Smoothing with Mixed Model Software

BY LONG NGO AND M.P. WAND

Department of Biostatistics, School of Public Health, Harvard University, 665 Huntington Avenue, Boston, Massachusetts 02115, U.S.A.

January 08, 2004

ABSTRACT

Smoothing methods that use basis functions with penalization can be formulated as fits in a mixed model framework. One of the major benefits is that software for mixed model analysis can be used for smoothing. We illustrate this for several smoothing models such as additive and varying coefficient models for both `S-PLUS` and `SAS` software. Code for each of the illustrations is available on the `Internet`.

Keywords: Additive mixed models; Additive models; Bivariate smoothing; Generalized additive models; Kriging; Scatterplot smoothing; Semiparametric mixed models; Semiparametric regression; Variance components; Varying coefficient models.

1 Introduction

Smoothing methodology offers a means by which non-linear relationships can be handled without the restrictions of parametric models. It has become a widely used tool for data analysis and inference and its integration into complex models and use in applications is becoming more and more pervasive.

When fitting models that involve smoothing the analyst has to choose between programming the method herself or using customized software. The latter can be somewhat restrictive. For example, generalized additive models can be handled in either `PROC GAM` in `SAS` or `gam()` in `S-PLUS`; but varying coefficient models cannot. On the other hand, self-implementation of smoothing models can be time consuming. In this article we demonstrate how mixed model representations of penalized splines can largely alleviate this problem. Most smoothing models in common use: nonparametric regression, kriging, additive models, varying coefficient models, additive mixed models; can be formulated as a mixed model. See, for example, Wahba (1978), Speed (1991), Verbyla (1994), O'Connell and Wolfinger (1997), Brumback, Ruppert and Wand (1999). This allows for their fitting to be achieved using software such as `PROC MIXED` in `SAS` (Littell *et al.*, 1996) and `lme()` in `S-PLUS` (Pinheiro and Bates, 2000). Mixed model software also provides automatic smoothing parameter choice via (restricted) maximum likelihood estimation of variance components. Finally we note that mixed model representations of smoothers allow for straightforward combination of smoothing with other modelling tools such as random effects for longitudinal data. Ruppert, Wand and Carroll (2003) provides more background and materials for the class of semiparametric regression models. Wand (2003) is a companion article to this paper and provides more details on the connections between smoothing and mixed models.

We provide `S-PLUS` and `SAS` code that illustrates the use of mixed model software to do smoothing for several models. Sections 2 — 8 treat increasing more sophisticated models, starting with the simple scatterplot smoothing, or nonparametric regression, model and finishing with varying coefficient models. Section 9 treats user specified amounts of smoothing, while Section 10 deals with standard error computation. Extensions to other basis functions and bivariate smoothing is treated in Sections 11 and 12. We close with discussion on generalized models in Section 13, plotting issues in Section 14 and some closing remarks in Section 15.

All of the code given in this article is available in text files on the Internet.

2 Scatterplot Smoothing

The formulation of penalized spline scatterplot smoothers as mixed model fits is fundamental to the thrust of this paper. Therefore we will spend a few paragraphs explaining this connection.

The data in each panel of Figure 1 is identical, and was generated as

$$y_i = f(x_i) + 0.4\varepsilon_i$$

where the x_i and ε_i are random samples from the uniform distribution on $(0,1)$ and the standard normal distribution respectively. The mean function f is $f(x) = \sin(3\pi x)$.

In each panel, linear models of the form

$$y_i = \beta_0 + \beta_1 x_i + \sum_{k=1}^{K} u_k (x_i - \kappa_k)_+ + \varepsilon_i \qquad (1)$$

have been fitted to the data. The function

$$(x - \kappa_k)_+ = \begin{cases} 0 & x \leqslant \kappa_k \\ x - \kappa_k & x > \kappa_k \end{cases}$$

represents a piecewise line with a join-point, or *knot*, at κ_k. The choice of the κ_k's is discussed in Section 3.

Here and throughout most of this paper we use the truncated line basis

$$1,\ x,\ (x-\kappa_1)_+, \ldots, (x-\kappa_K)_+$$

for smoothing. *This is for simplicity of exposition*. Other smoother bases can be used instead and these are discussed in Section 11. However, the truncated line basis can perform adequately in many circumstances.

The bar at the base of each panel shows the location of the knots. Panel (a) is just an ordinary least squares fit to the scatterplot; but is quite rough due to the large number of truncated line functions being fit. Panel (b) remedies this through one simple modification:

$$u_k \overset{\text{ind.}}{\sim} N(0, \sigma_u^2). \tag{2}$$

For $\sigma_u^2 < \infty$ this shrinks the u_k and leads to the smooth fit shown in Figure 1 (b).

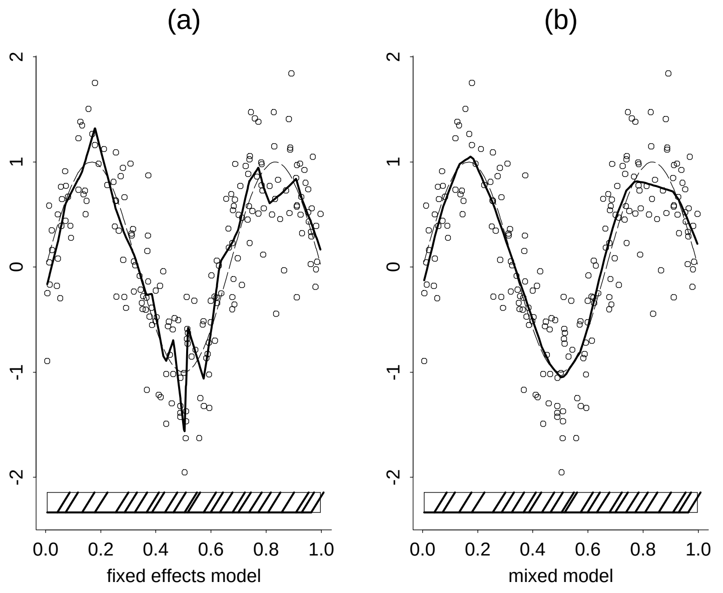


Figure 1: How mixed models do smoothing. In (a) all coefficients are fixed effects, while in (b) the coefficients of the knots are random effects. The solid curve is the estimated curve, while the dashed curve is the function from which the data were generated.

If we define the design matrices

$$\mathbf{X} = [1 \; x_i]_{1 \leqslant i \leqslant n}, \qquad \mathbf{Z} = [(x_i - \kappa_k)_+]_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant k \leqslant K}}$$

and set $\boldsymbol{\beta} = [\beta_0, \beta_1]^{\mathsf{T}}$, $\mathbf{u} = [u_1, \ldots, u_K]^{\mathsf{T}}$ then we can rewrite (1) and (2) as the linear mixed model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}, \qquad \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\varepsilon} \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \sigma_u^2 \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \sigma_\varepsilon^2 \mathbf{I} \end{bmatrix} \right). \tag{3}$$

Scatterplot smoothers of the type, where the number of basis functions is less than the sample size, presented in this section go back at least to Parker and Rice (1985), O'Sullivan (1986,1988), Gray (1992) and Kelly and Rice (1990). More recent references are Eilers and Marx (1996), Hastie (1996) and Ruppert and Carroll (2000) where the following names:

- P-splines,
- penalised splines,
- pseudosplines, and
- low-rank smoothers

have been coined. Each of these are virtually synonymous.

The next two subsections explain how (3) can be fit in the S-PLUS and SAS computing environments.

2.1 S-PLUS commands

For illustration of scatterplot smoothing we will use the fossil data described by Chaudhuri and Marron (1999). However, we will multiply the response variable (strontium ratio) by 100,000 to make the y-axis more readable.

Assign the scatterplot vectors x and y corresponding to the fossil data-frame:

```
x <- fossil$age
y <- 100000*fossil$strontium.ratio
```

The Z matrix requires a set of knots. For now we will take them to be

```
knots <- seq(94,121,length=25)
```

Section 3 describes good default choice of the knots for general x. However, it is important to realize that this default is not always appropriate and that selection of a good set of knots may need to be done manually.
Read in fossil data and assign to vectors x and y.

```
fossil <- read.table("fossil.dat",header=T)
x <- fossil$age
y <- fossil$strontium.ratio
```

Set up using the design matrices.

```
n <- length(x)
X <- cbind(rep(1,n),x)
Z <- outer(x,knots,"-")
Z <- Z*(Z>0)
```

Compute the mixed model fit using lme().

```
fit <- lme(y~-1+X,random=pdIdent(~-1+Z))
```

The estimated fixed and random coefficients and fitted values are:

```
beta.hat <- fit$coef$fixed
u.hat <- unlist(fit$coef$random)
f.hat <- X%*%beta.hat + Z%*%u.hat
```

The estimated standard deviation components are:

```
sig.eps.hat <- fit$sigma
sig.u.hat   <- sig.eps.hat*exp(unlist(fit$modelStruct))
```

Figure 2 shows the scatterplot using this code. Smoother fits can be obtained using the smoother basis functions as described in Section 11.

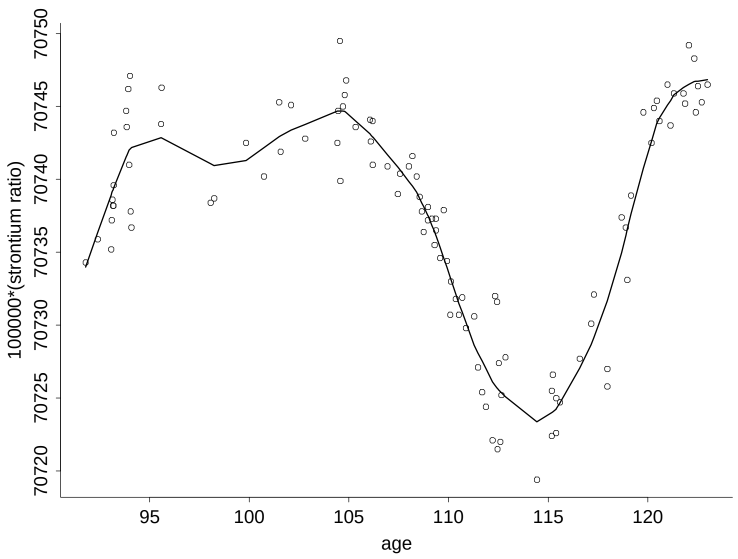


Figure 2: Linear penalized spline fit to the fossil data using the commands of Section 2.1.

2.2 SAS code

The following SAS code fits a linear penalized spline fit for given vectors of x_i and y_i values, along with a set of knots. Note that in order to use the enclosed SAS code, it is necessary to create the subdirectory and the referenced library name. In this case, a library name `paper` pointing to subdirectory `~ /test` has been created.

```
libname paper '~/test';
data paper.fossil;
  infile '~/test/fossil.dat' missover;
  input age ratio;
  ratio=ratio*100000;
  if age ne .;
run;

/********************************/
/*generate knots vector        */
/********************************/
```

```
data paper.knots;
  do i=0 to 24;
     knots=94+((121-94)/24)*i;
     output;
  end;
run;
data dataw;
  set paper.fossil;
  m=1;
run;
data kt1;
  set paper.knots nobs=nk;
  call symput(’nkt’,nk);
run;
proc transpose data=paper.knots prefix=knots out=knotst;
  var knots;
run;
data paper.knotst;
  set knotst;
  m=1;
run;


/********************************/
/* creating the Z matrix        */
/********************************/

data dataw;
  merge dataw paper.knotst;
  by m;
  %let nk=&nkt;
  array Z (&nk) Z1-Z&nk;
  array knots (&nk) knots1-knots&nk;
  do k=1 to &nk;
     Z(k)=age-knots(k);
     if Z(k) < 0 then Z(k)=0;
  end;
  drop knots1-knots&nk _name_;
run;
ods output CovParms=paper.varcomp;

/********************************/
/* fitting the mixed model      */
/********************************/
```

```
proc mixed;
  model ratio = age / solution outp=paper.yhat;
  random  Z1-Z&nk / type=toep(1) s;
run;

/********************************/
/* plotting the smoothed curve  */
/********************************/
proc sort;
  by age;
run;
symbol1 v=circle c=black i=j l=1;
symbol2 v=point  c=blue  i=j l=2;
goptions device=xcolor;
proc gplot;
  plot ratio*age pred*age / overlay;
run;
```

3 Default Knot Specification

A reasonable default rule for the knot locations is:

$$\kappa_k = \{(k+1)/(K+2)\}\text{th sample quantile of the unique } x_i\text{'s} \tag{4}$$

for $k = 1, \ldots, K$.

A simple default choice of K that usually works well is

$$K = \max\left(5, \min\left(\tfrac{1}{4} \times \text{number of unique } x_i\text{'s}, 35\right)\right). \tag{5}$$

See Ruppert (2002) for further discussion on default knot specification.

3.1 S-PLUS commands

The default choice of knots corresponding to (4) and (5) can be generated using the following S-PLUS function:

```
default.knots <- function(x,num.knots)
{
   if (missing(num.knots))
      num.knots <- max(5,min(floor(length(unique(x))/4),35))

   return(quantile(unique(x),seq(0,1,length=
                  (num.knots+2))[-c(1,(num.knots+2))]))
}
```

3.2 SAS code

The following SAS code obtains the default set of knots for given vector of x_i values. This algorithm does not produce identical knots that are generated by the Splus algorithm; however, as long as the underlying knots capture the variable's distribution, the smoothing results are quite similar. The algorithm selects a knot at every fifth value, and limits the number of knots generated. The option of specifying the number of knots to be selected is also allowed.

```
%macro default_knots(librefknots=,data=,knotdata=,varknots=,numknots=);
proc sort data=&data (keep=&varknots) out=q1;
  by &varknots;
run;
data q2;
  set q1;
  by &varknots;
  if first.&varknots;
run;
data &librefknots..&knotdata;
  set q2 nobs=n;
  knotsp=int(n/5);
  if knotsp>=35 then kmx=35; else
  if knotsp<35 then kmx=knotsp;
  %if &numknots ne  %then %do;
    ktemp=&numknots;
    if 1 <= ktemp <= 35 then kmx=ktemp;
  %end;
  kintrvl=round(n/kmx);
  knotsok=mod(_n_,kintrvl);
  knots=&varknots;
  if knotsok=0 or _n_=n-1 then output;
  keep knots;
run;
%mend;
```

4 Simple Semiparametric Regression

An example of a simple semiparametric regression model is

$$\log(\texttt{yield}_i) = \beta_0 + \beta_1 \texttt{PL}_i + f(\texttt{density}_i) + \varepsilon_i, \quad 1 \leqslant i \leqslant n,$$

where the $\texttt{yield}_i$ and $\texttt{density}_i$ respectively refer to the yields (g/plant) and densities of white Spanish onion plants (plants/m^2) grown in two locations: Purnong

Landing and Virginia, South Australia. The variable $\texttt{PL}_i$ is an indicator defined as

$$\texttt{PL}_i = \begin{cases} 0 \text{ if } i\text{th measurement is from Virginia} \\ 1 \text{ if } i\text{th measurement is from Purnong Landing.} \end{cases}$$

These onions data are taken from Ratkowsky (1983). A detailed semiparametric analysis of the data is given by Young and Bowman (1995).

We use the phrase "simple semiparametric" because the model has a parametric component (location term) and a nonparametric component (density term). Such models are also commonly referred to as "partially linear" (e.g. Härdle, Liang and Gao, 2000). The special case where the parametric component is binary is sometimes called a "binary offset model". The fitting of this model is a trivial extension of the content of Section 2: add a column to the $\mathbf{X}$ matrix corresponding to the offset indicators (the $\texttt{PL}_i$ in the onions example).

4.1 S-PLUS commands

```
onions <- read.table("onions.dat",header=T)

dens <- onions$density
log.yield <- log(onions$yield)
location <- onions$location
```

Set up design matrices for a binary offset model.

```
X <- cbind(rep(1,length(dens)),dens,location)

knots <- default.knots(dens)
Z <- outer(dens,knots,"-")
Z <- Z*(Z>0)
```

Obtain the fit using mixed model function `lme()`.

```
fit <- lme(log.yield~-1+X,random=pdIdent(~-1+Z))
beta.hat <- fit$coef$fixed
u.hat <- unlist(fit$coef$random)
```

Extract the estimated standard deviation components.

```
sig.eps.hat <- fit$sigma
sig.u.hat   <- sig.eps.hat*exp(unlist(fit$modelStruct))
```

4.2 SAS code

The following SAS code fits the above simple semiparametric regression model.

```
libname paper '˜/test';
data paper.onions;
  infile '˜/test/onions.dat' missover;
  input density yield location;
  logyield=log(yield);
  if density = . then delete;
run;
%include "default_knots.macro";
%default_knots(librefknots=paper,data=paper.onions,
               knotdata=onionsknots,varknots=density);
data dataw;
  set paper.onions (keep=logyield density location);
  m=1;
run;
data kt1;
  set paper.onionsknots nobs=nk;
  call symput('nkt',nk);
run;
proc transpose data=paper.onionsknots prefix=knots out=knotst;
  var knots;
run;
data paper.knotst;
  set knotst;
  m=1;
run;
data dataw;
  merge dataw paper.knotst;
  by m;
  %let nk=&nkt;
  array Z (&nk) Z1-Z&nk;
  array knots (&nk) knots1-knots&nk;
  do k=1 to &nk;
    Z(k)=density-knots(k);
    if Z(k) < 0 then Z(k)=0;
  end;
  drop knots1-knots&nk _name_;
run;
ods output CovParms=paper.varcomp;
proc mixed;
  model logyield = location density / solution outp=paper.yhat;
  random  Z1-Z&nk / type=toep(1) s;
run;
```

5 Additive Models

An example of an *additive model* is

$$\sqrt{\texttt{mortality}_t} = \beta_0 + \beta\,\texttt{TSP}_t + f_1(t) + f_2(\texttt{temperature}_t) + f_3(\texttt{humidity}_t) + \varepsilon_t \quad (6)$$

where, for day t, $\texttt{mortality}_t$ is the number of deaths, $\texttt{TSP}_t$ is the number of Total Suspended Particles, $\texttt{temperature}_t$ is the temperature and $\texttt{humidity}_t$ is the humidity for the city of Milan, Italy. Here we will fit just one year of data, so $1 \leqslant t \leqslant 365$.

An additive model differs from a simple semiparametric model in that there may be several nonparametric components entering the model additively. Model (6) has three nonparametric components.

Design matrices appropriate for fitting (6) are

$$\mathbf{X} = \begin{bmatrix} 1 & \texttt{TSP}_1 & 1 & \texttt{temperature}_1 & \texttt{humidity}_1 \\ 1 & \texttt{TSP}_2 & 2 & \texttt{temperature}_2 & \texttt{humidity}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \texttt{TSP}_{365} & 365 & \texttt{temperature}_{365} & \texttt{humidity}_{365} \end{bmatrix}$$

and

$$\mathbf{Z} = \Big[\underset{1\leqslant k\leqslant K_1}{(t-\kappa_k^1)_+} \quad \underset{1\leqslant k\leqslant K_2}{(\texttt{temperature}_t-\kappa_k^2)_+} \quad \underset{1\leqslant k\leqslant K_3}{(\texttt{humidity}_t-\kappa_k^3)_+}\Big]_{1\leqslant t\leqslant 365}.$$

Here κ_k^1, κ_k^2 and κ_k^3 are knot sequences of lengths K_1, K_2 and K_3 for handling t, `temperature` and `humidity` respectively.

The random effects have covariance matrix

$$\text{Cov}(\mathbf{u}) = \begin{bmatrix} \sigma_1^2\mathbf{I}_{K_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_2^2\mathbf{I}_{K_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma_3^2\mathbf{I}_{K_3} \end{bmatrix}$$

where $\mathbf{I}_K$ denotes the $K\times K$ identity matrix.

5.1 S-PLUS commands

```
milanmort <- read.table("milanmort.dat",header=T)

year.num <- 1
subinds <- (365*(year.num-1)+1):(365*year.num)
milanmort <- milanmort[subinds,]

y <- sqrt(milanmort$resp.mort)
x.1 <- milanmort$day.num
x.2 <- milanmort$mean.temp
x.3 <- milanmort$rel.humid
x.4 <- milanmort$TSP
```

Set up design matrices.

```
X <- cbind(rep(1,length(y)),x.1,x.2,x.3,x.4)


knots.1 <- default.knots(x.1)
Z.1 <- outer(x.1,knots.1,"-")
Z.1 <- Z.1*(Z.1>0)
K.1 <- length(knots.1)

knots.2 <- default.knots(x.2)
Z.2 <- outer(x.2,knots.2,"-")
Z.2 <- Z.2*(Z.2>0)
K.2 <- length(knots.2)

knots.3 <- default.knots(x.3)
Z.3 <- outer(x.3,knots.3,"-")
Z.3 <- Z.3*(Z.3>0)
K.3 <- length(knots.3)

Z <- cbind(Z.1,Z.2,Z.3)
```

Fit the additive model using `lme()`. First the block structure of the random effects covariance matrix must be specified and stored in the list `Z.block`.

```
re.block.inds <- list(1:K.1,(K.1+1):(K.1+K.2),
                     (K.1+K.2+1):(K.1+K.2+K.3))
Z.block  <-  list()
for (i in 1:length(re.block.inds))
   Z.block[[i]] <- as.formula(paste("~Z[,c(",paste(
             re.block.inds[[i]],collapse=","),")]-1"))

fit <- lme(y~-1+X,random=pdBlocked(Z.block,pdClass="pdIdent"))
beta.hat <- fit$coef$fixed
u.hat <- unlist(fit$coef$random)
```

Extract the estimated variance components.

```
sig.eps.hat <- fit$sigma
sig.u.hat  <- sig.eps.hat*exp(unlist(fit$modelStruct))
```

Print a summary of the fixed effects. The last row is the only one that has an interpretation and corresponds to the effect of air pollution (non-significant in this case).

```
print(summary(fit)$tTable)
```

5.2 SAS code

The following SAS code fits the above additive model.

```
libname paper '~/test';
data paper.milan1;
  infile '~/test/milanmort.dat' missover;
  input daynum dayweek holiday meantemp relhumid
  totmort respmort s02 tsp;
  y=sqrt(respmort);
  x1=daynum;
  x2=meantemp;
  x3=relhumid;
  x4=tsp;
  if daynum ne . ;
run;
data paper.milan2;
  set paper.milan1;
  if _n_ <= 365;
run;

/*********************************************/
/* creating knots for 3 smoothing variables */
/*********************************************/

%include "default_knots.macro";
%default_knots(librefknots=paper,data=paper.milan2,
               knotdata=knots1,varknots=x1);
%default_knots(librefknots=paper,data=paper.milan2,
               knotdata=knots2,varknots=x2);
%default_knots(librefknots=paper,data=paper.milan2,
               knotdata=knots3,varknots=x3);

data dataw;
  set paper.milan2 (keep=y x1-x4);
  m=1;
run;
data kt1;
  set paper.knots1 nobs=nk1;
  call symput('nkt1',nk1);
run;
proc transpose data=paper.knots1 prefix=knots1_ out=knotst1;
  var knots;
run;
data kt2;
  set paper.knots2 nobs=nk2;
  call symput('nkt2',nk2);
run;
proc transpose data=paper.knots2 prefix=knots2_ out=knotst2;
  var knots;
run;
data kt3;
  set paper.knots3 nobs=nk3;
  call symput('nkt3',nk3);
run;
proc transpose data=paper.knots3 prefix=knots3_ out=knotst3;
  var knots;
run;
data paper.knotst;
  merge knotst1 knotst2 knotst3;
  m=1;
run;

/************************************/
/* creating the Z matrix            */
/************************************/

data dataw;
  merge dataw paper.knotst;
  by m;
  %let nk1=&nkt1;
  %let nk2=&nkt2;
  %let nk3=&nkt3;
  array Z1a (&nk1) Z1_1-Z1_&nk1;
  array knots1a (&nk1) knots1_1-knots1_&nk1;
  do k=1 to &nk1;
     Z1a(k)=x1-knots1a(k);
     if Z1a(k) < 0 then Z1a(k)=0;
  end;
  array Z2a (&nk2) Z2_1-Z2_&nk2;
  array knots2a (&nk2) knots2_1-knots2_&nk2;
  do k=1 to &nk2;
     Z2a(k)=x2-knots2a(k);
     if Z2a(k) < 0 then Z2a(k)=0;
  end;
  array Z3a (&nk3) Z3_1-Z3_&nk3;
  array knots3a (&nk3) knots3_1-knots3_&nk3;
  do k=1 to &nk3;
     Z3a(k)=x3-knots3a(k);
     if Z3a(k) < 0 then Z3a(k)=0;
  end;
  drop knots1_1-knots1_&nk1 knots2_1-knots2_&nk2
       knots3_1-knots3_&nk3 _name_;
run;
ods output CovParms=paper.varcomp;

/************************************/
/* fitting the additive model       */
/************************************/

proc mixed;
  model y = x1-x4 / solution outp=paper.yhat;
  random  Z1_1-Z1_&nk1 / type=toep(1) s;
  random  Z2_1-Z2_&nk2 / type=toep(1) s;
  random  Z3_1-Z3_&nk3 / type=toep(1) s;
run;
```

6 Additive Mixed Models

The sitka data are listed in Table 1.2 and displayed in Figure 1.3 of Diggle, Liang and Zeger (1995). They correspond to measurements of log-size for 79 Sitka spruce trees grown in normal or ozone-enriched environments.

A useful model for these data is the *additive mixed* model

$$\log(\texttt{size}_{ij}) = U_i + \texttt{ozone}_{ij} + f(\texttt{days}_{ij}) + \varepsilon_{ij}, \quad 1 \leqslant j \leqslant n_i, \quad 1 \leqslant i \leqslant m \qquad (7)$$

where f is some smooth function. For the sitka spruce data $m = 79$ and $n_i = 26$ for all i. Note that $\texttt{ozone}_{ij}$ is an indicator variable corresponding to whether or not the trees are grown in normal or ozone-enriched environments.

Appropriate design matrices are

$$\mathbf{X} = \begin{bmatrix} 1 & \texttt{ozone}_{11} & \texttt{days}_{11} \\ \vdots & \vdots & \\ 1 & \texttt{ozone}_{1n_1} & \texttt{days}_{1n_1} \\ \vdots & \vdots & \\ 1 & \texttt{ozone}_{m1} & \texttt{days}_{m1} \\ \vdots & \vdots & \\ 1 & \texttt{ozone}_{mn_m} & \texttt{days}_{mn_m} \end{bmatrix},$$

$$\mathbf{Z} = \begin{bmatrix} 1 & \cdots & 0 & (\texttt{days}_{11}-\kappa_1)_+ & \cdots & (\texttt{days}_{11}-\kappa_K)_+ \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & 0 & (\texttt{days}_{1n_1}-\kappa_1)_+ & \cdots & (\texttt{days}_{1n_1}-\kappa_K)_+ \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & (\texttt{days}_{m1}-\kappa_1)_+ & \cdots & (\texttt{days}_{m1}-\kappa_K)_+ \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & (\texttt{days}_{mn_m}-\kappa_1)_+ & \cdots & (\texttt{days}_{mn_m}-\kappa_K)_+ \end{bmatrix}.$$

Note that the random effects vector is

$$\text{and} \qquad \mathbf{u} = \begin{bmatrix} U_1 \\ \vdots \\ U_m \\ u_1 \\ \vdots \\ u_K \end{bmatrix}.$$

We can simultaneously estimate variance components for the random intercept and the amount of smoothing for f through the mixed model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}, \quad \text{Cov}\begin{bmatrix} \mathbf{u} \\ \boldsymbol{\varepsilon} \end{bmatrix} = \begin{bmatrix} \sigma_U^2\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_u^2\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma_\varepsilon^2\mathbf{I} \end{bmatrix}. \qquad (8)$$

Here σ_U^2 measures the between subject variation, σ_ε^2 measures within subject variation and σ_u^2 controls the amount of smoothing done to estimate f.

6.1 S-PLUS commands

Read in the sitka spruce data:

```
sitka <- read.table("sitka_spruce.dat",header=T)
```

Extract data corresponding to the `sitka` data frame:

```
ozone <- sitka$ozone
days <- sitka$days
log.size <- sitka$log.size
idnum <- sitka$idnum
```

Construct the `y` response vector and `X` matrix.

```
y <- log.size
X <- cbind(rep(1,length(y)),days,ozone)
```

Create the spline component of the Z matrix. Note that the presence of `knots` for the `days` variable can be a known vector of knots. Notice that in the SAS code below, we use the knot vector generated by the Splus code. The estimates from both the Splus and SAS code are identical.

```
Z.spline <- outer(days,knots,"-")
Z.spline <- Z.spline*(Z.spline>0)
```

The component of the Z matrix corresponding to the random intercept does not need to be specified and can be handled through the identification numbers stored in `idnum`:

```
idnum <- factor(idnum)
fit <- lme(y~-1+X,random=pdBlocked(list(pdIdent(~-1+idnum),
                        pdIdent(~-1+Z.spline))))

beta.hat <- fit$coef$fixed
u.hat <- unlist(fit$coef$random)

sig.eps.hat <- fit$sigma
sig.u.hat  <- sig.eps.hat*exp(2*unlist(fit$modelStruct))
```

6.2 SAS code

The following SAS code fits the above additive mixed model.

```
libname paper '~/test';
data paper.sitka1;
  infile '~/test/sitka_spruce.dat' missover;
  input idnum order days logsize ozone;
  if idnum ne .;
run;

/*********************************************/
/* Creating knots for the smoothing variable:*/
/* these knots were obtained from the Splus  */
/* program sec6.1.s. The fixed effects       */
/* estimates are thus identical to those of  */
/* the Splus code.                           */
/*********************************************/
data paper.knots;
   input knots;
datalines;
196.5
247.6667
498.5
563.6667
```

```
617.3333
;
run;

data dataw;
  set paper.sitka1 (keep=idnum logsize days ozone);
  m=1;
run;
data kt;
  set paper.knots nobs=nk;
  call symput('nkt',nk);
run;
proc transpose data=paper.knots prefix=knots out=knotst;
  var knots;
run;

data paper.knotst;
  set knotst;
  m=1;
run;


/***********************************/
/* creating the Z matrix           */
/***********************************/

data dataw;
  merge dataw paper.knotst;
  by m;
  %let nk=&nkt;
  array Z (&nk) Z1-Z&nk;
  array knots (&nk) knots1-knots&nk;
  do k=1 to &nk;
     Z(k)=days-knots(k);
     if Z(k) < 0 then Z(k)=0;
  end;
  drop knots1-knots&nk _name_;
run;
ods output CovParms=paper.varcomp;

/************************************/
/* fitting the additive model       */
/************************************/

proc mixed;
```

```
  class idnum;
  model logsize = days ozone / solution outp=paper.yhat;
  random idnum / type=toep(1) s;
  random  Z1-Z&nk / type=toep(1) s;
run;
```

7 Additive Models with Interactions

Coull, Ruppert and Wand (2001) developed mixed model approaches to building in *factor by curve* interactions into additive models. The example concerning pollen counts given there required an overdispersed Poisson mixed model since the response variable was a count. For the purposes of this paper we tried to work with the square root response transformation, but found that the normality assumption was not reasonable. Therefore, we will use another data set with similar characteristics for which the square root response transformation does reasonably approximate normality. The data correspond to mortality counts for the city of Milan, Italy, as analysed by Zanobetti, Wand, Schwartz and Ryan (2000). The questions for these data are different for those arising in the pollen data, but we will ignore these for now. Our goal here is to simply illustrate the fitting of additive models with interactions.

Consider the model corresponding to daily measurements for the years 1984–1987.

$$\begin{aligned}\mathsf{E}(\sqrt{\texttt{mortality}_t}) = {}& \beta_1\texttt{TSP}_t + \beta_2\texttt{holiday}_t + f_1(\texttt{min.temp}_t) + f_2(\texttt{rel.humid}_t) \\ & + f_{\text{year}}(\texttt{day.of.year}_t)\end{aligned} \tag{9}$$

where, for day t, $\sqrt{\texttt{mortality}_t}$ is the number of respiratory mortalities, $\texttt{TSP}_t$ is the air pollution measure `Total Suspended Particles`, $\texttt{min.temp}_t$ is the minimum temperature, $\texttt{rel.humid}_t$ is the relative humidity. For the final term $\texttt{year} \in \{1984, 1985, 1986, 1987\}$ and $\texttt{day.of.year}_t$ is the number of day within the particular year and represents an interaction between the factor `year` and the overall seasonal effect.

Model (9) can be formulated as a linear mixed model (see Coull *et al.*, 2001 for details)

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}$$

where

$$\begin{aligned}\mathbf{X} = {}& [1\ \texttt{TSP}_t\ \texttt{holiday}_t\ \texttt{min.temp}_t\ \texttt{rel.humid}_t\ I(\texttt{year} = 1985)_t, \ldots, I(\texttt{year} = 1987)_t \\ & \texttt{day.of.year}_t I(\texttt{year} = 1985)_t, \ldots, \texttt{day.of.year}_t I(\texttt{year} = 1987)_t]_{1\leqslant t\leqslant 1460}\end{aligned}$$

and

$$\mathbf{Z} = [\underset{1\leqslant k\leqslant K_{mt}}{(\texttt{min.temp}_t - \kappa_k^{mt})_+}\ \underset{1\leqslant k\leqslant K_{rh}}{(\texttt{rel.humid}_t - \kappa_k^{rh})_+}\ I(\texttt{year} = 1984)_t\underset{1\leqslant k\leqslant K_{dy}}{(\texttt{day.of.year}_t - \kappa_k^{dy})_+}, \ldots,$$

$$I(\texttt{year} = 1987)_t(\texttt{day.of.year}_t - \kappa_k^{dy})_+]_{\substack{1 \leqslant t \leqslant 1460 \\ 1 \leqslant k \leqslant K_{dy}}}$$

where

$$I(\texttt{year} = j)_t = \begin{cases} 1 \ \texttt{year}_t = j \\ 0 \text{ otherwise,} \end{cases}$$

κ_k^{mt}, $1 \leqslant k \leqslant K_{mt}$ are knots for minimum temperature, κ_k^{rh}, $1 \leqslant k \leqslant K_{rh}$ are knots for relative humidity and κ_k^{dy}, $1 \leqslant k \leqslant K_{dy}$ are knots for day of the year. Also,

$$\text{Cov}(\mathbf{u}) = \text{blockdiag}(\sigma^2_{mt}\mathbf{I}, \sigma^2_{rh}\mathbf{I}, \sigma^2_{dy,1984}\mathbf{I}, \sigma^2_{dy,1985}\mathbf{I}, \sigma^2_{dy,1986}\mathbf{I}, \sigma^2_{dy,1987}\mathbf{I}).$$

Note that the fixed effects component has `year`=1984 as a reference group. However, the random effects component does not use a reference group and all years are on equal footing.

7.1 S-PLUS commands

Set up the design matrix X for the fixed effects with 1981 serving as the reference year:

```
X <- cbind(rep(1,n),holiday,day.in.seas,indic.1985,indic.1986,indic.1987,
              day.in.seas*indic.1985,
              day.in.seas*indic.1986,
              day.in.seas*indic.1987,TSP,temperature,rel.humid)
```

Set up Z matrix for temperature and relative humidity smoothing function:

```
K.temp <- 15
knots.temp <- quantile(unique(temperature),
              seq(0,1,length=K.temp+2))[-c(1,K.temp+2)]

K.relh <- 15
knots.relh <- quantile(unique(rel.humid),
                       seq(0,1,length=K.relh+2))[-c(1,K.relh+2)]

Z.temp <- outer(temperature,knots.temp,"-")
Z.temp <- Z.temp*(Z.temp>0)

Z.relh <- outer(rel.humid,knots.relh,"-")
Z.relh <- Z.relh*(Z.relh>0)
```

Set up Z matrix for day in season and interaction terms between day in season and year:

```
K <- 15
knots <- quantile(unique(day.in.seas),seq(0,1,length=K+2))[-c(1,K+2)]
Z.overall <- outer(day.in.seas,knots,"-")
Z.overall <- Z.overall*(Z.overall>0)

Z <- cbind(Z.overall,indic.1984*Z.overall,indic.1985*Z.overall,
             indic.1986*Z.overall,indic.1987*Z.overall)
```

Set up blocked components of the Z matrix:

```
re.block.inds <- list(1:K,(K+1):(2*K),(2*K+1):(3*K),
                         (3*K+1):(4*K),(4*K+1):(5*K),
                         (5*K+1):(5*K+K.temp),
                         (5*K+K.temp+1):(5*K+K.temp+K.relh))

   Z <- cbind(Z,Z.temp,Z.relh)

   Z.block  <-  list()
   for (i in 1:length(re.block.inds))
      Z.block[[i]] <- as.formula(paste("~Z[,c(",paste(
                re.block.inds[[i]],collapse=","),")]-1"))
```

Fit the additive mixed model with interactions:

```
fit <- lme(sqrt.mort~-1+X,random=pdBlocked(Z.block,pdClass="pdIdent"))
```

Extract the fixed effects estimates, the blups, the error variance, and the variance components correponding to the random effects:

```
beta.hat <- fit$coef$fixed
u.hat <- unlist(fit$coef$random)

sig.sq.eps <- fit$sigma^2
sig.sq.u   <- sig.sq.eps*exp(2*unlist(fit$modelStruct))
```

7.2 SAS code

The following SAS code fits the above additive model with interaction.

```
libname paper '~/test';
data milan1;
  infile '~/test/milanmort.dat' missover;
  input daynum dayweek holiday temperature relhumid
  totmort respmort s02 tsp;
  sqrtmort=sqrt(respmort);
  if daynum ne . ;
```

```
run;
data paper.milan1;
  set milan1;
  if 1 <= _n_ <= 4*365;

  if  1<= _n_ <= 365 then  do;
      indic1984=1;
      dayinseas=_n_;
  end; else
  if (365+1) <= _n_ <= 365*2 then do;
      indic1985=1;
      dayinseas=_n_-365;
  end; else
  if (365*2+1) <= _n_ <= 365*3 then do;
      indic1986=1;
      dayinseas=_n_-365*2;
  end; else
  if (365*3+1) <= _n_ <= 365*4 then do;
      indic1987=1;
      dayinseas=_n_-365*3;
  end;


  array ind (4) indic1984-indic1987;
  do i=1 to 4;
     if ind(i)=. then ind(i)=0;
  end;
  m=1;
 run;

%include "default_knots.macro";
%default_knots(librefknots=paper,data=paper.milan1,
               knotdata=knots1,varknots=temperature);
%default_knots(librefknots=paper,data=paper.milan1,
               knotdata=knots2,varknots=relhumid);
%default_knots(librefknots=paper,data=paper.milan1,
               knotdata=knots3,varknots=dayinseas);
data kt1;
  set paper.knots1 nobs=nk1;
  call symput(’nkt1’,nk1);
run;
proc transpose data=paper.knots1 prefix=knots1_ out=knotst1;
  var knots;
run;
data kt2;
```

```
  set paper.knots2 nobs=nk2;
  call symput('nkt2',nk2);
run;
proc transpose data=paper.knots2 prefix=knots2_ out=knotst2;
  var knots;
run;
data kt3;
  set paper.knots3 nobs=nk3;
  call symput('nkt3',nk3);
run;
proc transpose data=paper.knots3 prefix=knots3_ out=knotst3;
  var knots;
run;
data paper.knotst;
  merge knotst1 knotst2 knotst3;
  m=1;
run;
/************************************/
/* creating the Z matrix           */
/************************************/

data dataw;
  merge paper.milan1 paper.knotst;
  by m;
  %let nk1=&nkt1;
  %let nk2=&nkt2;
  %let nk3=&nkt3;
  array Z1a (&nk1) Z1_1-Z1_&nk1;
  array knots1a (&nk1) knots1_1-knots1_&nk1;
  do k=1 to &nk1;
     Z1a(k)=temperature-knots1a(k);
     if Z1a(k) < 0 then Z1a(k)=0;
  end;
  array Z2a (&nk2) Z2_1-Z2_&nk2;
  array knots2a (&nk2) knots2_1-knots2_&nk2;
  do k=1 to &nk2;
     Z2a(k)=relhumid-knots2a(k);
     if Z2a(k) < 0 then Z2a(k)=0;
  end;
  array Z3a (&nk3) Z3_1-Z3_&nk3;
  array knots3a (&nk3) knots3_1-knots3_&nk3;

  array intera1a  (&nk3) inter1_1-inter1_&nk3;
  array intera2a  (&nk3) inter2_1-inter2_&nk3;
  array intera3a  (&nk3) inter3_1-inter3_&nk3;
  array intera4a  (&nk3) inter4_1-inter4_&nk3;

  do k=1 to &nk3;
     Z3a(k)=dayinseas-knots3a(k);
     if Z3a(k) < 0 then Z3a(k)=0;
     *constructing interaction terms;
     intera1a(k) = indic1984*Z3a(k);
     intera2a(k) = indic1985*Z3a(k);
     intera3a(k) = indic1986*Z3a(k);
     intera4a(k) = indic1987*Z3a(k);
  end;
  drop knots1_1-knots1_&nk1 knots2_1-knots2_&nk2
       knots3_1-knots3_&nk3 _name_;
run;


ods output CovParms=paper.varcomp;

/************************************/
/* fitting the additive model       */
/************************************/

proc mixed;
  model sqrtmort =  holiday dayinseas
                    indic1985 indic1986 indic1987
                    indic1985*dayinseas indic1986*dayinseas
                    indic1987*dayinseas
                    tsp temperature relhumid
                    / solution outp=paper.yhat;
  random  Z3_1-Z3_&nk3 / type=toep(1) s;
  random inter1_1-inter1_&nk3 / type=toep(1) s;
  random inter2_1-inter2_&nk3 / type=toep(1) s;
  random inter3_1-inter3_&nk3 / type=toep(1) s;
  random inter4_1-inter4_&nk3 / type=toep(1) s;
  random  Z1_1-Z1_&nk1 / type=toep(1) s;
  random  Z2_1-Z2_&nk2 / type=toep(1) s;
run;
```

8 Varying Coefficient Models

Let x be a predictor variable that, for given values of a *modifying* predictor s, has a linear relationship with the mean of the response variable y. If (x_i, s_i, y_i), $1 \leqslant i \leqslant n$, are measurements on each then a *varying coefficient model* for these data is

$$y_i = \alpha(s_i) + \beta(s_i)x_i + \varepsilon_i. \tag{10}$$

The model allows the intercept and slope coefficients to be arbitrary smooth functions of s. The penalized linear spline version of this model is

$$y_i = \alpha_0 + \alpha_1 s_i + \sum_{k=1}^{K} u_k^{\alpha}(s_i - \kappa_k)_+ + \left\{ \beta_0 + \beta_1 s_i + \sum_{k=1}^{K} u_k^{\beta}(s_i - \kappa_k)_+ \right\} x_i + \varepsilon_i.$$

where $\kappa_1, \ldots, \kappa_K$ are knots over the range of the s_i values. A mixed model representation $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}$ is obtained by setting

$$\mathbf{X} = [1 \;\; s_i \;\; x_i \;\; s_i x_i]_{1\leqslant i\leqslant n}, \qquad \mathbf{Z} = [\underset{1\leqslant k\leqslant K}{(s_i - \kappa_k)_+} \;\; \underset{1\leqslant k\leqslant K}{x_i(s_i - \kappa_k)_+}]_{1\leqslant i\leqslant n},$$

$\mathbf{u} = [u_1^{\alpha}, \ldots, u_K^{\alpha}, u_1^{\beta}, \ldots, u_K^{\beta}]^{\mathsf{T}}$ and $\mathrm{Cov}(\mathbf{u}) = \mathrm{diag}\{\sigma_{\alpha}^2 \mathbf{1}_{K\times 1}, \sigma_{\beta}^2 \mathbf{1}_{K\times 1}\}$.

8.1 S-PLUS commands

Varying coefficient models will be demonstrated on the `ethanol` data set in `S-PLUS`. Type `help(ethanol)` to find out more about these data. Extract the data as follows.

```
z <- ethanol$E
x <- ethanol$C
y <- ethanol$NOx
```

Set up the design matrices.

```
X <- cbind(rep(1,length(y)),z,x,x*z)

knots <- default.knots(z)
K <- length(knots)
Z <- outer(z,knots,"-")
Z <- Z*(Z>0)
Z <- cbind(Z,x*Z)
```

Fit the model using `lme()`.

```
re.block.inds <- list(1:K,(K+1):(2*K))

Z.block  <-  list()
for (i in 1:length(re.block.inds))
   Z.block[[i]] <- as.formula(paste("~Z[,c(",paste(
             re.block.inds[[i]],collapse=","),")]-1"))

fit <- lme(y~-1+X,random=pdBlocked(Z.block,
           pdClass="pdIdent"))
```

The estimated fixed and random coefficients and fitted values are:

```
beta.hat <- fit$coef$fixed
u.hat <- unlist(fit$coef$random)
```

The estimated standard deviation components are:

```
sig.eps.hat <- fit$sigma
sig.u.hat   <- sig.eps.hat*exp(unlist(fit$modelStruct))
```

8.2 SAS code

The following SAS code fits the above varying coefficient model.

```
libname paper '~/test';
data paper.ethanol;
  infile '~/test/ethanol.dat' missover;
  input idnum nox c e;
  if idnum ne .;
run;

/*********************************************/
/* creating knots for the smoothing variable */
/*********************************************/

%include "default_knots.macro";
%default_knots(librefknots=paper,data=paper.ethanol,
               knotdata=knots,varknots=e);

data dataw;
  set paper.ethanol;
  m=1;
run;

data kt;
  set paper.knots nobs=nk;
  call symput('nkt',nk);
run;
proc transpose data=paper.knots prefix=knots out=knotst;
  var knots;
run;


data paper.knotst;
  set knotst;
  m=1;
run;
```

```
/**********************************/
/* creating the Z matrix          */
/**********************************/

data dataw;
  merge dataw paper.knotst;
  by m;
  %let nk=&nkt;
  array Z  (&nk) Z1-Z&nk;
  array XZ (&nk) XZ1-XZ&nk;
  array knots (&nk) knots1-knots&nk;
  do k=1 to &nk;
     Z(k)=e-knots(k);
     if Z(k) < 0 then Z(k)=0;
     XZ(k)=c*Z(k);
  end;
  drop knots1-knots&nk _name_;
run;

ods output CovParms=paper.varcomp;

/***********************************/
/* fitting the additive model      */
/***********************************/

proc mixed;
  model nox = e c e*c / solution ; *outp=paper.yhat;
  random  Z1-Z&nk  / type=toep(1) s;
  random XZ1-XZ&nk / type=toep(1) s;
run;
```

9 User Specified Smoothing Parameters

In the mixed model representation of smoothers described in Sections 2–8 the amount of smoothing is controlled by the variance components appearing in both $\text{Cov}(\mathbf{u})$ and $\text{Cov}(\boldsymbol{\varepsilon})$. Mixed model software usually defaults to the REML or ML estimates of these variance components. Thus, the amount of smoothing is chosen automatically. However, there are situations where the analyst would like to specify the amount of smoothing. A simple example is a sensitivity analysis for a simple semiparametric model (Section 4) where the sensitivity of the estimate of the offset coefficient β_1 to different amounts of smoothing in the estimate of f requires investigation (e.g. Bowman and Azzalini, 1997). Another is the feature significance methodology described by Chaudhuri and Marron (1999), for example.

In SAS the problem of user specified smoothing parameters is relatively easy to overcome using the PARMS option – see Section 9.3. However versions of S-PLUS's `lme()` known to us at the time of writing do not support user specified variance components and direct computation is required. We will show how this can be done in the scatterplot smoothing situation. Extensions to other models follows relatively straightforwardly.

Recall the setting and notation described in Section 2. For given values of σ_u^2 and σ_ε^2 application of ML and Best Prediction (BP) to obtain $\widehat{\boldsymbol{\beta}}$ and $\widehat{\mathbf{u}}$ is equivalent to solving the penalized least squares problem

$$\begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\mathbf{u}} \end{bmatrix} = \underset{\boldsymbol{\beta},\mathbf{u}}{\operatorname{argmin}} \left(\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u}\|^2 + \alpha\|\mathbf{u}\|^2 \right) \tag{11}$$

where $\alpha \equiv \sigma_\varepsilon^2/\sigma_u^2$ and, for a general vector $\mathbf{v}$, $\|\mathbf{v}\| \equiv \sqrt{\mathbf{v}^\mathsf{T}\mathbf{v}}$ (e.g. Robinson, 1991). This is an example of *penalized least squares* (e.g. Green, 1987) since minimisation of the least squares $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u}\|^2$ is subject to the penalty $\alpha\|\mathbf{u}\|^2$ being imposed on the coefficients in $\mathbf{u}$. The solution is easily shown to be

$$\begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\mathbf{u}} \end{bmatrix} = (\mathbf{C}^\mathsf{T}\mathbf{C} + \alpha\mathbf{D})^{-1}\mathbf{C}^\mathsf{T}\mathbf{y}$$

where $\mathbf{C} = [\mathbf{X}\ \mathbf{Z}]$ and $\mathbf{D} = \operatorname{diag}(0, 0, 1, \ldots, 1)$. The fitted values are then

$$\widehat{\mathbf{f}}_\alpha = \mathbf{C}(\mathbf{C}^\mathsf{T}\mathbf{C} + \alpha\mathbf{D})^{-1}\mathbf{C}^\mathsf{T}\mathbf{y} \tag{12}$$

9.1 Demmler-Reinsch orthogonalization

Algorithm 1 allows for fast and stable calculation of (12).

Algorithm 1

```
    Inputs: $\mathbf{y}$, $\mathbf{C}$, $\mathbf{D}$, $\alpha$.
(1) Obtain the singular value decomposition of $\mathbf{C}$:
        $\mathbf{C} = \mathbf{U}_{\text{C}}\text{diag}(\mathbf{d}_{\text{C}})\mathbf{V}_{\text{C}}^{\mathsf{T}}$.
(2) Form the symmetric matrix $\text{diag}(\mathbf{1}/\mathbf{d}_{\text{C}})\mathbf{V}_{\text{C}}^{\mathsf{T}}\mathbf{D}\mathbf{V}_{\text{C}}\text{diag}(\mathbf{1}/\mathbf{d}_{\text{C}})$ and obtain its singular value decomposition:
        $\text{diag}(\mathbf{1}/\mathbf{d}_{\text{C}})\mathbf{V}_{\text{C}}^{\mathsf{T}}\mathbf{D}\mathbf{V}_{\text{C}}\text{diag}(\mathbf{1}/\mathbf{d}_{\text{C}}) = \mathbf{U}_{\text{D}}\text{diag}(\mathbf{d}_{\text{D}})\mathbf{U}_{\text{D}}^{\mathsf{T}}$.
(3) Compute the matrix and vector
        $\mathbf{A} \equiv \mathbf{U}_{\text{C}}\mathbf{U}_{\text{D}}$ and $\mathbf{b} \equiv \mathbf{A}^{\mathsf{T}}\mathbf{y}$.
(4) The fitted values are then
        $\widehat{\mathbf{f}}_{\alpha} = \mathbf{A}\left(\frac{\mathbf{b}}{\mathbf{1} + \alpha\,\mathbf{d}_{\text{D}}}\right)$
    with corresponding degrees of freedom
        $df_{\text{fit}}(\alpha) = \mathbf{1}^{\mathsf{T}}\left(\frac{\mathbf{1}}{\mathbf{1} + \alpha\,\mathbf{d}_{\text{D}}}\right)$.
```

Once the matrix $\mathbf{A}$ and vectors $\mathbf{b}$ and $\mathbf{d}_{\text{D}}$ have been computed, the vector of fits, for different values of α, reduces to a matrix multiplication. Therefore, $\widehat{\mathbf{f}}_{\alpha}$ and $df_{\text{fit}}(\alpha)$ can be computed cheaply for several α values. This is particularly useful when solving for the α corresponding to a pre-specified number of degrees of freedom.

9.1.1 Justification of Algorithm 1.

Now

$$\text{diag}(\mathbf{1}/\mathbf{d}_{\text{C}})\mathbf{V}_{\text{C}}^{\mathsf{T}}\mathbf{D}\mathbf{V}_{\text{C}}\text{diag}(\mathbf{1}/\mathbf{d}_{\text{C}}) = \mathbf{U}_{\text{D}}\text{diag}(\mathbf{d}_{\text{D}})\mathbf{U}_{\text{D}}^{\mathsf{T}} \quad \text{with } \mathbf{U}_{\text{D}}^{\mathsf{T}}\mathbf{U}_{\text{D}} = \mathbf{I}.$$

Since $\mathbf{V}_{\text{C}}$ and $\mathbf{U}_{\text{D}}$ are square matrices $\mathbf{V}_{\text{C}}^{\mathsf{T}} = \mathbf{V}_{\text{C}}^{-1}$ and $\mathbf{U}_{\text{D}}^{\mathsf{T}} = \mathbf{U}_{\text{D}}^{-1}$ and so

$$\mathbf{D} = \mathbf{V}_{\text{C}}\text{diag}(\mathbf{d}_{\text{C}})\mathbf{U}_{\text{D}}\text{diag}(\mathbf{d}_{\text{D}})\mathbf{U}_{\text{D}}^{-1}\text{diag}(\mathbf{d}_{\text{C}})\mathbf{V}_{\text{C}}^{\mathsf{T}}.$$

Also, noting that $\mathbf{U}_{\rm C}^{\sf T}\mathbf{U}_{\rm C} = \mathbf{I}$,

$$\mathbf{C}^{\sf T}\mathbf{C} = \mathbf{V}_{\rm C}\text{diag}(\mathbf{d}_{\rm C})\mathbf{U}_{\rm C}^{\sf T}\mathbf{U}_{\rm C}\text{diag}(\mathbf{d}_{\rm C})\mathbf{V}_{\rm C}^{\sf T}$$

Thus,

$$\mathbf{C}^{\sf T}\mathbf{C} + \alpha\mathbf{D} = \mathbf{V}_{\rm C}\text{diag}(\mathbf{d}_{\rm C})\mathbf{U}_{\rm D}\{\mathbf{I} + \alpha\text{diag}(\mathbf{d}_{\rm D})\}\mathbf{U}_{\rm D}^{-1}\text{diag}(\mathbf{d}_{\rm C})\mathbf{V}_{\rm C}^{\sf T}$$

and

$$\begin{aligned}\widehat{\mathbf{f}}_\alpha &= \mathbf{U}_{\rm C}\text{diag}(\mathbf{d}_{\rm C})\mathbf{V}_{\rm C}^{\sf T}[\mathbf{V}_{\rm C}\text{diag}(\mathbf{d}_{\rm C})\mathbf{U}_{\rm D}\{\mathbf{I} + \alpha\text{diag}(\mathbf{d}_{\rm D})\}\mathbf{U}_{\rm D}^{-1}\text{diag}(\mathbf{d}_{\rm C})\mathbf{V}_{\rm C}^{\sf T}]^{-1} \\ &\qquad \times\{\mathbf{U}_{\rm C}\text{diag}(\mathbf{d}_{\rm C})\mathbf{V}_{\rm C}^{\sf T}\}^{\sf T}\mathbf{y} \\ &= (\mathbf{U}_{\rm C}\mathbf{U}_{\rm D})\{\text{diag}(\mathbf{1} + \alpha\text{diag}(\mathbf{d}_{\rm D}))\}^{-1}(\mathbf{U}_{\rm C}\mathbf{U}_{\rm D})^{\sf T}\mathbf{y} \\ &= \mathbf{A}\left(\frac{\mathbf{b}}{\mathbf{1} + \alpha\,\mathbf{d}_{\rm D}}\right)\end{aligned}$$

where $\mathbf{A} \equiv \mathbf{U}_{\rm C}\mathbf{U}_{\rm D}$ and $\mathbf{b} \equiv \mathbf{A}^{\sf T}\mathbf{y}$.

An alternative approach to handling the ridge regressions that arise in penalized spline models is through QR decomposition (e.g., Golub and Van Loan, 1983; Hastie, 1996). Algorithm A.2 provides another fitting procedure for (12).

Algorithm 2

Inputs: $\mathbf{y}$, $\mathbf{C}$, $\mathbf{D}$, α.

(1) Form the augmented matrices

$$\mathbf{C}_a = \begin{bmatrix}\mathbf{C} \\ \sqrt{\alpha}\,\mathbf{D}^{1/2}\end{bmatrix} \quad \text{and} \quad \mathbf{y}_a = \begin{bmatrix}\mathbf{y} \\ \mathbf{0}\end{bmatrix}.$$

(2) Obtain the QR decomposition of $\mathbf{C}_a$:

$$\mathbf{C}_a = \mathbf{QR}$$

and set

$$\mathbf{Q}_1 = \text{matrix consisting of first } n \text{ rows of } \mathbf{Q}.$$

(3) The fitted values are then

$$\widehat{\mathbf{f}}_\alpha = \mathbf{C}\mathbf{R}^{-1}\mathbf{Q}_1^{\sf T}\mathbf{y}.$$

9.2 S-PLUS commands

We now give S-PLUS commands for Algorithm 1.
Read in the fossil data and assign scatterplot vectors to x and y:

```
fossil <- read.table("fossil.dat",header=T)
x <- fossil$age
y <- 100000*fossil$strontium.ratio
```

Set the value of the smoothing parameter (variance ratio) alpha:

```
alpha <- 2
```

Set up design matrices, for linear splines in this case.

```
n <- length(x)
X <- cbind(rep(1,n),x)
knots <- default.knots(x)
Z <- outer(x,knots,"-")
Z <- Z*(Z>0)
```

Set up input matrices for Algorithm 1.

```
C.mat <- cbind(X,Z)
D.mat <- diag(c(rep(0,ncol(X)),rep(1,ncol(Z))))
```

Carry out Steps 1 and 2 of Algorithm 1.

```
svd.C <- svd(C.mat)
U.C <- svd.C$u
V.C <- svd.C$v
d.C <- svd.C$d
svd.D <- svd(t(t(t(V.C)%*%D.mat%*%V.C/d.C)/d.C))
d.D <- svd.D$d
```

Obtain $\mathbf{A}$ matrix and $\mathbf{b}$ vector.

```
A.mat <- U.C%*%svd.D$u
b.vec <- as.vector(t(A.mat)%*%y)
```

Obtain vector of fitted values.

```
f.hat <- A.mat%*%(b.vec/(1+alpha*d.D))
```

Note that if a scatterplot smooth corresponding to a different value of α is required then only the last command needs to be re-issued.

A meaningful measurement of the amount of smoothing being done is the *degrees of freedom* (e.g. Hastie and Tibshirani, 1990) which we denote by df_{fit}. For p th degree polynomial regression $df_{\text{fit}} = p + 1$. The value of df_{fit} for penalized splines is a simple by-product of the above code:

```
df.fit <- sum(1/(1+alpha*d.D))
```

If the last two lines of code are re-run for three different values of α:

$$\alpha = 400, 1.6, 0.15$$

then the fits shown in Figure 3 result. These have df_{fit} values of 4,13 and 20 respectively.

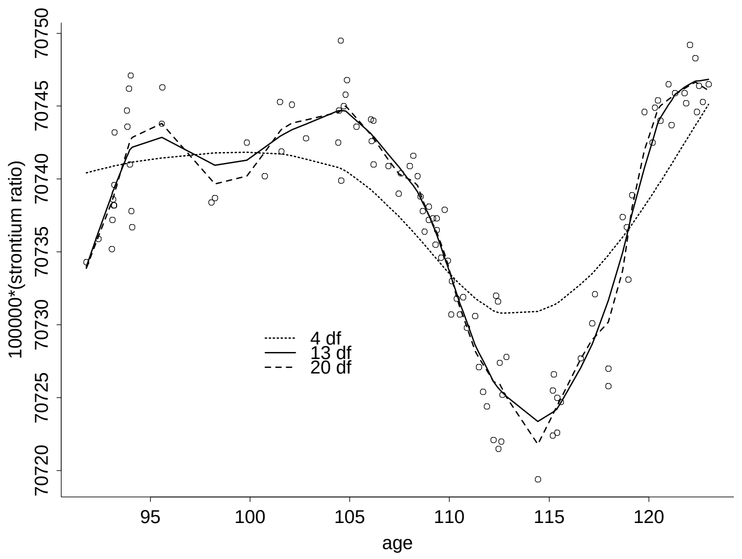


Figure 3: Linear penalized spline fit to the fossil data with differing degrees of freedom values.

9.3 SAS code

User specified smoothing parameter selection may be handled in SAS through the PARMS option. This is illustrated in the following SAS code. Notice the use of the PARMS option in body of the mixed model specification. The example is taken from section 2.2. Here the variance components are specified whose ratios are equal to the smoothing parameter values given in section 9.2. Note that if the degree of freedom is specified, then SAS/IML can be used to implement Algorithm 1 to obtain the estimate of the smoothing parameter. The last equation in Algorithm 1 can be solved by using the nonlinear procedure NLIN.

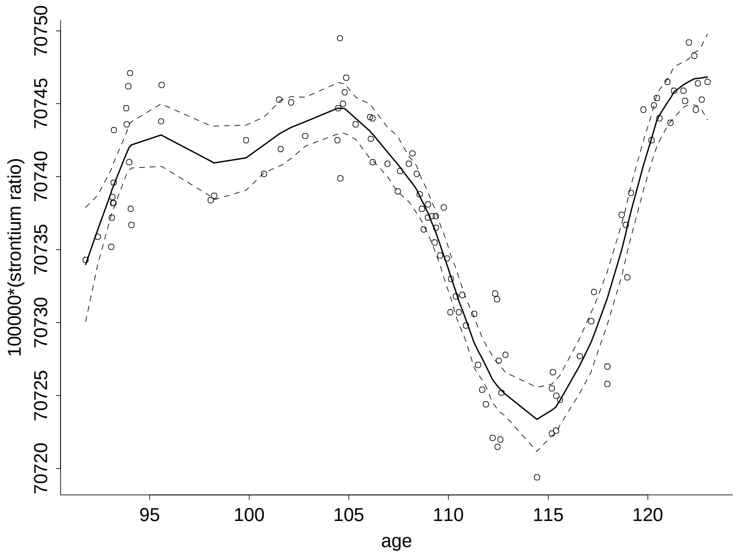
```
proc mixed noprofile; *noprofile stops the algorithm from profiling;
      *out the variance of the error term;
  model ratio = age / solution outp=paper.yhat;
  random  Z1-Z&nk / type=toep(1) s;
  *specifying residual and smoothing term variance components;
  *parms (400) (1)   / noiter; *noiter prevents Newton-Raphson iterative;
        *algorithm from changing variance components;
  *parms (3.2) (2)   / noiter;
```

```
   parms (15)  (100) / noiter;
run;
```

10 Variability Bars

A common embellishment to a scatterplot smooth such as the one shown in Figure 2 is to add variability bars, as shown in Figure 4.

Figure 4: Linear penalized spline fit to the fossil data with variability bar.



The dashed lines in Figure 4 correspond to plus and minus twice

$$\widehat{\text{st.dev.}}(\widehat{f}-\widehat{f}) = \widehat{\sigma}_\varepsilon\sqrt{\text{diagonal}\{(\mathbf{C}^\mathsf{T}\mathbf{C}+\alpha\mathbf{D})^{-1}\}}$$
$$= \widehat{\sigma}_\varepsilon\sqrt{\text{diagonal}\left\{\mathbf{A}\,\text{diag}\left(\frac{\mathbf{1}}{\mathbf{1}+\alpha\,\mathbf{d}_\text{D}}\right)\mathbf{A}^\mathsf{T}\right\}}$$

If α corresponds to the REML estimates of σ_u and σ_ε then $\widehat{\sigma}_\varepsilon$ can just be taken to be this REML estimate. For general α a reasonable estimate of σ_ε^2 is

$$\widehat{\sigma}_\varepsilon^2 = \text{RSS}(\alpha)/df_\text{res}(\alpha)$$

where $df_\text{res}(\alpha)$ can be computed as

$$df_\text{res}(\alpha) = n - 2\mathbf{1}^\mathsf{T}\left(\frac{\mathbf{1}}{\mathbf{1}+\alpha\,\mathbf{d}_\text{D}}\right) + \left\|\frac{\mathbf{1}}{\mathbf{1}+\alpha\,\mathbf{d}_\text{D}}\right\|^2.$$

10.1 S-PLUS commands

The following code computes lower and upper limits of variability bars:

```
RSS <- sum((y - f.hat)^2)
r.vec <- 1/(1+alpha*s.vec)
df.fit <- sum(r.vec)
df.res <- n - 2*df.fit + sum(r.vec^2)
sig.eps.hat <- sqrt(RSS/df.res)
st.dev.hat <- sig.eps.hat*sqrt(diag(A.mat%*%(r.vec*t(A.mat))))
var.bar.upp <- f.hat + 2*st.dev.hat
var.bar.low <- f.hat - 2*st.dev.hat
```

10.2 SAS code

The following SAS code shows the use of the `outp` option to obtain the standard error and the 95% confidence interval of the predicted value.

```
proc mixed;
  model ratio = age / solution outp=paper.yhat; *option outp gives the;
                                   *SE of the fitted for variability bar;
  random  Z1-Z&nk / type=toep(1) s;
run;
```

11 Extension to Other Bases

Up until now the only basis that has been used for mixed model-based penalized spline smoothing is the truncated line basis. For a predictor x this corresponds to the basis functions

$$1,\ x,\ (x-\kappa_1)_+, \ldots, (x-\kappa_K)_+. \tag{13}$$

We have done this to keep the presentation as simple as possible. Truncated line bases work reasonably well, but other bases have advantages such as smoothness and better handling of peaks and dips. An obvious extension of (13) is to use truncated polynomials of arbitrary degree p:

$$1,\ x, \ldots, x^p, \{(x-\kappa_1)_+\}^p, \ldots, \{(x-\kappa_K)_+\}^p.$$

Truncated polynomial bases are often scorned because of their numerical instability in regression settings. We have not found this to be a big problem in mixed model-based smoothing. One reason is that mixed model software transforms the basis functions internally to one that is more numerically stable (e.g. Pinheiro and Bates, 2000, Chapter 2). Algorithm 1 in Section 9 shows this phenomenon explicitly. The input matrix $\mathbf{C}$ corresponds to the truncated line basis, but it gets transformed to the design matrix $\mathbf{A}$ corresponding to the more stable Demmler-Reinsch basis. A second reason is that for $\alpha > 0$ the least squares problem gets replaced by a ridge regression problem which is usually more numerically stable (e.g. Draper and Smith, 1998).

An alternative to truncated polynomials with certain attractions are radial basis functions.

Penalised spline smoothers with radial bases, or *radial smoothers*, and their relationship to smoothing/thin plate splines and kriging are summarised in French, Kammann and Wand (2001). For $x_i \in \mathbb{R}$ a useful class of low-rank radial smoothers is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}_{\rm K}\mathbf{u} + \boldsymbol{\varepsilon}, \quad \text{Cov}(\mathbf{u}) = \sigma_u^2(\boldsymbol{\Omega}_{\rm K}^{-1/2})(\boldsymbol{\Omega}_{\rm K}^{-1/2})^{\sf T}$$

where $\mathbf{X} = [1\ x_i\ \dots\ x_i^{m-1}]_{1\leqslant i\leqslant n}$,

$$\mathbf{Z}_{\rm K} = [|x_i - \kappa_k|^{2m-1}]_{\substack{1\le i\le n\\ 1\le k\le K}} \quad \text{and} \quad \boldsymbol{\Omega}_{\rm K} = [|\kappa_k - \kappa_{k'}|^{2m-1}]_{1\le k,k'\le K}.$$

Using the transformation $\mathbf{Z} = \mathbf{Z}_{\rm K}\boldsymbol{\Omega}_{\rm K}^{-1/2}$ the model can be rewritten as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon}, \quad \text{Cov}\begin{bmatrix}\mathbf{u}\\ \boldsymbol{\varepsilon}\end{bmatrix} = \begin{bmatrix}\sigma_u^2\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \sigma_\varepsilon^2\mathbf{I}\end{bmatrix}. \tag{14}$$

This form allows fitting through standard mixed model software.

Note that

$$C(r) = (-1)^m|r|^{2m-1}$$

is a so-called *generalized covariance function* and could be replaced by any of the proper covariance functions used in kriging (e.g. Cressie 1993; O'Connell & Wolfinger 1997; Stein 1999).

11.1 S-PLUS code

Cubic radial basis functions can be used in `lme()` by setting up the `Z` matrix as follows:

```
svd.Omega <- svd(abs(outer(knots,knots,"-"))^3)
matrix.sqrt.Omega <- t(svd.Omega$v%*%(t(svd.Omega$u)*sqrt(svd.Omega$d)))
Z <- t(solve(matrix.sqrt.Omega,t(abs(outer(x,knots,"-")^3))))
```

11.2 SAS code

The following `SAS` code shows the use of `SAS/IML` to apply to the extension of other bases.

```
libname paper '˜/test';
data paper.fossil;
  infile '˜/test/fossil.dat' missover;
  input age ratio;
  ratio=ratio*100000;
  if age ne .;
run;

/*********************************/
/* calling macro to create knots */
/*********************************/

%include "default_knots.macro";
%default_knots(librefknots=paper,data=paper.fossil,
               knotdata=knots,varknots=age);
data dataw;
  set paper.fossil;
  m=1;
run;

data kt1;
  set paper.knots nobs=nk;
  call symput('nkt',nk);
  m=1;
run;
proc transpose data=paper.knots prefix=knots out=knotst;
  var knots;
run;
data paper.knotst;
  set knotst;
  m=1;
run;

/***********************************/
/* creating the Z(k) matrix        */
/***********************************/

data Zk;
  merge dataw paper.knotst;
  by m;
  %let nk=&nkt;
  array Z (&nk) Z1-Z&nk;
  array knots (&nk) knots1-knots&nk;
  do k=1 to &nk;
     Z(k)=(abs(age-knots(k)))**3;
  end;
  keep Z1-Z&nk;
run;

/************************************/
/* creating the O(k) matrix        */
/************************************/

data Ok;
  merge kt1 paper.knotst;
  by m;
  %let nk=&nkt;
  array O (&nk) O1-O&nk;
  array knotsa (&nk) knots1-knots&nk;
  do k=1 to &nk;
     O(k)=(abs(knots-knotsa(k)))**3;
  end;
  keep O1-O&nk;
run;

/************************************/
/* creating the Z matrix           */
/************************************/

proc iml;
  use Zk;
  read all var _num_ into Zk;
  use Ok;
  read all var _num_ into Ok;
  call svd(u,d,v,Ok);
  sqrtOk=u*sqrt(diag(d))*v‘;
  Z=Zk*inv(sqrtOk);
    create Z from Z[colname={col1 col2 col3 col4 col5 col6 col7
                       col8 col9 col10 col11 col12
                       col13 col14 col15 col16 col17 col18
                       col19 col20 col21}];
  append from Z;
quit;
run;

data dataw2;
  merge dataw Z;
run;

ods output CovParms=paper.varcomp;
```

```
/********************************/
/* fitting the mixed model      */
/********************************/

proc mixed;
  model ratio = age / solution outp=paper.yhat;
  random  COL1-COL&nk / type=toep(1) s;
run;

/********************************/
/* plotting the smoothed curve  */
/********************************/

proc sort;
  by age;
run;
symbol1 v=circle c=black i=j l=1;
symbol2 v=point  c=blue  i=j l=2;
goptions device=xcolor;
proc gplot;
  plot ratio*age pred*age / overlay;
run;
```

12 Multivariate Smoothing

For $\mathbf{x}_i \in \mathbb{R}^d$, $1 \leqslant i \leqslant n$, and $\boldsymbol{\kappa}_k \in \mathbb{R}^d$, $1 \leqslant k \leqslant K$, then higher dimension approximate smoothing splines (also called *thin plate* splines) with smoothness parameter m can be obtained by taking $\mathbf{X}$ to have columns spanning the space of all d-dimensional polynomials in the components of $\mathbf{x}_i$ with degree less than m and

$$\mathbf{Z} = [C(\|\mathbf{x}_i - \boldsymbol{\kappa}_k\|)]_{\substack{1\leqslant i\leqslant n \\ 1\leqslant k\leqslant K}} [C(\|\boldsymbol{\kappa}_k - \boldsymbol{\kappa}_{k'}\|)]_{1\leqslant k,k'\leqslant K}^{-1/2}$$

where

$$C(\mathbf{r}) = \begin{cases} \|\mathbf{r}\|^{2m-d} & d \text{ odd} \\ \|\mathbf{r}\|^{2m-d}\log\|\mathbf{r}\| & d \text{ even} \end{cases}$$

(e.g. Nychka, 2000).

Alternatively, $C(\cdot)$ could be a covariance function such as those used in kriging (e.g. Cressie 1993; O'Connell & Wolfinger 1997; Stein 1999).

The choice of the bivariate knots $\boldsymbol{\kappa}_k$, $1 \leqslant k \leqslant K$, is somewhat more challenging. We have had good experience with knots chosen via an efficient space filling algorithm (e.g. Johnson, Moore and Ylvisaker, 1990; Nychka and Saltzman, 1998). The

S-PLUS module FUNFITS (Nychka, Haaland, O'Connell and Ellner, 1998) supports space filling algorithms.

Figure 5 shows the result of applying such an algorithm to the (jittered) locations in the example used by Kammann and Wand (2003) for $d = 2$.

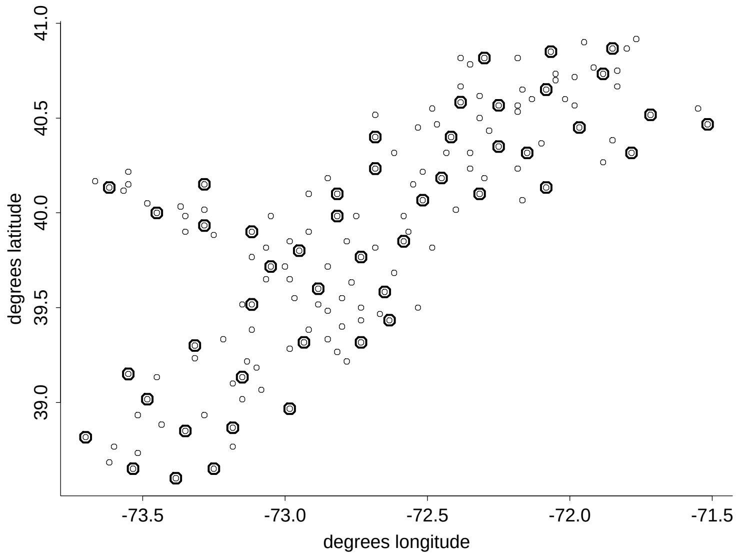


Figure 5: The smaller dots correspond to the geographical locations in the scallop reproductive data, with jittering to protect identity. The circled dots correspond to a representative subset of 48 locations for performing radial penalized spline smoothing.

12.1 S-PLUS commands

We will now illustrate mixed model-based bivariate smoothing using thin plate splines with $m = 2$ using lme() in S-PLUS.

First, define the function tps.cov() corresponding to the thin plate spline generalised covariance function

$$C(r) = r^2 \log|r|.$$

The function is a bit more complicated so that zero arguments and matrix and vector arguments are handled.

```
tps.cov <- function(r)
{
    r <- as.matrix(r)
    num.row <- nrow(r)
    num.col <- ncol(r)
    r <- as.vector(r)
    nzi <- (1:length(r))[r!=0]
    ans <- rep(0,length(r))
    ans[nzi] <- r[nzi]^2*log(abs(r[nzi]))
    if (num.col>1) ans <- matrix(ans,num.row,num.col)
    return(ans)
}
```

Set the point cloud variables to be smoothed.

```
scallop <- read.table("scallop.dat",header=T)
x1 <- scallop$lon
x2 <- scallop$lat
y <- log(scallop$tcatch + 1)
```

Read in the knots from a file. These were created using a space-filling algorithm.

```
knots <- as.matrix(read.table("scallop.knots",header=T))
K <- nrow(knots)
```

Set up the design matrices corresponding to a plane for `X` and thin plate spline basis functions for `Z`.

```
X <- cbind(rep(1,length(y)),x1,x2)

dist.mat <- matrix(0,K,K)
dist.mat[lower.tri(dist.mat)] <- dist(knots)
dist.mat <- dist.mat + t(dist.mat)

Omega <- tps.cov(dist.mat)

diffs.1 <- outer(x1,knots[,1],"-")
diffs.2 <- outer(x2,knots[,2],"-")
dists <- sqrt(diffs.1ˆ2+diffs.2ˆ2)

svd.Omega <- svd(Omega)
sqrt.Omega <- t(svd.Omega$v %*% (t(svd.Omega$u) * sqrt(svd.Omega$d)))
Z <- t(solve(sqrt.Omega,t(tps.cov(dists))))
```

Obtain the bivariate smooth using `lme()` and extract the coefficients.

```
fit <- lme(y˜-1+X,random=pdIdent(˜-1+Z))

beta.hat <- fit$coef$fixed
u.hat <- unlist(fit$coef$random)
```

12.2 SAS code

The following SAS code fits bivariate smoothing for the above model.

```
libname paper '˜/test';
data paper.scallop;
  infile '˜/test/scallop.dat' missover;
  input strata sample lat long tcatch prerec recruits;
  y=log(tcatch+1);
  m=1;
  if strata ne .;
  keep y lat long m;
run;
/*******************************************************/
/*Read in the knots data -- as used in the Splus module*/
/*******************************************************/
data knots;
  infile '~/test/scallop.knots' missover;
  input x1 x2;
  if x1 ne .;
run;
data knots;
  set knots nobs=nk;
  call symput('nkt',nk);
run;
%let numknots=&nkt;
proc transpose out=t1;
  var x1 x2;
run;
/********************************************************/
/*Compute the matrix Omega                             */
/********************************************************/
data d1 (keep=i j xt1) d2 (keep=xt2);
  set t1;
  array da (&numknots) col1-col&numknots;
  do i=1 to &numknots-1;
     do j=i+1 to &numknots;
        if _name_='x1' then do;
           xt1=(da(j)-da(i))**2;
           output d1;
        end; else
        if _name_='x2' then do;
           xt2=(da(j)-da(i))**2;
           output d2;
        end;
      end;
   end;
run;
data e1;
  merge d1 d2;
  dist=sqrt(xt1+xt2);
  omegaelm=dist*dist*log(dist);
  keep i j omegaelm;
run;
/********************************************************/
/*Construct the Zk matrix                               */
/********************************************************/
data t1a;
  set t1;
  if _name_='x1';
  m=1;
  drop _name_;
run;
data diffs1;
  merge paper.scallop (keep=long m) t1a;
  by m;
  array cola (&numknots) col1-col&numknots;
  array z1a  (&numknots) z1_1-z1_&numknots;
  do i=1 to &numknots;
     z1a(i)=long-cola(i);
  end;
  keep z1_1-z1_&numknots;
run;
data t2a;
  set t1;
  if _name_='x2';
  m=1;
  drop _name_;
run;
data diffs2;
  merge paper.scallop (keep=lat m) t2a;
  by m;
  array cola (&numknots) col1-col&numknots;
  array z2a  (&numknots) z2_1-z2_&numknots;
  do i=1 to &numknots;
     z2a(i)=lat-cola(i);
  end;
  keep z2_1-z2_&numknots;
run;
data dists;
  merge diffs1 diffs2;
  array z1a (&numknots) z1_1-z1_&numknots;
  array z2a (&numknots) z2_1-z2_&numknots;
  array dista (&numknots) dist1-dist&numknots;
  do i=1 to &numknots;
     temp1=sqrt(z1a(i)**2+z2a(i)**2);
     if temp1=0 then dista(i)=0; else
     dista(i)=(temp1**2)*log(temp1);
  end;
  keep dist1-dist&numknots;
run;
/************************************************************/
/*Construct the Z matrix from the Omega and Zk matrix     */
/************************************************************/
proc iml;
  use e1;
  read all var _num_ into e1;
  omega=j(&numknots,&numknots,0);
  do i=1 to (&numknots*(&numknots-1))/2;
    omega[e1[i,1],e1[i,2]]=e1[i,3];
    omega[e1[i,2],e1[i,1]]=e1[i,3];
  end;
  call svd(u,d,v,omega);
  sqrtomega=u*sqrt(diag(d))*v‘;
  use dists;
  read all var _num_ into Zk;
  Z=Zk*inv(sqrtomega);
  create Z from Z[colname={col1 col2 col3 col4 col5 col6 col7
                           col8 col9 col10 col11 col12
                           col13 col14 col15 col16 col17 col18
                           col19 col20 col21 col22 col23 col24 col25
                           col26 col27 col28 col29 col30 col31 col32
                           col33 col34 col35 col36 col37 col38 col39
                           col40 col41 col42 col43 col44 col45 col46
                           col47 col48}];
  append from Z;
quit;
data dataw2;
  merge paper.scallop Z;
run;
ods output CovParms=paper.varcomp;
/**********************************/
/* fitting the mixed model      */
/**********************************/
proc mixed;
  model y = long lat / solution outp=paper.yhat;
  random  col1-col&numknots / type=toep(1) s;
run;
```

13 Generalized models

The extension to generalized responses, such as binary and count variables, entails *generalized* mixed models. The most common is the *generalized linear mixed model* (GLMM) corresponding to the one-parameter exponential family and Gaussian random effects, for which

$$f(\mathbf{y}|\mathbf{u}) = \exp\{\mathbf{y}^{\mathsf{T}}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}) - \mathbf{1}^{\mathsf{T}}b(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}) + \mathbf{1}^{\mathsf{T}}c(\mathbf{y})\}$$

is the density of $\mathbf{y}$ given $\mathbf{u}$ and

$$\mathbf{u} \sim N(\mathbf{0}, \mathbf{G}).$$

The logistic-normal mixed model corresponds to

$$b(x) = \log(1 + e^x), \quad c(x) = 0$$

while the Poisson-normal mixed model corresponds to

$$b(x) = e^x, \quad c(x) = -\log(x!).$$

A very common extension is to allow for quasi-likelihood functions (e.g. Breslow and Clayton, 1993) McCulloch and Searle (2000) provides an excellent overview of GLMMs.

Fitting generalized linear mixed models is much more computationally challenging than the linear case (e.g. McCulloch and Searle, Chapter 10). The only software known to us for fitting GLMMs *but with the provision for general design matrices as needed for smoothing* is the `SAS` macro `glimmix`; although this relies on Laplace approximation of integrals (e.g. Wolfinger and O'Connell, 1993). We have recently learned from John Staudenmayer (University of Massachusetts) that the `R` version of `lme()` can be used to emulate `glimmix` because it allows for weights. Section 13.2 illustrates this for smoothing the scatterplot shown in Figure 6.

For a user specified degrees of freedom smoothing reduces to iteratively reweighted least squares ridge regression.

13.1 `S-PLUS` commands

Read in data and assign regression vectors and knots:

```
trade.union <- read.table("tradeunion.dat",header=T)

x <- trade.union$wage
y <- trade.union$union.member

knots <- default.knots(x)
n <- length(y)
```

Set the smoothing parameter:

```
alpha <- 1000
```

Set the design matrices for quadratic penalized splines:

```
X <- cbind(rep(1,n),x,x^2)

Z <- outer(x,knots,"-")
Z <- Z*(Z>0)
Z <- Z^2

C.mat <- cbind(X,Z)
D.mat <- diag(c(rep(0,ncol(X)),rep(1,ncol(Z))))
```

Find an initial estimate based on an ordinary ridge regression fit using Algorithm 1:

```
svd.C <- svd(C.mat)

U.C <- svd.C$u
V.C <- svd.C$v
d.C <- svd.C$d

svd.D <- svd(t(t(t(V.C)%*%D.mat%*%V.C/d.C)/d.C))

d.D <- svd.D$d

A.mat <- U.C%*%svd.D$u

b.vec <- as.vector(t(A.mat)%*%y)

eta.hat <- A.mat%*%(b.vec/(1+alpha*d.D))
```

Now do iteratively reweighted penalized fits

```
desired.accuracy <- 0.001

rel.error <- desired.accuracy+1
max.iter <- 100

iter.num <- 0
while((rel.error>desired.accuracy)&(iter.num<max.iter))
{
   eta.hat.old <- eta.hat

   wt.vec <- as.vector(exp(eta.hat)/((1+exp(eta.hat))^2))
   Cw.mat <- C.mat*sqrt(wt.vec)
   y.adj <- eta.hat + (y-f.hat)/wt.vec
   svd.C <- svd(Cw.mat)

   U.C <- svd.C$u
   V.C <- svd.C$v
   d.C <- svd.C$d

   svd.D <- svd(t(t(t(V.C)%*%D.mat%*%V.C/d.C)/d.C))
   d.D <- svd.D$d

   A.mat <- U.C%*%svd.D$u/sqrt(wt.vec)
   b.vec <-  as.vector(t(A.mat)%*%(wt.vec*y.adj))

   eta.hat <- A.mat%*%(b.vec/(1+alpha*d.D))

   rel.error <- sum(abs(eta.hat-eta.hat.old))/sum(abs(eta.hat.old))

   iter.num <- iter.num + 1
}
```

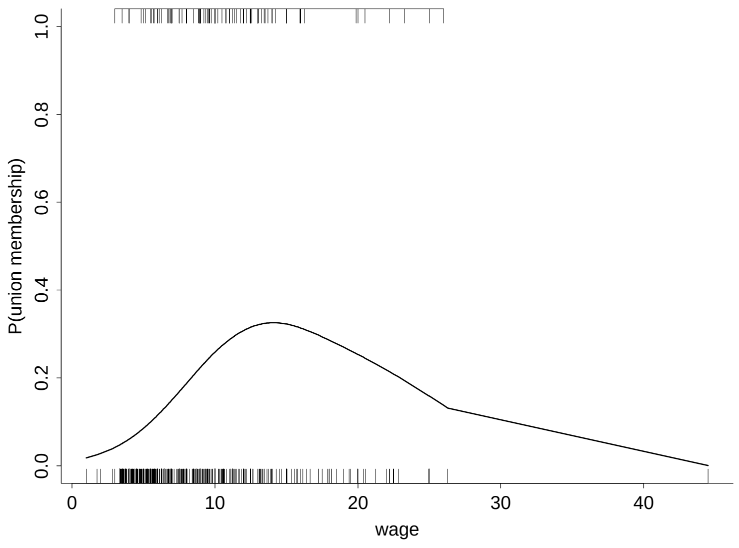


Figure 6: Quadratic spline fit to the traded union data.

13.2 SAS code

The following SAS code fits the trade union data using the SAS macro `glimmix`.

```
libname paper '~/test';
data paper.tradeunion;
```

```
  infile '˜/test/tradeunion.dat' missover;
  input yearedu south female yearsexpe member wage age
        race occupation sector
        married;
  m=1;
  wage2=wage**2;
  if member ne .;
  keep wage wage2 member m;
run;
/*********************************************/
/* creating knots for the smoothing variable */
/*********************************************/

options mprint;

%include "default_knots.macro";
%default_knots(librefknots=paper,data=paper.tradeunion,
               knotdata=knots,varknots=wage,numknots=35);

data dataw;
  set paper.tradeunion;
  m=1;
run;
data kt;
  set paper.knots nobs=nk;
  call symput('nkt',nk);
run;
proc transpose data=paper.knots prefix=knots out=knotst;
  var knots;
run;

data paper.knotst;
  set knotst;
  m=1;
run;


/***********************************/
/* creating the Z matrix           */
/***********************************/

data dataw;
  merge dataw paper.knotst;
  by m;
  %let nk=&nkt;
  array Z  (&nk) Z1-Z&nk;
  array knots (&nk) knots1-knots&nk;
  do k=1 to &nk;
     Z(k)=wage-knots(k);
     if Z(k) < 0 then Z(k)=0;
     Z(k)=Z(k)**2;
  end;
  drop knots1-knots&nk _name_;
run;
/***********************************************/
/*Fit generalized linear mixed model          */
/***********************************************/

%include ’glimmix.sas’;
%glimmix(data=dataw,procopt=method=reml,
         stmts=%str(
                    model member=wage wage2 / solution;
                    random Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9 Z10
                     Z11 Z12 Z13 Z14 Z15 Z16 Z17 Z18 Z19 Z20
                     Z21 Z22 Z23 Z24 Z25 Z26 Z27 Z28 Z29 Z30
                     Z31 Z32 Z33 Z34 Z35
                     / type=toep(1) s;),
                    error=binomial,
                    link=logit,
                    out=fitted);

proc print data=fitted;
var mu;
title ’Fitted Probabilities’;
run;
```

14 Plotting Issues

In the mixed model approach to smoothing the estimate may be plotted over a grid (or mesh in two dimensions) of arbitrarily fine resolution once the effects estimates $\widehat{\boldsymbol{\beta}}$ and $\widehat{\mathbf{u}}$ have been computed. In this section we provide some of the computational details.

14.1 Univariate plots

Suppose that the fossil data are smoothed using the truncated quadratic basis

$$1, x, x^2, \{(x-\kappa_1)_+\}^2, \ldots, , \{(x-\kappa_K)_+\}^2.$$

Let $\boldsymbol{g} = [g_1, \ldots, g_M]^\mathsf{T}$ denote a vector of grid points over which a plot of the fit is desired. Then set up the "grid-wise" design matrices

$$\mathbf{X}_{\boldsymbol{g}} = [1 \; g_\ell \; g_\ell^2], \qquad \mathbf{Z}_{\boldsymbol{g}} = [\{(g_i - \kappa_k)_+\}^2]_{\substack{1 \leqslant \ell \leqslant M \\ 1 \leqslant k \leqslant K}}.$$

The grid of $\widehat{f}(g_\ell)$ values, $1 \leqslant \ell \leqslant M$, is then

$$\widehat{f}_{\boldsymbol{g}} = \mathbf{X}_{\boldsymbol{g}} \widehat{\boldsymbol{\beta}} + \mathbf{Z}_{\boldsymbol{g}} \widehat{\mathbf{u}}.$$

The following `S-PLUS` code illustrates this for the fossil data:

```
x <- fossil$age
y <- fossil$strontium.ratio
knots <- default.knots(x)
n <- length(x)
X <- cbind(rep(1,n),x,xˆ2)
Z <- outer(x,knots,"-")
Z <- Z*(Z>0)
Z <- Zˆ2
fit <- lme(y˜-1+X,random=pdIdent(˜-1+Z))
beta.hat <- fit$coef$fixed
u.hat <- unlist(fit$coef$random)

num.grid <- 401
x.grid <- seq(min(x),max(x),length=num.grid)
X.grid <- cbind(rep(1,num.grid),x.grid,x.gridˆ2)
Z.grid <- outer(x.grid,knots,"-")
Z.grid <- Z.grid*(Z.grid>0)
Z.grid <- Z.gridˆ2
fhat.grid <- X.grid%*%beta.hat + Z.grid%*%u.hat

plot(x,y,pch=1)
lines(x.grid,fhat.grid)
```

14.2 Bivariate plots

Bivariate plotting is much more delicate. When using `S-PLUS` our preferred approach is through image plots, but it is recommended that the pixels corresponding to locations outside the range of the data be switched off. The following `S-PLUS` code illustrates this for the scallop data.

Obtain the surface estimate over a bivariate pixel mesh.

```
x1.grid <- seq(min(x1),max(x1),length=64)
x2.grid <- seq(min(x2),max(x2),length=64)
mesh <- expand.grid(x1.grid,x2.grid)
x1.mesh <- mesh[,1] ; x2.mesh <-  mesh[,2]

diffs.1 <- outer(x1.mesh,knots[,1],"-")
diffs.2 <- outer(x2.mesh,knots[,2],"-")

dists <- sqrt(diffs.1^2+diffs.2^2)

X.mesh <- as.matrix(cbind(rep(1,nrow(mesh)),mesh))
Z.mesh <-  t(solve(sqrt.Omega,t(tps.cov(dists))))

f.hat <- X.mesh%*%beta.hat + Z.mesh%*%u.hat
```

The remaining commands should produce an image plot of the surface estimate and show you the best places to fish for scallops! Note that only those pixels where there are data are switched on. Using the controls in the `motif` window, pick a colour scheme appropriate for image plots.

```
on.pixels <- scan("scallop.pixels")
f.hat[on.pixels==0] <- NA

f.hat.mat <- matrix(f.hat,64,64)

x1.width <- x1.grid[2] - x1.grid[1]
x1.frame <- c(x1.grid-x1.width/2,x1.grid[64]+x1.width/2)
x2.width <- x2.grid[2] - x2.grid[1]
x2.frame <- c(x2.grid-x2.width/2,x2.grid[64]+x2.width/2)

par(mfrow=c(1,1))
image(x1.frame,x2.frame,f.hat.mat,bty="l",
      xlab="degrees longitude",ylab="degrees latitude")
```

15 Closing remarks

The ability to fit smoothing-based models with mixed model software is an exciting development and can only lead to more widespread use and less time spent coding. For example, the analyses in Kammann and Wand (2002) were done entirely using `lme()` despite the complexity of the modelling. This paper has focussed chiefly on the case where normality of the response is reasonably assumed. The extension to generalized linear mixed models is the focus of ongoing research.

Appendix: Obtaining and running the code

The code in this paper is stored in ordinary text files that may be downloaded from the `Internet`. Several auxiliary files (e.g. those containing data sets) are also posted. The name of each code file corresponds to the section number. For example, the `S-PLUS` code in Section 6.1 is stored in the file `sec6.1.S` and the `SAS` code in Section 6.2 is stored in the file `sec6.2.sas`. In most cases, successful running of the `S-PLUS` code will require other files to be available to the current session. For example, to `sec7.1.S` requires the data file `milanmort.dat` to be available to the current session. Many of the `S-PLUS` scripts use the function `default.knots()` stored in `default.knots.sf`. The current location of the files is

`http://www.maths.unsw.edu.au/ ~ wand/papers.html`.

Acknowledgments

We are grateful for advice from Maria Durban, Mary Lindstrom, José Pinheiro and John Staudenmayer. This research was supported by US National Institutes of Health grant T32 ES07142-18.

References

Bowman, A.W. and Azzalini, A. (1997). *Applied Smoothing Techniques for Data Analysis*. Oxford: Clarendon Press.

Breslow, N.E. and Clayton, D.G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**, 9–25.

Brumback, B.A., Ruppert, D. and Wand, M.P. (1999). Comment on Shively, Kohn and Wood. *Journal of the American Statistical Association*, **94**, 794–797.

Chaudhuri, P. and Marron, J.S. (1999). SiZer for exploration of structures in curves. *Journal of the American Statistical Association*, **94**, 807–823.

Coull, B.A., Ruppert, D. and Wand, M.P. (2001). Simple incorporation of interactions into additive models. *Biometrics*, **57**, 539–545.

Cressie, N. (1993). *Statistics for Spatial Data.* New York: John Wiley & Sons.

Diggle, P., Liang, K.-L. and Zeger, S. (1995). *Analysis of Longitudinal Data*. Oxford: Oxford University Press.

Draper, N.R. and Smith, H. (1998). *Applied Regression Analysis (Third Edition).* New York: John Wiley & Sons.

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

French, J.L., Kammann, E.E. and Wand, M.P. (2001). Comment on Ke and Wang. *Journal of the American Statistical Association*, **96**, 1285–1288.

Golub, G.H. and Van Loan, C.F. (1983). *Matrix Computations (Third Edition)*. Baltimore: John Hopkins University Press.

Gray, R. J. (1992). Spline-based tests in survival analysis. *Biometrics*, **50**, 640–652.

Green, P.J. (1987). Penalized likelihood for general semi-parametric regression models. *International Statistical Review*, **55**, 245–259.

Kelly, C. and Rice, J. (1990). Monotone smoothing with application to dose-response curves and the assessment of synergism. *Biometrics*, **46**, 1071–1085.

Härdle, W., Liang, H. and Gao, J. (2000). *Partially Linear Models*. Heidelberg: Physica-Verlag.

Hastie, T.J. (1996). Pseudosplines. *Journal of the Royal Statistical Society, Series B*, **58**, 379–396.

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

Johnson, M.E., Moore, L.M. and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, **26**, 131–148.

Kammann, E.E. and Wand, M.P. (2003). Geoadditive models. *Applied Statistics*, **52**, 1–18.

Littell, R.C., Milliken, G.A., Stroup, W.W., Wolfinger, R.D. (1996). *SAS System for Mixed Models*, Cary, NC: SAS Institute Inc., 1996. 633 pp.

McCulloch, C.E., and Searle, S.R. (2000). *Generalized, Linear, and Mixed Models*. New York: John Wiley & Sons.

Nychka, D. and Saltzman, N. (1998). Design of Air Quality Monitoring Networks. In *Case Studies in Environmental Statistics Nychka* (D. Nychka, Cox, L., Piegorsch,

W. eds.), Lecture Notes in Statistics, Springer-Verlag, 51–76.

Nychka, D., Haaland, P., O'Connell, M., Ellner, S. (1998). FUNFITS, data analysis and statistical tools for estimating functions. In *Case Studies in Environmental Statistics* (D. Nychka, W.W. Piegorsch, L.H. Cox, eds.), New York: Springer-Verlag, 159–179.

Nychka, D.W. (2000). Spatial process estimates as smoothers. In *Smoothing and Regression* (M. Schimek, ed.). Heidelberg: Springer-Verlag.

O'Connell, M.A. and Wolfinger, R.D. (1997). Spatial regression models, response surfaces, and process optimization. *Journal of Computational and Graphical Statistics*, **6**, 224–241.

O'Sullivan, F. (1986). A statistical perspective on ill-posed inverse problems (with discussion). *Statistical Science*, **1**, 505–527.

O'Sullivan, F. (1988). Fast computation of fully automated log-density and log-hazard estimators. *SIAM Journal on Scientific and Statistical Computing*, **9**, 363–379.

Parker, R.L. and Rice, J.A. (1985). Discussion of "Some aspects of the spline smoothing approach to nonparametric curve fitting" by B.W. Silverman. *Journal of the Royal Statistical Society, Series B*, **47**, 40-42.

Pinheiro, J.C. and Bates, D.M. (2000). *Mixed-Effects Models in S and S-PLUS*. New York: Springer.

Ratkowsky, D. A. (1983). *Nonlinear Regression Modeling: A Unified Practical Approach.* New York: Marcel Dekker.

Robinson, G.K. (1991). That BLUP is a good thing: the estimation of random effects. *Statistical Science*, **6**, 15–51.

Ruppert, D. (2002). Selecting the number of knots for penalized splines. *Journal of Computational and Graphical Statistics*, in press.

Ruppert, D. and Carroll, R.J. (2000). Spatially-adaptive penalties for spline fitting. *Australian and New Zealand Journal of Statistics*, **42**, 205–224.

Ruppert, D., Wand, M. P. and Carroll, R.J. (2003). *Semiparametric Regression*. New York: Cambridge University Press.

Speed, T. (1991). Comment on paper by Robinson. *Statistical Science*, **6**, 42–44.

Stein, M.L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. New York: Springer.

Verbyla, A.P. (1994). Testing linearity in generalized linear models. *Contributed Pap. 17th Int. Biometric Conf., Hamilton, Aug. 8th-12th*, 177.

Wahba, G. (1978). Improper priors, spline smoothing and the problem of guarding against model errors in regression. *Journal of the Royal Statistical Society, Series B*, **40**, 364-372.

Wand, M. P. (2003). Smoothing and mixed models. *Computational Statistics*, to appear.

Wolfinger, R. and O'Connell, M. (1993). Generalized linear mixed models: a pseudo-likelihood approach. *Journal of Statistical Computation and Simulation*, **48**, 233–243.

Young, S. G. and Bowman, A. W. (1995). Non-parametric analysis of covariance. *Biometrics*, **51**, 920–931.

Zanobetti, A., Wand, M.P., Schwartz, J. and Ryan, L.M. (2000). Generalized additive distributed lag models. *Biostatistics*, **1**, 279–292.