Smoothing Splines and Rank Structured Matrices: Revisiting the Spline Kernel

Andersen, Martin S.; Chen, Tianshi

Published in:
SIAM Journal on Matrix Analysis and Applications

Link to article, DOI:
10.1137/19m1267349

Publication date:
2020

Document Version
Peer reviewed version

Smoothing Splines and Rank Structured Matrices: Revisiting the Spline Kernel

Martin S. Andersen* and Tianshi Chen†

April 2020

Abstract

We show that the spline kernel of order p is a so-called semiseparable function with semiseparability rank p. A consequence of this is that kernel matrices generated by the spline kernel are rank structured matrices that can be stored and factorized efficiently. We use this insight to derive new recursive algorithms with linear complexity in the number of knots for various kernel matrix computations. We also discuss applications of these algorithms, including smoothing spline regression, Gaussian process regression, and some related hyperparameter estimation problems.

1 Introduction

Spline functions play a key role in many areas of applied mathematics. They are flexible and have good approximation properties, and they arise naturally, e.g., in the so-called smoothing spline regression problem

$$\text{minimize} \quad \mathcal{J}(f) \equiv \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int_a^b |f^{(p)}(x)|^2\,dx\,. \tag{1}$$

The vector $y = (y_1, \ldots, y_n)$ and $a < x_1 < x_2 < \cdots < x_n < b$ are given, $\lambda \geq 0$ is a parameter, p is a natural number less than or equal to n, and the pth derivative of f is assumed to be square integrable. The second term of the functional $\mathcal{J}(f)$ may be viewed as a roughness penalty. Schoenberg [15] showed that the solution to (1) is a so-called natural spline of order $2p$ with knots $x_1, \ldots, x_n$. The set of such splines form a vector space of dimension n, and hence the variational problem (1) can be cast as an equivalent finite-dimensional problem by introducing a suitable set of basis functions; see, e.g., [16] for an introduction to spline functions. Alternatively, the solution to (1) can be found by solving a convex quadratic optimization problem of the form [20]

$$\text{minimize} \quad \frac{1}{n}\|y - (\Sigma\alpha + F\beta)\|_2^2 + \lambda\alpha^T\Sigma\alpha \tag{2}$$

with variables $\alpha \in \mathbf{R}^n$ and $\beta \in \mathbf{R}^p$. The matrix Σ is symmetric and positive semidefinite of order n with entries generated by the so-called spline kernel, and $F \in \mathbf{R}^{n\times p}$ is a Vandermonde matrix. Given an optimal solution $(\alpha^\star, \beta^\star)$ to (2), the natural spline of order $2p$ that interpolates $(x_1, \hat{y}_1), \ldots, (x_n, \hat{y}_n)$ with $\hat{y} = \Sigma\alpha^\star + F\beta^\star$ is the unique optimal solution to (1). It can be shown

*Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark. Email: mskan@dtu.dk

†School of Science and Engineering and Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, China. Email: tschen@cuhk.edu.cn

from the optimality condition associated with (2) that there exists a solution $(\alpha^\star, \beta^\star)$ that satisfies the system of equations

$$\begin{bmatrix} \Sigma + n\lambda I & F \\ F^T & 0 \end{bmatrix} \begin{bmatrix} \alpha^\star \\ \beta^\star \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}. \tag{3}$$

Forming this system explicitly requires $O(n^2)$ memory, and solving it using a standard, general-purpose method requires $O(n^3)$ floating-point operations (flops).

The main purpose of this paper is to show that the spline kernel is a so-called p-semiseparable function, and as a consequence, the matrix Σ is a rank structured matrix. Specifically, we show that Σ is a so-called extended generator representable p-semiseparable matrix (see definition 2). In other words, the lower-triangular part of Σ, which we will denote by $\mathbf{tril}(\Sigma)$, is generated by a rank p matrix, i.e., it can be expressed as

$$\mathbf{tril}(\Sigma) = \mathbf{tril}(UV^T), \qquad U, V \in \mathbf{R}^{n\times p}.$$

We show this in section 3, and it has two important consequences: (i) the matrix Σ admits an implicit representation (in terms of its generators U and V) that only requires $O(pn)$ memory, and (ii) the system (3) can be solved in $O(p^2n)$ flops. As a second contribution, we show in section 4 that key computations that involve Σ can be carried out efficiently (e.g., in $O(p^2n)$ flops rather than $O(n^3)$ flops) by exploiting the inherent structure. Specifically, we show that positive definite matrices of the form $\Sigma + \mathbf{diag}(d)$ may be factorized as LL^T where L is lower triangular and, like Σ, it has a memory-efficient implicit representation. We also derive an implicit representation of L^{-1} that we use to construct efficient recursive algorithms for computing the diagonal elements of matrices of the form $(\Sigma + \mathbf{diag}(d))^{-1}$ and the trace of, e.g., $(\Sigma + \mathbf{diag}(d))^{-1}\Sigma$. This is useful for model selection based on generalized cross validation (GCV) or generalized maximum likelihood (GML) estimation, e.g., in the context of smoothing spline regression. We provide a brief review of theory pertaining to smoothing spline regression in the next section, and in section 5, we discuss how the algorithms that we derive in section 4 can be applied to smoothing spline regression and Gaussian process regression. Finally, we provide a numerical example in section 6 that demonstrates the computational efficiency of some of the algorithms presented in section 4, and we conclude the paper in section 7.

2 Background and related work

Before we present our main results in sections 3 and 4, we briefly introduce notation, recall some results from the literature about splines and smoothing spline regression, and discuss related work. Interested readers may find a comprehensive introduction to smoothing spline regression in, e.g., the monographs by Wahba [20] and Schumaker [16].

2.1 Notation

We denote by $\mathbf{R}^n_+$ the set of elementwise nonnegative vectors in $\mathbf{R}^n$, and $\mathbf{R}^n_{++} = \mathbf{int}\,\mathbf{R}^n_+$ denotes its interior (i.e., elementwise positive vectors). Given n real numbers $x_1, \ldots, x_n$, we define $x = (x_1, \ldots, x_n)$ to be a column vector in $\mathbf{R}^n$. The identity matrix of order n is denoted I_n; we will simply use I when the order can be inferred from its context. The vector e_k denotes the unit vector whose kth element is equal to 1, and $\mathbf{1}$ is the vector of ones. Given a square matrix A of order n, $\mathbf{tril}(A)$ denotes the lower-triangular matrix obtained from A by setting all elements above the main diagonal to zero, and $\mathbf{tril}(A, k)$ is obtained from A by setting all elements above the kth superdiagonal to zero ($k > 0$ corresponds to a superdiagonal, $k < 0$ corresponds to a subdiagonal, and $\mathbf{tril}(A, 0) = \mathbf{tril}(A)$). Similarly, $\mathbf{triu}(A)$ denotes the upper triangular part of A and $\mathbf{triu}(A, k) = \mathbf{tril}(A^T, -k)^T$. Given a vector $x \in \mathbf{R}^n$, $\mathbf{diag}(x)$ is a diagonal matrix of order n with the elements of x as its diagonal entries. Finally, $L^2[a, b]$ denotes the space of square integrable functions defined on the interval $[a, b]$.

2.2 Splines and smoothing spline regression

We start by recalling the definition of a univariate polynomial spline. Given a set of n simple knots $\{x_1, \ldots, x_n\}$ and an interval $[a, b]$ such that $a = x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1} = b$, a polynomial spline $g(x)$ of order r is a real-valued function, defined on $[a, b]$, that satisfies the following conditions:

(i) g is a polynomial of degree at most $r-1$ on any subinterval $[x_j, x_{j+1}]$ $(j = 0, \ldots, n)$

(ii) g has $r-2$ continuous derivatives (i.e., $g \in C^{r-2}$).

We will denote the set of polynomial splines by $\mathcal{S}^r(\Delta)$ where $\Delta = \{x_0, x_1, \ldots, x_{n+1}\}$. The set of functions that satisfy the first condition has dimension $(n+1)r$, and since the second condition imposes $r-1$ constraints for each of the n knots, the dimension of $\mathcal{S}^r(\Delta)$ is $n+r$.

The set of natural polynomial splines of order $r = 2p$, which we will denote $\mathcal{S}^{2p}_{\text{nat}}(\Delta)$, is the n dimensional subset of $\mathcal{S}^{2p}(\Delta)$ that is obtained by imposing the additional condition that

(iii) g is a polynomial of degree at most $p-1$ on each of the subintervals $[a, x_1]$ and $[x_n, b]$.

As mentioned in the introduction, Schoenberg [15] showed that the solution to the smoothing spline regression problem (1) is a natural polynomial spline of order $2p$.

We now turn our attention to the problem (1) and let the functional $\mathcal{J}(f)$ be defined on the Sobolev space

$$W_p^2[a, b] = \left\{ f \colon f, f', \ldots, f^{(p-1)} \text{ absolutely continuous}, f^{(p)} \in L^2[a, b] \right\}$$

endowed with the inner product

$$\langle f, g \rangle = \sum_{k=0}^{p-1} f^{(k)}(a) g^{(k)}(a) + \int_a^b f^{(p)}(u) g^{(p)}(u)\, du, \qquad f, g \in W_p^2[a, b]. \tag{4}$$

It follows from Taylor's theorem that any $f \in W_p^2[a, b]$ can be expressed as

$$f(x) = \sum_{k=0}^{p-1} \frac{f^{(k)}(a)}{k!}(x-a)^k + \int_a^x \frac{(x-u)^{p-1}}{(p-1)!} f^{(p)}(u)\, du \tag{5}$$

where the last term is the integral form of the remainder. Furthermore, the inner product (4) implies that the space $W_p^2[a, b]$ can be decomposed as $W_p^2[a, b] = \mathcal{H}_0 \oplus \mathcal{H}_1$ where $\mathcal{H}_0$ and $\mathcal{H}_1$ are orthogonal complements and defined as [20]

$$\mathcal{H}_0 = \mathbf{span}\left\{\phi_1, \ldots, \phi_p\right\}, \qquad \phi_k(x) = \frac{(x-a)^{k-1}}{(k-1)!}, \quad k = 1, \ldots, p$$
$$\mathcal{H}_1 = \left\{ f \colon f^{(k)}(a) = 0,\ k = 0, \ldots, p-1,\ f^{(p)} \in L^2[a, b] \right\}.$$

As a consequence, every $f \in W_p^2[a, b]$ has a unique decomposition $f = f_0 + f_1$ where $f_0 \in \mathcal{H}_0$ and $f_1 \in \mathcal{H}_1$. It is easy to check that if $f \in \mathcal{H}_0$, then the remainder term in (5) is zero, and similarly, if $f \in \mathcal{H}_1$, then only the remainder term can be nonzero. If we let $\mathcal{P}_1 f$ denote the orthogonal projection of f onto $\mathcal{H}_1$, the roughness penalty in (1) can be expressed as

$$\lambda \int_a^b |f^{(p)}(u)|^2\, du = \lambda \langle \mathcal{P}_1 f, \mathcal{P}_1 f \rangle,$$

and hence it can be viewed as a penalty on the remainder term in the Taylor expansion (5).

Both $\mathcal{H}_0$ and $\mathcal{H}_1$ are reproducing kernel Hilbert spaces (RKHSs). Their corresponding reproducing kernels (RKs) are

$$\mathcal{K}_p^0(s,t) = \sum_{k=1}^{p} \phi_k(s)\phi_k(t) \tag{6}$$

$$\mathcal{K}_p^1(s,t) = \int_a^b G_p(s,u)G_p(t,u)\,du \tag{7}$$

where $G_p(s,u) = \max(0,s-u)^{p-1}/(p-1)!$ is the Green's function for the problem $D^p f = g$ with $f \in \mathcal{H}_1$, $g \in L^2[a,b]$, and where the operator D^p is the pth derivative; we refer the interested reader to [11] for an introduction to RKHSs. The kernel functions $\mathcal{K}_p^0$ and $\mathcal{K}_p^1$ are both positive semidefinite, and $\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{H}_1$ is itself a RKHS with RK $\mathcal{K}_p^0 + \mathcal{K}_p^1$. We remark that $\mathcal{K}_p^0(s,t)$ consists of p multiplicatively separable terms, and, as we will show in section 3, $\mathcal{K}_p^1(s,t)$ is a so-called semiseparable function with semiseparability rank p.

Kimeldorf & Wahba [9] showed that the solution to (1) can be expressed as

$$\hat{f}(x) = \sum_{k=1}^{p} \beta_k^\star\, \phi_k(x) + \sum_{j=1}^{n} \alpha_j^\star\, \xi_j(x) \tag{8}$$

where $\xi_j(x) = \mathcal{K}_p^1(x_j,x)$ and where the parameter vectors $\alpha^\star$ and $\beta^\star$ satisfy the system of equations (3). The matrix F in (3) is of size $n \times p$, and its (i,j) entry is $F_{ij} = \phi_j(x_i)$. Moreover, Σ is a kernel matrix generated by the kernel function $\mathcal{K}_p^1$, i.e., the (i,j) entry is $\Sigma_{ij} = \mathcal{K}_p^1(x_i,x_j)$. We will show in section 3 that the semiseparable structure of $\mathcal{K}_p^1$ carries over to Σ.

It can be shown that functions of the form (8), parameterized by two vectors α and β instead of $\alpha^\star$ and $\beta^\star$, are natural splines if $F^T\alpha = 0$; see [20]. Finally, we note that our assumption that $a < x_1 < x_2 < \cdots < x_n < b$ implies that Σ is positive definite; more generally, if we were to assume that $a \le x_1 \le x_2 \le \cdots \le x_n \le b$, then Σ will be positive semidefinite but not necessarily positive definite.

2.3 Stochastic processes

The smoothing spline regression problem (1) has strong ties to Bayesian estimation. Specifically, suppose $X(t)$ is a zero-mean Gaussian process defined on $[a,b]$ and with covariance function $\mathcal{K}_p^1$, and let

$$g(t) = \sum_{k=1}^{p} \theta_k\phi_k(t) + \nu X(t), \qquad t \in [a,b], \tag{9a}$$

$$y_i = g(t_i) + \epsilon_i, \qquad i = 1,2,\ldots,n \tag{9b}$$

where $\theta \in \mathcal{N}(0,\gamma I_p)$, $\epsilon \sim \mathcal{N}(0,\sigma^2 I_n)$, and θ and $X(t)$ are assumed to be independent. Moreover, if we let f_λ denote the solution to (1) with $\lambda = \sigma^2/(n\nu^2)$ and $x_i = t_i$, then the conditional expectation of $g(t)$ given the vector y is related to f_λ through the limit [20, Thm. 1.5.3]

$$\lim_{\gamma\to\infty} \mathrm{E}[g(t) \mid y] = f_\lambda(t), \qquad t \in [a,b].$$

The function $f_\lambda(t)$ interpolates the points $(t_i,\hat{y}_i)$ where $\hat{y} = (\hat{y}_1,\ldots,\hat{y}_n)$ can be expressed as $\hat{y} = H(\lambda)y$ and $H(\lambda)$ is the so-called influence matrix

$$H(\lambda) = I - n\lambda(M_\lambda^{-1} - M_\lambda^{-1}F(F^TM_\lambda^{-1}F)^{-1}F^TM_\lambda^{-1}), \quad M_\lambda = \Sigma + n\lambda I. \tag{10}$$

In section 5, we explain how the semiseparable structure of $\mathcal{K}_p^1$ can be exploited in computations that involve $H(\lambda)$ or matrices with similar structure. It can be shown that the covariance matrix associated with $\hat{y}$ is given by $\sigma^2 H(\lambda)$, and the ith diagonal element of this covariance matrix can be used to construct Bayesian credible intervals for $g(t_i)$. However, in practice, both σ and ν must somehow be selected or estimated in order to construct such credible intervals, but only the ratio $\lambda = \sigma^2/(n\nu^2)$ is necessary to compute $\hat{y}$.

2.4 Parameter selection

We now briefly review two commonly used parameter selection criteria, namely generalized cross validation (GCV) and generalized maximum likelihood (GML). GCV chooses the parameter λ as a minimizer of

$$\mathrm{GCV}(\lambda) = \frac{\frac{1}{n}\|(I - H(\lambda))y\|_2^2}{\left(\frac{1}{n}\,\mathbf{tr}(I - H(\lambda))\right)^2}. \tag{11}$$

We show in section 5 that $\mathbf{tr}(I - H(\lambda))$ can be computed in $O(p^2 n)$ flops using the recursive algorithms that we derive in section 4.

A GML estimate of λ is a maximizer of the likelihood function of λ given y, or equivalently, a minimizer of

$$\mathrm{GML}(\lambda) = \frac{y^T Q_2 (Q_2^T M_\lambda Q_2)^{-1} Q_2^T y}{\det(Q_2^T M_\lambda Q_2)^{-1/(n-p)}} \tag{12}$$

where $Q_2 \in \mathbf{R}^{n\times(n-p)}$ is a matrix whose columns form an orthonormal basis for the nullspace of F^T, i.e., $F^T Q_2 = 0$ and $Q_2^T Q_2 = I$. Furthermore, the GML estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{y^T (I - H(\hat{\lambda}))y}{n - p}$$

where $\hat{\lambda}$ denotes the GML estimate of λ. An implicit representation of the matrix Q_2 can be computed by means of a QR factorization

$$F = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \tag{13}$$

which requires $O(p^2 n)$ flops, but the cost of evaluating (12) by means of a Cholesky factorization of $Q_2^T M_\lambda Q_2$ is $O((n-p)^3)$ flops. We return to the GML objective (12) in section 5 where we show how it can be evaluated in $O(p^2 n)$ flops by exploiting the structure of M_λ.

2.5 Related work

It is well known that for a given parameter λ, the smoothing spline regression problem (1) can be solved in linear time. One of the earliest algorithms is due to Reinsch [13] who derived an $O(n)$ algorithm for the case where $p = 2$. This corresponds to f being a natural cubic spline, which is a popular choice in practice. Reinsch's method is based on a parameterization of natural cubic splines that leads to a banded system of equations of order $n-2$ with bandwidth 5. The approach can be generalized to other values of p, resulting in a system of equations with a banded matrix of order $n-p$ and bandwidth $2p+1$, as shown by Reinsch in a follow-up paper [14]. Hutchinson & de Hood [7] pointed out that only a partial inverse of this band matrix is needed to compute a band of the influence matrix, and it can be computed in $O(p^2 n)$ flops using the recursively technique developed by Erisman & Tinney [3]. Reinsch's approach can also be derived from a stochastic perspective, as shown by Kohn & Ansley [10]. By applying a modified Kalman filter and Kalman smoothing to a stochastic model by Wahba [19], they derived an $O(n)$ algorithm for smoothing spline regression and parameter estimation.

The algorithms that we present in section 4 draw upon an extensive body of literature on semiseparable and rank structured matrices; see, e.g., the books by Vandebril et al. [18, 17] for a comprehensive overview. We note that our algorithm for computing the Cholesky factorization of $\Sigma + \mathbf{diag}(d)$, algorithm 3, is very similar to an algorithm proposed by Foreman–Mackey et al. [4] which computes an LDL decomposition of an extended generator representable p-semiseparable matrix that arises from the so-called *celerite* kernel. However, our results pertaining to the implicit inverse of the Cholesky factor, theorem 2 and algorithm 4, appear to be new.

As a special case of the semiseparable structure of the spline kernel, Chen et al. [2] and Carli et al. [1] pointed out that for $p = 1$, the stable spline kernel generates a kernel matrix that has a tridiagonal inverse. The stable spline kernel is closely related to the spline kernel, and it has applications in system identification. We briefly discuss how our results apply to the stable spline kernel in section 5. Other stochastic processes with semiseparable covariance functions include Brownian motion and the Brownian bridge, and Keiner & Waterhouse [8] exploited the semiseparable structure of the covariance matrix associated with these processes to perform fast principal component analysis.

3 The spline kernel

In this section, we present our main results, namely that $\mathcal{K}_p^1(s,t)$ is a p-semiseparable function and that, as a consequence, kernel matrices generated by $\mathcal{K}_p^1(s,t)$ are rank structured matrices that possess a computationally favorable structure. We start with a formal definition of p-semiseparable functions.

Definition 1. A real-valued function $g(s,t)$ is said to be p-semiseparable (or semiseparable with semiseparability rank p) if

$$g(s,t) = \begin{cases} \sum_{k=1}^{p} u_k(s) v_k(t) & s \geq t \\ \sum_{k=1}^{p} p_k(s) q_k(t) & s < t \end{cases} \tag{14}$$

where u_k, v_k, p_k, and q_k, for $k = 1, \ldots, p$, are univariate functions.

Remark. If the function g is symmetric (i.e., $g(s,t) = g(t,s)$), then $q_k = u_k$ and $p_k = v_k$ for $k = 1, \ldots, p$.

To simplify our notation, we will restrict our attention to a "standardized" version of the spline kernel (7), defined on $[0,1] \times [0,1]$ and given by

$$\kappa_p(s,t) = \int_0^1 G_p(s,u) G_p(t,u)\, du, \quad s,t \in [0,1] \tag{15}$$

where $G_p(s,u) = \max(0, s-u)^{p-1}/(p-1)!$. Using integration by substitution, it is easy to check that $\mathcal{K}_p^1$, which is defined on $[a,b] \times [a,b]$, may be expressed in terms of κ_p as

$$\mathcal{K}_p^1(s,t) = \int_a^b G_p(s,u) G_p(t,u)\, du = (b-a)^{2p-1} \kappa_p\left(\frac{s-a}{b-a}, \frac{t-a}{b-a}\right), \quad s,t \in [a,b].$$

We are now ready to state our first result.

Theorem 1. *The spline kernel of order p can be expressed as*

$$\kappa_p(s,t) = \sum_{k=0}^{p-1} \frac{(-1)^k}{(p-1-k)!(p+k)!} (st)^{p-1-k} \min(s,t)^{2k+1}, \qquad s,t \in [0,1]. \tag{16}$$

The proof is provided in appendix A. As a corollary to theorem 1, we now show that κ_p is p-semiseparable.

Corollary 1. *The spline kernel $\kappa_p(s,t)$ is symmetric and semiseparable with semiseparability rank p.*

Proof. Using theorem 1, we may express $\kappa_p(s,t)$ as

$$\kappa_p(s,t) = \begin{cases} \sum_{k=0}^{p-1} (-1)^k \phi_{p-k}(s) \phi_{p+1+k}(t) & s \geq t \\ \sum_{k=0}^{p-1} (-1)^k \phi_{p-k}(t) \phi_{p+1+k}(s) & s < t \end{cases}$$

where $\phi_k(t) = t^{k-1}/(k-1)!$ for $k = 1, \ldots, 2p$. The desired result follows by observing that this is of the form (14) with $u_k(s) = \phi_{p+1-k}(s)$, $v_k(t) = (-1)^{k-1}\phi_{p+k}(t)$, $p_k(s) = v_k(s)$, and $q_k(t) = u_k(t)$ for $k = 1, \ldots, p$. □

We now turn our attention to symmetric kernel matrices of the form

$$(K_p)_{ij} = \kappa_p(x_i, x_j), \qquad i, j \in \{1, \ldots, n\},$$

i.e., K_p is a symmetric matrix with entries that are generated by the spline kernel κ_p and a sequence $x_1, \ldots, x_n$. Recall that the kernel function κ_p is positive semidefinite, and as a consequence K_p is always positive semidefinite. Moreover, as we will show next, the semiseparable structure of the spline kernel carries over to the corresponding kernel matrices. We will need the following definition before we state the result in corollary 2.

Definition 2. [18, p. 304] A square matrix A of order n is said to be extended $\{p, q\}$-generator representable semiseparable, with natural numbers $p \geq 0$ and $q \geq 0$, if

$$\mathbf{tril}(A) = \mathbf{tril}(UV^T) \tag{17}$$

$$\mathbf{triu}(A) = \mathbf{triu}(PQ^T) \tag{18}$$

where $U, V \in \mathbf{R}^{n\times p}$ and $P, Q \in \mathbf{R}^{n\times q}$ are so-called generators.

Remark. When A is symmetric (i.e., $Q = U$ and $P = V$) we will use the shorthand notation $A = S(U, V)$ where

$$S(U, V) = \mathbf{tril}(UV^T) + \mathbf{triu}(VU^T, 1), \qquad U, V \in \mathbf{R}^{n\times p},$$

and we will say that A is an extended generator representable p-semiseparable matrix, or equivalently, A is an extended generator representable semiseparable matrix with semiseparability rank p.

Corollary 2. *The kernel matrix K_p generated by the spline kernel $\kappa_p(s, t)$ and a monotonic sequence $x_1, \ldots, x_n$ is an extended generator representable semiseparable matrix with semiseparability rank p.*

Proof. Let $x = (x_1, \ldots, x_n)$ and $\phi_k(x) = \frac{1}{(k-1)!}(x_1^{k-1}, \ldots, x_n^{k-1})$ for $k = 1, \ldots, 2p$. We start by assuming that $x_1, \ldots, x_n$ is monotonically increasing. This implies that for $i \geq j$, we have

$$(x_i x_j)^{p-1-k} \min(x_i, x_j)^{2k+1} = x_i^{p-1-k} x_j^{p+k},$$

and hence

$$\mathbf{tril}(K_p) = \sum_{k=0}^{p-1} (-1)^k \, \mathbf{tril}(\phi_{p-k}(x)\phi_{p+1+k}(x)^T).$$

Similarly, if $x_1, \ldots, x_n$ is decreasing, then for $i \geq j$ we have that

$$(x_i x_j)^{p-1-k} \min(x_i, x_j)^{2k+1} = x_i^{p+k} x_j^{p-1-k}$$

and hence

$$\mathbf{tril}(K_p) = \sum_{k=0}^{p-1} (-1)^k \, \mathbf{tril}(\phi_{p+1+k}(x)\phi_{p-k}(x)^T).$$

It follows that K_p is a symmetric extended generator representable semiseparable matrix with semiseparability rank p. □

4 Algorithms

Corollary 2 establishes that the kernel matrix K_p is symmetric extended generator representable semiseparable with semiseparability rank p, i.e., K_p may be expressed as $S(U,V)$ for some $U,V \in \mathbf{R}^{n\times p}$. To simplify notation, we will henceforth write K instead of K_p when p is implied by the column dimension of the generators U and V. We remark that the generator representation is not unique: it is easy to verify from the definition of $S(U,V)$ that $S(U,V) = S(UC, VC^{-T})$ for any nonsingular matrix $C \in \mathbf{R}^{p\times p}$. However, any such generator representation allows us to store the matrix K implicitly using only $O(np)$ memory.

The generator representation $S(U,V)$ allows us to perform several operations such as the matrix–vector product Kx in $O(np)$ flops; see, e.g., [18]. To see this, note that the kth element of Kx can be expressed as

$$e_k^T K x = \sum_{j=1}^{k} u_k^T v_j x_j + \sum_{j=k+1}^{n} v_k^T u_j x_j = u_k^T \bar{v}_k + v_k^T \bar{u}_k$$

where $u_k = U^T e_k$, $v_k = V^T e_k$, $\bar{u}_k = \sum_{j=k+1}^{n} u_j x_j$, and $\bar{v}_k = \sum_{j=1}^{k} v_j x_j$. Notice that $\bar{u}_k$ and $\bar{v}_k$ may be computed recursively as

$$\left.\begin{aligned} \bar{u}_k &= \bar{u}_{k-1} - u_k x_k \\ \bar{v}_k &= \bar{v}_{k-1} + v_k x_k \end{aligned}\right\} \qquad k = 1, \ldots, n$$

where we define $\bar{u}_0 = U^T x$ and $\bar{v}_0 = 0$. Algorithm 1 exploits this recursive definition and evaluates the matrix-vector product Kx in $O(np)$ flops.

Algorithm 1 Matrix–vector product Kx

Input: $U, V \in \mathbf{R}^{n\times p}$ such that $K = S(U,V)$
Output: Overwrites x by Kx

```
Initialization: v̄ ← 0, ū ← U^T x
for k = 1, ..., n do
    v̄ ← v̄ + v_k x_k
    ū ← ū − u_k x_k
    x_k ← u_k^T v̄ + v_k^T ū
end for
```

The technique behind the efficient algorithm for evaluating the matrix–vector product Kx can be generalized to a number of other matrix operations. This, in turn, will allow us to efficiently solve (3) and evaluate the GCV and GML parameter selection criteria. Before we return to this in section 5, we now derive the necessary algorithms. Implementations of these algorithms are available here: `https://git.io/JvYbI`.

We start by showing that if $K = S(U,V)$ is positive definite, it has a Cholesky factorization $K = LL^T$ where L has a generator representation of the form

$$L = \mathbf{tril}(UW^T), \qquad W \in \mathbf{R}^{n\times p}. \tag{19}$$

A similar result was recently shown by Foreman–Mackey et al. [4] for kernel matrices generated by the the so-called celerite kernel. We start by showing that the matrix W must satisfy the equation $LW = V$. To see this, partition K, L, U, V, and W into conformable blocks, i.e.,

$$\begin{bmatrix} K_{11} & K_{21}^T \\ K_{21} & K_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix}, \quad U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}, \quad V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}, \quad W = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}.$$

It follows from the $(2,1)$ block that $K_{21} = L_{21}L_{11}^T$, or equivalently, using the fact that $K_{21} = U_2 V_1^T$, we arrive at the equation $L_{21} = U_2 W_1^T$ where $W_1 = L_{11}^{-1} V_1$. This has to hold for all

Algorithm 2 Cholesky factorization ($K = LL^T$)

Input: $U \in \mathbf{R}^{n\times p}$, $V \in \mathbf{R}^{n\times p}$ such that $K = S(U,V)$ is positive definite
Output: $W \in \mathbf{R}^{n\times p}$ such that $L = \mathbf{tril}(UW^T)$
  Initialization: $P \leftarrow 0$
  for $k = 1, \ldots, n$ **do**
    $w_k \leftarrow v_k - Pu_k$
    $w_k \leftarrow w_k / \sqrt{u_k^T w_k}$
    $P \leftarrow P + w_k w_k^T$
  end for

possible partitions, and since $L_{11} = \mathbf{tril}(U_1 W_1^T)$, we may compute $W = L^{-1}V$ recursively in $O(p^2 n)$ flops using algorithm 2. The generator representation of L may be used to compute matrix-vector products with L, L^T, L^{-1}, and L^{-T} in $O(pn)$ flops. We will omit the details, which can be found in [18], and note that the corresponding algorithms are special cases of the more general algorithms included in appendix C. We now turn our attention to a somewhat more general case.

4.1 Cholesky factorization

Suppose $K + D$ is positive definite where $K = S(U,V)$ is positive semidefinite and $D = \mathbf{diag}(d)$ for some $d \in \mathbf{R}^n_+$. The matrix $K + D$ is a so-called quasiseparable matrix with quasiseparability rank p, and the inverse of such a matrix is itself quasiseparable with quasiseparability rank p [18, Thm. 8.46]. We start by deriving an efficient Cholesky factorization of $K + D$ which requires $O(p^2 n)$ flops. Specifically, we will show that the Cholesky factorization $K + D = LL^T$ yields a factor L that has a generator representation of the form

$$L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c), \qquad W \in \mathbf{R}^{n\times p}, c \in \mathbf{R}^n_{++}. \tag{20}$$

To show this, we introduce the following conformable block partitions

$$U = \begin{bmatrix} U_1 \\ u_k^T \\ U_2 \end{bmatrix}, \quad V = \begin{bmatrix} V_1 \\ v_k^T \\ V_2 \end{bmatrix}, \quad W = \begin{bmatrix} W_1 \\ w_k^T \\ W_2 \end{bmatrix}, \quad L = \begin{bmatrix} L_{11} & 0 & 0 \\ u_k^T W_1^T & c_k & 0 \\ U_2 W_1^T & U_2 w_k & L_{22} \end{bmatrix},$$

where u_k^T, v_k^T, and w_k^T denote the kth row of U, V, and W, respectively. The first k columns of the matrix equation $K + D = LL^T$ can then be expressed as

$$\begin{bmatrix} K_{11} + D_1 & V_1^T u_k \\ u_k^T V_1^T & u_k^T v_k + d_k \\ U_2 V_1^T & U_2 v_k \end{bmatrix} = \begin{bmatrix} L_{11} L_{11}^T & L_{11} W_1 u_k \\ u_k^T W_1^T L_{11}^T & u_k^T W_1^T W_1 u_k + c_k^2 \\ U_2 W_1^T L_{11}^T & U_2(c_k w_k + W_1^T W_1 u_k) \end{bmatrix}$$

where $D_1 = \mathbf{diag}(d_1, \ldots, d_{k-1})$ and $W_1 = L_{11}^{-1} V_1$. It follows from the kth diagonal entry that

$$c_k = \left(u_k^T (v_k - W_1^T W_1 u_k) + d_k\right)^{1/2}, \tag{21}$$

and from the entries below the kth diagonal entry, we obtain the equation

$$w_k = (v_k - W_1^T W_1 u_k)/c_k. \tag{22}$$

Thus, we can compute W and c recursively by defining $P_0 = 0$ and

$$P_k = P_{k-1} + w_k w_k^T, \qquad k = 1, \ldots, n$$

such that $W_1^T W_1 = P_{k-1}$. The resulting algorithm, algorithm 3, computes $W = L^{-1}V$ and c in $O(p^2 n)$ flops. As a special case, note that if K is positive definite and $D = 0$, then (21) and (22) imply that $c_k = u_k^T w_k$, and hence L may also be expressed as (19). Finally, we note that the generator representation (20) allows us to compute the matrix–vector products Lx, $L^T x$, $L^{-1}x$, and $L^{-T}x$ in $O(pn)$ flops using algorithms 7 to 10, included in appendix C.

Algorithm 3 Cholesky factorization ($K + D = LL^T$)

Input: $U, V \in \mathbf{R}^{n\times p}$ and $d \in \mathbf{R}^n_+$ such that $K = S(U,V)$ and $D = \mathbf{diag}(d)$
Output: $W \in \mathbf{R}^{n\times p}$ and $c \in \mathbf{R}^n_{++}$ such that $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$

```
Initialization: P ← 0
for k = 1, ..., n do
    w_k ← v_k − P u_k
    c_k ← (u_k^T w_k + d_k)^{1/2}
    w_k ← w_k / c_k
    P ← P + w_k w_k^T
end for
```

4.2 Inverse of Cholesky factor

The Cholesky factor $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$ is a $\{p, 0\}$-quasiseparable matrix, and it follows from [18, Thm. 8.46] that L^{-1} is itself $\{p, 0\}$-quasiseparable. The following theorem provides a generator representation of L^{-1} when $K + D = LL^T$ and $D = \mathbf{diag}(d)$ for some $d \in \mathbf{R}^n_{++}$.

Theorem 2. *Let $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$ be the Cholesky factor of $K + \mathbf{diag}(d) \succ 0$ where $K = S(U,V)$ is positive semidefinite and $d \in \mathbf{R}^n_{++}$. The inverse of L can then be expressed as*

$$L^{-1} = \mathbf{tril}(YZ^T, -1) + \mathbf{diag}(c)^{-1} \tag{23}$$

where $Y = L^{-1}U$ and $Z = L^{-T}W(W^TY - I)^{-T}$.

Proof. We start by showing that the assumption that $d \in \mathbf{R}^n_{++}$ implies that $W^TY - I$ is nonsingular. Using Sylvester's determinant identity, we may express $\det(I - W^TY)$ as

$$\begin{aligned}
\det(I - L^{-1}UW^T) &= \det(L^{-1})\det(L - UW^T) \\
&= \det(L^{-1})\det(\mathbf{diag}(c) - \mathbf{triu}(UW^T)) \\
&= \left(\prod_{i=1}^n c_i^{-1}\right)\left(\prod_{k=1}^n (c_k - u_k^T w_k)\right).
\end{aligned}$$

It now follows from (21) and (22) that $c_k^2 = u_k^T w_k c_k + d_k$, or equivalently, $c_k - u_k^T w_k = d_k/c_k$ which implies that

$$\det(I - W^TY) = \prod_{k=1}^n \frac{d_k}{c_k^2}. \tag{24}$$

Thus, $W^TY - I$ must be nonsingular since $c, d \in \mathbf{R}^n_{++}$.

To show that the strictly lower-triangular part of L^{-1} is determined by Y and Z, we partition L into blocks as

$$L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}$$

such that L_{11} and L_{22} are both square matrices. The inverse of L may then be expressed as

$$L^{-1} = \begin{bmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}U_2W_1^TL_{11}^{-1} & L_{22}^{-1} \end{bmatrix} \tag{25}$$

where we have used the fact that $L_{21} = U_2 W_1^T$. Using conformable partitions of $Y = L^{-1}U$ and $Z = L^{-T}W(W^TY - I)^{-T}$, i.e.,

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} L_{11}^{-1}U_1 \\ L_{22}^{-1}U_2(I - W_1^T L_{11}^{-1}U_1) \end{bmatrix},$$
$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} L_{11}^{-T}W_1(I - U_2^T L_{22}^{-T}W_2) \\ L_{22}^{-T}W_2 \end{bmatrix} (W^TY - I)^{-T},$$

we may express $Y_2 Z_1^T$ as

$$Y_2 Z_1^T = L_{22}^{-1}U_2(I - W_1^T L_{11}^{-1}U_1)(W^TY - I)^{-1}(I - W_2^T L_{22}^{-1}U_2)W_1^T L_{11}^{-1}. \tag{26}$$

It follows from the definition of Y that the matrix $W^TY - I$ may be expressed as

$$\begin{aligned} W^TY - I &= \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}^T \begin{bmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} - I \\ &= W_1^T L_{11}^{-1}U_1 + W_2^T L_{22}^{-1}U_2 - W_2^T L_{22}^{-1}U_2 W_1^T L_{11}^{-1}U_1 - I \\ &= -(I - W_2^T L_{22}^{-1}U_2)(I - W_1^T L_{11}^{-1}U_1), \end{aligned}$$

and hence the right-hand side of (26) reduces to the $(2,1)$ block of L^{-1} in (25). The $(2,1)$ block of L^{-1} is therefore equal to $Y_2 Z_1^T$. This holds for all block partitions of the form (25), and hence we have that $\mathbf{tril}(L^{-1}, -1) = \mathbf{tril}(YZ^T, -1)$. To complete the proof, we note that the diagonal elements of L^{-1} are given by $1/L_{11}, \ldots, 1/L_{nn}$ (this readily follows from the diagonal of the equation $LL^{-1} = I$), and hence L^{-1} may be expressed as (23). □

Theorem 2 implies that L^{-1} has a generator representation (Y, Z, c) that only requires $O(np)$ memory, and this representation can be computed in $O(p^3 n)$ flops using algorithm 4. As we will show next, theorem 2 allows us to compute the diagonal elements of $(K + D)^{-1}$ and the trace of matrices of the form $(K + D)^{-1}(\widetilde{K} + \widetilde{D})$, where $\widetilde{K} = S(\widetilde{U}, \widetilde{V})$ and $\widetilde{D} = \mathbf{diag}(\tilde{d})$, in $O(p^3 n)$ flops.

Algorithm 4 Inverse of $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$

Input: $U, W \in \mathbf{R}^{n\times p}$, $c \in \mathbf{R}^n_{++}$
Output: $Y, Z \in \mathbf{R}^{n\times p}$ such that $L^{-1} = \mathbf{tril}(YZ^T, -1) + \mathbf{diag}(c)^{-1}$
Compute $Y \leftarrow L^{-1}U$ using algorithm 9
Compute $Z \leftarrow L^{-T}W$ using algorithm 10
Compute $Z \leftarrow Z(U^TZ - I)^{-1}$

Remark. The generator representation of the inverse Cholesky factor (23) requires that d is a positive vector. If $d = 0$ and $K = S(U, V)$ is positive definite, then the Cholesky factor of K still has a generator representation of the form (20), but its inverse is no longer generator representable of the form (23) since $W^TY - I$ is singular (see (24)). As a result, the representation (23) may require high numerical precision to accurately compute all elements of L^{-1} from its generators when one or more elements of d are small.

4.3 Additional algorithms

The kth diagonal element of $(K + D)^{-1}$ can be expressed as $e_k^T(K + D)^{-1}e_k = \|L^{-1}e_k\|_2^2$ where $K + D = LL^T$. We now show that given a generator representation of L^{-1}, all the diagonal elements of $(K + D)^{-1}$ can be computed in $O(p^2 n)$ flops. Indeed, using the generator representation of L^{-1}, we can express $\|L^{-1}e_k\|_2^2$ as

$$c_k^{-2} + \sum_{j=k+1}^{n} (y_j^T z_k)^2 = c_k^{-2} + z_k^T P_k z_k, \qquad P_k = \sum_{j=k+1}^{n} y_j y_j^T, \qquad k = 1, \ldots, n.$$

Noting that $P_n = 0$ and $P_k = P_{k+1} + y_i y_i^T$ for $k = 1, \ldots, n-1$, we can compute all the diagonal elements of $(K+D)^{-1}$ recursively in $O(p^2 n)$ flops using algorithm 5. In section 5, we show how

Algorithm 5 Diagonal elements of $(K+D)^{-1}$

Input: $Y, Z \in \mathbf{R}^{n\times p}$, $c \in \mathbf{R}^n_{++}$ such that $K + D = LL^T$ and

$$L^{-1} = \mathbf{tril}(YZ^T, -1) + \mathbf{diag}(c)^{-1}$$

Output: $b \in \mathbf{R}^n$ such that $b_k = e_k^T(K+D)^{-1}e_k = \|L^{-1}e_k\|_2^2$

```
Initialization: P ← 0
for k = n, ..., 1 do
    b_k ← c_k^{-2} + z_k^T P z_k
    P ← P + y_k y_k^T
end for
```

this algorithm can be used to efficiently compute the diagonal elements of matrices of the form $H(\lambda)$, defined in (10). This is useful for constructing Bayesian credible intervals, as mentioned in section 2.3. We also show how algorithm 5 can be used to efficiently evaluate $\mathbf{tr}(I - H(\lambda))$, and hence also the GCV function (11).

Next we consider the problem of computing the trace of matrices of the form $(K+D)^{-1}(\widetilde{K} + \widetilde{D})$ where $\widetilde{K} = S(\widetilde{U}, \widetilde{V})$ for some $\widetilde{U}, \widetilde{V} \in \mathbf{R}^{n\times\tilde{p}}$ and $\widetilde{D} = \mathbf{diag}(\tilde{d})$ for some $\tilde{d} \in \mathbf{R}^n$. In section 5, we will see that this can be used to efficiently evaluate a partial derivative of a certain log-likelihood function. Assuming that $K + D = LL^T$ with $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$ and $c \in \mathbf{R}^n_{++}$, we have that

$$\mathbf{tr}((K+D)^{-1}(\widetilde{K} + \widetilde{D}) = \sum_{k=1}^{n} e_k^T L^{-1}(\widetilde{K} + \widetilde{D})L^{-T}e_k.$$

The term $e_k^T L^{-1}(\widetilde{K} + \widetilde{D})L^{-T}e_k$ only involves the first k elements of the vector $L^{-T}e_k$ and the leading principal minor of $\widetilde{K} + \widetilde{D}$ of order k, i.e.,

$$\begin{aligned} e_k^T L^{-1}(\widetilde{K} + \widetilde{D})L^{-T}e_k &= \begin{bmatrix} Z_1 y_k \\ c_k^{-1} \end{bmatrix}^T \begin{bmatrix} \widetilde{K}_{11} + \widetilde{D}_1 & \widetilde{V}_1\tilde{u}_k \\ \tilde{u}_k^T\widetilde{V}_1^T & \tilde{u}_k^T\tilde{v}_k + \tilde{d}_k \end{bmatrix} \begin{bmatrix} Z_1 y_k \\ c_k^{-1} \end{bmatrix} \\ &= (\tilde{u}_k^T\tilde{v}_k + \tilde{d}_k)c_k^{-2} + y_k^T Z_1^T(\widetilde{K}_{11} + \widetilde{D}_1)Z_1 y_k + 2y_k^T Z_1^T\widetilde{V}_1\tilde{u}_k c_k^{-1} \end{aligned}$$

where $\widetilde{K}_{11}$ denotes the leading principal minor of $\widetilde{K}$ of order $k-1$, and $\widetilde{V}_1$ and Z_1 denote the first $k-1$ rows of $\widetilde{V}$ and Z, respectively. Now define $R_k = \sum_{i=1}^k z_i\tilde{v}_i^T$, or equivalently, employing a recursive definition,

$$R_k = R_{k-1} + z_k\tilde{v}_k^T, \qquad R_0 = 0.$$

Similarly, we define $P_k = Z^T E_k E_k^T(\widetilde{K} + \widetilde{D})E_k E_k^T Z$ where E_k denotes the first k columns of the identity matrix of order n. By expanding $Z^T E_k E_k^T(\widetilde{K} + \widetilde{D})E_k E_k^T Z$, we can obtain a recursive definition of P_k, i.e.,

$$\begin{aligned} P_k &= \begin{bmatrix} Z_1 \\ z_k^T \end{bmatrix}^T \begin{bmatrix} \widetilde{K}_{11} + \widetilde{D}_1 & \widetilde{V}_1\tilde{u}_k \\ \tilde{u}_k^T\widetilde{V}_1^T & \tilde{u}_k^T\tilde{v}_k + \tilde{d}_k \end{bmatrix} \begin{bmatrix} Z_1 \\ z_k^T \end{bmatrix} \\ &= P_{k-1} + (\tilde{u}_k^T\tilde{v}_k + \tilde{d}_k)z_k z_k^T + z_k\tilde{u}_k^T\widetilde{V}_1^T Z_1 + Z_1^T\widetilde{V}_1\tilde{u}_k z_k^T \\ &= P_{k-1} + (\tilde{u}_k^T\tilde{v}_k + \tilde{d}_k)z_k z_k^T + z_k\tilde{u}_k^T R_{k-1}^T + R_{k-1}\tilde{u}_k z_k^T \end{aligned}$$

where we define $P_0 = 0$. It follows that

$$e_k^T L^{-1}(\widetilde{K} + \widetilde{D})L^{-T}e_k = y_k^T P_{k-1} y_k + 2y_k^T R_{k-1}\tilde{u}_k c_k^{-1} + (\tilde{u}_k^T\tilde{v}_k + \tilde{d}_k)c_k^{-2},$$

and hence

$$\mathbf{tr}((K+D)^{-1}(\widetilde{K}+\widetilde{D})) = \sum_{k=1}^{n}\left(y_k^T P_{k-1} y_k + 2y_k^T R_{k-1}\tilde{u}_k c_k^{-1} + (\tilde{u}_k^T\tilde{v}_k + \tilde{d}_k)c_k^{-2}\right).$$

Algorithm 6 evaluates this in $O(p\tilde{p}n)$ flops using the recursive definitions of R_k and P_k. As a special case of this algorithm, we mention that letting $\widetilde{K} = 0$ and $\tilde{d} = \mathbf{1}$ yields the trace of $(K+D)^{-1}$. However, we note that algorithm 6 cannot be used to compute the diagonal elements of $(K+D)^{-1}$, so it cannot replace algorithm 5.

Algorithm 6 Trace of $L^{-1}(\widetilde{K}+\widetilde{D})L^{-T}$

Input: $\widetilde{U}, \widetilde{V} \in \mathbf{R}^{n\times\tilde{p}}$, $\tilde{d} \in \mathbf{R}^n$ and $Y, Z \in \mathbf{R}^{n\times p}$, $c \in \mathbf{R}^n_{++}$ such that

$$\widetilde{K} = S(\widetilde{U}, \widetilde{V}), \qquad \widetilde{D} = \mathbf{diag}(\tilde{d}), \qquad L^{-1} = \mathbf{tril}(YZ^T, -1) + \mathbf{diag}(c)^{-1}$$

Output: $b \in \mathbf{R}$ such that $b = \mathbf{tr}(L^{-1}(\widetilde{K}+\widetilde{D})L^{-T})$

Initialization: $b \leftarrow 0$, $P \leftarrow 0$, $R \leftarrow 0$

for $k = 1, \ldots, n$ **do**

$b \leftarrow b + y_k^T P y_k + 2y_k^T R\tilde{u}_k c_k^{-1} + (\tilde{u}_k^T\tilde{v}_k + \tilde{d}_k)c_k^{-2}$

$P \leftarrow P + (\tilde{u}_k^T\tilde{v}_k + \tilde{d}_k)z_k z_k^T + z_k(R\tilde{u}_k)^T + (R\tilde{u}_k)z_k^T$

$R \leftarrow R + z_k\tilde{v}_k^T$

end for

5 Applications

We now discuss some applications of the algorithms introduced in section 4. We start by revisiting the smoothing spline regression problem (1), and next we turn to applications in Gaussian process regression.

5.1 Smoothing spline regression

Recall that the solution to the smoothing spline regression problem (1) can be expressed as (8) where $(\alpha^\star, \beta^\star)$ is a solution to the system of equations (3). Rearranging (3) yields the equivalent system of equations

$$F^T M_\lambda^{-1} F\beta^\star = F^T M_\lambda^{-1} y$$
$$M_\lambda \alpha^\star = y - F\beta^\star.$$

The matrix $M_\lambda = \Sigma + n\lambda I$ can be factorized as $M_\lambda = LL^T$ using algorithm 3 in $O(p^2 n)$ flops, and we can also compute $B = L^{-1}F$ and the "thin" QR factorization $B = QR$, where $Q \in \mathbf{R}^{n\times p}$ and $R \in \mathbf{R}^{p\times p}$, in $O(p^2 n)$ flops. This allows us to reduce (3) to

$$R\beta^\star = Q^T L^{-1} y \tag{27}$$
$$L^T\alpha^\star = (I - QQ^T)L^{-1}y, \tag{28}$$

and hence we can solve for $(\alpha^\star, \beta^\star)$ in $O(p^2 n)$ flops. The resulting spline interpolates the points $(x_1, \hat{y}_1), \ldots, (x_n, \hat{y}_n)$ where $\hat{y} = \Sigma\alpha^\star + F\beta^\star = y - n\lambda\alpha^\star$.

The GCV objective (11) involves the matrix-vector product $(I - H(\lambda))y$ and the trace of $I - H(\lambda)$ where $H(\lambda)$ is the influence matrix (10). Using the factorizations $M_\lambda = LL^T$ and $L^{-1}F = QR$, we may rewrite $I - H(\lambda)$ as

$$\begin{aligned} I - H(\lambda) &= n\lambda(M_\lambda^{-1} - M_\lambda^{-1}F(F^T M_\lambda^{-1}F)^{-1}F^T M_\lambda^{-1}) \\ &= n\lambda L^{-T}(I - QQ^T)L^{-1}. \end{aligned} \tag{29}$$

Moreover, $(I - H(\lambda))y = n\lambda\alpha^\star$ and

$$\mathbf{tr}(I - H(\lambda)) = n\lambda \left(\mathbf{tr}(M_\lambda^{-1}) - \|L^{-T}Q\|_F^2\right).$$

The term $\mathbf{tr}(M_\lambda^{-1})$ can be evaluated in $O(p^3 n)$ flops using algorithms 4 and 5, and it requires $O(p^2 n)$ flops to compute $L^{-T}Q$ and its Frobenius norm. We note that in finite precision, algorithm 4 should be avoided when λ is small: the generator representation of the inverse Cholesky factor becomes unfavorable from a numerical point of view when λ approaches zero (see section 4.2). An alternative when λ is small is to compute $\mathbf{tr}(M_\lambda^{-1})$ as $\sum_{i=1}^n \|L^{-1}e_i\|^2$ in $O(pn^2)$ flops using algorithm 9.

To evaluate the GML objective (12), we note that it can be expressed as

$$\begin{aligned}\mathrm{GML}(\lambda) &= \frac{y^T(I - H(\lambda))y}{n\lambda[\det(M_\lambda)^{-1}\det(F^T M_\lambda^{-1}F)^{-1}\det(F^T F)]^{1/(n-p)}} \\ &\propto \frac{y^T(I - H(\lambda))y}{n\lambda \det(M_\lambda)^{-1/(n-p)}\det(F^T M_\lambda^{-1}F)^{-1/(n-p)}},\end{aligned} \tag{30}$$

as shown in appendix B. Using the fact that $(I - H(\lambda))y = n\lambda\alpha^\star$ and the factorizations $M_\lambda = LL^T$ and $L^{-1}F = QR$, the expression (30) can be simplified as

$$y^T\alpha^\star \det(L)^{2/(n-p)}\det(R)^{2/(n-p)} \tag{31}$$

which is readily evaluated in $O(n)$ flops.

5.2 Gaussian process regression

The semiseparable structure of Σ also has applications in Gaussian process regression. As an example, we consider the following generalization of the observation model (9)

$$g(t) = \sum_{k=1}^p \theta_k \phi_k(t) + \nu X(t), \qquad t \in [a, b], \tag{32a}$$

$$y_i = \mathcal{L}_i g + \epsilon_i, \qquad i = 1, \ldots, m, \tag{32b}$$

where $X(t)$ is a zero-mean Gaussian process with covariance function $\mathcal{K}_p^1$, the functionals $\mathcal{L}_1, \ldots, \mathcal{L}_m$ are bounded and linear, and $\theta \sim \mathcal{N}(0, \gamma I)$ and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Moreover, we will assume that θ and $X(t)$ are independent. It is easy to see that the model (9) is obtained as a special case of (32) if we let $m = n$ and define $\mathcal{L}_i g = g(t_i)$ for $i = 1, \ldots, n$. We note that the model (32) is closely related to the so-called *general* smoothing spline regression problem [20]

$$\text{minimize} \quad \widetilde{\mathcal{J}}(f) \equiv \frac{1}{m}\sum_{i=1}^n (y_i - \mathcal{L}_i f)^2 + \lambda \int_a^b |f^{(p)}(t)|^2\, dt$$

where the functional $\widetilde{\mathcal{J}}$ is defined on $W_p^2[a, b]$.

As a special case of (32), we will focus on an observation model with $\gamma = 0$ (implying that $\theta = 0$) and discrete observations

$$\mathcal{L}_i g = \sum_{j=1}^n A_{ij} g(t_j), \qquad i = 1, \ldots, m,$$

where $t_j \in [a, b]$ for $j = 1, \ldots, n$, and A_{ij} is the (i, j) entry of a given matrix $A \in \mathbf{R}^{m\times n}$. The vector of observations can then be expressed as

$$y = Ax + \epsilon \tag{33}$$

where the jth entry of $x \in \mathbf{R}^n$ is $x_j = g(t_j)$, and hence $x \sim \mathcal{N}(0, \nu^2\Sigma)$ where the (i,j) entry of Σ is given by $\mathcal{K}_p^1(t_i, t_j)$. It follows that the posterior distribution of x is given by

$$x \mid y, \sigma, \nu \sim \mathcal{N}(\sigma^{-2}\Sigma_{w|y}A^T y, \Sigma_{w|y})$$

where $\Sigma_{w|y} = (\nu^{-2}\Sigma^{-1} + \sigma^{-2}A^TA)^{-1}$. The posterior mean $\sigma^{-2}\Sigma_{w|y}A^Ty$ can also be expressed as $\hat{x} = \nu^2\Sigma\alpha^\star$ where

$$\alpha^\star = (\sigma^2 I + \nu^2 A\Sigma A^T)^{-1}y. \tag{34}$$

Moreover, the covariance function $\mathcal{K}_p^1$ is a reproducing kernel for $\mathcal{H}_1$, and the function

$$\hat{g}(t) = \nu^2 \sum_{i=1}^n \alpha_i^\star \mathcal{K}_p^1(t_i, t), \qquad t \in [a, b],$$

interpolates the points $(t_i, \hat{x}_i)$ for $i = 1, \ldots, n$.

The hyperparameters ν and σ can be estimated by maximizing the likelihood function associated with the marginal distribution

$$y \mid \sigma, \nu \sim \mathcal{N}(0, \sigma^2 I + \nu^2 A\Sigma A^T).$$

Expressing the covariance matrix as $\nu^2(A\Sigma A^T + m\lambda I)$ with $\lambda = \sigma^2/(m\nu^2)$, the negative log-likelihood (up to an additive constant) may be expressed as

$$\psi(\lambda, \nu \mid y) = \nu^{-2}y^T(A\Sigma A^T + m\lambda I)^{-1}y + \log\det(A\Sigma A^T + m\lambda I) - m\log(\nu^{-2}) \tag{35}$$

with domain $\mathbf{dom}\,\psi = \mathbf{R}_{++} \times \mathbf{R}_{++}$. Note that $\psi(\lambda, \nu \mid y)$ is strictly convex with respect to ν^{-2}. Taking the derivative with respect to ν^{-2} and setting it equal to zero yields

$$\nu^2 = \frac{y^T(A\Sigma A^T + m\lambda I)^{-1}y}{m},$$

and by substituting this expression for ν^2 in (35), we obtain the univariate function

$$\tilde{\psi}(\lambda \mid y) = m\log(y^T(A\Sigma A^T + m\lambda I)^{-1}y) + \log\det(A\Sigma A^T + m\lambda I) + m. \tag{36}$$

Minimizing (36) yields an estimate of λ, and a local minimum can be found using, e.g., Newton's method or a derivative-free method such as golden section search. We note that the positivity condition $\lambda > 0$ can be handled implicitly by means of a change of variables, e.g., by substituting e^μ for λ with $\mu \in \mathbf{R}$.

We now discuss how the semiseparable structure of Σ can be used to reduce the computational cost of estimating the hyperparameters (λ, ν) and the cost of computing $\alpha^\star$. We will assume that $a < t_1 < t_2 < \cdots < t_n < b$ such that Σ is positive definite and can be factorized as $\Sigma = LL^T$ in $O(p^2n)$ flops using algorithm 2. We will also assume that the rank of A is $r = \min(m, n)$, i.e., A has full rank. In the special case where $A = I$, the estimation problem can be solved in $O(p^3n)$ flops using the algorithms from section 4, as outlined in section 5.1. More generally, the matrix $B = AL$ can be computed in $O(mnp)$ flops by exploting the structure of L, and a "thin" singular value decomposition $B = USV^T$ can be computed in $O(\max(m,n)r^2)$ flops where $U \in \mathbf{R}^{m\times r}$, $S = \mathbf{diag}(\sigma_1, \ldots, \sigma_r)$, and $V \in \mathbf{R}^{n\times r}$; see, e.g., [6]. We start by considering the case where $m \leq n$. This implies that $r = m$ and

$$A\Sigma A^T + m\lambda I = BB^T + r\lambda I = U(S^2 + r\lambda I)U^T. \tag{37}$$

Letting $\tilde{y} = U^Ty$, which can be computed in $O(r^2)$ flops, we can express (36) as

$$\tilde{\psi}(\lambda \mid y) = r\log\left(\sum_{i=1}^r \frac{\tilde{y}_i^2}{\sigma_i^2 + r\lambda}\right) + \sum_{i=1}^r \log(\sigma_i^2 + r\lambda) + r, \tag{38}$$

and hence the complexity of evaluating $\tilde{\psi}$ is $O(r)$. Furthermore, using (37), we have that

$$\nu^2 = \frac{\tilde{y}^T(S^2 + r\lambda I)^2\tilde{y}}{r}, \qquad \alpha^\star = \frac{1}{\nu^2}U(S^2 + r\lambda I)^{-1}\tilde{y}.$$

Interestingly, minimizing (38) can be viewed as minimizing the ratio of the geometric mean (GM) to the harmonic mean (HM) of the sequence $(\sigma_i^2 + r\lambda)/\tilde{y}_i^2$ for $i = 1, \ldots, r$, i.e., (38) satisfies

$$\exp\left(\tilde{\psi}(\lambda \mid y)\right) \propto \frac{\left(\prod_{i=1}^r \frac{\sigma_i^2+r\lambda}{\tilde{y}_i^2}\right)^{1/r}}{\left(\sum_{i=1}^r \frac{\tilde{y}_i^2}{\sigma_i^2+r\lambda}\right)^{-1}},$$

provided that $\tilde{y}_i \neq 0$ for $i = 1, \ldots, n$. The GM-HM inequality implies that the GM-to-HM ratio is greater than or equal to 1. Moreover, the arithmetic mean of a positive sequence is an upper bound on its GM, and this leads to the following upper bound

$$\frac{\left(\prod_{i=1}^r \frac{\sigma_i^2+r\lambda}{\tilde{y}_i^2}\right)^{1/r}}{\left(\sum_{i=1}^r \frac{\tilde{y}_i^2}{\sigma_i^2+r\lambda}\right)^{-1}} \leq \frac{\sum_{i=1}^r(\sigma_i^2 + r\lambda)/\tilde{y}_i^2}{r\left(\sum_{i=1}^r \frac{\tilde{y}_i^2}{\sigma_i^2+r\lambda}\right)^{-1}}. \tag{39}$$

The harmonic mean is a concave function on $\mathbf{R}^n_{++}$, and hence the upper bound is a quasiconvex function of λ. However, it is not clear if the left-hand side of (39) is itself a quasiconvex function of λ.

Next, we consider the case where $m \geq n$. We then have $r = n$ and

$$\begin{aligned}(A\Sigma A^T + m\lambda I)^{-1} &= \frac{1}{m\lambda}(I - B(m\lambda I + B^TB)^{-1}B^T) \\ &= \frac{1}{m\lambda}(I - US(m\lambda I + S^2)^{-1}SU^T)\end{aligned} \tag{40}$$

which follows from the Woodbury identity and the decomposition $B = USV^T$. Letting $\tilde{y} = U^Ty$, which can be computed in $O(mn)$ flops, we can express (36) as

$$\tilde{\psi}(\lambda \mid y) = m\log\left(\|y\|_2^2 - \sum_{i=1}^r \frac{\tilde{y}_i^2\sigma_i^2}{\sigma_i^2 + m\lambda}\right) - \sum_{i=1}^r \log\left(\frac{m\lambda}{\sigma_i^2 + m\lambda}\right) + m$$

which has evaluation complexity $O(r)$. Finally, using (40), we arrive at

$$\nu^2 = \frac{\|y\|_2^2 - \tilde{y}^TS(m\lambda + S^2)^{-1}S\tilde{y}}{m^2\lambda}, \qquad \alpha^\star = \frac{1}{m\lambda\nu^2}(y - US(m\lambda I + S^2)^{-1}S\tilde{y}).$$

Thus, the computational cost of estimating the hyperparameters and computing $\alpha^\star$ is therefore at most $O(\max(m,n)r^2)$, regardless of m and n. Note that without exploiting the structure of Σ, the computational bottleneck is either forming and factorizing $A\Sigma A^T$ (which costs $O(r\max(m,n)^2 + m^3)$ flops) or forming and factorizing $B = AL$ (which costs $O(n^3 + mn^2 + \max(m,n)r^2)$ flops). Consequently, if the semiseparable structure of Σ is ignored, the computational cost is $O(mn^2)$ instead of $O(r^2\max(m,n))$.

5.3 Kernel warping

New kernel functions can be constructed from the spline kernel κ_p by means of a technique known as kernel warping. Specifically, by introducing a transformation $\eta\colon I \to [0,1]$ where I is a subset of $\mathbf{R}$, we can construct a new kernel as

$$\widetilde{\mathcal{K}}(s,t) = \kappa_p(\eta(s), \eta(t)), \qquad s,t \in I.$$

An example of such a transformation is the monotonic transformation $\eta(t) = e^{-\rho t}$, defined on $I = [0, \infty)$ and with parameter $\rho > 0$, which yields the so-called stable spline kernel [12]

$$\kappa_p^{\text{ss}}(s, t; \rho) = \kappa_p(e^{-\rho s}, e^{-\rho t}), \qquad s, t \in [0, \infty). \tag{41}$$

The stable spline kernel was introduced in the context of system identification as a way to construct a prior that ensures stability of a dynamic system. Corollary 2, combined with the monotonicity of the transformation $t \mapsto e^{-\rho t}$, implies that the kernel matrix $K_p^{\text{ss}}(\rho)$ generated by the stable spline kernel $\kappa_p^{\text{ss}}(s, t; \rho)$ and a monotonic sequence $t_1, \ldots, t_n$ inherits the semiseparable structure of the spline kernel. In other words, $K_p^{\text{ss}}(\rho)$ is an extended generator representable semiseparable matrix with semiseparability rank p, and hence $K_p^{\text{ss}}(\rho) = S(U, V)$ for some $U, V \in \mathbf{R}^{n \times p}$. We note that for a general transformation η (not necessarily monotonic) and a sequence $t_1, \ldots, t_n$, the warped kernel $\widetilde{\mathcal{K}}$ generates an extended generator representable matrix with semiseparability rank p up to a symmetric permutation. We note that a recent example of the use of kernel warping to derive new kernels from the spline kernel can be found in [5].

We now outline how the algorithms from section 4 can be useful for Gaussian process regression using a warped version of the spline kernel as covariance function. As an example, we will consider an instance of the Gaussian linear model (33) where the covariance matrix associated with x is $\nu^2 K_p^{\text{ss}}(\rho)$ instead of $\nu^2 \Sigma$. Moreover, we will treat the parameter ρ as an unknown that should be estimated along with ν and λ. Eliminating ν from the likelihood function, we arrive at

$$\tilde{\psi}(\lambda, \rho \mid y) = m \log(y^T (A K_p^{\text{ss}}(\rho) A^T + m\lambda I)^{-1} y) + \log\det(A K_p^{\text{ss}}(\rho) A^T + m\lambda I) + m$$

which can be evaluated in $O(\max(m, n) r^2)$ flops using the same approach as in section 5.2. Moreover, the partial derivatives of $\tilde{\psi}$ with respect to λ and ρ can be expressed as

$$\begin{aligned}
\frac{\partial}{\partial \lambda} \tilde{\psi}(\lambda, \rho \mid y) &= -m^2 \frac{\|\tilde{c}\|_2^2}{y^T \tilde{c}} + m \,\mathbf{tr}\left(C^{-1}\right) \\
\frac{\partial}{\partial \rho} \tilde{\psi}(\lambda, \rho \mid y) &= -m \frac{\tilde{c}^T A \frac{dK_p^{\text{ss}}(\rho)}{d\rho} A^T \tilde{c}}{y^T \tilde{c}} + \mathbf{tr}\left(C^{-1} \frac{dK_p^{\text{ss}}(\rho)}{d\rho}\right)
\end{aligned}$$

where $C = A K_p^{\text{ss}}(\rho) A^T + m\lambda I$ and $\tilde{c} = C^{-1} y$. The derivative of the stable spline kernel with respect to the parameter ρ is itself a semiseparable function with semiseparability rank $2p - 1$. To see this, note that for $s \geq t$, theorem 1 implies that

$$\begin{aligned}
\frac{d}{d\rho} \kappa_p^{\text{ss}}(s, t; \rho) &= \sum_{k=0}^{p-1} \frac{(-1)^k}{(p-1-k)!(p+k)!} \frac{d}{d\rho} e^{-\rho(p+k)s} e^{-\rho(p-1-k)t} \\
&= -\sum_{k=0}^{p-1} \frac{(-1)^k ((p+k)s + (p-1-k)t) e^{-\rho(p+k)s} e^{-\rho(p-1-k)t}}{(p-1-k)!(p+k)!}.
\end{aligned}$$

It is easy to check that the right-hand side is a sum of $2p - 1$ multiplicatively separable terms (since $(p - 1 - k)t$ vanishes when $k = p - 1$), and by the symmetry of κ_p^{ss}, the derivative $\frac{d}{d\rho} \kappa_p^{\text{ss}}$ is therefore semiseparable with semiseparability rank at most $2p - 1$. Consequently, $\frac{d}{d\rho} K_p^{\text{ss}}(\rho)$ may be represented as

$$\frac{d}{d\rho} K_p^{\text{ss}}(\rho) = S(\widetilde{U}, \widetilde{V}), \qquad \widetilde{U}, \widetilde{V} \in \mathbf{R}^{n \times (2p-1)},$$

and hence $\mathbf{tr}\left(C^{-1} \frac{dK_p^{\text{ss}}(\rho)}{d\rho}\right)$ can be computed in $O(mnp)$ flops given a singular value decomposition $B = USV^T$ where $B = AL$ and $K_p^{\text{ss}}(\rho) = LL^T$. We note that in the special case where $A = I$, algorithms 4 and 6 can be used to evaluate the trace of $(K_p^{\text{ss}}(\rho) + n\lambda I)^{-1} \frac{d}{d\rho} K_p^{\text{ss}}(\rho)$ in $O(p^3 n)$ flops without forming the matrix product.

6 Numerical example

To illustrate the efficiency of the algorithms derived in this paper, we now compare the execution time for solving (3) and computing $\hat{y}$ when $p = 2$ using (i) Reinsch's algorithm [13] and (ii) the semiseparable structure of the spline kernel, as outlined in section 5.1. We implemented both algorithms in MATLAB in an attempt to make a fair comparison. Our implementation of Reinsch's algorithm is based on sparse matrices and MATLAB's built-in sparse Cholesky factorization. The implementation of our algorithm is based on algorithms 3 and 9, both of which we implemented as MATLAB MEX files written in C using a row-major representation of the generators U, V, and W. Table 1 shows the average execution time in milliseconds as a function of n and based on $10^7/n$ repetitions. We used $\lambda = 10^{-9}$, and for each value of n, we generated a problem instance with observations

$$x_i = \frac{i-1}{n-1}, \qquad y_i = \cos(2\pi x_i) + 0.3\sin(10\pi x_i) + \epsilon_i, \qquad i = 1, \ldots, n,$$

where $\epsilon_1, \ldots, \epsilon_n$ are realizations of a zero-mean Gaussian random variable with standard deviation 0.1. The results confirm that the complexity is linear in n for both algorithms, and while our algorithm is roughly 3-5 times faster than Reinsch's algorithm, we note that an implementation of Reinsch's algorithm based on band storage and suitable LAPACK routines would likely improve its performance.

n	Reinsch	Semiseparable	Ratio
1000	0.78	0.23	3.4
2000	1.59	0.39	4.0
4000	2.99	0.81	3.7
8000	7.29	1.54	4.7
16000	15.47	2.97	5.2
32000	32.55	6.15	5.3
64000	70.29	13.10	5.4

Table 1: Average execution time in milliseconds.

7 Conclusions

We have shown that the spline kernel of order p is a semiseparable function with semiseparability rank p. Building on this result, we have constructed efficient, recursive algorithms for key computations that arise in smoothing spline regression, Gaussian process regression, and related hyperparameter estimation problems. The complexity of these algorithms grows linearly with the number of knots, and hence they match the complexity of the best, known algorithms for smoothing spline regression such as Reinsh's algorithm [13, 14]. More importantly, theorem 2 and the algorithms derived in section 4 are not limited to kernel matrices generated by the spline kernel, so their potential reach may extend beyond that of existing methods for smoothing spline regression.

A natural next step would be to extend our results to tensor-product splines defined on d-dimensional rectilinear grids. In two dimensions, a tensor-product spline would result in a kernel matrix that can be expressed as a Kronecker product of two rank structured matrices. A potential application of this is spatial–temporal modeling where it may be natural to assume that that the spatial and temporal dimensions are separable.

A Proof of Theorem 1

Proof. Recall the definition of the spline kernel (15). This can also be expressed as

$$\kappa_p(s,t) = \int_0^{\min(s,t)} \phi_p(s;u)\phi_p(t;u)\,du, \qquad s,t \in [0,1], \tag{42}$$

where $\phi_k(t;u) = \frac{(t-u)^{k-1}}{(k-1)!}$ for k integral and positive, and we define $\phi_1(t;t) = 1$. To simplify the notation when $u = 0$, we define $\phi_k(t) = \phi_k(t;0)$.

We start by noting that for $p = 1$, we have $\kappa_1(s,t) = \min(s,t)$. For $p \geq 2$, we may use integration by parts combined with the fact that

$$\frac{d}{du}\phi_k(t;u) = \begin{cases} -\phi_{k-1}(t;u) & k \geq 2 \\ 0 & k = 1 \end{cases}$$

to express (42) as

$$\kappa_p(s,t) = \Big[-\phi_p(s;u)\phi_{p+1}(t;u)\Big]_{u=0}^{\min(s,t)} - \int_0^{\min(s,t)} \phi_{p-1}(s;u)\phi_{p+1}(t;u)\,du\,.$$

Expanding the integral on the right-hand side by repeated use of integration by parts, we arrive at the expression

$$\kappa_p(s,t) = \sum_{k=0}^{p-1}(-1)^k\Big[-\phi_{p-k}(s;u)\phi_{p+1+k}(t;u)\Big]_{u=0}^{\min(s,t)}$$

which for $s \geq t$ simplifies to

$$\kappa_p(s,t) = \sum_{k=0}^{p-1}(-1)^k\phi_{p-k}(s)\phi_{p+1+k}(t), \qquad s \geq t.$$

Using the fact that $\kappa_p(s,t) = \kappa_p(t,s)$ and the definition of ϕ_k, we arrive at

$$\kappa_p(s,t) = \sum_{k=0}^{p-1}\frac{(-1)^k}{(p-1-k)!(p+k)!}(st)^{p-1-k}\min(s,t)^{2k+1}, \qquad s,t \in [0,1], \tag{43}$$

which holds for $p \geq 1$. □

B Generalized likelihood function

We now consider the stochastic process defined in section 2.3 and derive the generalized likelihood function associated with the conditional distribution of y given the parameters λ, ν, and γ. We have that

$$y \mid \lambda, \nu, \gamma \sim \mathcal{N}(0, \nu^2 M_\lambda + \gamma F F^T),$$

and we are interested in the case where $\gamma \to \infty$, corresponding to an improper prior on the parameter vector θ. The negative log-likelihood function may be expressed as

$$\psi(\lambda, \nu, \gamma \mid y) = \frac{1}{2}y^T(\nu^2 M_\lambda + \gamma F F^T)^{-1}y + \frac{1}{2}\log\det(\nu^2 M_\lambda + \gamma F F^T) + \frac{n}{2}\log(2\pi),$$

and it is easy to check that its limit as $\gamma \to \infty$ is unbounded. The standard approach to this problem is to project y onto the nullspace of F^T. Specifically, if we let $w = Q_2^T y$, where Q_2 is obtained from the QR factorization (13), we may consider the conditional distribution

$$w \mid \lambda, \nu^2 \sim \mathcal{N}(0, \nu^2 Q_2^T M_\lambda Q_2)$$

and the corresponding negative log-likelihood function

$$\begin{aligned}\psi(\lambda,\nu \mid w) = \frac{\nu^{-2}}{2} w^T (Q_2^T M_\lambda Q_2)^{-1} w + \frac{1}{2}\log\det(Q_2^T M_\lambda Q_2) \\ + \frac{n-p}{2}\log(\nu^2) + \frac{n}{2}\log(2\pi).\end{aligned} \tag{44}$$

Setting the derivative with respect to ν^{-2} equal to zero and solving for ν^2 yields the optimality condition

$$\nu^2 = \frac{w^T (Q_2^T M_\lambda Q_2)^{-1} w}{n-p},$$

and using this expression in (44), we arrive at the one-dimensional profile

$$\tilde{\psi}(\lambda \mid w) = \frac{1}{2}\log\det(Q_2^T M_\lambda Q_2) + \frac{n-p}{2}\log(w^T (Q_2^T M_\lambda Q_2)^{-1} w) + \zeta \tag{45}$$

where ζ is a constant. It is easy to check that the function $\mathrm{GML}(\lambda)$, defined in (12), is proportional to $\exp(\tilde{\psi}(\lambda \mid w))$.

The function (45) may be rewritten as

$$\tilde{\psi}(\lambda \mid y) = \frac{1}{2}\log\det(M_\lambda)\det(F^T M_\lambda^{-1} F) + \frac{n-p}{2}\log(\lambda^{-1} y^T (I - H(\lambda))y) + \tilde{\zeta} \tag{46}$$

where $\tilde{\zeta}$ is a constant. To see this, first note that Schur's determinant identity implies that

$$\det\left(\begin{bmatrix}\nu^2 M_\lambda & F \\ -F^T & \gamma^{-1} I\end{bmatrix}\right) = \det(\nu^2 M_\lambda)\det(\gamma^{-1} I + \nu^{-2} F^T M_\lambda^{-1} F). \tag{47}$$

Now, by applying the similarity transformation

$$\begin{bmatrix}Q & 0 \\ 0 & I\end{bmatrix}^T \begin{bmatrix}\nu^2 M_\lambda & F \\ -F^T & \gamma^{-1} I\end{bmatrix} \begin{bmatrix}Q & 0 \\ 0 & I\end{bmatrix} = \begin{bmatrix}\nu^2 Q_1^T M_\lambda Q_1 & \nu^2 Q_1^T M_\lambda Q_2 & R_1 \\ \nu^2 Q_2^T M_\lambda Q_1 & \nu^2 Q_2^T M_\lambda Q_2 & 0 \\ -R_1^T & 0 & \gamma^{-1} I\end{bmatrix}$$

where $Q = \begin{bmatrix}Q_1 & Q_2\end{bmatrix}$ is the QR factorization (13) of F, we obtain an equivalent expression by applying Schur's determinant identity to the $(2,2)$ block of the right-hand side, i.e.,

$$\begin{aligned}\det\left(\begin{bmatrix}\nu^2 M_\lambda & F \\ -F^T & \gamma^{-1} I\end{bmatrix}\right) &= \det(\nu^2 Q_2^T M_\lambda Q_2)\det\left(\begin{bmatrix}C & R_1 \\ -R_1^T & \gamma^{-1} I\end{bmatrix}\right) \\ &= \det(\nu^2 Q_2^T M_\lambda Q_2)\det(C)\det(\gamma^{-1} I + R_1^T C^{-1} R_1)\end{aligned} \tag{48}$$

where

$$C = \nu^2 Q_1^T M_\lambda Q_1 - \nu^2 Q_1^T M_\lambda Q_2 (Q_2^T M_\lambda Q_2)^{-1} Q_2^T M_\lambda Q_1.$$

Equating (47) and (48), and taking the limit as $\gamma \to \infty$, we arrive at

$$\det(Q_2^T M_\lambda Q_2) = \det(M_\lambda)\det(F^T M_\lambda^{-1} F)\det(F^T F)^{-1} \tag{49}$$

where $\det(F^T F) = \det(R_1^T R_1)$. Finally, to show that $w^T (Q_2^T M_\lambda Q_2)^{-1} w \propto \lambda^{-1} y^T (I - H(\lambda))y$, first note that the Woodbury identity implies that

$$(\nu^2 M_\lambda + \gamma F F^T)^{-1} = \nu^{-2}(M_\lambda^{-1} - M_\lambda^{-1} F(\nu^2 \gamma^{-1} I_p + F^T M_\lambda^{-1} F)^{-1} F^T M_\lambda^{-1}),$$

and hence

$$\begin{aligned}\lim_{\gamma\to\infty}(\nu^2 M_\lambda + \gamma F F^T)^{-1} &= \nu^{-2}(M_\lambda^{-1} - M_\lambda^{-1} F(F^T M_\lambda^{-1} F)^{-1} F^T M_\lambda^{-1}) \\ &= \frac{\nu^{-2}}{n\lambda}(I - H(\lambda)).\end{aligned} \tag{50}$$

Moreover, it is staightforward (but tedious) to show that

$$\lim_{\gamma\to\infty}(Q^T(\nu^2 M_\lambda + \gamma FF^T)Q)^{-1} = \nu^{-2}\begin{bmatrix}0 & 0\\ 0 & (Q_2^T M_\lambda Q_2)^{-1}\end{bmatrix},$$

and this implies that

$$\lim_{\gamma\to\infty} Q(Q^T(\nu^2 M_\lambda + \gamma FF^T)Q)^{-1}Q^T = \nu^{-2}Q_2(Q_2^T M_\lambda Q_2)^{-1}Q_2^T. \tag{51}$$

Combining (50) and (51), we conclude that

$$w^T(Q_2^T M_\lambda Q_2)^{-1}w = (n\lambda)^{-1}y^T(I - H(\lambda))y.$$

C Additional algorithms

Given the Cholesky factorization $K + D = LL^T$ where $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$ with $U, W \in \mathbf{R}^{n\times p}$ and $c \in \mathbf{R}^n_{++}$, the matrix–vector products Lx, L^Tx, $L^{-1}x$, and $L^{-T}x$ can be evaluated in $O(pn)$ flops using algorithms 7 to 10.

Algorithm 7 Triangular product (Lx)

Input: $x \in \mathbf{R}^n$, $U, W \in \mathbf{R}^{n\times p}$, and $c \in \mathbf{R}^n_{++}$ such that $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$
Output: $y = Lx$
  Initialization: $z \leftarrow 0$
  for $k = 1, \ldots, n$ **do**
    $y_k \leftarrow c_k x_k + u_k^T z$
    $z \leftarrow z + w_k x_k$
  end for

Algorithm 8 Adjoint triangular product (L^Tx)

Input: $x \in \mathbf{R}^n$, $U, W \in \mathbf{R}^{n\times p}$, and $c \in \mathbf{R}^n_{++}$ such that $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$
Output: $y = L^Tx$
  Initialization: $z \leftarrow 0$
  for $k = n, \ldots, 1$ **do**
    $y_k \leftarrow c_k x_k + w_k^T z$
    $z \leftarrow z + u_k x_k$
  end for

Algorithm 9 Forward substitution (solve $Lx = b$)

Input: $b \in \mathbf{R}^n$, $U, W \in \mathbf{R}^{n\times p}$, and $c \in \mathbf{R}^n_{++}$ such that $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$
Output: $x = L^{-1}b$
  Initialization: $z \leftarrow 0$
  for $k = 1, \ldots, n$ **do**
    $x_k \leftarrow (b_k - u_k^T z)/c_k$
    $z \leftarrow z + w_k x_k$
  end for

Algorithm 10 Backward substitution (solve $L^T x = b$)

Input: $b \in \mathbf{R}^n$, $U, W \in \mathbf{R}^{n \times p}$, and $c \in \mathbf{R}^n_{++}$ such that $L = \mathbf{tril}(UW^T, -1) + \mathbf{diag}(c)$
Output: $x = L^{-T} b$

```
Initialization: z ← 0
for k = n, ..., 1 do
    x_k ← (b_k − w_k^T z)/c_k
    z ← z + u_k x_k
end for
```

References

[1] F. P. Carli, T. Chen, and L. Ljung. Maximum entropy kernels for system identification. *IEEE Transactions on Automatic Control*, 62(3):1471–1477, Mar. 2017.

[2] T. Chen, T. Ardeshiri, F. P. Carli, A. Chiuso, L. Ljung, and G. Pillonetto. Maximum entropy properties of discrete-time first-order stable spline kernel. *Automatica*, 66:34–38, Apr. 2016.

[3] A. M. Erisman and W. F. Tinney. On computing certain elements of the inverse of a sparse matrix. *Communications of the ACM*, 18(3):177–179, Mar. 1975.

[4] D. Foreman-Mackey, E. Agol, S. Ambikasaran, and R. Angus. Fast and scalable Gaussian process modeling with applications to astronomical time series. *The Astronomical Journal*, 154(6):220, nov 2017.

[5] Y. Fujimoto and T. Chen. On the coordinate change to the first-order spline kernel for regularized impulse response estimation, 2019.

[6] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 2013.

[7] M. F. Hutchinson and F. R. de Hoog. Smoothing noisy data with spline functions. *Numerische Mathematik*, 47(1):99–106, Mar. 1985.

[8] J. Keiner and B. J. Waterhouse. Fast principal components analysis method for finance problems with unequal time steps. In *Monte Carlo and Quasi-Monte Carlo Methods 2008*, pages 455–465. Springer Berlin Heidelberg, 2009.

[9] G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, Jan. 1971.

[10] R. Kohn and C. F. Ansley. A new algorithm for spline smoothing based on smoothing a stochastic process. *SIAM Journal on Scientific and Statistical Computing*, 8(1):33–48, Jan. 1987.

[11] J. H. Manton and P.-O. Amblard. A primer on reproducing kernel Hilbert spaces. *Foundations and Trends in Signal Processing*, 8(1-2):1–126, 2015.

[12] G. Pillonetto and G. De Nicolao. A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93, Jan. 2010.

[13] C. H. Reinsch. Smoothing by spline functions. *Numerische mathematik*, 10(3):177–183, 1967.

[14] C. H. Reinsch. Smoothing by spline functions. II. *Numerische Mathematik*, 16(5):451–454, Feb. 1971.

[15] I. J. Schoenberg. Spline functions and the problem of graduation. *Proceedings of the National Academy of Sciences*, 52(4):947–950, Oct. 1964.

[16] L. Schumaker. *Spline Functions: Basic Theory.* Cambridge University Press, Sept. 2007.

[17] R. Vandebril, M. van Van Barel, and N. Mastronardi. *Matrix Computations and Semiseparable Matrices: Eigenvalue and Singular Value Methods*, volume 2. Johns Hopkins University Press, 2008.

[18] R. Vandebril, M. van Van Barel, and N. Mastronardi. *Matrix Computations and Semiseparable Matrices: Linear Systems*, volume 1. Johns Hopkins University Press, 2008.

[19] G. Wahba. Improper priors, spline smoothing and the problem of guarding against model errors in regression. *Journal of the Royal Statistical Society. Series B*, 40(3):364–372, 1978.

[20] G. Wahba. *Spline Models for Observational Data.* SIAM, Jan. 1990.