MINIMIZING GCV/GML SCORES WITH MULTIPLE SMOOTHING PARAMETERS VIA THE NEWTON METHOD*

CHONG GU† AND GRACE WAHBA‡

Abstract. The (modified) Newton method is adapted to optimize generalized cross validation (GCV) and generalized maximum likelihood (GML) scores with multiple smoothing parameters. The main concerns in solving the optimization problem are the speed and the reliability of the algorithm, as well as the invariance of the algorithm under transformations under which the problem itself is invariant. The proposed algorithm is believed to be highly efficient for the problem, though it is still rather expensive for large data sets, since its operational counts are \((2/3)kn^3 + O(n^2)\), with \(k\) the number of smoothing parameters and \(n\) the number of observations. Sensible procedures for computing good starting values are also proposed, which should help in keeping the execution load to the minimum possible. The algorithm is implemented in Rkpack [RKPACK and its applications: Fitting smoothing spline models, Tech. Report 857, Department of Statistics, University of Wisconsin, Madison, WI, 1989] and illustrated by examples of fitting additive and interaction spline models. It is noted that the algorithm can also be applied to the maximum likelihood (ML) and the restricted maximum likelihood (REML) estimation of the variance component models.

Key words. additive/interaction spline models, gradient, Hessian, invariance, Newton method, smoothing parameters, starting values

AMS(MOS) subject classifications. 65D07, 65D10, 65D15, 65K10, 65V05

1. Introduction. Suppose we observe

\[y_j = L_j f + e_j \quad j = 1, \ldots, n,\]

where the \(L_j\)'s are bounded linear functionals in a Hilbert space \(\mathcal{H}\), and the \(e_j\)'s are independently and identically distributed Gaussian noise with possibly unknown variance \(\sigma^2\). Such setups encompass a broad range of smoothing and indirect sensing problems. The solution \(f_\lambda\) to the variational problem

\[
\min \frac{1}{n} \sum_{j=1}^{n} (y_j - L_j f)^2 + \lambda \|P_1 f\|^2
\]

is called a spline in a general sense (see [14]), where \(P_1\) is a projection operator to a subspace \(\mathcal{H}_1\) with codimension \(M\), \(\| \cdot \|\) is the norm in \(\mathcal{H}\), and \(\lambda\) is the so-called smoothing parameter. The parameter \(\lambda\) controls the trade-off between the residual sum of squares \(\sum_{j=1}^{n} (y_j - L_j f_\lambda)^2\) and the penalty \(\|P_1 f_\lambda\|^2\). \(f_\lambda\) is called a regularized estimate of \(f\) in the literature of ill-posed problems, and \(\lambda\) the regularization parameter. See, e.g., [17] and references cited therein.

Let \(\xi_j\) denote the representer of \(L_j\). It is derived by Kimeldorf and Wahba [14] that \(f_\lambda = \sum_{j=1}^{n} c_j (P_1 \xi_j) + \sum_{\nu=1}^{M} d_\nu \phi_\nu\), where \(\{\phi_\nu\}_{\nu=1}^{M}\) span \(\mathcal{H}_0\), the null space of \(P_1\).

\(c = (c_1, \ldots, c_n)\); \(d = (d_1, \ldots, d_M)\) are solutions to the minimization problem

\[
\min \frac{1}{n} \|y - Sd - \tilde{Q}c\|^2 + \lambda c^T \tilde{Q}c,
\]
where \( \tilde{Q} = (\langle P_1 \xi_i, P_1 \xi_j \rangle) \) and \( S = (L_0 \phi_i) \), with \( \langle \cdot, \cdot \rangle \) indicating the inner product in \( \mathcal{H} \). It can be shown (see [21]) that the solution to the linear system

\[
(\tilde{Q} + nI)c + Sd = y,
\]

is a minimizer of (2), and when \( \tilde{Q} \) is of full rank, it is the unique minimizer.

Consider an orthogonal decomposition of \( \mathcal{H} \) into more than two components, \( \mathcal{H} = \bigoplus_{i=0}^{k} \mathcal{H}_i \). A direct generalization of (1) is

\[
\min \frac{1}{n} \sum_{j=1}^{n} (y_j - L_jf)^2 + \sum_{i=1}^{k} \lambda_i \|P_i f\|^2,
\]

where the \( \lambda_i \)'s are a set of smoothing parameters and \( P_i \) is the orthogonal projection operator onto \( \mathcal{H}_i \). Writing \( \lambda_i = \lambda / \theta_i \), we can rewrite (4) as

\[
\min \frac{1}{n} \sum_{j=1}^{n} (y_j - L_jf)^2 + \lambda \|P_i f\|^2_\theta,
\]

where \( P_i = \sum_{j=1}^{k} P_i \) is the projection operator onto \( \mathcal{H}_i = \bigoplus_{j=0}^{k} \mathcal{H}_i \), and \( \|f\|^2_\theta = \|P_i f\|^2 + \sum_{j=1}^{k} \theta_j \|P_j f\|^2 \) is a modified norm indexed by \( \theta \), where \( \|\cdot\| \) is the original norm. It can be shown that the representer of \( L_f \) under the norm \( \|\cdot\|_\theta \) is \( \xi_\theta = G(P_0 \xi) + \sum_{i=1}^{k} \theta_i (P_i \xi) \), where \( \xi_i \) is its representer under the norm \( \|\cdot\| \). Denoting \( \langle \cdot, \cdot \rangle_\theta \) as the inner products corresponding to the norms \( \|\cdot\|, \|\cdot\|_\theta \), respectively, we have \( \tilde{Q}_\theta = (\langle P_0 \xi, P_0 \xi \rangle_\theta) = \sum_{i=1}^{k} \theta_i \tilde{Q}_i \), where \( \tilde{Q}_i = (\langle P_i \xi, P_i \xi \rangle) \). Thus the solution to (4) can be written as \( f_\lambda = \sum_{j=1}^{n} \xi_j \xi_j + \sum_{j=1}^{M} d_j \phi_j \), with \( c, d \) determined by

\[
\min \frac{1}{n} \|y - Sd - \tilde{Q}_\theta c\|^2 + \lambda c^T \tilde{Q}_\theta c.
\]

Choosing appropriate smoothing parameters \( \lambda \) is crucial for effectively recovering truth functions from the data by fitting spline models. Two of the commonly recognized data-driven methods for choosing smoothing parameters are the generalized cross-validation (GCV) and the generalized maximum likelihood (GML) methods. Writing \( \tilde{Q}_\theta c + Sd = (L_1 f_\lambda, \cdots, L_n f_\lambda)^T = A(\lambda) y \), the GCV method seeks \( \lambda \) to minimize

\[
V(\lambda) = \frac{(1/n) \| (I - A(\lambda)) y \|^2}{(1/n) \text{tr} (I - A(\lambda))^2}.
\]

\( A(\lambda) \) is the so-called influence matrix. Letting

\[
S = FR = (F_1, F_2) \begin{pmatrix} R_1 \\ 0 \end{pmatrix}
\]

be the QR-decomposition of \( S \), it can be shown (see [21]) that \( I - A(\lambda) = n\lambda F_2 (\tilde{Q}_\theta F_2 + n\lambda I)^{-1} F_2^T \), hence

\[
V(\lambda) = V(\lambda, \theta) = n z^T (Q_\theta^T + n\lambda I)^{-2} z \left[ \frac{\text{tr} (Q_\theta^T + n\lambda I)^{-1}}{\text{tr} (Q_\theta^T + n\lambda I)^{-2}} \right]^2,
\]

where \( z = F_2^T y \) and \( Q_\theta = F_2^T \tilde{Q}_\theta F_2 = \sum_{i=1}^{k} \theta_i (F_2^T \tilde{Q}_i F_2) = \sum_{i=1}^{k} \theta_i Q_i \), where \( Q_i = F_2^T \tilde{Q}_i F_2 \). The GCV method is proposed by Craven and Wahba [6] and is shown to be asymptotically optimum for minimizing predictive mean square error ([6] and [15]). See also [24], where it is shown that this method is also good for minimizing solution mean square error in a variety of circumstances.
Based on the Bayesian interpretation of the smoothing spline models [20], Wahba [22] also derives the GML method which seeks the minimizer of

\[ M(\lambda) = \frac{y^T(I - A(\lambda))y/n}{[\text{det}^+(I - A(\lambda))]^{1/n_1}}, \]

where \( \text{det}^+(I - A) \) is the product of the \( n_1 = n - M \) nonzero eigenvalues of \( I - A \). In parallel to (7), we have

\[ M(\lambda) = M(\lambda, \theta) = \frac{\theta^T(Q_0^\theta + n\lambda I)^{-1}\theta/n}{[\text{det}(Q_0^\theta + n\lambda I)^{-1}]^{1/n_1}}. \]

A theoretical comparison of the GCV and GML methods can be found in [22]. The purpose of this article is to present a numerical algorithm for the computation of selecting the smoothing parameters according to the GCV or GML criterion. Our main concern is to deal with problems with \( k > 1 \). The method is developed on the basis of the \( k = 1 \) algorithm of Gu et al. [11].

2. Preliminaries. To minimize the functions \( V(\lambda, \theta) \) or \( M(\lambda, \theta) \) with respect to \( \theta \) and \( n\lambda \), we wish to iterate on the following cycle:

1) For fixed \( \theta \), minimize \( V(\lambda|\theta) \) or \( M(\lambda|\theta) \) with respect to \( n\lambda \).
2) Update \( \theta \) using information from the current estimates.

Step 1) above can be implemented through the single smoothing parameter GCV/GML algorithm based on the Householder tridiagonalization, as proposed by Gu et al. [11]. To carry out step 2), we will evaluate the gradient and the Hessian of \( V(\theta|\lambda) \) or \( M(\theta|\lambda) \) with respect to \( \eta = \log(\theta) \), then apply the modified Newton method (see [8]) to update the \( \eta \). In this section, we first discuss the choices for the scaling of the variables (\( \eta \)) and the scaling of the objective functions (\( V(\cdot) \) of \( M(\cdot) \)). Then we present the expressions of the gradient and the Hessian for later use.

We choose the variables \( \eta \) instead of \( \theta \) mainly for their invariance. Looking at the formulas of \( V(\cdot) \) and \( M(\cdot) \) ((7) and (8)), we can see that what really matter are the matrices \( \theta_iQ_i \)'s, while the “face values” of the \( \theta_i \)'s are subject to rescaling when the matrices \( Q_i \)'s are multiplied by some positive constants. Notice that the problem itself is not changed by such transformations. Invariance under this kind of transformation is hence essential to any sensible algorithm. It is easy to see that the derivatives of \( V(\cdot) \) and \( M(\cdot) \) with respect to \( \eta \) are invariant, and hence the methods based on these derivatives are invariant. Standard calculations show that the derivatives of \( V(\cdot) \) and \( M(\cdot) \) with respect to \( \theta \) are in general not invariant in this sense, which disqualifies them for serving as the basic variables. Another immediate numerical merit from adopting \( \eta \) instead of \( \theta \) is that we change a constrained optimization problem (\( \theta \geq 0 \)) to an unconstrained one, which allows much simpler treatment.

The objective functions can always be rescaled by monotone transformations without changing the optima of the problems. In our problem, it is observed that log \( V(\cdot) \) and log \( M(\cdot) \) have simpler derivative formulas. However, this transformation is not adopted for the following reason. To minimize the objective function efficiently, we want the objective function to be as convex as possible, since most optimization methods are modeled after convex functions. When the convexity is violated, we have to modify the methods and to suffer lower efficiency, and the methods may even fail to converge. Since log \( f(\cdot) \) is concave, we can expect log \( f(\cdot) \) to be “less convex” than \( f(\cdot) \) for general \( f(\cdot) > 0 \). Actually, it can be shown that a positive-definite Hessian of log \( f(\cdot) \) implies a positive-definite Hessian of \( f(\cdot) \), but the reverse is not true.
Having chosen the objective functions and the basic variables, we now collect the formulas of the gradient and the Hessian. Define \( D = \sum_{i=1}^{k} (e_i Q_i) + n \lambda I = Q^0 + n \lambda I \). We write \( V(\eta | \lambda) = n z^T D^{-2} z / (\text{tr } D^{-1})^2 \) and \( M(\eta | \lambda) = z^T D^{-1} z / n (\text{det } D^{-1})^{1/n} \). We have the following lemmas.

**Lemma 2.1.**

\[
\begin{align*}
\frac{\partial V}{\partial \eta_i} &= n \left( \frac{\dot{r}_i}{t^2} - \frac{2 \dot{r}_i}{t^3} \right), \\
\frac{\partial^2 V}{\partial \eta_i \partial \eta_j} &= n \left( \frac{\ddot{r}_{ij}}{t^2} - \frac{2 \dot{r}_i \dot{r}_j}{t^3} - \frac{2 \dot{r}_j \dot{r}_i}{t^3} + \frac{6 \dot{r}_i \dot{r}_j}{t^4} \right),
\end{align*}
\]

where

\[
\begin{align*}
\dot{r}_i &= \frac{\partial r}{\partial \eta_i} = \frac{\partial}{\partial \eta_i} (z^T D^{-2} z) = -2 z^T D^{-2} (e_i Q_i) D^{-1} z, \\
\dot{r}_{ij} &= \frac{\partial^2 r}{\partial \eta_i \partial \eta_j} = \frac{\partial}{\partial \eta_i} (\text{tr } D^{-1}) = -\text{tr} (D^{-1} (e_i Q_i) D^{-1}),
\end{align*}
\]

\[
\begin{align*}
\ddot{r}_i &= 2 [z^T D^{-2} (e_i Q_i) D^{-1}] + z^T D^{-1} (e_i Q_i) D^{-2} (e_i Q_i) D^{-1} z + z^T D^{-1} (e_i Q_i) D^{-2} (e_i Q_i) D^{-1} z \\
&+ z^T D^{-1} (e_i Q_i) D^{-1} (e_i Q_i) D^{-2} z + \delta_{i-j} \dot{r}_i,
\end{align*}
\]

where \( \delta_i \) is Kronecker's delta.

**Lemma 2.2.**

\[
\begin{align*}
\frac{\partial M}{\partial \eta_i} &= \frac{1}{n} \left( \frac{\dot{r}_i}{t^{1/n}} - \frac{1}{n} \frac{\dot{r}_i}{t^{1+1/n}} \right), \\
\frac{\partial^2 M}{\partial \eta_i \partial \eta_j} &= \frac{1}{n} \left( \frac{\ddot{r}_{ij}}{t^{1/n}} - \frac{1}{n} \frac{\ddot{r}_{ij}}{t^{1+1/n}} - \frac{1}{n} \frac{\dot{r}_i \dot{r}_j}{t^{1+1/n}} + \frac{1}{n^2} \frac{\dot{r}_i \dot{r}_j}{t^{2+1/n}} \right),
\end{align*}
\]

where

\[
\begin{align*}
\dot{r}_i &= \frac{\partial r}{\partial \eta_i} = \frac{\partial}{\partial \eta_i} (z^T D^{-1} z) = -z^T D^{-1} (e_i Q_i) D^{-1} z, \\
\dot{r}_{ij} &= \frac{\partial^2 r}{\partial \eta_i \partial \eta_j} = \frac{\partial}{\partial \eta_i} (\det D^{-1}) = -\det (D^{-1}) \text{tr} (e_i Q_i D^{-1}),
\end{align*}
\]

\[
\begin{align*}
\ddot{r}_i &= 2 [z^T D^{-1} (e_i Q_i) D^{-1}] + z^T D^{-1} (e_i Q_i) D^{-2} (e_i Q_i) D^{-1} z + \delta_{i-j} \dot{r}_i, \\
\ddot{r}_{ij} &= \det (D^{-1}) \text{tr} ((e_i Q_i) D^{-1} (e_i Q_i) D^{-1}) + \det (D^{-1}) \text{tr} (e_i Q_i D^{-1}) \text{tr} (e_i Q_i D^{-1}) + \delta_{i-j} \dot{r}_i,
\end{align*}
\]

where \( \delta_i \) is Kronecker's delta.

The proofs to the above two lemmas are straightforward and tedious. We omit them here.

### 3. Algorithms.

In this section, we will specify the main algorithm and discuss its various aspects. More discussions will be collected in § 6 as remarks.
Algorithm 3.1. Assuming inputs of the null-space matrix $S$, the response vector $y$, the matrices $\tilde{Q}_i$, $i = 1, \cdots, k$, and the starting values $\eta_0$, we propose performing the following:

1. Initialization:
   (a) Compute the QR-decomposition of $S = FR = (F_1, F_2)(R_0)$. 
   (b) Compute $z = F_2^T y$ and $Q_i = F_2^T \tilde{Q}_i F_2$. 
   (c) Set $\Delta \eta = 0$, $\eta_+ = \eta_0$, $V_- = \infty$ (or $M_- = \infty$).

2. Iteration:
   (a) For the current trial value $\eta = \eta_+ + \Delta \eta$, collect $\hat{D} = Q_0^* = \sum_{i=1}^k \theta_i Q_i$.
   (b) Compute $\hat{D} = U\hat{T}U^T$, where $U$ is orthogonal and $\hat{T}$ is tridiagonal. Compute $x = U^T z$.
   (c) Minimize

   \[
   V(\lambda|\eta) = \frac{n x^T (\hat{T} + n \lambda I)^{-2} x}{[\text{tr} (\hat{T} + n \lambda I)^{-1}]^2}
   \]
   or

   \[
   M(\lambda|\eta) = \frac{x^T (\hat{T} + n \lambda I)^{-1} x}{n \det (\hat{T} + n \lambda I)^{-1/2}}.
   \]

   If $V > V_-$ (or $M > M_-$), set $\Delta \eta = \Delta \eta / 2$, goto (a); else proceed.
   (d) Evaluate the gradient $g = (\partial / \partial \eta) V(\eta|\lambda)$ (or $(\partial / \partial \eta) M(\eta|\lambda)$) and the Hessian $H = (\partial^2 / \partial \eta \partial \eta^T) V(\eta|\lambda)$ (or $(\partial^2 / \partial \eta \partial \eta^T) M(\eta|\lambda)$). Calculate the increment $\Delta \eta = -H^{-1} g$, where $H = H + \text{diag} (e)$ is positive definite. If $H$ itself is positive definite "enough," $e$ is simply 0.
   (e) Check convergence conditions. If the conditions fail, set $\eta_+ = \eta$, $V_- = V$ (or $M_- = M$), goto (a); else proceed.

3. Calculate the optimal model:
   (a) If $\Delta \eta_i < -\gamma$, set $\eta_i = -\infty$, where $\gamma$ is a "large" number, say, $\gamma \in (.5, .9)$.
   (b) Collect $\hat{D} = \sum_{i=1}^k \theta_i Q_i$. Calculate the model minimizing $V(\lambda|\eta)$ (or $M(\lambda|\eta)$).

In the above specification, most items clearly explain themselves, except steps 2(d), 2(e) and 3(a), which we will explain in turn. Step 2(d) is the major part of this algorithm. Define $D = \hat{D} + n \lambda I$ (this definition is consistent with the one in §2), $T = \hat{T} + n \lambda I$, and $K_i = e^{\eta_i} U^T Q_i U$. We can write (11)-(14) as

\[
\hat{r}_i = -2x^T T^{-2} K_i T^{-1} x = -2(T^{-1} x)^T (T^{-1} K_i)(T^{-1} x),
\]

\[
i_i = -\text{tr} (T^{-2} K_i),
\]

\[
\hat{r}_{ij} = 2[(T^{-1} x)^T (T^{-1} K_i)(T^{-1} K_j)(T^{-1} x) + (T^{-1} x)^T (T^{-1} K_i)(T^{-1} K_j)(T^{-1} x)] + \delta_{i-j} \hat{r}_i,
\]

\[
i_{ij} = 2 \text{tr} ((T^{-2} K_i)(T^{-1} K_j)) + \delta_{i-j} \hat{r}_i.
\]

Similarly, (17)-(20) become

\[
i_i = -x^T (T^{-1} K_i)(T^{-1} x),
\]

\[
i_i = -\text{det} (T^{-1}) \text{tr} (T^{-1} K_i),
\]

\[
\hat{r}_{ij} = 2x^T (T^{-1} K_i)(T^{-1} K_j)(T^{-1} x) + \delta_{i-j} \hat{r}_i.
\]
\[ t_{ij} = \det (T^{-1}) \text{tr} ((T^{-1}K_i)(T^{-1}K_j)) + \det (T^{-1}) \text{tr} (T^{-1}K_i) \text{tr} (T^{-1}K_j) + \delta_{i,-j}t_i. \]

The gradients and the Hessians presented in Lemmas 2.1 and 2.2 will be calculated via (23)-(26) and (27)-(30). The modified Newton method based on the modified Cholesky decomposition, as described in [8], is adopted to calculate the update direction \( \Delta \eta = -\hat{H}^{-1}g \). Since the score evaluation is expensive, we choose not to perform a step length search, but simply to pick unity as the default step length. We execute an Armijo-type procedure of halving the step to insure an improved score in each iteration, where improvement is guaranteed by the fact that \( -\hat{H}^{-1}g \) is a descent direction.

Given a user-supplied precision requirement \( \epsilon_A \), the algorithm is thought to have converged if at least one of the following criteria is satisfied:
1) (a) \( V_\gamma - V < \epsilon_A(1 + V) \), and (b) \( \|g\|_\infty < \sqrt{\epsilon_A (1 + V)} \).
2) \( \|g\|_\infty < \epsilon_A \).

The criteria are modifications of the suggestions by Gill et al. [8, pp. 306-307]. Our rule 1(a) is identical to their U1, while our rule 1(b) is expected to do what their U2–U3 do. In our problem some optimal \( \eta_i \) could be at \(-\infty\). In such a situation, the algorithm will keep wanting to move towards the optimum with big steps even when the score values are well within the required precision of the optimal value. This is a typical ill-conditioned situation where their U2 will never be satisfied. By adopting a more stringent version of their U3 (our rule 1(b)) instead of their U2–U3 combination, we can avoid such endless iteration and yet deliver qualified termination, since our rule 1(b) always implies the satisfaction of their U2 when the problem is well conditioned (see [8, p. 307]). Our rule 2 is simply their U4. Their U5 is discarded for the same reason discussed above. At this point, the motivation of step 3(a) of the algorithm should have become clear.

We now briefly discuss the operational counts of the algorithm. Steps 1(a), 1(b), and 2(a) can be executed with \( O(n^2) \) operations, provided \( M \) (the rank of the null space matrix \( S \)) and \( k \) are both of constant order. Step 2(b) can be implemented via the Householder tridiagonalization algorithm, which in general takes about \( (2/3)n^3 \) operations, while some time-saving is possible through the distributed truncation proposed by Gu et al. [11]. Step 2(c) is usually performed by a golden section search on \( \log (n\lambda) \); each evaluation of the score functions \( V(\cdot) \) or \( M(\cdot) \) via formulas (21) or (22) requires only \( O(n) \) operations (see [11]). As we mentioned earlier, step 2(d) is the major burden on the algorithm. To calculate each of the \( K_i = e^\eta_i U^{-1}Q_i U \), noticing that \( U \) is available from step 2(b) in a factored form as the product of a series of Householder transforms, we could successively apply the Householder transforms from both sides of \( Q_i \), which need approximately the same number of operations as step 2(b). Making use of the identity \( \sum_{i=1}^k K_i = \hat{T} \), we need a total of \( (2/3)(k-1)n_i^3 \) operations for the \( K_i \)'s. Since the linear system \( Tx = b \) can be solved with \( O(n) \) operations, \( T^{-1}K_i \)'s and \( T^{-2}K_i \)'s can be obtained with \( O(n^2) \) operations. Hence the total number of operations needed for each iteration is in general \( (2/3)kn_i^3 + O(n^2) \). For each failure trial with \( V > V_\gamma \) or \( M > M_\gamma \), we have to spend \( (2/3)n_i^3 \) operations (step 2(b)) before discovering it. Step 3(b) needs another \( (2/3)n_i^3 \) to calculate the final results. The above operational counts are based on related discussions in [7] and [9]. See also [11].

Good starting values are important to Newton-type iterative optimization methods. They are even more crucial to our algorithm since our iteration is extremely expensive for large \( n \). Deriving good starting values is not a mere numerical problem but something
to do with the original setup of the model. Below, we will propose two sensible
approaches for obtaining good starting values for the algorithm. Both of them are
based on the background problem formulation presented in § 1.

The first proposal is built on the assumption that the optimal smoothing parameters
\( \lambda_i \)'s share approximately the same decreasing rate as the number of observations
increases. We can then randomly select a subset of the observations available, calculate
the optimal \( \theta \) for the subset, and use these as the starting value for some bigger subset
or the complete data set. The idea here is to perform a "crude" search with a smaller
problem size to save execution time. If this assumption is strongly believed, then a
large size iteration can be avoided by simply adopting the \( \theta \) obtained from a subset
run and going ahead to perform only step 3(b) of the algorithm on the complete data
set.

The second proposal is more involved. Suppose we knew the underlying truth
function \( f_\gamma \), and it could be decomposed as the sum of projections onto orthogonal
subspaces \( \mathcal{H}_i \)'s, \( f_\gamma = \sum_{i=0}^{k} P_i f_\gamma \), where \( P_i \) denotes the projection operator onto subspace
\( \mathcal{H}_i \). When using the smoothing spline model to retrieve the truth function from data,
it is sensible to adjust the face values of the norms \( \| P_i f \| \) by the "strength" of the
corresponding components of the truth function, \( \| P_i f \| \), to balance the roughness
penalties put on different components. Specifically, when \( \| P_1 f \| = .07 \) and \( \| P_2 f \| = 7 \),
say, we might regard \( P_1 f \) with \( \| P_2 f \| = .07 \) as rough as \( P_2 f \) with \( \| P_2 f \| = 5 \). This heuristic
leads to the following formulation with one smoothing parameter:

\[
\min \frac{1}{n} \sum_{j=1}^{n} (y_j - L_j f)^2 + \lambda \sum_{i=1}^{k} \| P_i f \|^2,
\]
or equivalently, \( \theta_i = \| P_i f \|^2 \). \( \theta \) chosen this way should be close to optimal, and hence
be a good starting value for the iteration. Of course in practice we will never know
the truth function \( f_\gamma \), and in turn the factors \( \| P_i f \|^2 \). However, all we need here is
just a set of starting values for the iterative algorithm, and some estimates of the factors
\( \| P_i f \|^2 \) should suffice. The resulting starting value procedure, which is made default
in the implementation, is thus Algorithm 3.2.

**Algorithm 3.2.** If no starting values are specified, we calculate the default
starting values as follows:

1. Set \( \theta_i = (\text{tr} (Q_i))^{-1} \), fit the one smoothing parameter spline model by minimizing
   \( V(\theta_i \lambda) \) or \( M(\theta_i \lambda) \), calculate the parameters \( \mathbf{c} \).
2. Estimate \( \| P_i f \|^2 \) by \( \theta_i \mathbf{c}^T \hat{Q_i} \mathbf{c} \), and set the starting values of Algorithm 3.1
to be \( \eta_0 = \log (\theta_0) \).

The choice of \( \theta_i \)'s in step 1 above is arbitrary but invariant in the sense we discussed
in § 2; other invariant selections might be equally appropriate. We estimate \( \| P_i f \|^2 \)
simply by replacing \( P_i f \) with \( P_i f = \sum_j c_i \theta_i (P_i \xi_j) \), where \( \xi_j \) is the representer of the
linear functional \( L_i \) under the original norm \( \| \cdot \| \). Remember that \( \langle P_i \xi_j, P_i \xi_{j'} \rangle = (\hat{Q}_i)_{j,j'} \).
Algorithm 3.2 takes about \( (2/3)n^3 + O(n^2) \) operations for execution.

4. **Examples: Additive/interaction spline models.** To test the algorithms presented
in the previous section, we apply them to fit the additive/interaction spline models,
which were proposed by Barry [2], [3] and Wahba [23], and were first illustrated by
Gu et al. [11] in the \( k = 2 \) case where a grid search on the one-dimensional \( \theta \) was
conducted. In this section, we will report some of our numerical experiments with the
iterative multiple smoothing parameter algorithm proposed in this article. Various
statistical aspects of the additive/interaction spline models are currently under study.
The findings of the study will be presented elsewhere. For a review of the additive
models via the backfitting algorithm without automatic smoothing parameter selection, see [13] and [4].

The formulation of the additive/interaction spline models is based on the tensor product Hilbert space formulation. Take the component Hilbert spaces (on [0, 1]) as the reproducing kernel Hilbert space $W_0^n = \{ f: f(x) \text{ abs cont.}, \; \nu = 0, \cdots, m-1, \int (f^{(\nu)})^2 < \infty \}$ with norm $\|f\|^2 = \sum_{\nu=0}^{m-1} (\int_0^1 f^{(\nu)})^2 + \int_0^1 (f^{(m)})^2$ (see [6]). We let $H$ be the tensor product Hilbert space $H = \bigotimes_{l=1}^d H_l$, with $H_l = W_0^m$. Readers are referred to [1] for technical details on tensor products of reproducing kernel Hilbert spaces. We note that the component spaces of the tensor product space need not to be of the same form. For example, we might specify different $m$'s for different $H_l$'s for our formulation here, though we choose not to do so to keep the notation simple. Write $W_0^n = N \oplus P_{m-1} \oplus S_m$, where $N$ is the space of constant functions with square norm $(\int_0^1 f)^2$, $P_{m-1}$ is the space of all polynomials with degrees less than $m$ which integrate to zero, with square norm $\sum_{\nu=0}^{m-1} (\int_0^1 f^{(\nu)})^2$, and $S_m$ is the space of functions with square integrable $m$th derivative and satisfy $\int_0^1 f^{(\nu)} = 0, \nu = 0, \cdots, m-1$, with square norm $\int_0^1 (f^{(m)})^2$. When $m=1$, the space $P_0$ vanishes. (Further decomposition of $P_{m-1}$ for $m>2$ is sometimes useful (see [11]), though we will not give much detail here.) From the direct sum decompositions of the component spaces formulated above, the space $H = \bigotimes_{l=1}^d H_l = \bigotimes_{l=1}^d (N_l \oplus P_{m-1} \oplus S_m)$ can be represented as the direct sum of $3^d$ orthogonal subspaces when $m>1$, and $2^d$ subspaces when $m=1$. Various statistical model formulations can be specified from this structure. For example, the additive models are obtained by eliminating the subspaces with more than one nonconstant tensor component (i.e., with fewer than $d-1$ $N_l$'s as components).

We first present a simulated additive spline example on $[0, 1]^4$. The truth function is specified by

$$f(x) = f(x_1, x_2, x_3, x_4) = 10 \sin (\pi x_2) + \exp (3 x_3) + 10^6 x_1^4 (1 - x_4) + 10^4 x_1 (1 - x_4)^{10}.$$  

The bounded linear functionals $L_j$'s are chosen as the evaluation functionals $[x_j]f = f(x_j)$. We generated $n = 100$ sampling points in $[0, 1]^4$ randomly, and computed the observations by $y = f(x) + \varepsilon_j$, $j = 1, \cdots, n$, where $\varepsilon_j$'s are independent Gaussian pseudorandom numbers with mean 0 and variance 1. The sampling points were generated using the Fortran routine uni of the Core Mathematics Library (Cmlib) from the National Bureau of Standards, with mdig = 32 and seed 2,375; the first 400 (after a null call to pass the seed) random variables were cut into four segments of length 100, and formed the first to fourth coordinates of the sampling points in the natural order. The $\varepsilon_j$'s were generated by the routine rnor of Cmlib with mdig = 32, and we took the 100 outcomes after the null call which passed the seed 5,732 to the routine. The scatter plot matrix of $x$ and $y$ is shown in Fig. 1. We fitted models with $m = 2, 1$, respectively. For $m = 2$, we selected $H_0 = \text{span} \{ 1, x_1 -.5, x_2 -.5, x_3 -.5, x_4 -.5 \}$, where $1$ spans $N_1 \otimes \cdots \otimes N_4$, $x_1 -.5$ span $((\otimes_{i=1}^d N_i) \otimes P_{1})$, $i = 1, \cdots, 4$. The penalized spaces were $H_i = ((\otimes_{i=1}^d N_i) \otimes S_2)$, $i = 1, \cdots, 4$. For $m = 1$, we chose $H_0 = \{ 1 \}$ and $H_i = ((\otimes_{i=1}^d N_i) \otimes S_1)$, $i = 1, \cdots, 4$. Other subspaces were deleted. The evaluation formulas for the matrices $S$, $Q_i$, $i = 1, \cdots, 4$, and for the representers $P_i \xi$ can be found from [11] for the $m = 2$ setting directly, and the $m = 1$ formulas can be similarly derived. Figures 2 and 3 illustrate the fitted additive components (solid lines) compared to the true additive components (dotted lines) for $m = 2$ (respectively, 1) with the vertical positions of the solid lines being adjusted to make the solid and dotted lines of the same frames integrate to the same number, as they should. The $m = 1$ curves (broken lines) look more wiggly than the $m = 2$ curves (cubic splines), although the main features are the same. The criterion for selecting the smoothing parameters was the
GCV score. The GCV scores and the corresponding mean square errors (evaluated at
the sampling points) of each iteration are listed in Table 1. In Table 1, the iterations
numbered 0 correspond to the starting values, and the last iterations (number 5 for
m = 2 and number 3 for m = 1) in the two cases correspond to the models illustrated
in Fig. 2 and Fig. 3, respectively. It appears that the starting value procedure (Algorithm
3.2) worked pretty well, and the convergence of the iteration was very fast. It can also
be seen that both V and mean square error (MSE) are smaller for m = 2, indicating
that m = 2 should have been chosen as a better model according to the GCV score,
and actually it is better according to the MSE. An execution for the k = 4, n = 100
case with five iterations (first column of Table 1; the amount of work is that of six
main loop executions) ran 724 cpu seconds on a Sun-3/280 (without a floating point
accelerator) in the Department of Statistics, University of Wisconsin–Madison. We
have also tried n = 200 and n = 400 for the same model. An n = 200 execution with
four iterations ran 3,628 cpu seconds on the same machine, and an n = 400 run with
three iterations took 16,371 cpu seconds.
The second example has a nonnull interaction component. This time we worked on the unit cube \([0, 1]^3\), with truth function

\[
f(x) = f(x_1, x_2, x_3) = 10 \sin(\pi x_2) + \exp(3x_3) + 5 \cos(2\pi(x_1 - x_2)),
\]
and \(y_j = f(x_j) + \varepsilon_j\). We used similar procedures and the same seeds to generate the sampling points and noise with the same mean and variance as in the additive model example, with \(n = 400\) and \(d = 3\). The scatter plot matrix of \(x_i\)'s and \(y\) is omitted since it is no more informative than Fig. 1. We chose \(m = 1\) for the interaction model example for its simpler formulation, though we would expect the plots of the estimation to look rather wrinkled. Eliminating the 3-factor interaction, we were left with seven subspaces of the tensor product Hilbert space \(\mathcal{H} = \bigotimes_{l=1}^3 \mathcal{H}^l\), namely, \(\mathcal{H}_0 = \{1\} = \bigotimes_{l=1}^3 \mathcal{N}^l\), \(\mathcal{H}_i = (\bigotimes_{l\neq i} \mathcal{N}^l) \otimes \mathcal{F}^l\), \(i = 1, 2, 3\), and \(\mathcal{H}_{(i)} = (\bigotimes_{l\neq i} \mathcal{F}^l) \otimes \mathcal{N}^l\), \(i = 1, 2, 3\), where \(\mathcal{H}_0\) was the null space and the other \(k = 6\) subspaces were penalized. We know that the truth function has null projections in \(\mathcal{H}_1\), \(\mathcal{H}_{(2)}\), and \(\mathcal{H}_{(3)}\). Projecting the estimated function to the
FIG. 3. Additive model: $m = 1$.

<table>
<thead>
<tr>
<th>Iter. No.</th>
<th>$m = 2$</th>
<th>$m = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>V</td>
<td>MSE</td>
</tr>
<tr>
<td>0</td>
<td>0.291176</td>
<td>1.80939</td>
</tr>
<tr>
<td>1</td>
<td>1.50180</td>
<td>1.75559</td>
</tr>
<tr>
<td>2</td>
<td>1.27318</td>
<td>1.74514</td>
</tr>
<tr>
<td>3</td>
<td>1.4512</td>
<td>1.74504</td>
</tr>
<tr>
<td>4</td>
<td>1.41040</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.40893</td>
<td></td>
</tr>
</tbody>
</table>
subspaces, we can compare the truth and the estimated components individually. For the subspace \( \mathcal{H}_1 \), the estimated component \( f_1(x_1) \) ranged between \([-0.14, 0.12]\) on grids \( x_1 = 0(.01)1 \). On \( \mathcal{H}_2 \), the estimated component \( f_{1,3}(x_1, x_3) \) ranged between \([-0.38, 0.38]\) on the tensor product mesh \( x_1, x_3 = 0(.033)1 \). And on \( \mathcal{H}_3 \), \( f_{2,3}(x_2, x_3) \) ranged between \([-0.18, 0.15]\) on the tensor product mesh \( x_2, x_3 = 0(.033)1 \). All three null component estimates were well below the standard deviation of the \( \varepsilon_j \)'s. The other two additive components \( f_2 \in \mathcal{H}_2 \) and \( f_3 \in \mathcal{H}_3 \) are plotted in Fig. 4 in the same manner as in Fig. 3, and the truth and the estimation of the interaction component \( f_{1,2} \in \mathcal{H}_3 \) are plotted in Fig. 5. The iteration sequence of this example is shown in Table 2. The iteration number 0 is again the starting value model. The algorithm converged at the third iteration in the sense that the optimal GCV value was within .1 percent of the achieved one. The total execution time was 25,859 cpu seconds.

**Fig. 4. Interaction model: Additive components.**

**Fig. 5. Interaction model: Interaction component.**
Table 2
Iterations of the interaction model.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.40266</td>
<td>.376505</td>
<td>2</td>
<td>1.29839</td>
<td>.291201</td>
</tr>
<tr>
<td>1</td>
<td>1.31241</td>
<td>.303801</td>
<td>3</td>
<td>1.29585</td>
<td>.285957</td>
</tr>
</tbody>
</table>

5. Variance component models. Consider the variance component model (see [18, § 4j])

\[y = S\beta + \sum_{i=1}^{k} B_i \xi_i + \epsilon,\]

where \(S\beta\) is the fixed effect of dimension \(M\), the \(B_i\)'s are known matrices with \(B_iB_i^T = \tilde{Q}_i\), \(\xi_i \sim N(0, b_iI)\), \(\epsilon \sim N(0, \sigma^2 I)\), and \(\xi_i\)'s and \(\epsilon\) are mutually independent. Let the QR-decomposition of \(S\) be

\[S = FR = (F_1, F_2) \begin{pmatrix} R_1 \\ 0 \end{pmatrix}.\]

Since \(z = F_2^Ty\) are \(n - M\) linearly independent contrasts of \(y\), the restricted maximum likelihood (REML) estimates of \(b_i\)'s and \(\sigma^2\) are the minimizers of the restricted log likelihood

\[l_b(b, \sigma^2 | z) = -\frac{1}{2} \log \left[ \det \left( \sum b_iQ_i + \sigma^2 I \right) \right] - \frac{1}{2} z^T \left( \sum b_iQ_i + \sigma^2 I \right)^{-1} z + C_1,\]

where \(Q_i = F_2^T \tilde{Q}_i F_2\) are as defined in earlier sections, and \(C_1\) is a constant. See also [12]. Reparameterizing the problem by \(b = b\theta_i\), \(\sigma^2 = b(n\lambda)\), and maximizing \(l_b(\cdot)\) with respect to \(b\), we get \(\hat{b} = z^T(\sum \theta_i Q_i + n\lambda I)^{-1} z / (n - M)\), and the profile restricted log likelihood is

\[l_b(\theta, \lambda | \hat{b}) = -\frac{1}{2} \log \left[ \det \left( \sum \theta_i Q_i + n\lambda I \right) \right] - \frac{n - M}{2} \log (\hat{b}) - C_2,\]

where \(C_2\) is another constant. (By profile likelihood we mean the likelihood function where some of the parameters have already been optimized.) This leads to the equivalent criterion of minimizing

\[M(\theta, \lambda | z) = \frac{z^T(\sum \theta_i Q_i + n\lambda I)^{-1} z / n}{\left[ \det \left( \sum \theta_i Q_i + n\lambda I \right)^{-1} \right]^{1/(n - M)}}.\]

Hence, our algorithm is directly applicable to the REML estimation of the usual variance component models.

For the maximum likelihood (ML) estimation, writing \(r = y - S\beta\), we have the log likelihood

\[l(b, \sigma^2 | \beta) = -\frac{1}{2} \log \left[ \det \left( \sum b_i\tilde{Q}_i + \sigma^2 I \right) \right] - \frac{1}{2} r^T \left( \sum b_i\tilde{Q}_i + \sigma^2 I \right)^{-1} r + C_3,\]

where \(C_3\) is a constant. After the reparameterization \(b_i = b\theta_i\), \(\sigma^2 = b(n\lambda)\), we can again solve for \(b\) explicitly, leading to the minimization of a log likelihood \(l(\lambda, \theta, \beta)\). We can then compute the ML estimation by alternating parameters as

1) Maximize \(l(\lambda, \beta | \theta)\): Tridiagonalization.
2) Maximize \(l(\theta | \lambda, \beta)\): Newton update.
In step 1), we can perform an inner iteration loop to optimize \( \lambda \) by grid search and \( \beta \) by Gauss-Markov estimation. Specifically, we perform a tridiagonalization \( \sum \theta_i \hat{Q}_i = UTU^T \), and compute \( x = U^T y, W = U^T S \). The optimal \( \lambda \) given \( \beta \) is the minimizer of the equivalent score

\[
M(\lambda | \beta, \theta) = \frac{(x - W\beta)^T (T + n\lambda I)^{-1} (x - W\beta) / n}{\det (T + n\lambda I)^{-1/n}}.
\]

The Gauss-Markov estimator of \( \beta \) given \( \lambda \) can be computed through

\[
\beta = [W^T (T + n\lambda I)^{-1} W]^{-1} W^T (T + n\lambda I)^{-1} x.
\]

Overall, step 1) is a \((2/3)n^3 + O(n^2)\) operation. Step 2) is as before a \((2/3)(k-1)n^3\) operation. The corresponding ML algorithm is trivial to specify.

Various properties of the ML and REML estimation of variance component models and their relation with other estimation methods are discussed in [12]. See also [19]. Surprisingly, the simple variable transformation \( B = \log \theta \), which results in the invariant and constraint-free numerical procedures for the REML and ML estimation, is not recognized in previous works summarized by Harville [12] and Rao and Kleffe [19]. In a closely related context, Lindstrom and Bates [16] illustrate how to apply the Newton method to compute the REML and ML estimates of parameters including a covariance matrix, where they employ a Cholesky decomposition as the variable transformation to ensure the nonnegative-definiteness of the covariance matrix. Numerically, we should admit that our method is too general to make use of possible special structures of \( B_i \) (hence \( Q_i \)) to reduce computational load, as well as too restrictive to handle models where the variance-covariance matrix of \( y \) depends non-linearly on the unknown parameters, though in the later case the model is no longer a variance component model literally.

6. Remarks. We now remark on some further points of interest about the algorithms.

(1) Methodology. Our iterative method is different from standard derivative-based optimization methods. It does not operate on all parameters simultaneously. Instead, it cycles between the two sets of parameters (\( \theta \) and \( \lambda \)), conditioning one on the latest version of the other, and uses different strategies for updating the two sets of parameters. Loosely speaking, each iteration targets on the profile score with respect to \( \theta \) where \( \lambda \) is being optimized, though the updates of \( \theta \) are not based on the derivatives of the profile score function since they are not available. Hence the method may be considered as a hybrid method adapted from the Newton methods on either the profile score or the score itself. In general, the idea of alternating parameters illustrated in this article can be expected to prove useful in tackling other statistical computing problems similar to the ones considered here. It is particularly appropriate in the context of GCV/GML functions since it is our experience, and seems to be part of the folklore in the related area of components of variance, that the value of these functions near their minima is substantially more sensitive to the value of \( \lambda \) than to the relative sizes of the \( \theta \)'s. The idea of cycling between parameters appeared some time ago in the psychometric literature in a different context (see, for example, [5]).

(2) Convergence. The goal of the algorithm is to locate a set of smoothing parameters which offer minimum GCV/GML scores and in turn a near-optimal smoothing. If the (true, but unknown) predictive mean square error of the estimate of \( f \) as a function of the smoothing parameters has a flat bottom near its minimizer, then the GCV score will also tend to have a flat bottom. (See [6], for example.) When that is
the case we would be glad to pick any one point from the bottom, and the exact optimal \( \theta_i \)'s are not particularly important. In this sense the convergence property of our algorithm is at least as good as the standard Newton method for minimizing the GCV/GML scores with respect to \( \log \lambda_i \)'s, since for fixed \( \lambda \) \( \log \theta_i \)'s are only a constant shift from the \( \log \lambda_i \)'s and the \( \eta = \log \theta \) updating is via the standard Newton method.

(3) Optimal parameter of infinity. A solution with some \( \theta_i = 0 \) (i.e., \( \lambda_i = \infty \)) is of special interest. It indicates the absence of a certain "effect" and hence gives a more parsimonious statistical model which is often desirable. In the current algorithm we set an optimal parameter of infinity \( \eta_i = -\infty \) after discovering that the score function is close enough to the minimum (i.e., converged), but the algorithm still drives \( \eta_i \) with a large step towards \(-\infty\), indicating a flat bottom extending to \( \eta_i = -\infty \). In all of the examples we tried where such an optimal parameter of infinity was set (not presented here), the scores with the parameter set to infinity did reduce further compared to the scores at the point where convergence was declared. From a purely statistical point of view, there is a certain probability that, even though a certain component of \( f \) is absent, and hence its theoretically optimal \( \eta_i \) is \(-\infty\), the exact \( \eta_i \) minimizing the GCV/GML function will be greater than \(-\infty\). Other methods for appropriately setting \( \lambda_i = \infty \) (which are not necessarily GCV/GML optimal) but have appropriate statistical properties, are under study.

(4) Invariance. The invariance property of concern in this article is the invariance of the algorithm. It is different from the invariance properties encountered in most of the statistical literature which enforce the invariance of the end results of statistical procedures. In contrast, our concern is to enforce the invariance of numerical iteration sequences which we hope will converge to the invariantly defined end results.

(5) Numerical efficiency. Our iterative algorithm is very expensive for large \( n \), but it is believed to be highly efficient for the problem. The algorithm is efficient in the sense that its operational count, \( (2/3)kn^3 + O(kn^2) \) flops per iteration, is pretty much the least that can be reasonably expected, given the fact that the single evaluation of the profile score is a \( (2/3)n^3 + O(n^2) \) flop operation, which is itself the best available in general smoothing spline settings (see [11]). Some Monte Carlo approximations to the derivatives (read the trace terms in (12), (14), (18), and (20)), which require only \( O(n^2) \) extra operations beyond the profile score evaluation, have also been tried. They were eventually discarded because the derivative-based methods are very sensitive to the errors in the derivative evaluations [8], and the approximations will lose most or all significant digits due to cancellation as the optimum is being approached.

(6) Other applications and the starting values. It is obvious that our iterative algorithm applies to all settings which result in minimization problems with similar form to equation (7) or (8). However, our starting value procedures are derived from the setting of (4), and hence may not be appropriate for other settings such as the variance component models.

(7) Step length search. There is no step length search included in the current version of the algorithm except a safeguard procedure of Armijo type in step 2(c) of Algorithm 3.1. As the associate editor suggested, including a step length search will lead to a more reliable algorithm. However, since the cost for score evaluation is comparable to derivative computation in our setting, a step length search will consume considerable execution time, and the overall procedure may not be as efficient. On the other hand, since we have good starting values and since the \( \lambda \)-step of the algorithm keeps us close to the optimal area where the scores behave reasonably well (see the plots in [11]), we found the current algorithm quite satisfactory on the examples we tried. (As a matter of fact, the \( \lambda \)-step of the algorithm is an exact line search, though
it is on a different line.) It will be worthwhile to investigate the pros and cons of including a step length search in further study.

Acknowledgments. The original idea of the starting value procedure Algorithm 2 was inspired by a conversation with Zehua Chen, to whom we owe thanks. We also wish to acknowledge the effort of our Computer Committee in the Department of Statistics, University of Wisconsin–Madison, led by Doug Bates, for maintaining a very comfortable computing environment in the department.

REFERENCES