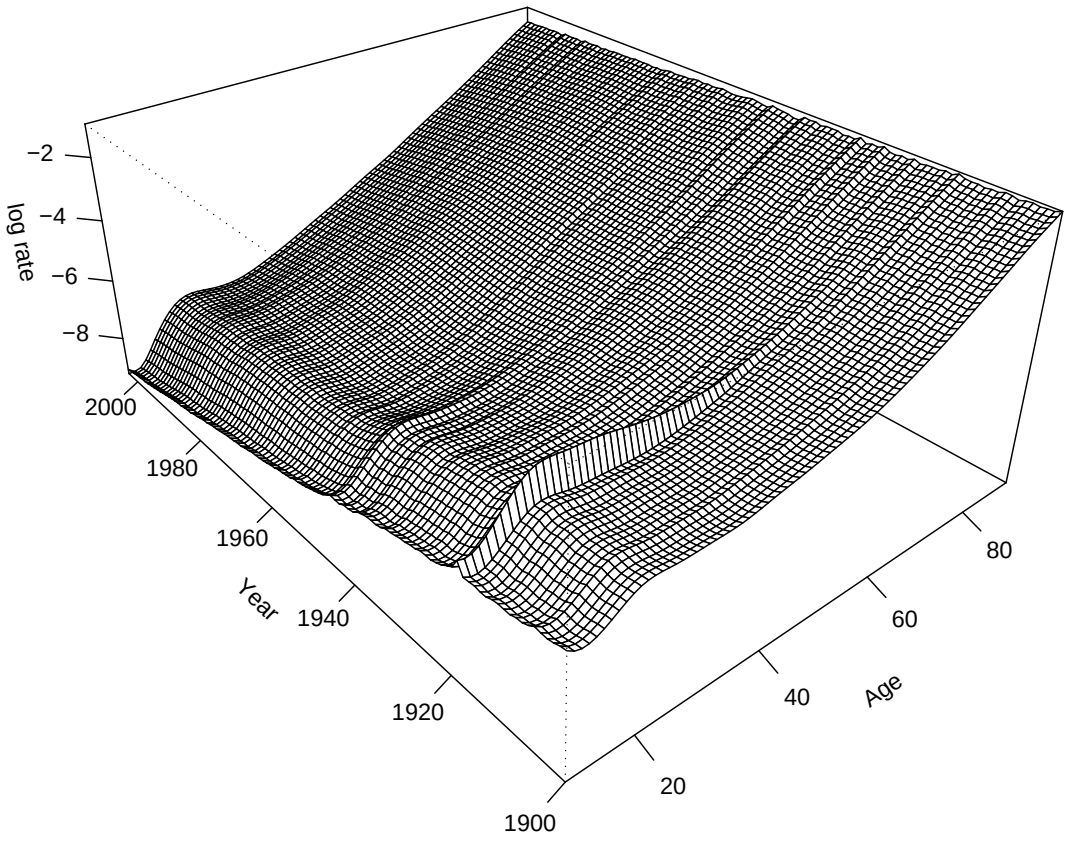


Figure 5.13: The fitted mortality surface for the Swedish Males.

as 1918 it significantly effects the mortality surface in Fig. 5.8, so even without the severity of 1918 it seems the additive model would be an improvement. Such smaller scale shocks may be common to other data sets in the form of disease epidemics or other events; for example, it is known that strange effects in adult mortality were observed in East Germany after reunification, so this may be a fruitful application of the additive model.

A slightly different application could be to model digit preferencing. For population data in developing countries (and for older data from developed countries) the exact age of some individuals is not known. When recording an event of interest (a death, mother's age at birth) a guess of the subject's age is often recorded. Typically this guess will be a round figure, so in raw population statistics for developing populations we often observe spikes in the number of events at the ages occurring at ages which are a multiple of five. As recording gets better over time, the size of these "rounding errors" tends to improve, so a time dependent shock (a within-age spline) could be used effectively to take these anomalies in the data into account.

Table 5.3: Various statistics for the three models (5.28), (5.29) and (5.30) for the Swedish male data.

Model	λ_a	λ_y	λ_r	Trace	Deviance	BIC
Basic	10.00	7.00	-	293	21226	23871
Mean shock	0.05	30.00	2000.00	367	15538	18852
Smooth shock	0.01	1900.00	850.00	489	9670	14089

5.3.2 Over-dispersion

There are some connections between the models described in Sections 5.1 and 5.2 and the over-dispersion models described in Perperoglou and Eilers (2006 (personal communication). These authors account for over-dispersion by fitting an extra parameter for each data point. Their approach differs from the usual method of over-dispersion modelling which assumes the specification of the variance is not correct. In the standard approach estimation usually proceeds by using quasi-likelihood to re-specify the mean-variance relationship. The two approaches correspond to different underlying structures in the data: the Perperoglou and Eilers (2006 (personal communication) approach would be appropriate in the case of an underlying smooth trend shocked by individual random effects, while the second corresponds to a genuinely over-dispersed distribution for each data point. In one dimensional problems, it is difficult to distinguish between the two structures, and the quasi-likelihood approach has the advantage of being able to deal with under-dispersion. In higher dimensional problems, as here, we can distinguish between the two structures provided that the shocks follow some kind of systematic pattern. In both the cases described in this chapter, the basic smooth model appeared defective to the eye, so the extension to an additive model was natural. In other situations close inspection of the data may lead to identification of additive effects that may otherwise be put down to over-dispersion.

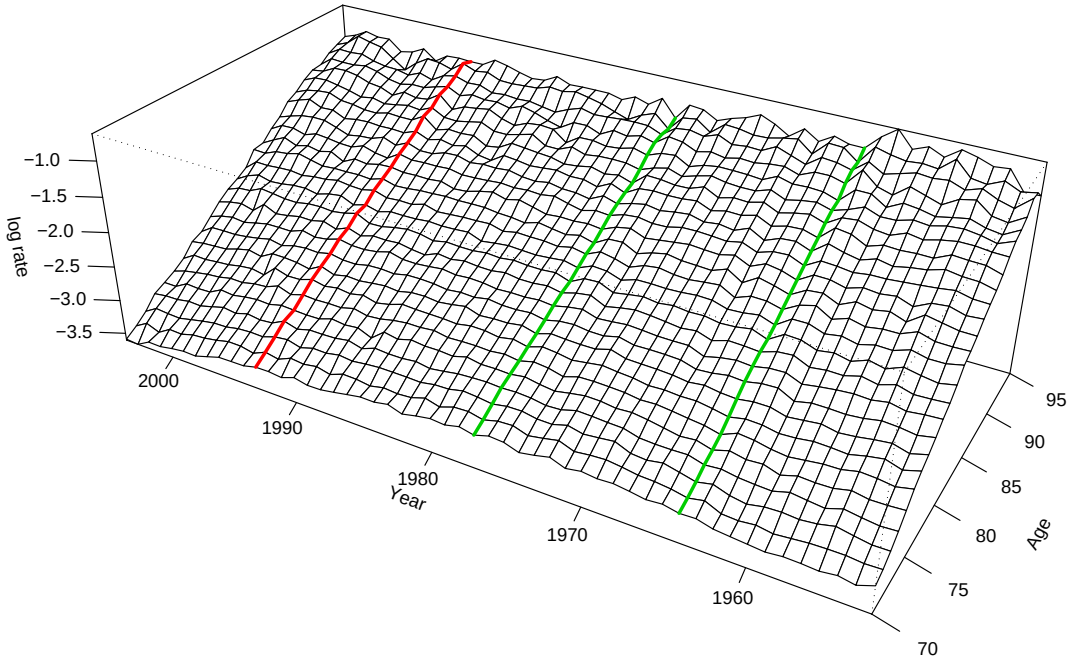


Figure 5.14: A strip of the observed mortality surface for the highest ages in the assured lives data showing similar period shocks to those modelled in the Swedish male data.

5.3.3 Further work

The first area where the additive model could be improved is to fit within the mixed model framework. The shock effects could then be considered genuine random effects, and estimates of the variance parameters would have an interpretation as the size of deviations a shock might produce from the under-lying smooth trend. Using the mixed model representation Section 2.5.3 we could fit the under-lying smooth surface, selecting the smoothing parameters by REML. Adding the within-year spline parameters as random effects on top of this surface then gives them the random interpretation. In this case we would be fitting using the PQL approximation, and we may do better by fitting a full Bayesian model discussed in Section 2.7.

The cohort shock model described in Section 5.2 could be extended by replacing the mean shock with an age dependent effect. This type of model cannot be fitted without the use of array methods, so in order to fit such a model one could re-arrange the data as shown in Section 4.6 and fit the under-lying smooth by age and year of birth. This would then give a Kronecker product structure, and the fast array methods could then be used within the penalized scoring algorithm. Alternatively the model

could be fitted with a back-fitting algorithm keeping the under-lying smooth in an age and period format, but on the iterations to update the within-cohort splines, one would change to a re-arranged version of the data allowing the use of the array methods in both iterations.

Chapter 6

Further Topics

In this chapter we will cover two topics which are separate from the rest of the thesis. Firstly, we will discuss the Lee-Carter model which was mentioned in Chapter 1. Secondly, we will discuss the outcome of the CASE studentship that initiated this thesis.

6.1 The Lee-Carter Model

As mentioned in Section 1.2 the most widely used model of mortality in recent times was that proposed by Lee and Carter (1992); as we shall see below, the model is particularly suitable for forecasting future mortality rates. Indeed, Lee and Carter designed their model with the express purpose of predicting life expectancy from US national mortality data. Since their initial paper the Lee-Carter method has been applied to many national data sets with varying degrees of success; see Booth et al. (2006).

The model has a bi-linear structure, to be precise we have a linear predictor

$$\log \mu_{x,t} = \eta_{x,t} = \alpha_x + \beta_x \kappa_t, \tag{6.1}$$

where α_x can be interpreted as the average mortality rate at age x over the n_y years; this is modified by an age-adjusted time trend with age adjustment β_x (at age x) and the time trend κ_t in year t. The linear predictor can be written in matrix notation

$$\log \boldsymbol{M} = \boldsymbol{H} = \boldsymbol{\alpha}\mathbf{1}' + \boldsymbol{\beta}\boldsymbol{\kappa}'. \tag{6.2}$$

In their original paper Lee and Carter estimate $\boldsymbol{\alpha}$ as the row average of the log of the raw rates

$$\hat{\boldsymbol{\alpha}} = \text{ave} \log \boldsymbol{T}_{x,\cdot}. \tag{6.3}$$

where $\boldsymbol{T} = \boldsymbol{T}_{x,t} = \boldsymbol{Y}_{x,t}/\boldsymbol{E}_{x,t}$ is the matrix of raw mortality rates; see section 1.3.1. The estimate $\hat{\boldsymbol{\alpha}}$ is then subtracted from each column of the log raw rates

$$\tilde{\boldsymbol{H}} = \log \boldsymbol{T} - \hat{\boldsymbol{\alpha}}\mathbf{1}' \tag{6.4}$$

to give the age adjusted raw rates. Lee and Carter then computed the singular value decomposition of $\tilde{\boldsymbol{H}}$, and estimated $\boldsymbol{\beta}$ and $\boldsymbol{\kappa}$ by the eigenvectors corresponding to the first singular value. This method allowed Lee and Carter to summarize the dynamics of the mortality surface with a one-dimensional time-series. Once the time-series $\hat{\boldsymbol{\kappa}}$ has been estimated forecasting follows as a second stage, by modelling $\hat{\boldsymbol{\kappa}}$ with an ARIMA time series (Lee and Carter recommended a first order auto-regressive model with drift); during the time series modelling process the age parameters $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ remain fixed at their estimated values.

One obvious extension to this model is to replace the rank one approximation to $\tilde{\boldsymbol{H}}$ with a higher order approximation; however this would remove the key benefit of the model: the resulting single time series. A significant improvement to the model was made by Brouhns et al. (2002) who brought the model within the likelihood framework. They started with the Poisson assumption discussed in Section 1.3.4, and then, using maximum likelihood, they showed that estimation of the parameters could be achieved using what they termed a one-step Newton method. An alternative method with the Poisson distribution assumption is to iterate between two conditional GLMs as follows. First, with $\boldsymbol{\kappa}$ at some current value $\tilde{\boldsymbol{\kappa}}$ we have a GLM with linear predictor

$$\boldsymbol{\eta} = \text{vec}(\boldsymbol{H}) = \boldsymbol{X}_y \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix}, \tag{6.5}$$

which can be used to update the estimates of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$; here the model matrix is $\boldsymbol{X}_y = [\mathbf{1}_{n_y} \vdots \tilde{\boldsymbol{\kappa}}] \otimes \boldsymbol{I}_{n_a}$. Second, with $\boldsymbol{\beta}$ at some current value $\tilde{\boldsymbol{\beta}}$ we have a GLM with linear predictor

$$\boldsymbol{\eta} = \boldsymbol{X}_a \boldsymbol{\kappa}, \tag{6.6}$$

where $\boldsymbol{X}_a = \boldsymbol{I}_{n_y} \otimes \tilde{\boldsymbol{\beta}}$, and on this iteration we use $\boldsymbol{\alpha}$ at some current value $\tilde{\boldsymbol{\alpha}}$ as an offset in the model. Clearly with a Kronecker product as the basis for both models we are able compute the model efficiently using the array methods described in the previous chapter. Currie (2008) investigates this approach.

An important consideration when fitting the model using either the one-step Newton method or the iterative GLM method is identifiability of the model. With a bilinear model of this type we have to identify a scale and location for the parameters, as for any set of parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\kappa}$ and a constant c then

$$\boldsymbol{\alpha}\mathbf{1}' + \boldsymbol{\beta}\boldsymbol{\kappa}' = \tilde{\boldsymbol{\alpha}}\mathbf{1}' + \tilde{\boldsymbol{\beta}}\tilde{\boldsymbol{\kappa}}' \tag{6.7}$$

for $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha} - c\boldsymbol{\beta}$ and $\tilde{\boldsymbol{\kappa}} = \boldsymbol{\kappa} + c$, or $\tilde{\boldsymbol{\beta}} = c\boldsymbol{\beta}$ and $\tilde{\boldsymbol{\kappa}} = \boldsymbol{\kappa}/c$. The problem of identifiability was avoided in Lee and Carter's original solution because the two-stage procedure identifies a location for $\boldsymbol{\alpha}$, and the singular value approximation selects an arbitrary scale for $\boldsymbol{\beta}$ and $\boldsymbol{\kappa}$. Without constraints and with the Poisson approach the model will still converge but to estimates of the parameters which depend on the initial values; of course, the estimated fitted values are unique. The `gnm` package in R of Turner and Firth (2008) can be used to fit the Lee-Carter model in this unconstrained fashion with the parameterization generated by random starting values; see the `gnm` manual for an example.

Although the model can be fitted without constraints, enforcing constraints after each update speeds up convergence, so we will follow the recommendation of Brouhns et al. (2002) and use the location constraint $\sum \kappa_t = 0$ and the scale constraint $\sum \beta_x = 1$. Figure 6.1 shows plots of the fitted parameters together with the mortality for three ages with the Poisson-based model for the CMI data. We make the obvious remark that the age profiles as shown in Fig. 6.1 are scaled and shifted copies of the time series $\hat{\boldsymbol{\kappa}}$.

Figure 6.1 suggests that the discrete estimates of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\kappa}$ can be replaced by smooth functions. Forecasting the mortality table can be achieved by using the penalty to forecast the $\boldsymbol{\kappa}$ values, as shown in section 2.8. This is in the spirit of Lee and Carter's original suggestion that forecasting of the mortality table can be reduced to the forecasting of a single time series function. We parameterize the components

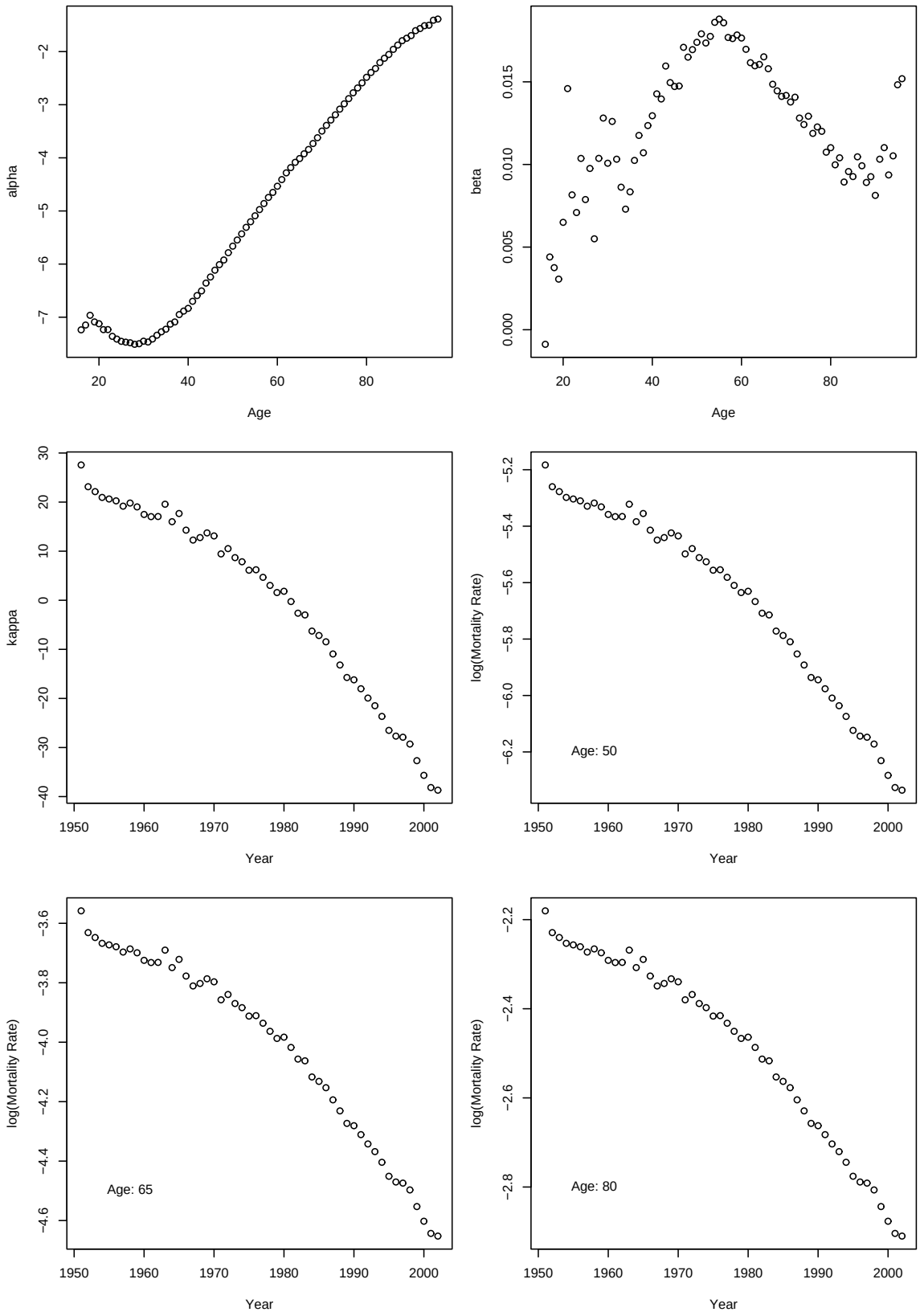


Figure 6.1: Output of Lee-Carter model. Components of the model: $\boldsymbol{\alpha}$ (top-left), $\boldsymbol{\beta}$ (top-right), and $\boldsymbol{\kappa}$ (middle-left). Log mortality rates for three different ages: 50 (middle-right), 65 (bottom-left), and 80 (bottom-right).

as B-spline functions

$$\boldsymbol{\alpha} = \boldsymbol{B}_a\boldsymbol{\theta}_\alpha \quad \text{and} \quad \boldsymbol{\beta} = \boldsymbol{B}_a\boldsymbol{\theta}_\beta \quad \text{and} \quad \boldsymbol{\kappa} = \boldsymbol{B}_y\boldsymbol{\theta}_\kappa. \tag{6.8}$$

Note that the same basis functions are used to smooth both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$; using the same basis functions simplifies the mathematics and there seems no good reason to use different bases. With this parameterization the same iterative GLM approach can be used as in the discrete case by replacing the design matrices in (6.5) and (6.6) with the matrices $\boldsymbol{X}_y = [\mathbf{1}_{n_y} \vdots \tilde{\boldsymbol{\kappa}}] \otimes \boldsymbol{I}_{n_a}$ and $\boldsymbol{X}_a = \boldsymbol{B}_y \otimes \tilde{\boldsymbol{\beta}}$. We tune the smoothness for each component by applying a roughness penalty to $\boldsymbol{\theta}_\alpha$, $\boldsymbol{\theta}_\beta$, and $\boldsymbol{\theta}_\kappa$. It follows that instead of the scoring algorithm we now use the penalized scoring algorithm for each update of the parameters, with the penalty matrices given by:

$$\boldsymbol{P}_A = \begin{bmatrix} \lambda_\alpha \boldsymbol{P}_a & \mathbf{0} \\ \mathbf{0} & \lambda_\beta \boldsymbol{P}_a \end{bmatrix}, \quad \boldsymbol{P}_a = \boldsymbol{D}_a'\boldsymbol{D}_a \tag{6.9}$$

and

$$\boldsymbol{P}_Y = \lambda_\kappa \boldsymbol{P}_y, \quad \boldsymbol{P}_y = \boldsymbol{D}_y'\boldsymbol{D}_y. \tag{6.10}$$

We consider the effect of the parameterization and penalty on the penalized likelihood. First, we consider a change of scale. We rescale the parameters so that $\tilde{\boldsymbol{\beta}} = c\boldsymbol{\beta}$ and $\tilde{\boldsymbol{\kappa}} = \boldsymbol{\kappa}/c$, and then rescale the smoothing parameters so that $\tilde{\lambda}_\beta = \lambda_\beta/c^2$ and $\tilde{\lambda}_\kappa = c^2\lambda_\kappa$: the penalized likelihood remains the same. However, a change of location results in a change of the penalized likelihood. Consider the location transformation $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha} - c\boldsymbol{\beta}$ and $\tilde{\boldsymbol{\kappa}} = \boldsymbol{\kappa} + c$. We consider the penalty on $\boldsymbol{\kappa}$ first: the original penalty is

$$p_Y = \lambda_\kappa \boldsymbol{\theta}_\kappa' \boldsymbol{P}_y \boldsymbol{\theta}_\kappa$$

and the new penalty is

$$\begin{aligned} \tilde{p}_Y &= \lambda_\kappa \tilde{\boldsymbol{\theta}}_\kappa' \boldsymbol{P}_y \tilde{\boldsymbol{\theta}}_\kappa \\ &= \lambda_\kappa \boldsymbol{\theta}_\kappa' \boldsymbol{P}_y \boldsymbol{\theta}_\kappa + \lambda_\kappa c^2 \mathbf{1}' \boldsymbol{P}_y \mathbf{1} \\ &= \lambda_\kappa \boldsymbol{\theta}_\kappa' \boldsymbol{P}_y \boldsymbol{\theta}_\kappa \\ &= p_Y, \end{aligned} \tag{6.11}$$

so the penalty on $\boldsymbol{\kappa}$ is unchanged. However for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ the initial penalty is

$$p_A = \lambda_\alpha \boldsymbol{\theta}'_\alpha \boldsymbol{P}_a \boldsymbol{\theta}_\alpha + \lambda_\beta \boldsymbol{\theta}'_\beta \boldsymbol{P}_a \boldsymbol{\theta}_\beta$$

but after the change in location the new penalty is

$$\begin{aligned}
\tilde{p}_A &= \lambda_\alpha \tilde{\boldsymbol{\theta}}'_\alpha \boldsymbol{P}_a \tilde{\boldsymbol{\theta}}_\alpha + \lambda_\beta \tilde{\boldsymbol{\theta}}'_\beta \boldsymbol{P}_a \tilde{\boldsymbol{\theta}}_\beta \\
&= \lambda_\alpha (\boldsymbol{\theta}_\alpha - c\boldsymbol{\theta}_\beta)' \boldsymbol{P}_a (\boldsymbol{\theta}_\alpha - c\boldsymbol{\theta}_\beta) + \lambda_\beta \boldsymbol{\theta}'_\beta \boldsymbol{P}_a \boldsymbol{\theta}_\beta \\
&= \lambda_\alpha \boldsymbol{\theta}'_\alpha \boldsymbol{P}_a \boldsymbol{\theta}_\alpha - 2c\lambda_\alpha \boldsymbol{\theta}'_\alpha \boldsymbol{P}_a \boldsymbol{\theta}_\beta + (\lambda_\beta + \lambda_\alpha c^2) \boldsymbol{\theta}'_\beta \boldsymbol{P}_a \boldsymbol{\theta}_\beta \qquad (6.12) \\
&= p_A - 2c\lambda_\alpha \boldsymbol{\theta}'_\alpha \boldsymbol{P}_a \boldsymbol{\theta}_\beta + \lambda_\alpha c^2 \boldsymbol{\theta}'_\beta \boldsymbol{P}_a \boldsymbol{\theta}_\beta \\
&\neq p_A.
\end{aligned}$$

In conclusion, not only is the penalty different, but due to the cross product between $\boldsymbol{\theta}_\alpha$ and $\boldsymbol{\theta}_\beta$ there is no way to adjust λ_α or λ_β in order to recover the same overall penalty.

This inability to re-locate the parameters alerts us to a problem with the original penalty. By selecting a different set of constraints we are able to change the penalty for the model, because of the unpenalized interaction between $\boldsymbol{\theta}_\alpha$ and $\boldsymbol{\theta}_\beta$. To ensure that the penalized likelihood is not effected by an arbitrary choice of constraint, we could include a cross penalty on the $\boldsymbol{\theta}_\alpha$ and $\boldsymbol{\theta}_\beta$. The penalty would then have the form

$$\boldsymbol{P}_A \begin{bmatrix} \lambda_\alpha \boldsymbol{P}_a & \lambda_{\alpha\beta} \boldsymbol{P}_a \\ \lambda_{\alpha\beta} \boldsymbol{P}_a & \lambda_\beta \boldsymbol{P}_a \end{bmatrix}. \qquad (6.13)$$

By adding the cross term to the penalty we can find values of smoothing parameters which result in the same penalized likelihood regardless of the choice of parameterization (i.e. choice of c). However, with the parameterization constraint acting on $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\kappa}$ and the penalty acting on the B-spline parameters, there can be numerical problems when we try to convert from one parameterization to another. A better solution, proposed by Currie (2008), is direct penalization of neighbouring data points of the mortality surface, using the array interpretation of the penalty from equation (4.44) we obtain the penalty

$$\boldsymbol{P} = \lambda_1 \big(\mathrm{vec}(\boldsymbol{D}_A \boldsymbol{\Theta})\big)' \mathrm{vec}(\boldsymbol{D}_A \boldsymbol{M}) + \lambda_2 \big(\mathrm{vec}(\boldsymbol{\Theta} \boldsymbol{D}'_Y)\big)' \mathrm{vec}(\boldsymbol{M} \boldsymbol{D}'_Y). \qquad (6.14)$$

where the matrices $\boldsymbol{D}_A$ and $\boldsymbol{D}_Y$ are $(n_a - k) \times n_a$ and $(n_y - k) \times n_y$, and k is the order of the differences. Currie (2008) shows how these penalties can be incorporated into the standard penalized scoring algorithm from (2.75).

In summary, the Lee-Carter model is an important model for the forecasting of mortality rates. Delwarde et al. (2007) consider smoothing the $\boldsymbol{\beta}$ component of the model and leaving the $\boldsymbol{\alpha}$ and $\boldsymbol{\kappa}$ free; this avoids any problems with the invariance of the penalty with respect to different parameterizations. Our first contribution is to point out that smoothing all three parameters runs into difficulties since penalization is not invariant with respect to the parameterization of the model. We suggest a solution based on a modification of the penalty. Our second contribution is to show that the Lee-Carter model can be fitted using array methods by iterating between two GLMs each with an array structure.

6.2 CASE Studentship

The funding for this thesis was based on a CASE studentship which was partly funded by the CMI (Continuous Mortality Investigation). As mentioned in Chapter 1, the CMI is a body set-up on behalf of the life assurance industry to collate and analyze the UK assured lives population. In particular, the studentship was awarded to help the work of the CMI's mortality projections working party. In their recent publications the CMI, CMI (1990) and CMI (1999), had favoured publication of a set of base mortality tables along with a set of *reduction factors*. Reduction factors are used to adjust a mortality table to allow for improvements in mortality over time. The concept is easily illustrate with an example. CMI (1999) gave a set of mortality tables which were based in the year 1992 and, along with the base tables, they published another table of reduction factors of the form $RF(x, t)$ for the integer ages, x, and integer annual time intervals, t. In order to calculate the projected $q_{65,2000}$ (the probability that somebody age 65 at the start of the year 2000 will die over the next year) we simply take the q_{65} ($q_{65,1992}$) from the base table and multiply by the appropriate reduction factor, so we have

$$q_{65,2000} = q_{65} \times RF(65, 2000 - 1992). \tag{6.15}$$

Producing reduction factors is a convenient way for actuaries to account for mortality dynamics in their calculations. However, having standard reduction factors seems to reduce the responsibility of the individual actuary to take account of the uncertainty surrounding mortality projection, and may possibly lead to future understatement of liabilities in the same manner as described in Section 1.1.

In order to place responsibility for setting assumptions back onto the individual, the mortality projection working party decided that rather than publish a table of standard reduction factors, they would release a software package to enable actuaries to produce their own mortality projection bases. It was also decided that the ability to quantify the uncertainty surrounding projections would be both informative and useful. To this end, we produced a Microsoft Excel based software package which used R as a calculation engine for the P-spline and Lee-Carter models. The P-spline model was an R implementation of the two-dimensional model described in Section 3.3.2, with the array methods described in Chapter 4; the Lee-Carter model was implemented with the improvements suggested by Brouhns et al. (2002) using the Poisson likelihood. Both methods were then made available through an Excel interface by using the R-(D)COM interface by Baier and Neuwirth (2007). Various control parameters were made available to the user (such as the degree of the splines and the order of the penalty for the P-spline model). Figures 6.2 and 6.3 show the input screens of the graphical user interface for the P-spline model in the CMI mortality projection software. An example of the output of the software is shown in Fig. 6.4.

The methods used to give an idea of the uncertainty were slightly different for each method. For the P-spline method we are able to get an idea of the parameter uncertainty by using the estimate of the covariance matrix given in (2.76). Based on these variance estimates scenarios could be generated simply by simulating from a standard normal variable, multiplying by the estimated standard error, $\hat{\boldsymbol{\Sigma}}$ (computed with the array formula (4.25)), and adding this to corresponding value of smooth surface, $\log \hat{\boldsymbol{T}}$. The ith scenario had the form

$$\log \boldsymbol{T}_i = \log \hat{\boldsymbol{T}} + z_i \hat{\boldsymbol{\Sigma}} \tag{6.16}$$

where z_i is a simulated realization from a standard normal distribution. Of course, this

method does not take into account any of the model uncertainty about the selection of the smoothing parameter for the model that produced $\hat{\boldsymbol{T}}$.

Due to the bi-linear structure of the Lee-Carter model getting an estimate of the parameter uncertainty explicitly isn't possible, a fact which is rarely acknowledged in the literature, and the only concession to any uncertainty is in the estimation of the ARIMA parameters. In order to get some idea of the parameter uncertainty, a non-parametric bootstrap methodology was used to simulate "new" datasets from the deviance residuals of the P-spline model. The Lee-Carter model was then fitted to these datasets which could be considered as scenarios from the model. These scenarios can be output into separate files which can then be used by the user for various actuarial purposes.

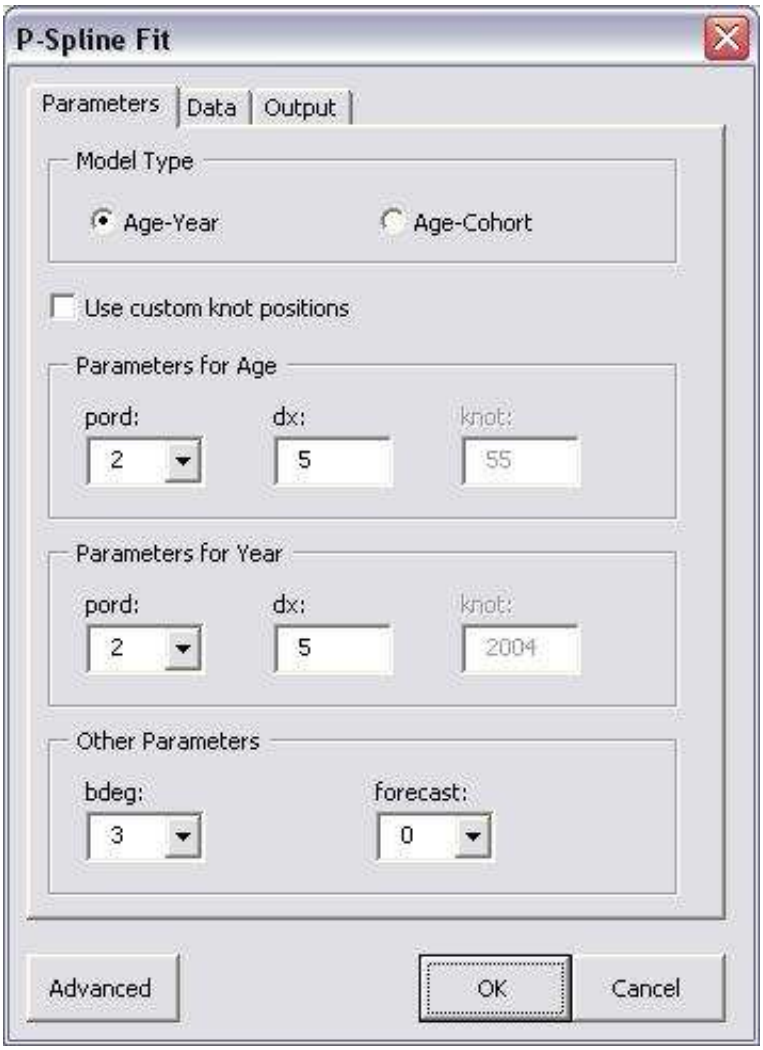


Figure 6.2: The parameter entry interface for the CMI software.

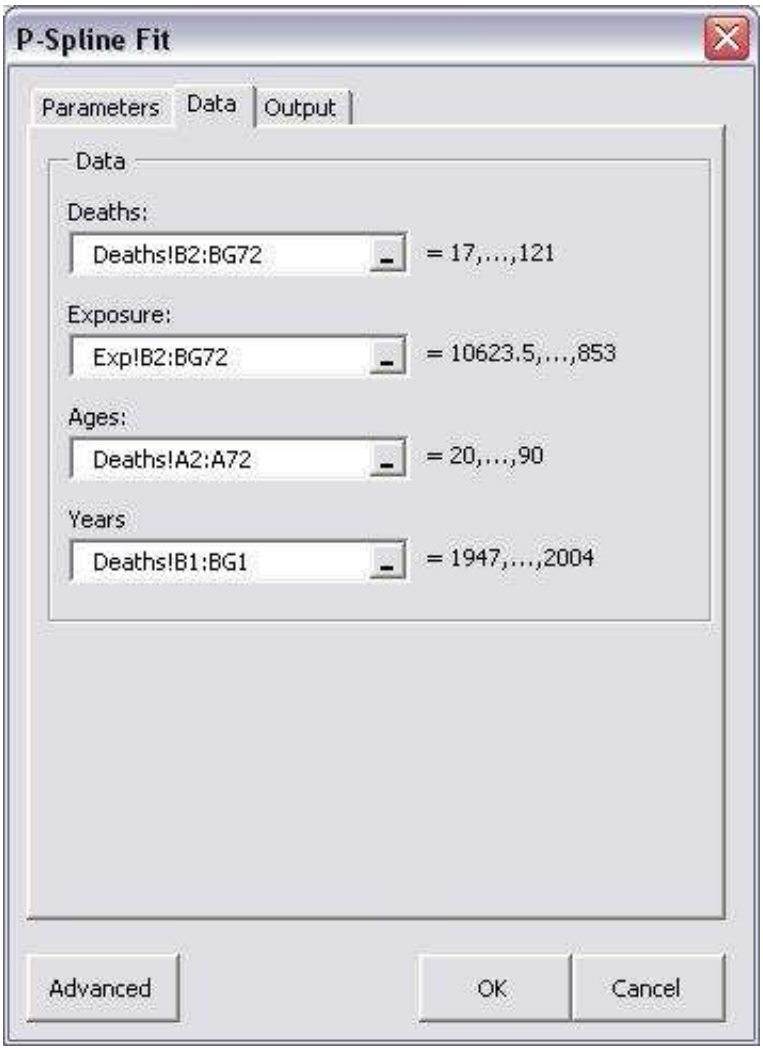


Figure 6.3: The data selection interface for the CMI software.

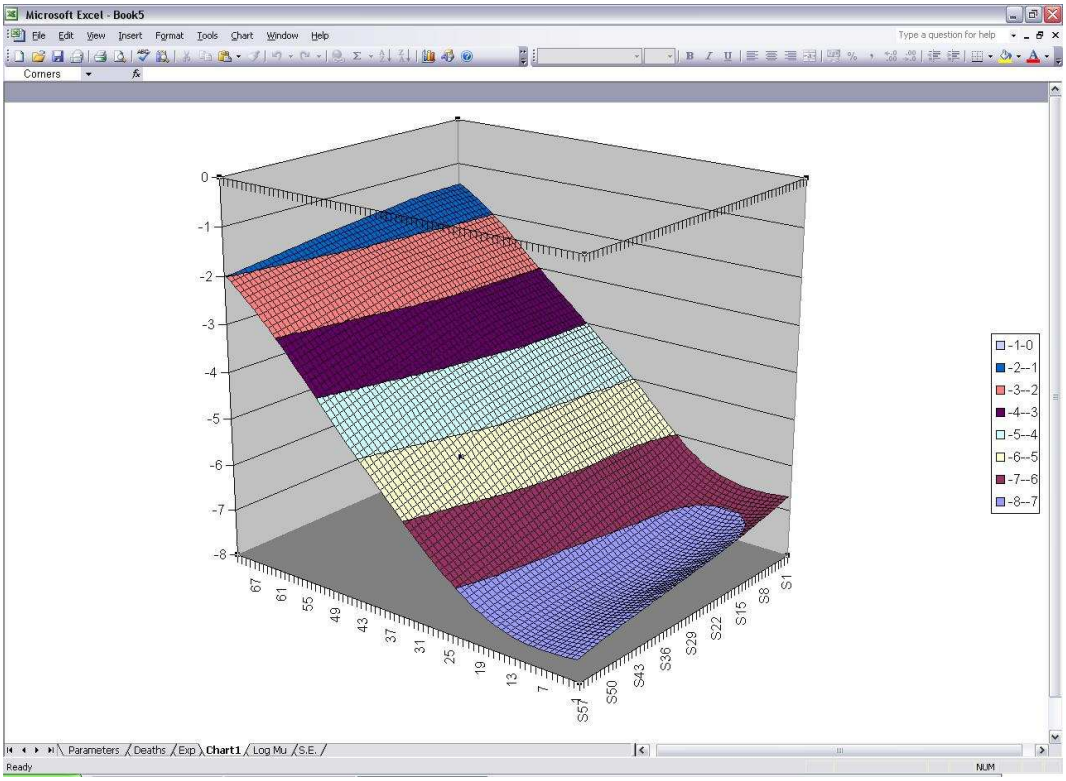


Figure 6.4: An example of the graphical output available from the CMI mortality smoothing software (the log mortality surface).

Chapter 7

Conclusions

In this chapter we will summarize the results of this thesis, and suggest areas of further work.

7.1 Summary

The aim of this thesis was to investigate multi-dimensional smoothing methods that could be applied to mortality data. The process of conducting the research led us to focus on a particular methodology, the P-splines described in Section 2.3; we focused on the application of the array methods described in Chapter 4. Despite the focus on these particular areas, there are some broad comments that we would like to make before we highlight the main technical conclusions of the thesis.

The first point is a general comment arising from Chapter 2: for one-dimensional data the selected smoothing parameter or bandwidth has a greater impact on the result than the method used to perform the smoothing; consequently the choice of model selection criterion is very important. Figure 2.13 showed a comparison of the main methods considered in Chapter 2, and there is little to choose between the methods graphically. In contrast, Fig. 2.11(a) showed that different model selection criteria, even when using the same smoothing method (in this example P-splines), will select noticeably different levels of smoothing. The importance of the smoothing parameter and model selection are highlighted further in Section 2.8.1, when we attempt forecasting using P-splines, Fig. 2.12 clearly illustrates the sensitivity of both the central

forecast and the confidence intervals to the choice of smoothing parameter.

In terms of results there seems to be little to choose between the various one-dimensional methods. There are some conceptual advantages to some of the methods. For the P-spline method, the link between the coefficients and the corresponding B-spline basis functions provides an easy way to tailor the penalty for special situations; the ability to select different orders of the penalty also makes P-spline method more general than the penalized TPFs. The argument supporting smoothing splines is very compelling; if one considers the measure of smoothness in (2.17) appropriate, then the optimal properties of the natural cubic spline clearly make it the best smoother in one dimension.

A trend in smoothing literature has been to show the connection between penalized regression and mixed models, as shown in the case of P-splines in Section 2.5.3. Expressing the models in this framework enables us to take advantage of the existing software developed for mixed models. However, it is difficult to find justification for making the distinction between the "random" and fixed parts, when in most cases the fixed part is arbitrarily chosen as a low order polynomial. Without a justification for the split, mixed model splines can only be considered a convenient computational method (with an embedded model selection criteria) and do not deserve a special position among smoothing techniques, especially given the importance of model selection on the final result. Further, if we are modelling a generalized response, as with the mortality data where we are assuming a Poisson response, then moving to a mixed model no longer offers a computational advantage because the solutions of generalized linear mixed model are approximate anyway.

In Chapter 3 we moved on to the main subject of this thesis: multi-dimensional smoothing. We discussed various full rank methods for smoothing multi-dimensional data, and thin plate spline and Kriging models were fitted to the assured lives mortality data. However, we were not able to fit these methods using a Poisson response, as this does not appear available in the `R` libraries we considered. In Section 2.3 we showed how the penalized spline models could be fitted simply by adding a penalty term to the likelihood, and using the appropriate functions in the basis. Penalized splines in multiple dimensions can be constructed using the same formula. For ex-

ample, the P-spline model in two dimensions can be fitted using the same penalized scoring algorithm from (2.75); all we need do is specify the two-dimensional B-spline basis shown in Fig. 3.5 and the appropriate penalty, for example (3.28) or (3.36). P-splines also have the advantage of being a low rank method; the other models discussed in Chapter 3 contain large numbers of parameters, and hence suffer a large computational burden. Wood (2006) uses approximations to thin plate splines which make the computations more manageable. Using P-splines we have a low number of parameters relative to data points which enables us to deal with the large data sets such as the mortality data used to illustrate Chapter 3.

In Chapter 4 we reviewed and developed the array methods of Currie et al. (2006). Our first contribution on this topic was to develop our understanding of how the algorithm (4.23) works; importantly, we noted that using the row-tensors, $\boldsymbol{G}_i$, enabled us to pre and post-multiply by the matrices, $\boldsymbol{B}_i$, simultaneously. We then showed that we can efficiently multiply a Kronecker product matrix onto any other matrix by re-arranging the elements of the target matrix into a multi-dimensional array using equation (4.29). Finally, we showed how these new methods could be combined with the methods of Currie et al. (2006) to deal with block diagonal matrices.

The multi-dimensional P-spline model produces a pleasing smooth surface when applied to the mortality data described in Section 1.3. However, when it comes to forecasting, the method seems to produce erratic results. The discussion of model uncertainty in Section 2.8.1 highlights that the central forecast, and confidence interval around that forecast, was heavily dependent on the smoothing parameters selected with reference to the data. This may be considered a weakness of the method, but we merely interpret this erratic behaviour as a reflection of the task at hand. The extrapolation of a two-dimensional surface using semi- or non-parametric methods will inevitably produce inconsistent results; if it did not, one would have to question if the method was flexible enough to make it suitable for data-modelling purposes. Models such as the Lee-Carter model, on the other hand, which have a rigid parametric structure are bound to have more consistent extrapolation properties. Lee and Carter (1992) developed their model with the express purpose of forecasting mortality; to this end, it seems to produce sensible results. However, critically analyzing the model we

must raise several concerns. Firstly, there appears to be some difficulty in quantifying the parameter uncertainty of the $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\kappa}$ in the first stage of the fitting procedure. Secondly, the parameter uncertainty surrounding $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is ignored in the second stage of the procedure. Finally, there is no consideration paid to model uncertainty in the projection. These three factors lead to the strange, mean-reverting, confidence intervals that are shown in the much of the Lee-Carter literature; see for example Lee and Carter (1992) and Renshaw and Haberman (2006).

When the mortality projection software described in Section 6.2 was presented to users at two meetings in 2006 there was scepticism from many of the actuaries in the audience that the confidence intervals produced by the P-spline model, such as those shown in Fig. 3.9, were plausible. The mortality projection bases using reduction factors published in CMI (1990) have consistently under-stated the realized improvements. Even the long, medium, and short cohort projections used by many companies in their pension scheme valuation calculations have a long term rate of mortality improvement approaching zero. These projections do seem to encapsulate the views of many actuaries: that mortality rates simply cannot carry on improving forever. One can find justification for this point of view; the ONS publishes the numbers of deaths in the UK split by individual causes, and evidence points to a trend of rapid reductions in the numbers of deaths caused by circulatory diseases, but relatively modest reductions for other causes. This leads many people to believe that once circulatory diseases are eradicated as a major cause of death in the UK, we will find that improvements in the aggregate death rates are more modest. Another, possibly associated, reason why we might expect less rapid improvements (or even increase in the death rate) is due to increases in obesity rates, particularly among younger people, which can lead heart disease and diabetes. On the other hand, we can find justifications for continued improvements at the current rate. A possible reason for the recent reductions in deaths due to circulatory disease could be the focus, from the government down, on improving treatment of this disease; for example research into the use of statins and the subsequent prescription of these drugs on a large scale has been a key reason for the reduction in heart disease. However, if we get to the stage where circulatory diseases are no longer a major cause of death in the

UK, the focus and resources applied to circulatory diseases will be pointed elsewhere, and we may see similar reductions in other diseases. Recently there have been claims that biogerontologists have isolated the major causes of ageing and will be able to extend human life-span indefinitely, these views are summarized by De Grey (2005). Although these views seem a little far-fetched, the theory suggests that the process of ageing in humans can be categorized into a handful of sub-processes and that progress has already been made in preventing and even reversing a number of these.

The task of forecasting mortality rates, as with any extrapolation exercise is quite subjective. Even in simple parametric models, where we are modelling a direct relationship between two variables, extrapolation of the relationship beyond the area of observation is reliant on the same relationship being maintained in the extrapolation region. In semi- and non-parametric models the data are used to determine the structure of the model; this means that we cannot infer much about the relationship beyond the limits of the data that we observe, and any extrapolation has to revert to some limiting property of the model. For the P-spline model, the structure of the penalty will determine the form of the extrapolation, and the amount of smoothing selected with reference to the data determines the width of the confidence intervals. We should not blindly follow any methodology (statistical or otherwise) for forecasting mortality; planning for the adverse financial effects of extended longevity in the population should be done on a prudent basis. Eventually, as markets for mortality derivatives develop we may see a market price for mortality, which should reflect current expectations of future mortality rates as well as a risk margin to account for the uncertainty. Provided the longevity risk associated with these financial instruments can be diluted in the market so that no party bares a disproportionate level of risk, it would allow pension schemes, life assurance companies, and even the government to hedge out their longevity risk exposure. Mortality forecasting can then assume its place as a tool to aid speculation on the value of mortality derivatives.

7.2 Further Work

In Section 4.5 we compared the computational overhead of performing calculations using array methods as opposed to matrix methods. This comparison was rather unsatisfactory because neither test was a fair reflection of the difference between the two methods: simply comparing the number of scalar multiplications does not fairly reflect the additional overhead required for the organization of the multiplications in the array methods; on the other hand, a comparison of the calculation times in `R` does not reflect fairly on the array methods because the calculations could be made more efficiently in a lower level language. In `R` the functionality of the libraries built into the application is much more efficient than code written in the `R` language, and loops coded in `R` are particularly inefficient. The algorithms in (4.14), (4.23), (4.25), and (4.33) are all designed to optimize calculation using the matrix and array libraries available in `R` by rotating the multidimensional array (using `aperm`) and then (in the implementation) flattening the array into a matrix to perform the multiplications, and so avoiding costly custom code with large loops. However, the rotation of the arrays requires a re-ordering of the elements, which requires a large number of read-write transactions, which are relatively costly. If we were to implement the methods in a lower level language where loops were not as costly, we could avoid the rotation by multiplying each element of the Kronecker product directly onto the appropriate dimension of the array. Formally, we would define a function to perform multiplication as follows.

Definition 7.1 *For the $n_t \times c_t$ matrix $\boldsymbol{A}$ and the $c_1 \times \cdots \times c_t \times \cdots \times c_d$ (where $1 \leq t \leq d$) array $\boldsymbol{X}$, the product $m(\boldsymbol{A}, \boldsymbol{X}, t)$ is the $c_1 \times \cdots \times n_t \times \cdots \times c_d$ array whose $i_1 i_2 \ldots i_d^{th}$ element is*

$$\sum_{s=1}^{c_t} a_{i_1,s} x_{i_1,\ldots,i_{t-1},s,i_{t+1},\ldots,i_d}. \tag{7.1}$$

Using this definition we could re-write any of the array algorithms in Chapter 4; for example we could re-write (4.14) as

$$\boldsymbol{AX} = m\Big(\boldsymbol{A}_d, \ldots m\big(\boldsymbol{A}_2, m(\boldsymbol{A}_1, \boldsymbol{X}, 1), 2\big) \ldots, d\Big). \tag{7.2}$$

This algorithm does not require rotation of the array, so provided the class structure of any implementation allowed efficient access to the elements of arrays the additional overhead for calculation of matrix-array products compared to matrix-matrix products should be relatively small.

Smoothing parameter selection within the P-spline model is another area where efficiency could be improved. Currently, we rely on "off-the-shelf" numerical optimization methods to optimize the smoothing parameters with respect to the optimization criterion. In one or two-dimensional applications this is not a big problem, and we find that for grid data using the efficient array algorithms with the built in numerical optimization routines in `R` still out performs other two-dimensional smoothing methods (in many cases the other methods described in Section 3.2.4 simply cannot complete the calculations). However, the method of differentiating the scoring algorithm described by Wood (2007) offers the possibility of further reducing calculation time. Using this method we can find derivatives of the optimization criterion with respect to the smoothing parameters, and armed with this information we are able to produce a targeted optimization algorithm that would reduce the number of combinations of smoothing parameters that need to be fitted in order to find the optimum smoothing. Used in combination with the array methods, this would offer a highly efficient and powerful multi-dimensional smoothing method.

In Chapter 4 we were able to generalize the array methods of Currie et al. (2006); however, we gave no examples of the practical application of these methods. Clearly the methods offer no benefit in a standard penalized GLM, because the weight matrix, $\boldsymbol{W}$, in the scoring algorithm (2.75) is a diagonal matrix which means the inner product is most efficiently calculated using the row-tensors in (4.23). Immediately, we begin to consider what type of model would lead to an estimation algorithm like (2.75) where the weight matrix is a non-diagonal matrix. Using a quasi-likelihood approach, Zeger and Liang (1986) and Zeger et al. (1988) showed how the standard GLM could be extended to allow for correlated data, and to find solutions for these models they developed *Generalized Estimating Equations* (GEE). In the Poisson case, the GEEs are exactly equivalent to the scoring algorithm with a non-diagonal weight matrix. We could use the GEEs as an alternative to the model described in Sec-

tion 5.2 by specifying a correlation structure acting only on data points within the same year. This would lead to a block-diagonal covariance structure which would be most efficiently tackled using the same algorithm used in the example starting on p.93. Currie et al. (2006) focused on using the array methods within statistical models and smoothing methods in particular, but the array methods have a scope beyond statistical modelling and would be useful in any application where a Kronecker product matrix is used.

The additive models described in Chapter 5 could also be improved. Both of the examples are good candidates for mixed models: although there is no particular reason to decompose the underlying surface into fixed and random parts, there is a clear case for treating the additive shock components as random. The models could be fitted as a mixed model simply by decomposing the B-spline basis of the smooth surface into a mixed model basis with a fixed and random part (as described in Section 3.3.3), and then treating the parameters of the additive shock components as random components, with the smoothing parameters selected by REML or ML. However there is no reason why we need to use the same basis in the fitting procedure as in the smoothing parameter selection calculations. Depending on our preference for mixed models, we could fit the model using the standard B-spline basis and the methods described in Sections 5.1 and 5.2 and then simply use the mixed model basis in our calculations of the REML or the ML. Further progress has already been made on the model described in Section 5.2 by Kirkby and Currie (2009 (to appear). They show how using a separate smoothing or shrinkage parameter for each within-year spline can improve the model by allowing the splines to fully explain shocks to the surface without being restricted by the overall level of shrinkage.

There are demographic models that require estimates of rates of change and derivatives of the mortality (fertility or migration) rates; see for example Keyfitz and Caswell (2005). In the UK, since Willets (1999) highlighted the "cohort" effect in UK mortality experience, actuaries have also been focusing on rates of change, particularly mortality improvement rates. For example, the CMI have been illustrating all of its recent publications with heat maps showing the rate of improvements, and mortality improvements seem to be the prime focus for inference. However if our primary in-

terests are the rates of improvement rather than the mortality rates themselves, then we should use a model explicitly for the improvements, rather than estimating the smooth rates of mortality and then calculating the improvements rates from these. Ramsay and Silverman (2005) show how derivatives of an unknown function can be estimated using semi-parametric methods; if these methods could be applied within the efficient array framework it would make applying them to the large mortality datasets viable. A more ambitious under-taking would be to use data on mortality, fertility and migration to flesh out the population models described by Keyfitz and Caswell (2005).

Appendix A

Notation

A.1 Symbols

The following symbols are used in the thesis.

$b_{i,j,\boldsymbol{t}}(x)$	the i^{th} B-spline in a basis of degree j for a set of knots $\boldsymbol{t}$. The j and $\boldsymbol{t}$ subscripts maybe dropped when obvious or unnessary in the context.
$B_{j,\boldsymbol{t}}(x)$	a vector valued function which evaluates each B-spline in a basis at x, ie. $B_{j,\boldsymbol{t}}(x) = [b_{1,j,\boldsymbol{t}}(x), \ldots, b_{k,j,\boldsymbol{t}}(x)]$.
$\boldsymbol{B}$	a matrix of B-splines.
c	the number of columns in a matrix.
$\boldsymbol{D}_d$	a difference matrix of order d.
k	the number of knots in a basis.
n	the number of rows in a matrix.

$p_r(x)$	a polynomial of degree r.
$\boldsymbol{P}_d$	a penalty matrix of order d.
$\boldsymbol{t}$	a set of knots.
$\boldsymbol{X}$	a fixed effects design matrix.
$\boldsymbol{Z}$	a random effects design matrix.
$\boldsymbol{\alpha}$	random effect parameters in a mixed model (also used as the spline coefficients in Truncated Power Function regression model).
$\boldsymbol{\beta}$	fixed effect parameters.
λ	smoothing parameter
$\boldsymbol{\theta}$	parameters in a P-spline regression model.

Appendix B

Linear Algebra

This appendix gives a brief introduction to the use of Kronecker products and other unusal matrix operations used in this thesis.

B.1 Kronecker Product

In this section we will briefly define and state some properties of the Kronecker product. For a more detailed reference see Harville (1997). We denote the Kronecker product of two matrices by:

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix}. \tag{B.1}$$

The Kronecker product is associative and distributive over addition:

$$\boldsymbol{A} \otimes (\boldsymbol{B} \otimes \boldsymbol{C}) = (\boldsymbol{A} \otimes \boldsymbol{B}) \otimes \boldsymbol{C} \tag{B.2}$$

$$\boldsymbol{A} \otimes (\boldsymbol{B} + \boldsymbol{C}) = (\boldsymbol{A} \otimes \boldsymbol{B}) + (\boldsymbol{A} \otimes \boldsymbol{C}). \tag{B.3}$$

We will denote collected operations as:

$$\bigotimes_{i=1}^{n} \boldsymbol{A}_i = \boldsymbol{A}_n \otimes \boldsymbol{A}_{n-1} \otimes \cdots \otimes \boldsymbol{A}_1. \tag{B.4}$$

(NOTE - The order above is non-standard).

The Kronecker product has some useful properties. Provided the matrix multiplications conform:

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = \boldsymbol{AC} \otimes \boldsymbol{BD}. \tag{B.5}$$

The matrix $\boldsymbol{A} \otimes \boldsymbol{B}$ is non-singular if and only if $\boldsymbol{A}$ and $\boldsymbol{B}$ are non-singular, with:

$$(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}. \tag{B.6}$$

If $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ are eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{B}$ with corresponding eigenvalues λ_A and λ_B, it follows from B.5 that:

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{x}_A \otimes \boldsymbol{x}_B) = \lambda_A \lambda_B (\boldsymbol{x}_A \otimes \boldsymbol{x}_B) \tag{B.7}$$

and thus $\boldsymbol{x}_A \otimes \boldsymbol{x}_B$ is an eigenvector of $\boldsymbol{A} \otimes \boldsymbol{B}$ with corresponding eigenvalue $\lambda_A \lambda_B$. Following on from this if we find the singular value decompositions

$$\boldsymbol{A} = \boldsymbol{P}_A \boldsymbol{D}_A \boldsymbol{Q}'_A \qquad \text{and} \qquad \boldsymbol{B} = \boldsymbol{P}_B \boldsymbol{D}_B \boldsymbol{Q}'_B \tag{B.8}$$

then $\boldsymbol{A} \otimes \boldsymbol{B}$ may be decomposed as:

$$\boldsymbol{A} \otimes \boldsymbol{B} = (\boldsymbol{P}_A \otimes \boldsymbol{P}_B)(\boldsymbol{D}_A \otimes \boldsymbol{D}_B)(\boldsymbol{Q}'_A \times \boldsymbol{Q}'_B), \tag{B.9}$$

where $\boldsymbol{P}_A \otimes \boldsymbol{P}_B$ is a matrix of eigenvectors with corresponding eigenvalues $\text{diag}(\boldsymbol{D}_A \otimes \boldsymbol{D}_B)$.

B.2 Row Tensor

We will also define the Row Tensor operation which multiplies every column of one matrix by every column of another as

$$\boldsymbol{E} \square \boldsymbol{F} = (\boldsymbol{E} \otimes \boldsymbol{1}') * (\boldsymbol{1}' \otimes \boldsymbol{F}). \tag{B.10}$$

Appendix C

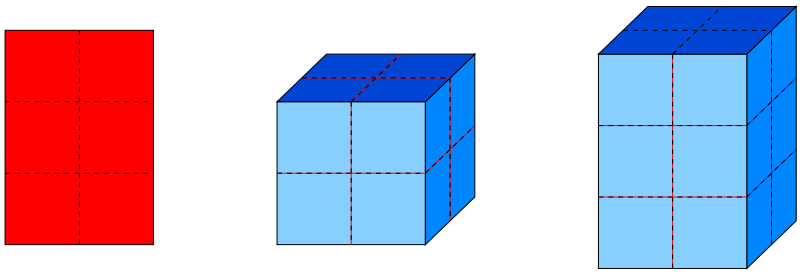
Array Methods

C.1 Graphical Representation

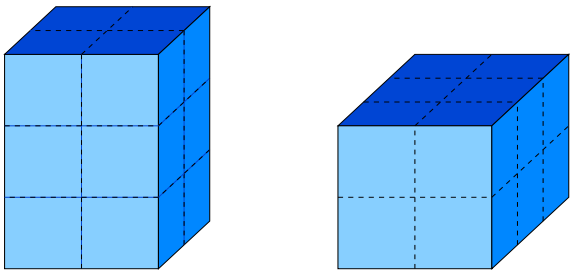
Here we will show a step-by-step graphical interpretation of the multiplication of a Kronecker product onto an array. In this case, the Kronecker product will be:

$$(\mathbf{A_3} \otimes \mathbf{A_2} \otimes \mathbf{A_1}) \tag{C.1}$$

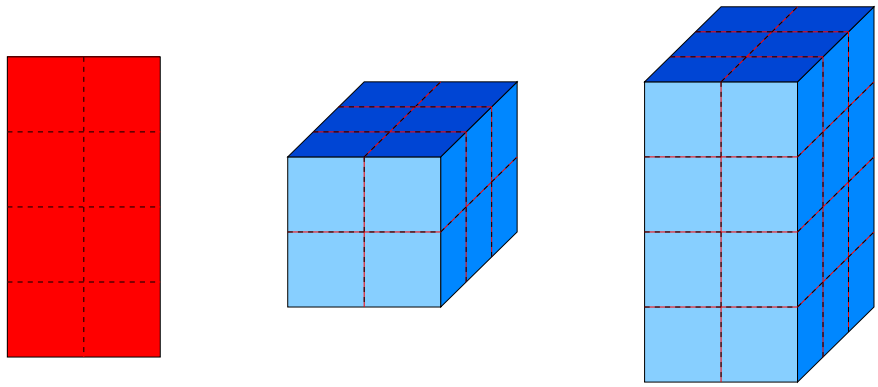
where $\mathbf{A_1}$, $\mathbf{A_2}$ and $\mathbf{A_3}$ are 3×2, 4×2 and 5×2 matrices respectively. $\mathbf{B}$ is therefore a $2 \times 2 \times 2$ array. We start by multiplying $\mathbf{A_1}$ onto $\mathbf{B}$:



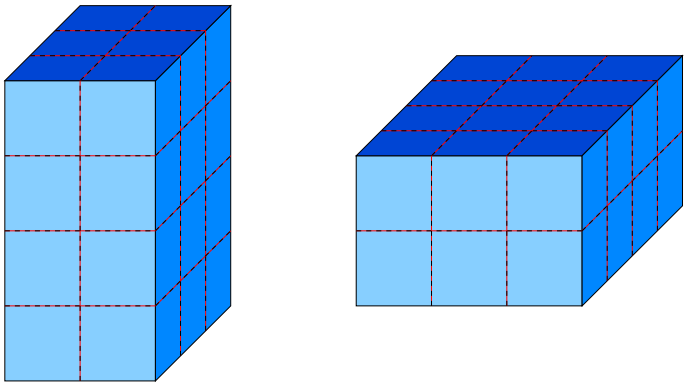
The result is a $3 \times 2 \times 2$. We then transpose the array so the next matrix can be multiplied onto the corresponding dimension of $\mathbf{B}$.



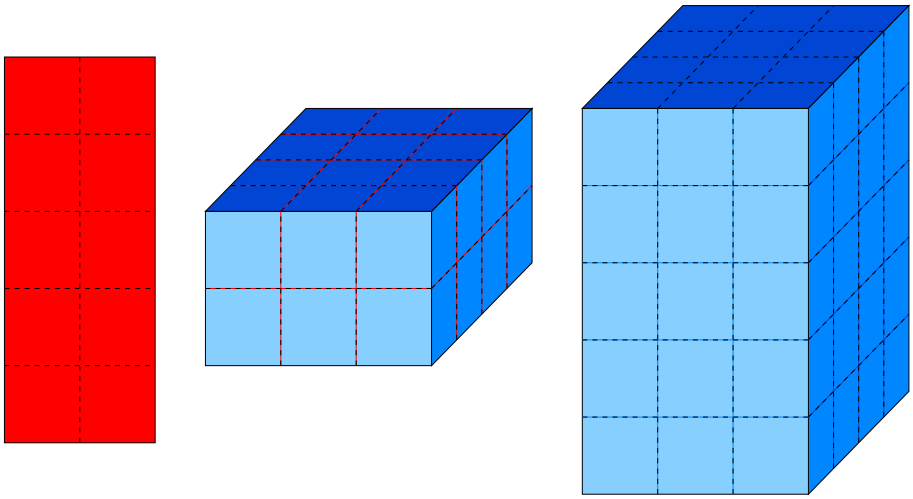
We now have a $2 \times 2 \times 3$ array, and can perform the next multiplication. We multiply $\mathbf{A_2}$ onto the array:



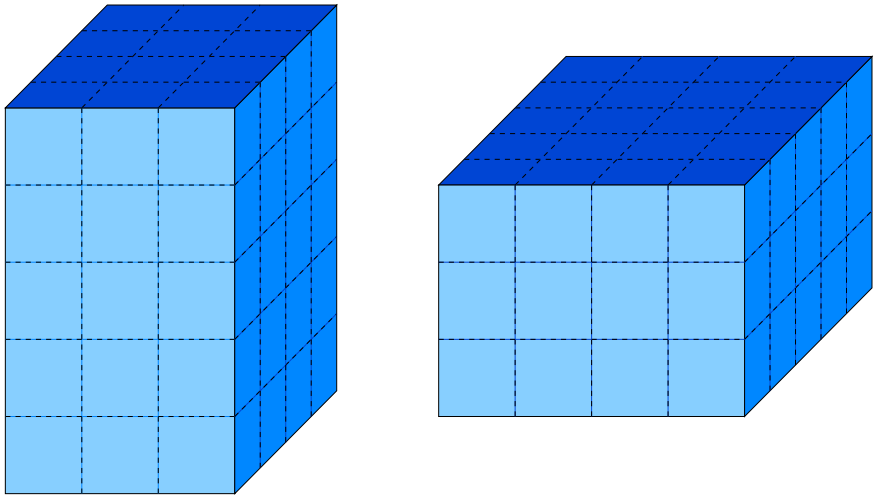
We now have to transpose again to line up $\mathbf{A_3}$ with its dimension.



Perform the final multiplication:



And finally transpose again to get the dimensions back in their original order.



Note if we had tried to represent the full matrix **A** graphically, it would not have fitted on the page!

Bibliography

H. Akaike. A new look at the statistical identification model. *IEEE Transactions on Automatic Control*, 19:716–723, 1971.

T. Baier and E. Neuwirth. Excel::COM:*R*. *Computational Statistics*, 22:91–108, 2007.

H. Booth, R. J. Hyndman, L. Tickle, and P. de Jong. Lee-Carter mortality forecasting: a multi-country comparison of variants and extensions. *Demographic Research*, 15: 289–310, 2006.

A. W. Bowman and A. Azzalini. *Applied smoothing techniques for data analysis: the kernel approach with S-plus illustrations*. Oxford University Press, 1997.

N. E. Breslow and D. G. Clayton. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88:9–24, 1993.

J. W. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, 25:772–781, 1978.

N. Brouhns, M. Denuit, and J. D. Vermunt. A Poisson log-bilinear regression approach to the construction of projected life tables. *Insurance: Mathematics and Economics*, 31:373–393, 2002.

C. Chatfield. Model uncertainty, data mining, and statistical inference. *Journal of the Royal Statistical Society, Series A*, 158:419–466, 1995.

CMI. Standard tables of mortality based on the 1979-82 experiences. Technical Report 10, Continuous Mortality Investigation, 1990.

CMI. Standard tables of mortality based on the 1991-94 experiences. Technical Report 17, Continuous Mortality Investigation, 1999.

P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 31:377–403, 1979.

I. Currie and M. Durban. Flexible smoothing with P-splines: a unified approach. *Statistical Modelling*, 2:333–349, 2002.

I. D. Currie. Smoothing overparameterized regression models. In *Proceedings of the 23st International Workshop on Statistical Modelling*, pages 194–199, 2008.

I. D. Currie, M. Durban, and P. H. C. Eilers. Smoothing and forecasting mortality rates. *Statistical Modelling*, 4:279–298, 2004.

I. D. Currie, M. Durban, and P. H. C. Eilers. Generalized linear array models with applications to multidimensional smoothing. *Journal of the Royal Statistical Society, Series B*, 68:259–280, 2006.

C. de Boor. Efficient computer manipulation of tensor products. *ACM Transactions on Mathematical Software*, 5:173–182, 1979.

C. de Boor. *A practical guide to splines*. Springer, 2nd edition, 2001.

A. D. N. J. De Grey. The foreseeability of real anti aging medicine. *Anti-Aging Medical Therapeutics*, 7:59–68, 2005.

A. Delwarde, M. Denuit, and P. Eilers. Smoothing the Lee-Carter and Poisson log-bilinear models for mortality forecasting. *Statistical Modelling*, 7:29–48, 2007.

A. Dobson. *An introduction to generalized linear models*. London: Chapman and Hall, 2nd edition, 2002.

S. Efromovich. *Nonparametric curve estimation*. Springer, 1994.

P. H. C. Eilers and B. D. Marx. *Splines, knots, and penalties*, 2004. URL `http://www.stat.lsu.edu/faculty/marx/splines_knots_penalties.pdf`. Unpublished manuscript downloaded November 2007.

P. H. C. Eilers and B. D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11:89–121, 1996.

P. H. C. Eilers, I. D. Currie, and M. Durban. Fast and compact smoothing on large multidimensional grids. *Computational Statistics and Data Analysis*, 50:61–76, 2006.

D. O. Forfar, J. J. McCutcheon, and A. D. Wilkie. On graduation by mathematical formula. *Journal of the Institute of Actuaries*, 115:1–149, 1988.

I. G. Good. The interaction algorithm and practical Fourier analysis. *Journal of the Royal Statistical Society, Series B*, 20:361–372, 1958.

J. Gower. The Yates algorithm. *Utilitas Mathematica*, 21:99–115, 1982.

P. Green. Linear models for field trials, smoothing and cross-validation. *Biometrika*, 72:527–537, 1985.

P. J. Green and B. W. Silverman. *Nonparametric regression and generalized linear models*. London: Chapman and Hall, 1994.

R. A. Harshman. An index formalism that generalizes the capabilities of matrix notation and algebra to n-way arrays. *Journal of Chemometrics*, 15:689–714, 2001.

D. A. Harville. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*, 72:320–338, 1977.

D. A. Harville. *Matrix algebra from a statistician's perspective*. Springer, 1997.

T. J. Hastie and R. J. Tibshirani. *Generalized additive models*. London: Chapman and Hall, 1990.

N. L. Hjort and G. Claeskens. Frequentist model average estimators. *Journal of the American Statistical Association*, 98:879–899, 2003.

Human Mortality Database. *Swedish Mortality Data*. Univerisity of California, Berkley (USA), and Max Planck Institute for Demographic Research (Germany). URL `http://www.mortality.org`. Downloaded 9th December 2006.

C. Hurvich and C. Tsai. Regression and time series model selection in small samples. *Biometrika*, 76:297–307, 1989.

G. Kauermann. A note on smoothing parameter selection for penalized spline smoothing. *Journal of statistical planning and inference*, 127:53–69, 2005.

N. Keyfitz and H. Caswell. *Applied mathematical demography*. Springer, 3rd edition, 2005.

J. G. Kirkby and I. D. Currie. Modelling mortality on the Lexis diagram. In *Proceedings of the 21st International Workshop on Statistical Modelling*, pages 279–285, 2006.

J. G. Kirkby and I. D. Currie. Smooth models of mortality with period shocks. In *Proceedings of the 22st International Workshop on Statistical Modelling*, pages 374–379, 2007.

J. G. Kirkby and I. D. Currie. Smooth models of mortality with period shocks. *Statistical Modelling*, 2009 (to appear).

P. Lambert and P. Eilers. Bayesian multi-dimensional density estimation with *P*-splines. In *Proceedings of the 21st International Workshop on Statistical Modelling*, pages 313–320, 2006.

R. D. Lee and L. R. Carter. Modeling and forecasting U.S. mortality. *Journal of the American Statistical Association*, 87:537–549, 1992.

P. McCullagh and J. A. Nelder. *Generalized linear models*. London: Chapman and Hall, 1990.

J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society*, 135:370–384, 1972.

D. Nychka. *fields: Tools for spatial data*, 2007. URL `http://www.image.ucar.edu/GSP/Software/Fields`. R package version 3.5.

Y. Pawitan. *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press, 2001.

A. Perperoglou and P. H. C. Eilers. Overdispersion modelling with individual random effects and penalized likelihood. 2006 (personal communication).

J. C. Pinheiro and D. M. Bates. *Mixed-effects models in S and S-plus*. Springer, 2000.

R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, 2nd edition, 2005.

A. Renshaw and S. Haberman. On the forecasting of mortality reduction factors. *Insurance: Mathematics and Economics*, 32:379–401, 2003a.

A. Renshaw and S. Haberman. Lee-Carter mortality forecasting: a parallel generalized linear modeling approach for England and Wales mortality projections. *Applied Statistics*, 52:119–137, 2003b.

A. Renshaw and S. Haberman. A cohort-based extension to the Lee-Carter model for mortality reduction factors. *Insurance: Mathematics and Economics*, 38:556–570, 2006.

S. Richards and G. Jones. Financial aspects of longevity risk. Presented to the Staple Inn Actuarial Society, 2004.

S. J. Richards. Detecting year-of-birth mortality patterns with limited data. *Journal of the Royal Statistical Society, Series A*, 171:1–20, 2008.

S. J. Richards, J. G. Kirkby, and I. D. Currie. The importance of year of birth in two-dimensional mortality data. *British Actuarial Journal*, 12:5–61, 2006.

G. K. Robinson. That BLUP is a good thing: the estimation of random effects. *Statistical Science*, 6:15–51, 1991.

D. Ruppert, M. P. Wand, and R. J. Carroll. *Semiparametric regression*. Cambridge University Press, 2003.

G. Schwarz. Estimating the dimension of a model. *Annals of statistics*, 6:461–464, 1978.

S. R. Searle, G. Casella, and C. E. McCulloch. *Variance components*. Wiley, 1992.

B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric curve fitting. *Journal of the Royal Statistical Society, Series B*, 47:1–52, 1985.

B. W. Silverman. *Density estimation for statistics and data analysis*. London: Chapman and Hall, 1986.

Statistiches Bundesamt. *German Lexis Mortality Data*, 2006. Downloaded from http://www.mortality.org, December 2006.

S. Tuljapurkar, N. Li, and C. Boe. A universal pattern of mortality decline for G7 countries. *Nature*, 405:789–792, 2000.

H. Turner and D. Firth. *gnm: Generalized Nonlinear Models*, 2008. URL `http://go.warwick.ac.uk/gnm`. R package version 0.9.

W. N. Venables and B. D. Ripley. *Modern applied statistics with S*. Springer, 4th edition, 2002.

M. Wand and M. Jones. *Kernel Smoothing*. London: Chapman and Hall, 1995.

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

M. P. Wand. Vector differential calculus in statistics. *American Statistician*, 56:55–62, 2002a.

M. P. Wand. Smoothing and mixed models. *American Statistician*, 56:55–62, 2002b.

S. Welham, B. Cullis, M. Kenwood, and R. Thompson. A comparison of mixed model splines. *Australian and New Zealand Journal of Statistics*, 49:1–23, 2007.

S. J. Welham. *Smoothing spline methods within the mixed model framework*. PhD thesis, University of London, 2005.

R. C. Willets. Mortality in the next millennium. Paper presented to the Staple Inn Actuarial Society, 1999.

S. Wood. *Generalized additive models: an introduction with R.* London: Chapman and Hall, 2006.

S. N. Wood. Tensor product smooth interaction terms in generalized additive mixed models. 2004 (unpublished).

S. N. Wood. Fast stable direct fitting and smoothness selection for generalized additive models. *Journal of the Royal Statistical Society, Series B*, 70:495–518, 2007.

F. Yates. The design and analysis of factorial experiments. Technical Report 35, Commonwealth Bureau of Soils, Harpenden, 1937.

S. L. Zeger and K.-Y. Liang. Longitudinal data analysis for discrete and continuous outcomes. *Biometrics*, 42:121–130, 1986.

S. L. Zeger, K.-Y. Liang, and P. S. Albert. Models for longitudinal data: a generalized estimating equation approach. *Biometrics*, 44:1049–1060, 1988.