Journal of Machine Learning Research 20 (2019) 1-27 Submitted 9/18; Revised 10/19; Published 11/19

# Fast Automatic Smoothing for Generalized Additive Models

**Yousra El-Bachir** YOUSRA.ELBACHIR@GMAIL.COM
**Anthony C. Davison** ANTHONY.DAVISON@EPFL.CH
*EPFL-FSB-MATH-STAT*
*Ecole Polytechnique Fédérale de Lausanne*
*Station 8, CH-1015 Lausanne, Switzerland*

**Editor:** Manfred Opper

## Abstract

Generalized additive models (GAMs) are regression models wherein parameters of probability distributions depend on input variables through a sum of smooth functions, whose degrees of smoothness are selected by $L_2$ regularization. Such models have become the de-facto standard nonlinear regression models when interpretability and flexibility are required, but reliable and fast methods for automatic smoothing in large data sets are still lacking. We develop a general methodology for automatically learning the optimal degree of $L_2$ regularization for GAMs using an empirical Bayes approach. The smooth functions are penalized by hyper-parameters that are learned simultaneously by maximization of a marginal likelihood using an approximate expectation-maximization algorithm. The latter involves a double Laplace approximation at the E-step, and leads to an efficient M-step. Empirical analysis shows that the resulting algorithm is numerically stable, faster than the best existing methods and achieves state-of-the-art accuracy. For illustration, we apply it to an important and challenging problem in the analysis of extremal data.

**Keywords:** Automatic $L_2$ Regularization, Empirical Bayes, Expectation-maximization Algorithm, Generalized Additive Model, Laplace Approximation, Marginal Maximum Likelihood

## 1. Introduction

State-of-the-art machine learning methods achieve impressive accuracy, but their opacity can make them unattractive when their results contradict intuition or lack a ready explanation. Methods that are interpretable by humans and easy to diagnose and debug are needed for difficult decision-making problems in healthcare, legal settings, finance, risk management, and many other areas. One possibility is the use of generalized additive models, a class of interpretable data-driven models used in smooth regression (Duvenaud et al., 2011; Ba et al., 2012; Tsang et al., 2018; Mutny and Krause, 2018).

Generalized additive models (GAMs) are a class of supervised learning tools that describe the relationship between output variables and inputs using a sum of smooth functions (Hastie et al., 2009, Chapter 9). They were introduced by Hastie and Tibshirani (1986), who represented the smooth functions by scatterplot smoothers and trained them sequentially by backfitting (Breiman and Friedman, 1985). The corresponding `R` (R Core Team, 2019) package `gam` implements the methods in Hastie and Tibshirani (1990), which select the level of smoothness by stepwise regression using approximate distributional results. Backfitting allows smooth terms to be represented by local regression smoothers (Cleveland et al., 1993), but inference based on the resulting fit is awkward. Yee and Wild (1996) later proposed modified vector backfitting, whereby several smooth functions are learned simultaneously. Their method, embodied in the `R` package `VGAM`, first learns the linear components and then learns the nonlinear part by training a vector additive model on the resulting partial residuals. In the `R` package `gamlss`, Rigby and Stasinopoulos (2005) learn the smooth functions sequentially by combining backfitting with two algorithms, which optimize the penalized likelihood of the regression weights. The first algorithm generalizes that of Cole and Green (1992), whereas the second generalizes that of Rigby and Stasinopoulos (1996) and is preferable when the parameters of the distribution are orthogonal with respect to the information matrix. All these approaches invoke backfitting, which dissociates learning of the regression model from that of the smoothing hyper-parameters. This may be statistically inefficient, and accuracy may be increased by learning the appropriate degree of smoothing as part of the regression training. An alternative representation of the smooth functions that enables automatic smoothing is via basis function expansion using reduced rank smoothing; this is the foundation upon which we build our methodology.

### 1.1. Model Set-up

We suppose that independent observations come from a probability distribution whose parameters depend on generalized additive models. Let $Y_i$ denote a random variable with realized value $y_i$ and probability distribution function $F_i(y_i;\boldsymbol{\theta}_i)$ that depends on a parameter vector $\boldsymbol{\theta}_i = (\theta_i^{(1)},\dots,\theta_i^{(P)}) \in \mathbb{R}^P$; so for the training set $\boldsymbol{y} = (y_1,\dots,y_n)^T$, the full parameter vector is $\boldsymbol{\theta} = (\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_n)^T \in \mathbb{R}^{nP}$ with sub-vectors $\boldsymbol{\theta}^{(p)} = (\theta_1^{(p)},\dots,\theta_n^{(p)})^T \in \mathbb{R}^n$ for $p = 1,\dots,P$. In the Gaussian model for example, $P=2$, $\boldsymbol{\theta}^{(1)} = \boldsymbol{\mu}$ is the mean and $\boldsymbol{\theta}^{(2)} = \boldsymbol{\sigma}$ is the standard deviation, and we have $\boldsymbol{\theta} = (\mu_1,\sigma_1,\dots,\mu_n,\sigma_n)^T$.

For a probability distribution with $P$ parameters, each $\boldsymbol{\theta}^{(p)}$ has an additive structure, which we now describe. Let $\boldsymbol{X}_i^{(p)^*}$ denote the $i$-th row of a feature matrix corresponding to a regression weight vector $\boldsymbol{w}^{(p)^*}$ that includes an offset. The superscript $^*$ refers to an intermediate notation for the regression elements that will become clear when we define the full feature matrices corresponding to the untransformed input variables and the smooth functions. Let $q_p \geqslant 0$ denote the number of unknown smooth functions $f_j^{(p)}$ contributing to $\boldsymbol{\theta}^{(p)}$, and let $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots$ denote the vectors of input variables. In spatio-temporal settings for example, $\boldsymbol{x}_1$ may denote the spatial location of a site and $\boldsymbol{x}_2$ may denote the time. The components $\theta_i^{(p)}$ of $\boldsymbol{\theta}^{(p)}$ are represented using the additive structure

$$\theta_i^{(p)} = \boldsymbol{X}_i^{(p)^*}\boldsymbol{w}^{(p)^*} + \sum_{j=1}^{q_p} f_j^{(p)}(x_{i1}, x_{i2}, \dots), \quad i = 1,\dots,n. \tag{1}$$

Each of the $f_j^{(p)}$ can be a function of one or more inputs, and is parametrized by an expansion of basis functions $b_k^{(p)}(x)$,

$$f_j^{(p)}(x) = \sum_{k=1}^{K} \tilde{w}_k^{(p)} b_k^{(p)}(x) = \tilde{\boldsymbol{w}}^{(p)^T}\boldsymbol{b}^{(p)}(x), \tag{2}$$

where $\tilde{\boldsymbol{w}}^{(p)} = (\tilde{w}_1^{(p)}, \ldots, \tilde{w}_K^{(p)})^T \in \mathbb{R}^K$ is the column vector of unknown regression weights that expand $f_j^{(p)}$, and $\boldsymbol{b}^{(p)}(x) = \{b_1^{(p)}(x), \ldots, b_K^{(p)}(x)\}^T \in \mathbb{R}^K$ is a $K$-dimensional vector of basis functions selected from a dictionary of readily-available functions, such as cubic regression splines. The basis dimension $K$ is chosen manually, and typically is allowed to grow slowly with the size $n$ of the training set. We assume that the smooth functions are subject to a sum-to-zero identifiability constraint

$$\sum_{i=1}^{n} f_j^{(p)}(x_{ij}) = 0, \tag{3}$$

which leads to good coverage for confidence intervals (Nychka, 1988). In this setting, the components of $\boldsymbol{\theta}^{(p)}$ in (1) become $\theta_i^{(p)} = \boldsymbol{X}_i^{(p)} \boldsymbol{w}^{(p)}$, where $\boldsymbol{w}^{(p)} \in \mathbb{R}^{d_p}$ denotes the regression weights including their parametric part indicated with superscript $^*$ and smooth part indicated by a tilde in (2), and $\boldsymbol{X}^{(p)} \in \mathbb{R}^{n \times d_p}$ denotes the corresponding feature matrix. We assume that some of the columns of $\boldsymbol{X}^{(p)}$ have been centered to absorb the identifiability constraint (3) on the smooth functions before training.

A GAM is characterized by the additivity of its functions of inputs and by their smoothness. The latter is controlled by penalizing the curvature of the functions, using roughness penalties on the $f_j^{(p)}$ of the form

$$\mathrm{PEN}(\lambda_j^{(p)}) = \lambda_j^{(p)} \int \left\{ f_j^{(p)\prime\prime}(t) \right\}^2 \mathrm{d}t \in \mathbb{R}, \tag{4}$$

where the regularization hyper-parameter $\lambda_j^{(p)} > 0$ controls the degree of smoothness, and the integral is taken over the input range values. If the basis functions are twice differentiable, the element-wise integration

$$\boldsymbol{S}_j^{(p)} = \int \boldsymbol{b}^{(p)\prime\prime}(x) \boldsymbol{b}^{(p)\prime\prime T}(x) \, \mathrm{d}x$$

defines a known symmetric and semi-positive definite smoothing matrix $\boldsymbol{S}_j^{(p)} \in \mathbb{R}^{K \times K}$. The penalty in (4) thus becomes quadratic,

$$\mathrm{PEN}(\lambda_j^{(p)}) = \lambda_j^{(p)} \boldsymbol{w}^{(p)T} \boldsymbol{S}_j^{(p)} \boldsymbol{w}^{(p)}.$$

On defining analogous quantities for any of the parameter vectors $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(P)}$, and stacking the regression weights and the smoothing hyper-parameters to form $\boldsymbol{w} \in \mathbb{R}^d$ and $\boldsymbol{\lambda} \in \mathbb{R}^q$ with $d = \sum_{p=1}^{P} d_p$ and $q = \sum_{p=1}^{P} q_p$, the full functional parameter vector and curvature penalties are parametrized by

$$\boldsymbol{\theta}_{\boldsymbol{w}} = \boldsymbol{X}\boldsymbol{w} \in \mathbb{R}^{nP}, \quad \mathrm{PEN}(\boldsymbol{\lambda}) = \sum_{p=1}^{P} \sum_{j=1}^{q_p} \mathrm{PEN}(\lambda_j^{(p)}) = \boldsymbol{w}^T \boldsymbol{S}_{\boldsymbol{\lambda}} \boldsymbol{w} \in \mathbb{R}, \tag{5}$$

where the $i$-th row block of the full feature matrix $\boldsymbol{X} \in \mathbb{R}^{nP \times d}$ is

$$\boldsymbol{X}_i = \mathrm{diag}\left( \boldsymbol{X}_i^{(1)}, \ldots, \boldsymbol{X}_i^{(P)} \right) \in \mathbb{R}^{P \times d},$$

and the full smoothing matrix

$$\boldsymbol{S}_{\boldsymbol{\lambda}} = \operatorname{diag}\left(\lambda_1^{(1)}\boldsymbol{S}_1^{(1)}, \dots, \lambda_{q_P}^{(P)}\boldsymbol{S}_{q_P}^{(P)}\right) \in \mathbb{R}^{d\times d} \tag{6}$$

is block diagonal.

Learning the regression weights involves balancing the conflicting goals of providing a good fit to the data and avoiding overfitting. For a given $\boldsymbol{\lambda}$, this is achieved by maximizing the penalized log-likelihood for $\boldsymbol{w}$,

$$\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}) = \ell_{\mathrm{L}}\left(\boldsymbol{\theta}_{\boldsymbol{w}};\boldsymbol{y}\right) - \frac{1}{2}\,\boldsymbol{w}^T\boldsymbol{S}_{\boldsymbol{\lambda}}\boldsymbol{w}, \tag{7}$$

where the log-likelihood $\ell_{\mathrm{L}}$ may be written equivalently in terms of $\boldsymbol{\theta}$ or of $\boldsymbol{w}$. Let $\boldsymbol{U}_{\mathrm{L}}(\boldsymbol{w};\boldsymbol{y}) \in \mathbb{R}^d$ and $\boldsymbol{H}_{\mathrm{L}}(\boldsymbol{w};\boldsymbol{y}) \in \mathbb{R}^{d\times d}$, or $\boldsymbol{U}_{\mathrm{L}}(\boldsymbol{\theta};\boldsymbol{y}) \in \mathbb{R}^{Pn}$ and $\boldsymbol{H}_{\mathrm{L}}(\boldsymbol{\theta};\boldsymbol{y}) \in \mathbb{R}^{Pn\times Pn}$, denote the gradient and negative Hessian of $\ell_{\mathrm{L}}$ with respect to $\boldsymbol{w}$ or to $\boldsymbol{\theta}$. The corresponding penalized quantities are

$$\begin{aligned} \boldsymbol{U}_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}) &= \boldsymbol{U}_{\mathrm{L}}(\boldsymbol{w};\boldsymbol{y}) - \boldsymbol{S}_{\boldsymbol{\lambda}}\boldsymbol{w}, & \boldsymbol{H}_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}) &= \boldsymbol{H}_{\mathrm{L}}(\boldsymbol{w};\boldsymbol{y}) + \boldsymbol{S}_{\boldsymbol{\lambda}}, \\ &= \boldsymbol{X}^T\boldsymbol{U}_{\mathrm{L}}(\boldsymbol{\theta};\boldsymbol{y}) - \boldsymbol{S}_{\boldsymbol{\lambda}}\boldsymbol{w}, & &= \boldsymbol{X}^T\boldsymbol{H}_{\mathrm{L}}(\boldsymbol{\theta};\boldsymbol{y})\boldsymbol{X} + \boldsymbol{S}_{\boldsymbol{\lambda}}. \end{aligned} \tag{8}$$

The negative Hessian is used to compute standard errors and confidence intervals. For simplicity we sometimes suppress the dependence upon $\boldsymbol{y}$ and $\boldsymbol{\lambda}$ from the notation. Maximization of the penalized log-likelihood (7) provides an estimate for $\boldsymbol{w}$ for a given value of the smoothing hyper-parameters $\boldsymbol{\lambda}$. In Section 1.2 we briefly summarize the main classical optimization methods for embodying learning of $\boldsymbol{\lambda}$ in that of the regression weights.

## 1.2. Classical Optimization

The two learning strategies for generalized additive models are performance iteration (Gu, 1992) and outer iteration (O'Sullivan et al., 1986), and optimize a criterion for the smoothing hyper-parameters whilst updating the regression weights. The updating step in performance iteration consists of one iteration for the regression weights, often performed by iterative least squares (Nelder and Wedderburn, 1972), followed by one full optimization for the smoothing hyper-parameters. Since the smoothness selection is applied to an intermediate estimate of the regression model, its score changes from iteration to iteration and convergence of the overall optimization is not guaranteed, as shown by Wood (2008, 2011). The updating step in outer iteration comprises one update for the smoothing hyper-parameters followed by one full optimization for the regression weights. Since the former are obtained from a converged and so a fixed regression model, the convergence of outer iteration is guaranteed, but each updating step is computationally more expensive, and the dependence between the regression weights and the smoothing hyper-parameters makes the calculations much more challenging.

Once the strategy for automatic smoothing is chosen, the classical approach for learning its regularization hyper-parameters is to minimize measures of prediction error such as the Akaike or Bayesian information criteria (AIC or BIC), or the generalized cross validation (GCV) criterion. The first criteria tend to overfit, and GCV can generate multiple minima and unstable estimates that may lead to substantial underfitting (Reiss and Ogden, 2009; Wood, 2011). Use of marginal likelihood overcomes these limitations but involves intractable integrals. Despite the widespread use of GAMs, automatic learning of their smoothing hyper-parameters remains an open problem. The reliable method (Wood, 2011) and its generalization (Wood et al., 2016), implemented in the `R` package `mgcv`, combine the advantages of the marginal likelihood approach with the good convergence of outer iteration. However, these approaches are challenging to set up, difficult to extend to new families of distributions, and computationally expensive for large samples. Methods specifically designed for large (Wood et al., 2015) and big (Wood et al., 2017) data sets are based on performance iteration, and so offer no guarantee of convergence to a local optimum. In this paper we overcome these limitations by presenting an approach that is simpler, faster and achieves state-of-the-art accuracy.

The rest of the paper is organized as follows. Section 2 introduces our proposed automatic smoothness selection procedure, which is based on an approximate expectation-maximization algorithm. Section 3 assesses its performance with a simulation study. Section 4 provides a real data analysis on extreme temperatures, and Section 5 closes the paper with a discussion.

## 2. Automatic Smoothing

The Bayesian perspective provides an interpretation for the roughness penalty that underlies the weighted $L_2$ regularization in the penalized log-likelihood (7), as we now describe. Let $\boldsymbol{S}_{\boldsymbol{\lambda}}^{-}$ denote the generalized inverse of $\boldsymbol{S}_{\boldsymbol{\lambda}}$, and suppose that the regression weights have an improper $\mathcal{N}(0, \boldsymbol{S}_{\boldsymbol{\lambda}}^{-})$ multivariate Gaussian prior density (Kimeldorf and Wahba, 1970; Silverman, 1985),

$$\pi(\boldsymbol{w};\boldsymbol{\lambda}) = (2\pi)^{-(d-m)/2}\ |\boldsymbol{S}_{\boldsymbol{\lambda}}|_{+}^{1/2} \exp\left(-\frac{1}{2}\boldsymbol{w}^T\boldsymbol{S}_{\boldsymbol{\lambda}}\boldsymbol{w}\right), \tag{9}$$

where $m$ is the number of zero eigenvalues of $\boldsymbol{S}_{\boldsymbol{\lambda}}$, and $|\boldsymbol{S}_{\boldsymbol{\lambda}}|_{+}$ is the product of its positive eigenvalues. With $f$ here denoting the density of the data, the log-posterior density for $\boldsymbol{w}$ is

$$\begin{aligned}\ell(\boldsymbol{w} \mid \boldsymbol{y};\boldsymbol{\lambda}) &= \log\left\{f(\boldsymbol{y}\mid\boldsymbol{w};\boldsymbol{\lambda})\ \pi(\boldsymbol{w};\boldsymbol{\lambda})\right\} - \log f(\boldsymbol{y};\boldsymbol{\lambda}) \\ &= \ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}) + \frac{1}{2}\log|\boldsymbol{S}_{\boldsymbol{\lambda}}|_{+} - \frac{d-m}{2}\log(2\pi) - \log f(\boldsymbol{y};\boldsymbol{\lambda}). \end{aligned}\tag{10}$$

The roughness penalty in (5) now appears as the key component of the logarithm of the prior (9), and the penalized log-likelihood (7) appears as the log-posterior (10) (up to a constant depending upon $\boldsymbol{\lambda}$ but not upon $\boldsymbol{w}$). The smoothing hyper-parameters can hence be learned from the last term on the right of (10), the marginal density of $\boldsymbol{y}$,

$$L_{\mathrm{M}}(\boldsymbol{\lambda};\boldsymbol{y}) = f(\boldsymbol{y};\boldsymbol{\lambda}) = \int f(\boldsymbol{y},\boldsymbol{w};\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{w} = \int f(\boldsymbol{y}\mid\boldsymbol{w};\boldsymbol{\lambda})\ \pi(\boldsymbol{w};\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{w}.$$

This integral is equivalent to $L_{\mathrm{M}}(\boldsymbol{\lambda};\boldsymbol{y}) = E_{\pi(\boldsymbol{w};\boldsymbol{\lambda})}\{f(\boldsymbol{y}\mid\boldsymbol{w};\boldsymbol{\lambda})\}$, which is the expectation of the likelihood function under the prior density of the regression weight vector.

A fully Bayesian approach would involve choosing a prior density for $\boldsymbol{\lambda}$ and integrating out over it, but instead we take an empirical Bayes approach and transform the smoothness selection problem to an optimization problem, where the optimal $\boldsymbol{\lambda}$ maximize the log-marginal likelihood

$$\ell_{\mathrm{M}}(\boldsymbol{\lambda};\boldsymbol{y}) \quad = \quad \log L_{\mathrm{M}}(\boldsymbol{\lambda};\boldsymbol{y}) \equiv \frac{1}{2}\log|\boldsymbol{S}_{\boldsymbol{\lambda}}|_{+} + \log\int \exp\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{w}, \tag{11}$$

where $\equiv$ indicates equality up to a constant not depending upon $\boldsymbol{\lambda}$. The multidimensional integral over $\boldsymbol{w}$ is generally intractable, and its accurate and efficient computation is challenging. The two classical strategies for this are stochastic integration methods based on importance sampling, and deterministic methods based on quadrature or the Laplace method. Importance sampling is a Monte Carlo integration technique under which the integral is treated as an expectation, but its performance relies on the choice of the distribution from which to sample, and its accuracy increases only with the number of samples. Quadrature involves a discretization of the integrand over the domain of integration, and amounts to calculating a weighted sum of the values of the integrand. Both methods perform well when the dimension $d$ of the regression weight vector $\boldsymbol{w}$ is small, but become computationally infeasible when $d$ becomes large (Vonesh et al., 2002). The most commonly used deterministic integration method is Laplace approximation, which yields an analytical expression for (11) by exploiting a quadratic Taylor expansion of the log-integrand around the maximum penalized likelihood estimate,

$$\ell_{\mathrm{M}}(\boldsymbol{\lambda};\boldsymbol{y}) \quad = \quad \frac{1}{2}\log|\boldsymbol{S}_{\boldsymbol{\lambda}}|_{+} + \ell_{\mathrm{P}}(\hat{\boldsymbol{w}}_{\boldsymbol{\lambda}};\boldsymbol{y},\boldsymbol{\lambda}) - \frac{1}{2}\log\det\boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_{\boldsymbol{\lambda}};\boldsymbol{y},\boldsymbol{\lambda}) + O(n^{-1}), \tag{12}$$

where $\hat{\boldsymbol{w}}_{\boldsymbol{\lambda}}$ denotes the maximizer of $\ell_{\mathrm{P}}$ for a given $\boldsymbol{\lambda}$. However, optimization of the resulting approximate log-marginal likelihood (12) is awkward. Each updating step includes intermediate maximizations, involves unstable terms that need careful and computationally expensive decompositions, and requires the fourth derivatives of the log-likelihood to obtain the Hessian matrix of the log-marginal likelihood in the Newton–Raphson method (Wood, 2011; Wood et al., 2016). These difficulties make Laplace approximation computationally demanding for smoothness selection, and limit its extension to complex models. In Section 2.1 we present a simpler alternative that is easier to implement, faster and very accurate.

Although we illustrate our ideas on generalized additive models, our automatic smoothing method extends to the general setting of $L_2$ regularization if we replace the smoothing matrices with identity matrices. Hyper-parameter optimization based on marginal likelihood in other contexts is a long-standing problem addressed by MacKay (1992, 1999), Neal (1996), Tipping (1999, 2001), Faul and Tipping (2001), Qi et al. (2004), McHutchon and Rasmussen (2011) and many others.

### 2.1. Approximate Expectation-maximization

We directly maximize the log-marginal likelihood (11) with respect to the smoothing hyper-parameters using the expectation-maximization (EM) algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2008), which circumvents evaluation of the objective function. The EM algorithm is an iterative method for computing maximum likelihood estimators for difficult functions, by alternating between an expectation step, the E-step, and its maximization, the M-step, at every iteration until convergence. Ignoring the constant term, taking conditional expectations of expression (10) with respect to the posterior $\pi(\boldsymbol{w} \mid \boldsymbol{Y} = \boldsymbol{y};\boldsymbol{\lambda}_k)$

at the best estimate $\boldsymbol{\lambda}_k$ at iteration $k$ yields

$$\ell_{\mathrm{M}}(\boldsymbol{\lambda};\boldsymbol{y}) \equiv Q(\boldsymbol{\lambda};\boldsymbol{\lambda}_k) - K(\boldsymbol{\lambda};\boldsymbol{\lambda}_k),$$

where $\equiv$ indicates equality up a constant, and

$$\begin{aligned} Q(\boldsymbol{\lambda};\boldsymbol{\lambda}_k) &= E_{\pi(\boldsymbol{w}\mid\boldsymbol{Y};\boldsymbol{\lambda}_k)}\left\{\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{Y},\boldsymbol{\lambda}) + \frac{1}{2}\log|\boldsymbol{S}_{\boldsymbol{\lambda}}|_+\right\}, \qquad (13)\\ K(\boldsymbol{\lambda};\boldsymbol{\lambda}_k) &= E_{\pi(\boldsymbol{w}\mid\boldsymbol{Y};\boldsymbol{\lambda}_k)}\left\{\ell(\boldsymbol{w}\mid\boldsymbol{Y};\boldsymbol{\lambda})\right\} = E_{\pi(\boldsymbol{w}\mid\boldsymbol{Y};\boldsymbol{\lambda}_k)}\left\{\log\pi(\boldsymbol{w}\mid\boldsymbol{Y};\boldsymbol{\lambda})\right\}. \end{aligned}$$

The E-step corresponds to the analytic calculation of the function $Q$, which is maximized with respect to $\boldsymbol{\lambda}$ at the M-step to provide $\boldsymbol{\lambda}_{k+1}$, as input for the next EM iteration. Jensen's inequality implies that $K(\boldsymbol{\lambda};\boldsymbol{\lambda}_k) \leqslant K(\boldsymbol{\lambda}_k;\boldsymbol{\lambda}_k)$ for all $\boldsymbol{\lambda}$, and since $Q(\boldsymbol{\lambda}_{k+1};\boldsymbol{\lambda}_k) \geqslant Q(\boldsymbol{\lambda}_k;\boldsymbol{\lambda}_k)$, we have $\ell_{\mathrm{M}}(\boldsymbol{\lambda}_{k+1};\boldsymbol{y}) \geqslant \ell_{\mathrm{M}}(\boldsymbol{\lambda}_k;\boldsymbol{y})$. The EM algorithm thus transfers optimization of the log-marginal likelihood $\ell_{\mathrm{M}}$ to that of $Q$, and ensures that $\ell_{\mathrm{M}}$ increases after every M-step. Under mild conditions, the algorithm is guaranteed to reach at least a local maximum (Dempster et al., 1977). We first construct the function $Q$ used at the E-step.

## 2.2. E-step

Applying Bayes' rule to the posterior for $\boldsymbol{w}$, the non-trivial element of the function $Q$ in (13) is

$$\begin{aligned} E_{\pi(\boldsymbol{w}\mid\boldsymbol{Y};\boldsymbol{\lambda}_k)}\left\{\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{Y},\boldsymbol{\lambda})\right\} &= \int \ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda})\ \pi(\boldsymbol{w}\mid\boldsymbol{Y}=\boldsymbol{y};\boldsymbol{\lambda}_k)\,\mathrm{d}\boldsymbol{w}\\ &= \frac{\displaystyle\int \ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda})\exp\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}_k)\,\mathrm{d}\boldsymbol{w}}{\displaystyle\int \exp\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}_k)\,\mathrm{d}\boldsymbol{w}}. \qquad (14) \end{aligned}$$

Both integrals are intractable. Since $\ell_{\mathrm{P}}$ may not be positive, the numerator cannot be expressed as the integral of an exponential function, so direct Laplace approximation is impracticable. Tierney et al. (1989) overcome this by approximating similar ratios using the moment generating function, since (14) is the expectation of a scalar function, $\ell_{\mathrm{P}}$, of the regression weights, which are here treated as random variables with joint probability density proportional to $\exp\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}_k)$, the posterior density conditional on $\boldsymbol{y}$ as seen from (10). For any $\boldsymbol{w}$, let

$$\ell_t(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k) = t\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}) + \ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}_k), \quad t\in\mathbb{R}.$$

The conditional moment generating function of $\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{Y},\boldsymbol{\lambda})$ is thus

$$\begin{aligned} \mathrm{M}(t) = E_{\pi(\boldsymbol{w}\mid\boldsymbol{Y};\boldsymbol{\lambda}_k)}\left[\exp\left\{t\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{Y},\boldsymbol{\lambda})\right\}\right] &= \frac{\int\exp\left\{t\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda})\right\}f(\boldsymbol{y},\boldsymbol{w};\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{w}}{\int f(\boldsymbol{y},\boldsymbol{w};\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{w}}\\ &= \frac{\displaystyle\int\exp\ell_t(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)\,\mathrm{d}\boldsymbol{w}}{\displaystyle\int\exp\ell_0(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)\,\mathrm{d}\boldsymbol{w}}. \qquad (15) \end{aligned}$$

Expression (15) is a ratio of two intractable integrals, each of which can be approximated using the Laplace method. Let

$$\hat{\boldsymbol{w}}_t = \arg\max_{\boldsymbol{w}} \ell_t(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k), \quad \hat{\boldsymbol{w}}_k = \arg\max_{\boldsymbol{w}} \ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}_k) = \arg\max_{\boldsymbol{w}} \ell_0(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)$$

denote the maximizers of $\ell_t(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)$ and $\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}_k)$, and write the negative Hessian matrix as $\boldsymbol{H}_t(\boldsymbol{w};\boldsymbol{\lambda},\boldsymbol{\lambda}_k) = t\boldsymbol{H}_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{\lambda}) + \boldsymbol{H}_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{\lambda}_k)$, where $\boldsymbol{H}_{\mathrm{P}}$ is given in (8). Second-order Taylor expansion of $\ell_t(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)$ around $\hat{\boldsymbol{w}}_t$ yields the following approximation for the numerator of (15),

$$\begin{aligned}\int \exp \ell_t(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)\,\mathrm{d}\boldsymbol{w} &= \exp \ell_t(\hat{\boldsymbol{w}}_t;\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k) \int \exp\left\{-\frac{1}{2}(\boldsymbol{w}-\hat{\boldsymbol{w}}_t)^T \boldsymbol{H}_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)(\boldsymbol{w}-\hat{\boldsymbol{w}}_t)\right\}\mathrm{d}\boldsymbol{w}\,\{1+O(n^{-1})\} \\ &= (2\pi)^{d/2} \det \boldsymbol{H}_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)^{-1/2} \exp \ell_t(\hat{\boldsymbol{w}}_t;\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)\{1+O(n^{-1})\},\end{aligned}$$

where the determinant is well-defined because $\boldsymbol{H}_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)$ is positive definite at convergence. On similarly applying Laplace approximation to the denominator of (15), the conditional moment generating function becomes

$$\mathrm{M}(t) = \exp\left\{\ell_t(\hat{\boldsymbol{w}}_t;\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k) - \ell_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{y},\boldsymbol{\lambda}_k)\right\} \frac{\det \boldsymbol{H}_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)^{-1/2}}{\det \boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k)^{-1/2}}\{1+O(n^{-2})\}, \tag{16}$$

where the relative approximation error is $O(n^{-2})$ rather than $O(n^{-1})$ because the error terms in the numerator and denominator almost cancel (Tierney et al., 1989, Theorem 1). The conditional expectation (14) is obtained by differentiating (16) with respect to $t$ and evaluating the result at $t = 0$.

Whereas Tierney et al. (1989) and Steele (1996) suggest numerical computation of such derivatives, we shall calculate them analytically. We need $\mathrm{d}\ell_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)/\mathrm{d}t$ and $\mathrm{d}\det \boldsymbol{H}_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)/\mathrm{d}t$, both evaluated at $t = 0$. To simplify the notation we use $\mathrm{d}_0\cdot/\mathrm{d}t$ to denote $\mathrm{d}\cdot/\mathrm{d}t\,|_{t=0}$, and similarly for $\partial_0\cdot/\partial t$.

*Calculation of* $\mathrm{d}_0\ell_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)/\mathrm{d}t$.

Since $\hat{\boldsymbol{w}}_t$ depends upon $t$,

$$\frac{\mathrm{d}\ell_t(\hat{\boldsymbol{w}}_t;\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)}{\mathrm{d}t} = \ell_{\mathrm{P}}(\hat{\boldsymbol{w}}_t;\boldsymbol{y},\boldsymbol{\lambda}) + t\frac{\partial\hat{\boldsymbol{w}}_t}{\partial t}\cdot\frac{\partial\ell_{\mathrm{P}}(\hat{\boldsymbol{w}}_t;\boldsymbol{y},\boldsymbol{\lambda})}{\partial\boldsymbol{w}} + \frac{\partial\hat{\boldsymbol{w}}_t}{\partial t}\cdot\frac{\partial\ell_{\mathrm{P}}(\hat{\boldsymbol{w}}_t;\boldsymbol{y},\boldsymbol{\lambda}_k)}{\partial\boldsymbol{w}},$$

where $\cdot$ denotes the scalar product. Since $\hat{\boldsymbol{w}}_t = \hat{\boldsymbol{w}}_k$ at $t = 0$, and $\hat{\boldsymbol{w}}_k$ maximizes $\ell_{\mathrm{P}}(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda}_k)$, we obtain

$$\frac{\mathrm{d}_0\ell_t(\hat{\boldsymbol{w}}_t;\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)}{\mathrm{d}t} = \ell_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{y},\boldsymbol{\lambda}). \tag{17}$$

*Calculation of* $\mathrm{d}_0\det \boldsymbol{H}_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)/\mathrm{d}t$.

This requires $\partial_0\hat{\boldsymbol{w}}_t/\partial t$, which we obtain by implicit differentiation of $\ell_t(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)$. At $\boldsymbol{w} = \hat{\boldsymbol{w}}_t$, we have $\partial\ell_t(\boldsymbol{w};\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\lambda}_k)/\partial\boldsymbol{w}\,|_{\boldsymbol{w}=\hat{\boldsymbol{w}}_t} = \mathbf{0}_d \in \mathbb{R}^d$, so differentiating with respect to $t$ and setting $t = 0$ yields

$$\boldsymbol{U}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}) - \boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k)\cdot\frac{\partial_0\hat{\boldsymbol{w}}_t}{\partial t} = \mathbf{0}_d. \tag{18}$$

As $\boldsymbol{U}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}) = \boldsymbol{U}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k) + \boldsymbol{S}_{\boldsymbol{\lambda}_k}\hat{\boldsymbol{w}}_k - \boldsymbol{S}_{\boldsymbol{\lambda}}\hat{\boldsymbol{w}}_k = \left(\boldsymbol{S}_{\boldsymbol{\lambda}_k} - \boldsymbol{S}_{\boldsymbol{\lambda}}\right)\hat{\boldsymbol{w}}_k$, we get from (18) that

$$\frac{\partial_0 \hat{\boldsymbol{w}}_t}{\partial t} \;=\; \boldsymbol{H}_{\mathrm{P}}^{-1}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k)\left(\boldsymbol{S}_{\boldsymbol{\lambda}_k} - \boldsymbol{S}_{\boldsymbol{\lambda}}\right)\hat{\boldsymbol{w}}_k. \tag{19}$$

Let $\hat{\boldsymbol{\theta}}_t = \boldsymbol{X}\hat{\boldsymbol{w}}_t \in \mathbb{R}^{nP}$ and $\hat{\boldsymbol{\theta}}_k = \boldsymbol{X}\hat{\boldsymbol{w}}_k$. Using the chain rule and (19), the derivative of the negative Hessian $\boldsymbol{H}_{\mathrm{L},i}$ of the unpenalized log-likelihood corresponding to a data-point $y_i$ is

$$\frac{\mathrm{d}_0 \boldsymbol{H}_{\mathrm{L},i}(\hat{\boldsymbol{\theta}}_{t,i};y_i)}{\mathrm{d}t} \;=\; \sum_{p=1}^{P} \boldsymbol{X}_i^{(p)} \cdot \frac{\partial_0 \hat{\boldsymbol{w}}_t^{(p)}}{\partial t}\frac{\partial \boldsymbol{H}_{\mathrm{L},i}(\hat{\boldsymbol{\theta}}_{k,i};y_i)}{\partial \theta_i^{(p)}}, \quad i = 1,\ldots,n. \tag{20}$$

Applying Jacobi's formula to $\mathrm{d}\det \boldsymbol{H}_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)/\mathrm{d}t$ and evaluating the result at $t = 0$ yields

$$\frac{\mathrm{d}_0 \det \boldsymbol{H}_t(\hat{\boldsymbol{w}}_t;\boldsymbol{\lambda},\boldsymbol{\lambda}_k)}{\mathrm{d}t} = \det \boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k) \times \mathrm{Tr}\left[\boldsymbol{H}_{\mathrm{P}}^{-1}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k)\left\{\boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}) + \frac{\mathrm{d}_0 \boldsymbol{H}_{\mathrm{L}}}{\mathrm{d}t}(\hat{\boldsymbol{w}}_t)\right\}\right], \tag{21}$$

where the last derivative term can be computed using (20).

On inserting (21) and (17) into the derivative of (16) with respect to $t$ and evaluating the result at $t = 0$, we find after a little algebra that

$$Q(\boldsymbol{\lambda};\boldsymbol{\lambda}_k) = \frac{1}{2}\log|\boldsymbol{S}_{\boldsymbol{\lambda}}|_+ + \left\{-\frac{1}{2}\,\mathrm{Tr}\left[\boldsymbol{H}_{\mathrm{P}}^{-1}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k)\left\{\boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}) + \frac{\mathrm{d}_0 \boldsymbol{H}_{\mathrm{L}}(\hat{\boldsymbol{w}}_t)}{\mathrm{d}t}\right\}\right] + \ell_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{y},\boldsymbol{\lambda})\right\}\left\{1 + O(n^{-2})\right\}.$$

The order $O(n^{-2})$ of the relative error in $Q$ over the usual $O(n^{-1})$ error for Laplace approximation may suggest that the approximate E-step provides a potentially better approximation to the function to be maximized for the smoothing hyper-parameters. We discuss this further in Appendix A.

The proposed approach is clearly an outer iteration optimization, since $Q$ is defined in terms of the maximizer $\hat{\boldsymbol{w}}_k$ rather than its intermediate estimate, as in the performance iteration optimization; see Section 1.2. As we shall see, this approximate E-step greatly simplifies the M-step; the crux is that $\hat{\boldsymbol{w}}_k$ depends on $\boldsymbol{\lambda}_k$ alone, and not on $\boldsymbol{\lambda}$.

### 2.3. M-step

The M-step entails the calculation of the gradient and Hessian matrix of $Q$ with respect to the smoothing hyper-parameters $\boldsymbol{\lambda}$. We first show that the derivative of $\partial_0\hat{\boldsymbol{w}}_t/\partial t$ in (19) with respect to $\boldsymbol{\lambda}$ equals $\partial\hat{\boldsymbol{w}}_k/\partial\boldsymbol{\lambda}_k$. As $\hat{\boldsymbol{w}}_k$ is the solution to the equation $\boldsymbol{U}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k) = \boldsymbol{0}_d$, taking the derivative with respect to the $j$-th component $\lambda_{k,j}$ of $\boldsymbol{\lambda}_k$ yields

$$\frac{\partial \hat{\boldsymbol{w}}_k}{\partial \lambda_{k,j}} = -\boldsymbol{H}_{\mathrm{P}}^{-1}(\hat{\boldsymbol{w}}_k;\boldsymbol{\lambda}_k)\boldsymbol{S}_j\hat{\boldsymbol{w}}_k = \frac{\partial\partial_0 \hat{\boldsymbol{w}}_t}{\partial \lambda_j \partial t}, \tag{22}$$

since $\partial \boldsymbol{S}_{\boldsymbol{\lambda}_{k,j}}/\partial\lambda_{k,j} = \boldsymbol{S}_j = \partial\boldsymbol{S}_{\boldsymbol{\lambda}}/\partial\lambda_j$. Combining (22) with (20) yields

$$\frac{\partial \mathrm{d}_0 \boldsymbol{H}_{\mathrm{L},i}(\hat{\boldsymbol{\theta}}_{t,i};y_i)}{\partial \lambda_j \mathrm{d}t} \;=\; \sum_{p=1}^{P} \boldsymbol{X}_i^{(p)} \cdot \frac{\partial \hat{\boldsymbol{w}}_k^{(p)}}{\partial \lambda_{k,j}}\,\frac{\partial \boldsymbol{H}_{\mathrm{L},i}(\hat{\boldsymbol{\theta}}_{k,i};y_i)}{\partial \theta_i^{(p)}} = \frac{\partial \boldsymbol{H}_{\mathrm{L},i}(\hat{\boldsymbol{\theta}}_{k,i};y_i)}{\partial \lambda_{k,j}},$$

and

$$\frac{\partial \mathrm{d}_0 \boldsymbol{H}_{\mathrm{L}}}{\partial \lambda_j \mathrm{d}t}(\hat{\boldsymbol{w}}_t) = \frac{\partial \boldsymbol{H}_{\mathrm{L}}}{\partial \lambda_{k,j}}(\hat{\boldsymbol{w}}_k). \tag{23}$$

Let $\hat{\boldsymbol{w}}_k^{(j)}$ denote the block of $\hat{\boldsymbol{w}}_k$ corresponding to $\boldsymbol{S}_j$ and the smooth function $f_j$. Using (23), the components of the gradient of the function $Q$ at the E-step are

$$G_j(\boldsymbol{\lambda}; \boldsymbol{\lambda}_k) \quad = \quad \frac{\partial Q(\boldsymbol{\lambda}; \boldsymbol{\lambda}_k)}{\partial \lambda_j} = \frac{1}{2}\left\{ \mathrm{Tr}\left( \boldsymbol{S}_{\boldsymbol{\lambda}}^{-} \boldsymbol{S}_j \right) - c_{k,j} \right\}, \tag{24}$$

where

$$c_{k,j} = \hat{\boldsymbol{w}}_k^{(j)T} \boldsymbol{S}_j \hat{\boldsymbol{w}}_k^{(j)} + \mathrm{Tr}\left[ \boldsymbol{H}_{\mathrm{P}}^{-1}(\hat{\boldsymbol{w}}_k; \boldsymbol{\lambda}_k) \left\{ \boldsymbol{S}_j + \frac{\partial \boldsymbol{H}_{\mathrm{L}}}{\partial \lambda_{k,j}}(\hat{\boldsymbol{w}}_k) \right\} \right] \in \mathbb{R}.$$

By construction in (6), $\boldsymbol{S}_{\boldsymbol{\lambda}}$ is a block-diagonal matrix whose blocks are of the general form $\boldsymbol{S} = \lambda_j \boldsymbol{S}_j$, which implies that $\mathrm{Tr}(\boldsymbol{S}_{\boldsymbol{\lambda}}^{-} \boldsymbol{S}_j) = \mathrm{rank}(\boldsymbol{S}_j)/\lambda_j$, and yields the closed form

$$\hat{\lambda}_{k+1,j} = \frac{\mathrm{rank}(\boldsymbol{S}_j)}{c_{k,j}}, \quad j = 1, \ldots, q, \tag{25}$$

where $c_{k,j} > 0$ by positivity of the smoothing hyper-parameters. The Hessian matrix of $Q$ is therefore diagonal with negative elements $-\mathrm{rank}(\boldsymbol{S}_j)/(2\lambda_j^2) < 0$, so (25) are always maximizers. The components of $\boldsymbol{\lambda}$ are positive, so it might be thought necessary to set $\boldsymbol{\rho} = \log \boldsymbol{\lambda}$ component-wise before the approximate EM optimization, and then back-transform afterwards. This would have led to finding the roots of

$$\tilde{G}_j(\boldsymbol{\rho}; \boldsymbol{\rho}_k) = \frac{\partial Q(\exp \boldsymbol{\rho}; \exp \boldsymbol{\rho}_k)}{\partial \rho_j} = G_j(\boldsymbol{\lambda}; \boldsymbol{\lambda}_k) \exp \rho_j,$$

which are also the roots of $G_j$ in (24), with Hessian components $-(c_{k,j} \exp \rho_j)/2 < 0$, so the positivity constraint need not be explicitly included. The diagonality of the Hessian matrix of $Q$ allows embarrassingly parallel computation of the M-step, which provides substantial speedup when $q$, the number of smooth functions, is large.

Overall, the $k$-th iteration of the approximate EM algorithm consists in

1. using the current best estimate $\boldsymbol{\lambda}_k$ to maximize the penalized log-likelihood (7) to get $\hat{\boldsymbol{w}}_k$;

2. computing $\boldsymbol{\lambda}_{k+1}$, possibly in parallel, using (25);

3. updating $k+1$ to $k$.

Learning of the regression weights is incorporated into Step 1, which is based on the Newton–Raphson algorithm. Given the trial value $\boldsymbol{w}_l$, each iteration involves

a) making $\boldsymbol{H}_{\mathrm{P}}(\boldsymbol{w}_l; \boldsymbol{\lambda}_k)$ positive definite;

b) evaluating the updating step

$$\boldsymbol{w}_{l+1} = \boldsymbol{w}_l + \alpha \boldsymbol{\Delta}_l, \quad \boldsymbol{\Delta}_l = \boldsymbol{H}_{\mathrm{P}}^{-1}(\boldsymbol{w}_l; \boldsymbol{\lambda}_k) \boldsymbol{U}_{\mathrm{P}}(\boldsymbol{w}_l; \boldsymbol{\lambda}_k),$$

where $\alpha$ is the learning rate. At step a), the positive definiteness of $\boldsymbol{H}_{\mathrm{P}}(\boldsymbol{w}_l; \boldsymbol{\lambda}_k)$ is guaranteed by increasing eigenvalues smaller than a certain positive tolerance to that tolerance. The stability of the algorithm is ensured by successively halving $\alpha$ at Step b) until the penalized log-likelihood increases.

At convergence of the Newton–Raphson algorithm, $\hat{\boldsymbol{w}}_k = \boldsymbol{w}_{l+1}$, and the identifiability of the regression weights must be checked to ensure that $\boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k; \boldsymbol{\lambda}_k)$ is invertible, since this matrix is required for calculating the smoothing hyper-parameters. By definition, the regression model is identifiable if and only if its weights are linearly independent, so we deal with lack of identifiability by keeping only the $r := \operatorname{rank}\left\{\boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k; \boldsymbol{\lambda}_k)\right\} \leqslant d$ linearly independent regression weights. An efficient and stable method to find these is QR decomposition with column pivoting (Golub and Van Loan, 2013, Section 5.4.2). The QR factorization finds a permutation matrix $\boldsymbol{P} \in \mathbb{R}^{d \times d}$ such that $\boldsymbol{H}_{\mathrm{P}}(\hat{\boldsymbol{w}}_k; \boldsymbol{\lambda}_k)\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{R}$, where the first $r$ columns of $\boldsymbol{Q}$ form an orthonormal basis for $\boldsymbol{H}_{\mathrm{P}}$. As the permutation matrix tracks the moves of the columns of $\boldsymbol{H}_{\mathrm{P}}$, the $r$ identifiable weights are the first $r$ components of the re-ordered vector $\boldsymbol{P}^T \hat{\boldsymbol{w}}_k$. Thus the remaining $d - r$ weights are linearly dependent and should be excluded from the model, together with the corresponding columns of $\boldsymbol{X}$ and rows and columns of $\boldsymbol{S}_{\boldsymbol{\lambda}_k}$.

Steps 1–3 are iterated until the gradient of the approximate log-marginal likelihood is sufficiently small. Oakes (1999) showed that this gradient can be written in terms of that of $Q$, as $\partial \ell_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y})/\partial \boldsymbol{\lambda} = G(\boldsymbol{\lambda}; \boldsymbol{\lambda}_k)|_{\boldsymbol{\lambda}_k = \boldsymbol{\lambda}}$. Since $G(\boldsymbol{\lambda}_{k+1}; \boldsymbol{\lambda}_k) = 0$, the convergence criterion is equivalent to checking that for each $j$,

$$G_j(\lambda_{k+1,j}) = \frac{1}{2}(c_{k,j} - c_{k+1,j}) < \epsilon,$$

where $\epsilon$ is a small tolerance. Furthermore, the diagonality of the Hessian of $Q$ allows one to check convergence independently for each smoothing hyper-parameter, so that only unconverged ones need be updated. In practice, the smoothing hyper-parameters may be large enough that significant changes in some components of $\boldsymbol{\lambda}$ yield insignificant changes in the penalized log-likelihood, which suggests deeming convergence when there is no significant change in $\ell_{\mathrm{P}}$. The full optimization is summarized in the three-step iteration, whose leading computational costs in the worst-case scenario are $O(nd^2)$ for the computation of the Hessian of the log-likelihood, $O(d^3)$ for its inversion, and $O(nd^2)$ for its derivative.

The approximate EM algorithm provides an elegant and straightforward approach to maximization of the log-marginal likelihood. We obtained an accurate E-step based on the approximation of Tierney et al. (1989) with relative error $O(n^{-2})$, and derived a closed form for the M-step that circumvents evaluation of the expensive and numerically unstable function $Q$. This indirect approach leads to an important simplification of the learning procedure compared to the direct Laplace approach. Since the M-step is upward, no learning rate tuning is required as in the Newton–Raphson algorithm: there is no need for intermediate evaluation of the log-marginal likelihood or its Hessian matrix. The former circumvents inner maximizations of the penalized log-likelihood and evaluation of the log-generalized

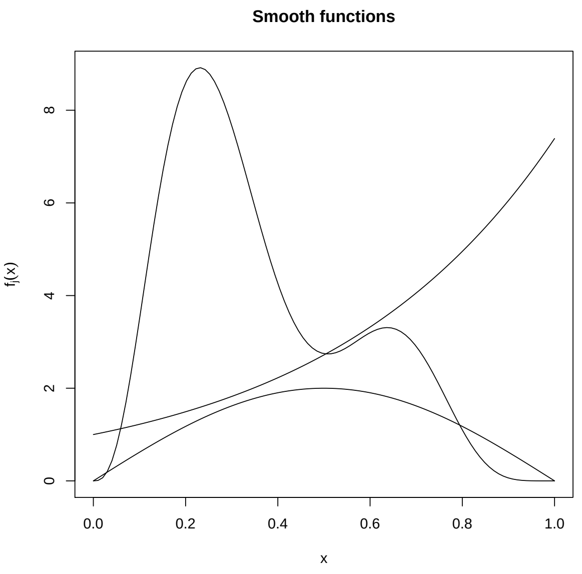

Figure 1: Original $f_j$ for $j = 1, \dots, 3$.

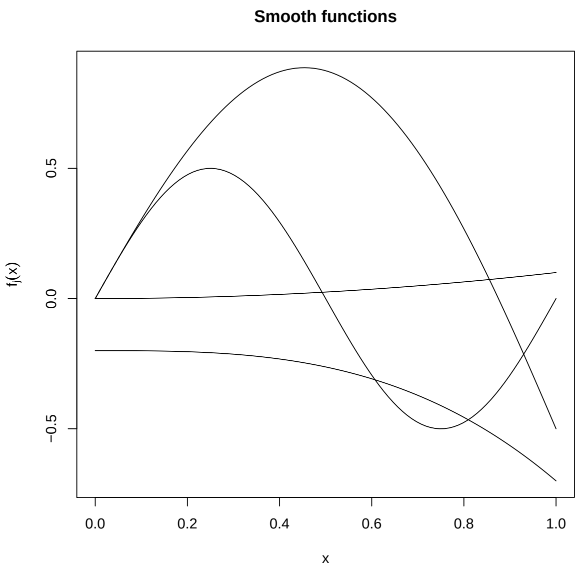

Figure 2: Original $f_j$ for $j = 4, \dots, 7$.

determinant in (12), which is unstable when the components of $\boldsymbol{\lambda}$ differ in magnitude. Furthermore, avoiding evaluation of the Hessian matrix of $\ell_{\mathrm{M}}$ bypasses computation of the fourth derivatives of the log-likelihood $\ell_{\mathrm{L}}$, which may be difficult to calculate, computationally expensive and numerically unstable (Wood, 2011; Wood et al., 2016). Moreover, the diagonality of the Hessian matrix of $Q$ allows parallelization of the M-step and update of the unconverged smoothing hyper-parameters only, giving an additional shortcut.

We assess the performance of the proposed methodology in Section 3.

## 3. Simulation Study

We generated $R = 100$ replicates of training sets of $n = 25000$ examples from a variety of probability distributions with parameters that depend on smooth functions of inputs. Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_7$ be independent vectors of $n$ identically distributed standard uniform variables. Figures 1 and 2 illustrate the seven smooth functions we considered,

$$\begin{aligned} f_1(x) &= 10^4 x^3 (1-x)^6 \left\{(1-x)^4 + 20x^8\right\}, \quad f_2(x) = 2\sin(\pi x), \quad f_3(x) = \exp(2x), \\ f_4(x) &= 0.1x^2, \quad f_5(x) = \sin(2\pi x)/2, \quad f_6(x) = -0.2 - x^3/2, \quad f_7(x) = -x^2/2 + \sin(\pi x). \end{aligned}$$

With the functional parameters

$$\mu(x_{i1}, x_{i2}, x_{i3}) = \sum_{j=1}^{3} f_j(x_{ij}), \quad \sigma(x_{i4}, x_{i5}, x_{i6}) = \sum_{j=4}^{6} f_j(x_{ij}), \quad \xi(x_{i7}) = f_7(x_{i7}), \quad i = 1, \dots, n,$$

we generated $n$ training examples from the following models:

- Gaussian distribution with mean $\mu(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ and standard deviation $\exp\{\sigma(\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6)/2\}$,

- Poisson distribution with rate $\exp\{\mu(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)/6\}$,
- Exponential distribution with rate $\exp\{\mu(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)/6\}$,
- Gamma distribution with shape $\exp\{\mu(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)/6\}$ and scale $\exp\{-\sigma(\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6)\}$,
- Binomial distribution with probability of success $1/[1 + \exp\{-\mu(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) + 5\}/6]$,
- Generalized extreme value (GEV) distribution with location $\mu(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$, scale $\exp \sigma(\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6)$ and shape $\xi(\boldsymbol{x}_7)$; see Section 4.1 for further details.

We fit the six models using cubic regression splines with evenly spaced knots in the input range values. We used ten basis functions for each of the smooth functions $f_j$. We computed the integrated mean squared error between the true and learned functional parameters, represented by hats, for each of the $r$ replicates

$$\text{MSE}(\hat{\boldsymbol{\theta}}^{(p)[r]}) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_i^{(p)[r]} - \hat{\theta}_i^{(p)[r]}\right)^2,$$

where $\hat{\boldsymbol{\theta}}^{(p)}$ is $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\sigma}}$ or $\hat{\boldsymbol{\xi}}$. Table 1 summarizes the results for the proposed approach, `multgam`, and three state-of-the-art methods implemented in the `R` packages `mgcv` gam (Wood, 2011; Wood et al., 2016) using the Newton–Raphson method, `mgcv` bam (Wood et al., 2015), and `INLA` (Rue et al., 2009). The routines used from `mgcv` are based on Version 1.8-22, and those used from `INLA` are based on Version 18.07.12 run with eight threads, using log-gamma priors and the random walk parametrization of order two. We also tried both Stan algorithms (Carpenter et al., 2017), a fully Bayesian approach with Markov chain Monte Carlo sampling and an approximate variational Bayes approach, through the `R` package `brms` (Burkner, 2017) Version 2.4.0, but a single replicate for a single functional parameter model run with four cores took five and three hours respectively, so the full simulation study would have taken more than four months, which is infeasible. Another widely-used `R` package, `VGAM`, does not offer automatic smoothing, and choosing $\boldsymbol{\lambda}$ manually for each $f_j$ for each model would have been tedious and error-prone. Use of the `R` package `gamlss` Version 5.1-0 turned out to be infeasible. Some results for the Gauss, Gamma and GEV models are missing from Table 1 because the corresponding packages do not support them. Moreover, `multgam` failed on 17 replicates for the GEV model, whereas `mgcv` gam failed on 46 replicates, so the values shown are based on 83 and 54 training sets respectively.

Table 1 shows that `multgam` is the only package which supports all the classical models, and its small errors and low variances demonstrate the high accuracy and reliability of its estimates. The proposed method is competitive with both methods in `mgcv`, whereas `INLA` is less accurate. The new method is considerably better for the GEV model; it could fit 83 of the replicates, compared to 54 for `mgcv` gam, and the estimates themselves were more accurate and less variable. The only model where all the methods give equally poor results is the binomial.

Table 2 gives a timing comparison for training sets of different sizes generated from the models described above. The computations were performed on a 2.80 GHz Intel i7-7700HQ laptop using Ubuntu. The proposed method is always fastest, more so for large training sets, and substantially outperforms `mgcv` gam and `INLA`. Moreover, it can fit the GEV

| Model | Package | $\hat{\mu}$ | $\hat{\sigma}$ | $\hat{\xi}$ |
|---|---|---|---|---|
| Gauss | `multgam` | $2.47_{2.93}$ | $0.04_{0.01}$ | – |
| | `mgcv` gam | $2.47_{2.93}$ | $0.04_{0.01}$ | – |
| Poisson | `multgam` | $1.63_{4.56}$ | – | – |
| | `mgcv` gam | $1.61_{4.67}$ | – | – |
| | `mgcv` bam | $1.62_{7.33}$ | – | – |
| | `INLA` | $9.91_{7.48}$ | – | – |
| Exponential | `multgam` | $3.67_{114.02}$ | – | – |
| | `mgcv` gam | $3.75_{115.26}$ | – | – |
| | `mgcv` bam | $3.69_{123.74}$ | – | – |
| | `INLA` | $11.95_{87.06}$ | – | – |
| Gamma | `multgam` | $1.78_{9.27}$ | $0.02_{0.01}$ | – |
| Binomial | `multgam` | $38.51_{0.99}$ | – | – |
| | `mgcv` gam | $38.51_{0.99}$ | – | – |
| | `mgcv` bam | $38.51_{0.99}$ | – | – |
| | `INLA` | $38.51_{0.99}$ | – | – |
| GEV | `multgam` | $3.58_{8.61}$ | $0.15_{0.11}$ | $0.41_{1.06}$ |
| | `mgcv` gam | $3.63_{8.56}$ | $0.27_{77.63}$ | $0.67_{336.54}$ |

Table 1: Means ($\times 10^{-2}$) over 100 replicates of the integrated mean squared errors of the learned functional parameters for a variety of models and `R` packages. The variances ($\times 10^{-6}$) appear as subscripts.

model at sizes unmatched by existing software. The package `INLA` fails with a half-million observations for all the models. Surprisingly, the proposed method is faster than `mgcv` bam, which is specifically designed for large data sets and exploits parallel computing, whereas `multgam` performs the M-step serially for fair comparison. Furthermore, the speed of `mgcv` bam should be balanced by lack of reliability of the performance iteration algorithm; see Section 1.2. Table 2 shows that speed and reliability need not be exclusive. One reason why `mgcv` gam is slow is that it evaluates the fourth derivatives of the log-likelihood. Except for the GEV model, these are not difficult to compute, but they can entail significant computational overhead, shown by the difference in performance between `multgam` and `mgcv` gam.

Overall, the new approach gives a large gain in speed with no loss in accuracy, and in some cases, it is the sole approach feasible. In Section 4 we apply it to real data.

## 4. Data Analysis

We analyze monthly maxima of temperature, which are non-stationary. Using stationary models to fit them could result in underestimation of risk, with serious potential conse-

| Model | Package | $2.5^4$ | $1^5$ | $5^5$ | $R_{2.5^4}$ | $R_{1^5}$ | $R_{5^5}$ |
|---|---|---|---|---|---|---|---|
| Gauss | `multgam` | 3.38 | 17.40 | 33.92 | 1 | 1 | 1 |
| | `mgcv` gam | 51.22 | 220.96 | 1269.16 | 15.15 | 12.70 | 37.42 |
| | `brms` MCMC | $\times$ | $359140^{(\times)}$ | $387242^{(\times)}$ | $\times$ | $\times$ | $\times$ |
| | `brms` VB | $\times$ | $384702^{(\times)}$ | $394031^{(\times)}$ | $\times$ | $\times$ | $\times$ |
| Poisson | `multgam` | 0.33 | 4.05 | 16.39 | 1 | 1 | 1 |
| | `mgcv` gam | 4.25 | 60.27 | 2157.61 | 12.88 | 14.88 | 131.64 |
| | `mgcv` bam | 1.37 | 9.83 | 30.30 | 4.15 | 2.43 | 1.85 |
| | `INLA` | 459.71 | 12077.35 | $\times$ | 1393.06 | 2982.06 | $\times$ |
| Exponential | `multgam` | 0.71 | 5.03 | 19.89 | 1 | 1 | 1 |
| | `mgcv` gam | 4.67 | 56.55 | 340.15 | 6.58 | 11.24 | 17.10 |
| | `mgcv` bam | 1.49 | 6.83 | 33.13 | 2.10 | 1.36 | 1.67 |
| | `INLA` | 466.77 | $\times$ | $\times$ | 657.42 | $\times$ | $\times$ |
| Gamma | `multgam` | 2.67 | 30.75 | 97.04 | 1 | 1 | 1 |
| Binomial | `multgam` | 0.38 | 7.90 | 19.27 | 1 | 1 | 1 |
| | `mgcv` gam | 3.33 | 58.51 | 463.46 | 8.76 | 7.41 | 24.05 |
| | `mgcv` bam | 1.23 | 8.81 | 22.60 | 3.24 | 1.12 | 1.17 |
| | `INLA` | 299.91 | 11543.32 | $\times$ | 789.24 | 1461.18 | $\times$ |
| GEV | `multgam` | 8.24 | 440.22 | $\times$ | 1 | 1 | $\times$ |
| | `mgcv` gam | 167.13 | $\times$ | $\times$ | 20.28 | $\times$ | $\times$ |

Table 2: Times (s) for a variety of models and three training set sizes $n = 2.5 \times 10^4$, $5 \times 10^5$, $5 \times 10^5$. The times correspond to averages over 100 replicates when $n = 2.5 \times 10^4$. The notation $x^y$ means $x \times 10^y$. The notation $t^{(\times)}$ means the computation failed to converge after $t$ seconds, and $\times$ indicates failure to converge or that computations turned out to be infeasible. The ratios $R_{x^y}$ are with respect to `multgam`, which does not benefit from the parallelization of the M-step.

quences. The generalized extreme-value distribution, widely used for modeling maxima and minima, will serve as our underlying probability model.

### 4.1. Model

Let $y_1, \ldots, y_n$ be the maxima of blocks of observations from an unknown probability distribution. Extreme value theory (Fisher and Tippett, 1928; de Haan and Ferreira, 2006) implies that as the block size increases and under mild conditions, each of the $y_i$ follows a $\text{GEV}(\mu_i, \sigma_i, \xi_i)$ distribution with parameters the location $\mu_i \in \mathbb{R}$, the scale $\sigma_i > 0$ and the shape $\xi_i \in \mathbb{R}$,

$$F(y_i;\mu_i,\sigma_i,\xi_i) \quad = \quad \begin{cases} \exp\left[-\left\{1+\xi_i\left(\dfrac{y_i-\mu_i}{\sigma_i}\right)\right\}_+^{-1/\xi_i}\right], & \xi_i \neq 0, \\ \exp\left[-\exp\left\{-\left(\dfrac{y_i-\mu_i}{\sigma_i}\right)\right\}\right], & \xi_i = 0, \end{cases}$$

where $a_+ = \max(a,0)$. This encompasses the three classical models for maxima (Jenkinson, 1955): if $\xi_i > 0$, the distribution is Fréchet; if $\xi_i < 0$, it is reverse Weibull; and if $\xi_i = 0$, it is Gumbel. The shape parameter is particularly important since it controls the tail properties of the distributions. The expectation of $Y_i$ is

$$E(Y_i) = \begin{cases} \mu_i + \dfrac{\sigma_i}{\xi_i}\left\{\Gamma(1-\xi_i)-1\right\}, & \xi_i \neq 0,\ \xi_i < 1, \\ \mu_i + \gamma\sigma_i, & \xi_i = 0, \\ \infty, & \xi_i \geqslant 1, \end{cases} \tag{26}$$

where $\gamma$ is Euler's constant. Non-stationarity of (26) could stem from changes in any of the parameters, and as interpretability is priority in risk assessment, a GEV model with functional parameters having flexible additive structure is well justified.

Most data analyses involving non-stationary extremes use a parametric or semi-parametric form in the location and/or scale parameters while keeping the shape a fixed scalar (Chavez-Demoulin and Davison, 2012, Section 4), even though it may be plausible that this varies. Seasonal effects, for example, may stem from different physical processes with different extremal behaviors. Fixing the shape parameter in such cases is a pragmatic choice driven by the difficulty of learning it from limited data in a numerically stable manner. Chavez-Demoulin and Davison (2005) learn a functional shape parameter for the generalized Pareto distribution, but their approach has some drawbacks. First, training is based on backfitting, which does not allow automatic smoothness selection, and so involves manual tuning of the smoothing hyper-parameters. Second, the optimization is in the spirit of performance iteration, with one updating step for the regression model followed by one full optimization for smoothing; limitations of this were discussed in Section 1.2. Third, optimization is sequential rather than simultaneous, by alternating a regression step for each smooth term when there are several, and alternating backfitting steps for each functional parameter separately. Fourth, convergence may only be guaranteed when the functional parameters are orthogonal, meaning that the methodology may not extend to more than two. Moreover, the smoothing method is applied to orthogonalized distribution parameters that may be awkward to interpret. We learn a functional shape in a generic and stable manner; this is of separate interest for the modeling of non-stationary extremes.

In our earlier general terms, $\theta_i^{(1)} = \mu_i$, $\theta_i^{(2)} = \exp\tau_i$, where $\tau_i = \log\sigma_i$ to ensure positivity of the scale, and $\theta_i^{(3)} = \xi_i$. Let $\Omega$ and $\Omega_0$ denote the partition of the support as

$$\begin{aligned} \Omega &= \left\{y_i \in \mathbb{R} : \xi_i > 0, y_i > \mu_i - \exp\tau_i/\xi_i\right\} \cup \left\{y_i \in \mathbb{R} : \xi_i < 0, y_i < \mu_i - \exp\tau_i/\xi_i\right\}, \\ \Omega_0 &= \left\{(y_i,\xi_i) :\ y_i \in \mathbb{R}, \xi_i = 0\right\}. \end{aligned}$$

The corresponding log-likelihood is then

$$\ell_{\mathrm{L}}(\boldsymbol{\mu}, \boldsymbol{\tau}, \boldsymbol{\xi}; \boldsymbol{y}) \;=\; \sum_{i=1}^{n} \ell_{\mathrm{L}}^{(i)}(\mu_i, \tau_i, \xi_i; y_i),$$

where the individual contributions are

$$\ell_{\mathrm{L}}^{(i)}(\mu_i, \tau_i, \xi_i; y_i) \;=\; \begin{cases} -\tau_i - \left(1 + \dfrac{1}{\xi_i}\right) \log(1 + z_i) - (1 + z_i)^{-1/\xi_i}, & y_i \in \Omega, \\ -\tau_i - \exp(-z_i) - z_i, & (y_i, \xi_i) \in \Omega_0, \end{cases}$$

with

$$z_i = \begin{cases} (y_i - \mu_i)\xi_i \exp(-\tau_i), & y_i \in \Omega, \\ (y_i - \mu_i) \exp(-\tau_i), & (y_i, \xi_i) \in \Omega_0. \end{cases}$$

This log-likelihood becomes numerically unstable when $\xi_i$ and $z_i$ are close to zero, while overflow is amplified as the order of the derivatives increases. The proposed approximate EM method requires the third derivatives of the log-likelihood, which involve terms like $\xi_i^{-5}$. When $\xi_i \approx 0$, the threshold below which the absolute value of the shape parameter should be set to zero is therefore troublesome. Its value should reflect the compromise between stability of the derivatives and the switch from the general GEV form to the Gumbel distribution. The numerical instability is even more problematic in the `mgcv` gam method, which requires the fourth log-likelihood derivatives; the lower the order of the derivatives, the fewer unstable computations. In our implementation, we set $\xi_i = 0$ whenever $|\xi_i| \leq \varpi^{3/10}$, with $\varpi$ the machine precision. This sets the order of the threshold to $10^{-5}$, while allowing negative exponents of $\xi_i$ terms to grow up to $10^{25}$, which is within the range of precision of all modern machines.

## 4.2. Application

We analyze monthly maxima of the daily Central England Temperature (CET)[1] series from January 1772 to December 2016. Figure 3 shows yearly maxima and suggests that the recent years are the warmest, while panel a) in Figure 4 indicates that any increase is most apparent at the end of the year. Figure 4 exhibits obvious seasonality, which we represent using 12 basis functions from cyclic cubic regression splines for each of the location, scale and shape parameters of the GEV model; we use ten basis functions from thin plate splines (Wood, 2003) in the location for the trend visible in Figure 3. We initially included long-term trends in the scale and shape, but they were not significant. We also tried more complex models with larger basis dimensions, but these did not improve the accuracy of the estimates. To our knowledge, this is the only paper modeling a variable shape parameter for this data set. Neither of the algorithms in Stan (Carpenter et al., 2017) using the `R` package `brms` (Burkner, 2017) converged, and the variational Bayes approach faced numerical instabilities.

Panel a) of Figure 5 shows annual variation of 11°C, similar to that seen in Figure 4, and panel b) of Figure 5 shows a non-linear trend with a drop from 1772 to 1800 and a sharp increase from the 1960s onwards. The pattern between is hard to discern in Figure 3,

1. The data can be downloaded at `https://www.metoffice.gov.uk/hadobs/hadcet/data/download.html`

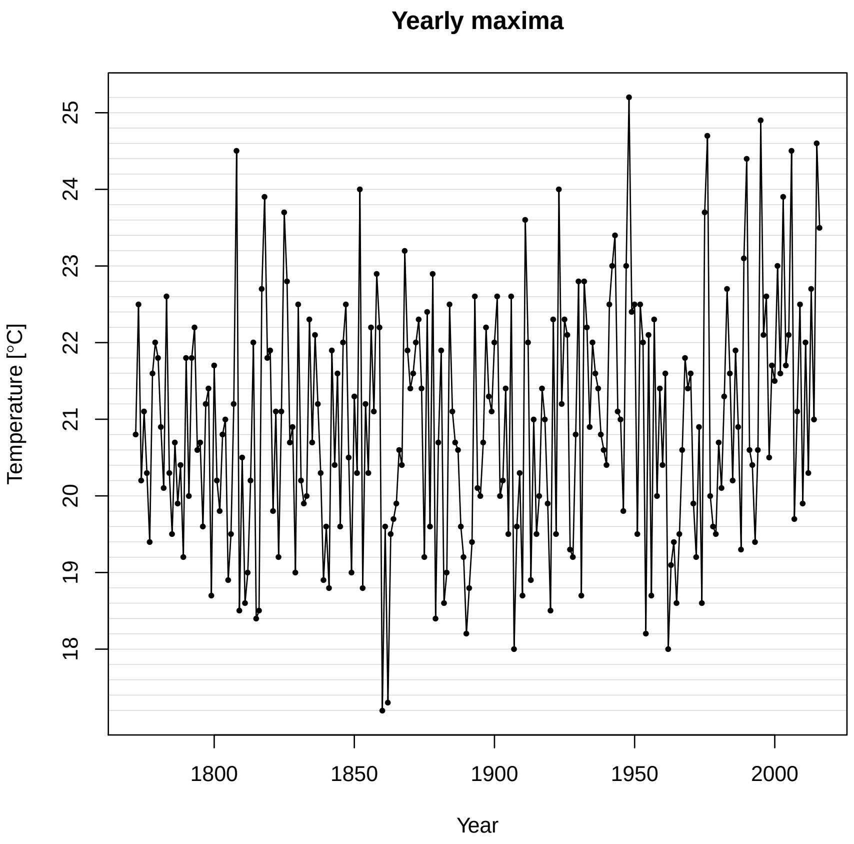

Figure 3: Annual maxima for the CET data.

but panel b) shows an overall increase of about 1.5°C from 1800 onwards and peaks over the last few decades.

The scale and shape parameters in Figure 5, whose functional forms vary significantly through the year, give insight into the seasonality. They are negatively correlated except in mid-June to September, where the increase in the shape is much slower and weaker than the drop in the scale. We can distinguish two cycles within the year, with similar patterns but different intensities: the extended winter from September to April, and the extended weak summer, from April to September. Each incorporates two antagonistic phases that are negatively correlated, alternating between decrease and increase for the shape, and vice-versa for the scale.

Figure 5 summarizes the influences of the scale and the shape parameters on the seasonality of the CET data as follows: whether the temperature is increasing or decreasing seems to be smoothly related to the direction of the shape in the winter, and to that of the scale in the summer. Since the former controls the tail of the distribution and is always significantly negative here, the temperature is bounded above throughout the year; the strongest

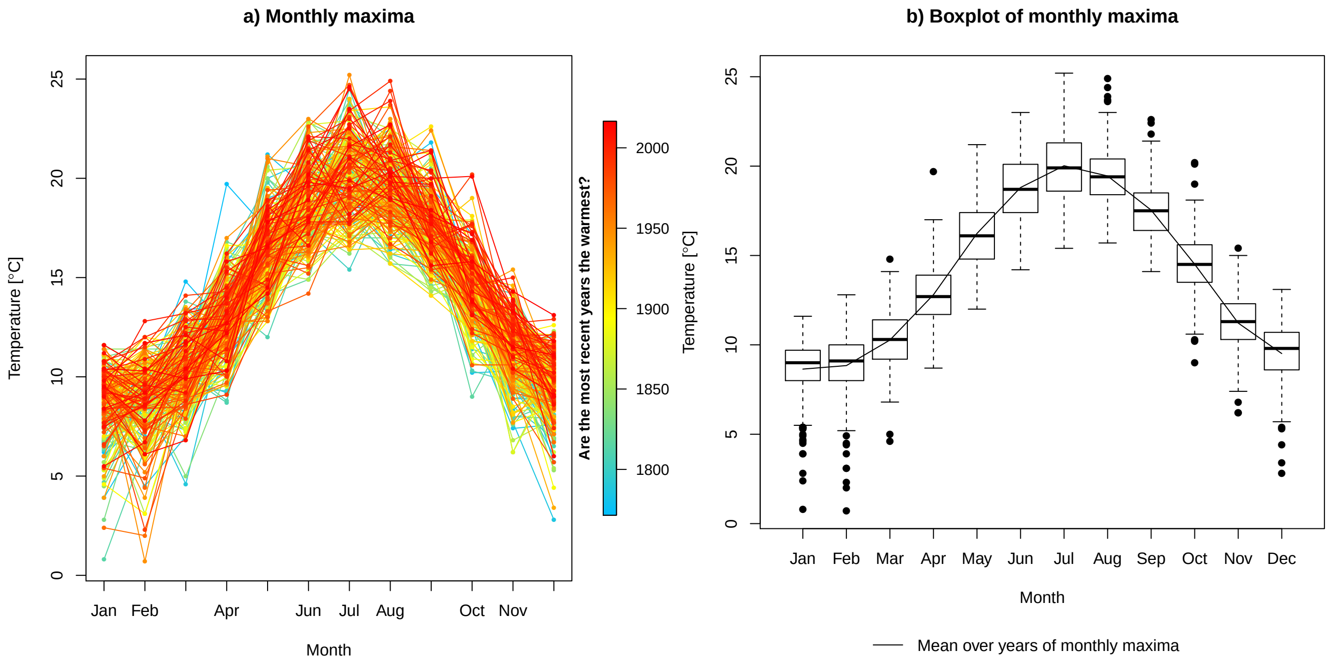

Figure 4: Monthly maxima for the CET data. Left: annual cycles with the year coded by color according to the scale to the right of the panel. Right: boxplots for monthly maxima.

increase of the shape occurs in February to mid-April, early spring, stabilizing around its highest values, $-0.2$ or so, in the summer. This stabilization and the negative correlation between the scale and the shape explain why the sharper fluctuations of the scale have more impact on the temperature in the summer than the near-constant shape. The rather narrow point-wise confidence intervals suggest that there is very strong evidence for seasonal variation of the shape, and less strong but still appreciable evidence of such variation for the scale. Davison and Ramesh (2000) fitted a local likelihood smooth model to the five largest daily data (after declustering) for each year up to 1996, using linear polynomials for the location and the scale parameters and a constant shape parameter. Their results are consistent with ours in that the changes in the upper extremes are driven by the changes in the scale parameter.

Figures 6 and 7 illustrate model fit diagnostics. Figure 6 shows that the true maxima are within the range of those simulated from the learned model. Figure 7 represents the predicted 0.95, 0.98 and 0.99 quantiles for monthly maxima. Based on the model for 1916, only one value from previous years, 24.5°C in July 1808, exceeded the maximum of the 0.99 quantile curve, 24.4°C, in July; all other exceedances occur after 1916. The maximum of the 0.99 quantile curve in 2016 occurred in July at 25.4°C, and no higher temperature has been observed.

Overall, the model does not seem unrealistic, although it may underestimate slightly the uncertainty, as it assumes independence of maxima in successive months. A possible

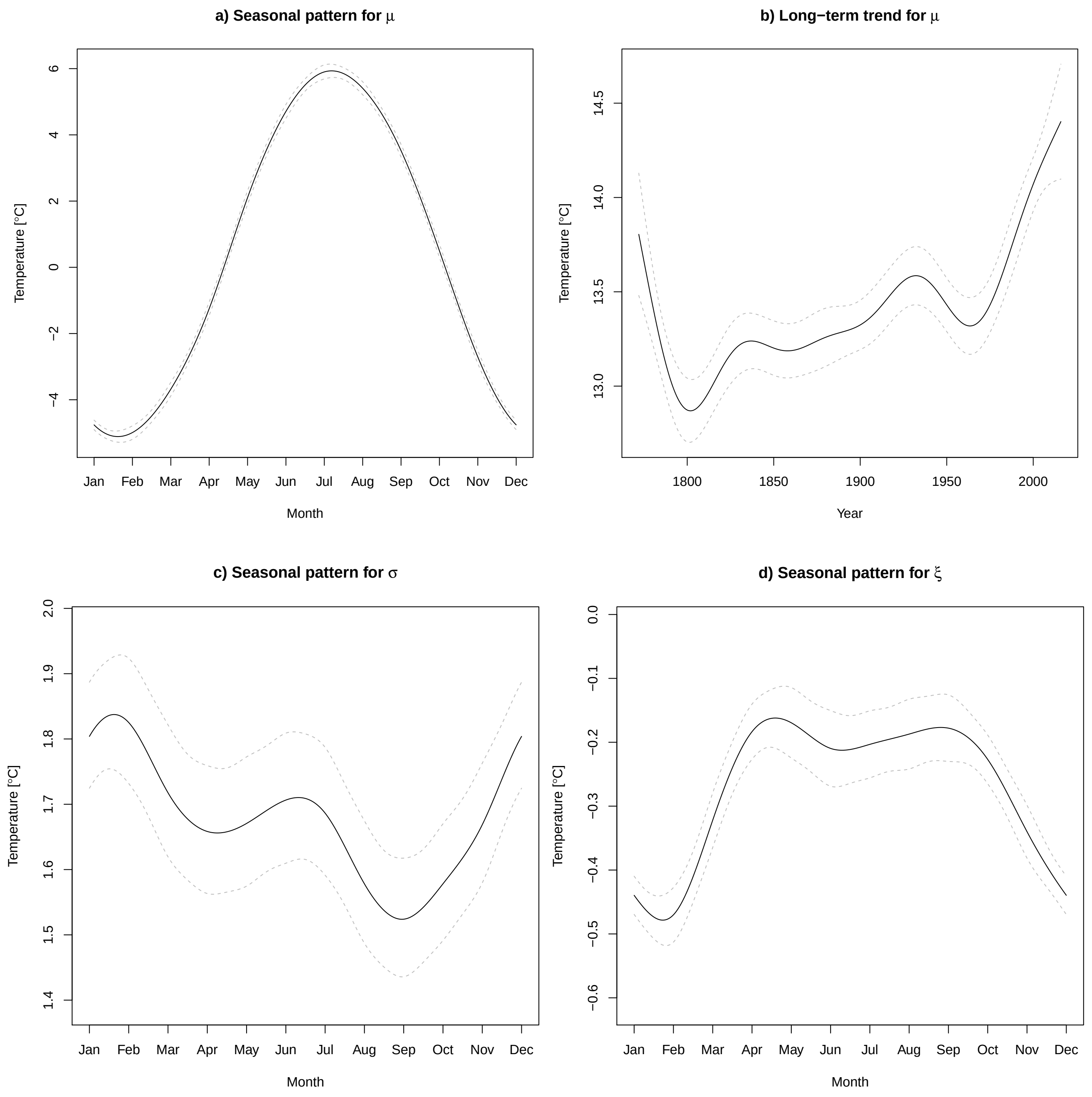

Figure 5: Learned functional parameters, with 95% point-wise confidence intervals (dashes) obtained from the Hessian of the penalized log-likelihood.

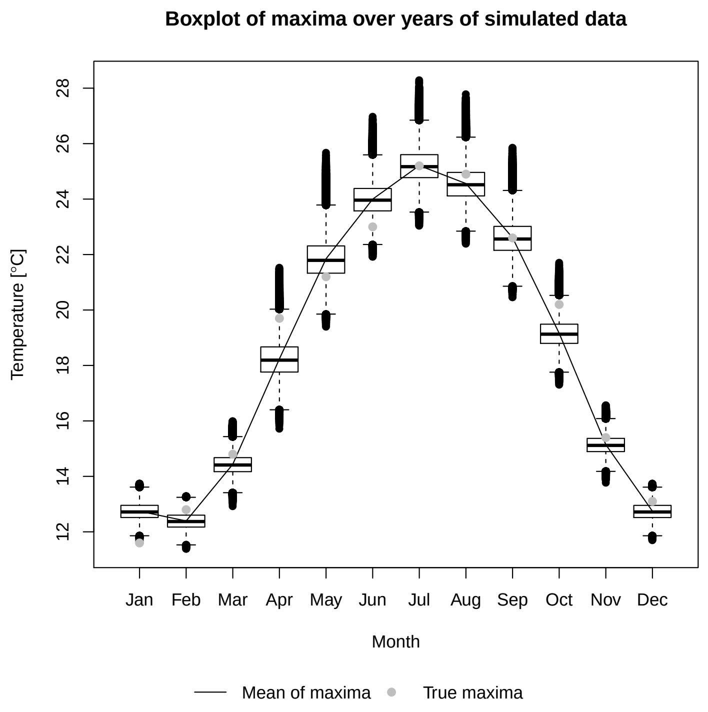

Figure 6: Monthly maxima simulated from the learned GEV model.

improvement would be the use of dependent errors, but this is outside the scope of the present paper.

## 5. Discussion

This paper makes contributions to optimal and automatic smoothing for generalized additive models using an empirical Bayes approach. The roughness penalty corresponds to a weighted $L_2$ regularization interpreted as a Gaussian prior on the regression weights, and the penalized log-likelihood is the associated posterior. We maximize the resulting log-marginal likelihood with respect to the smoothing hyper-parameters using an EM algorithm, made tractable via double Laplace approximation of the moment generating function underlying the E-step. The proposed approach transfers maximization of the log-marginal likelihood to a function whose maximizer has a closed form, and avoids evaluation of expensive and numerically unstable terms. The only requirement is that the log-likelihood has third derivatives. The proposed method is stable, accurate and fast. Its stability is ensured both by the EM algorithm and by its need for fewer derivatives, making the proposed method broadly applicable for complex models. Its high accuracy is established theoretically by Tierney et al. (1989), with an $O(n^{-2})$ relative error in the Laplace approximation at the E-step, and we show in Appendix A that this leads to an error of order $O(n^{-3})$ in the learned smoothing hyper-parameters. Its serial implementation is substantially faster than the best existing

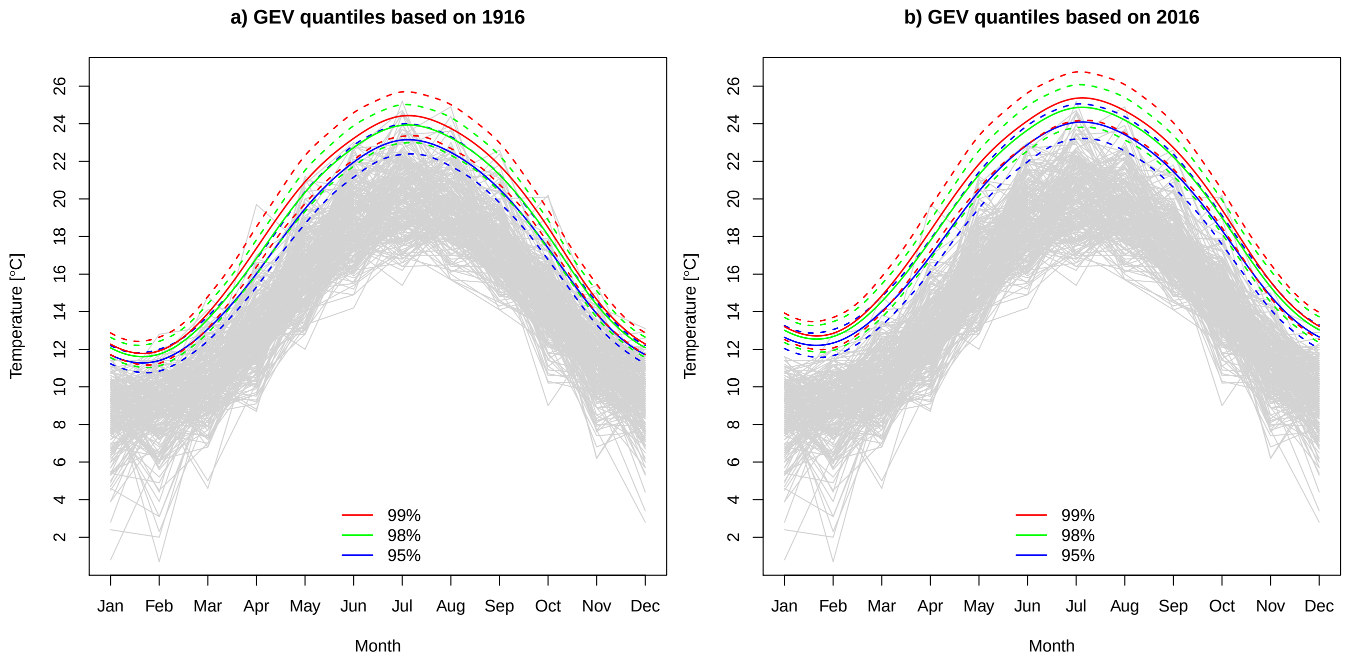

Figure 7: Superposition of the original data (grey) and quantiles of the GEV models, with 95% point-wise confidence intervals (dashes) obtained from the Hessian of the penalized log-likelihood.

methods and achieves state-of-the-art accuracy. It can easily be parallelized, making it appealing for extension to big-data settings, where no reliable method yet exists.

These advantages are balanced by potential difficulties. First, the EM algorithm can be slow around the optimum, though the simulations of Section 3 required no more than fifty iterations. Tests show that deceleration occurs when certain smoothing hyper-parameters become so large that their corresponding smooth functions are linear, and their updates no longer change the penalized log-likelihood. At that point, we declare convergence for those components of $\boldsymbol{\lambda}$, though they may keep changing without affecting the regression weights. Validating convergence for a portion of smoothing hyper-parameters and updating the remainder is supported by the diagonality of the Hessian matrix of the function $Q$ at the E-step. Second, the EM algorithm is known to suffer from local optima, though we found none in the data sets and the simulated models we analyzed, perhaps because the log-likelihood is fairly quadratic for large samples.

The proposed method is implemented in a C++ library that uses Eigen (Guennebaud et al., 2018) for matrix decompositions, is integrated into the `R` package `multgam` through the interface `RcppEigen` (Bates and Eddelbuettel, 2013), and makes addition of further probability models straightforward.

## Acknowledgments

The work was financially supported by the ETH domain Competence Center Environment and Sustainability (CCES) project Osper.

## Appendix A. Approximation error of the smoothing hyper-parameters

We discuss the approximation error on the final estimate of the smoothing hyper-parameters implied by the approximate E-Step in Section 2.2. Let $\ell^*_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y})$ denote the Laplace approximation to the log-marginal likelihood $\ell_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y})$ based on $n$ independent observations. If we write the relative error of the Laplace approximation for the corresponding density functions as $1 + A(\boldsymbol{\lambda}; \boldsymbol{y})/n^a + O(n^{-a-1})$ for some positive $a$, then on taking logs we obtain

$$\ell_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y}) = \ell^*_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y}) + A(\boldsymbol{\lambda}; \boldsymbol{y})/n^a + O(n^{-a-1}), \tag{27}$$

where $\ell_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y})$ and $\ell^*_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y})$ are each essentially sums of $n$ terms, and thus are $O(n)$, and $A(\boldsymbol{\lambda}; \boldsymbol{y})$ is $O(1)$.

Now suppose that $\ell_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y})$ has two continuous derivatives and an invertible Hessian matrix throughout an open convex subset $O \subset \mathbb{R}^q$ surrounding the maximum likelihood estimate $\hat{\boldsymbol{\lambda}}$. Then for any $\boldsymbol{\lambda}, \boldsymbol{\lambda}^* \in O$, the mean value theorem implies that we can write

$$\frac{\partial \ell_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y})}{\partial \lambda_j} = \frac{\partial \ell_{\mathrm{M}}(\boldsymbol{\lambda}^*; \boldsymbol{y})}{\partial \lambda_j} + \mathrm{D}\left\{ \frac{\partial \ell_{\mathrm{M}}(\boldsymbol{\lambda}^\dagger_{(j)}; \boldsymbol{y})}{\partial \lambda_j} \right\} \cdot (\boldsymbol{\lambda} - \boldsymbol{\lambda}^*), \quad j = 1, \dots, q,$$

where $\mathrm{D}\{\partial \ell_{\mathrm{M}}(\boldsymbol{\lambda}^\dagger_{(j)}; \boldsymbol{y})/\partial \lambda_j\} \in \mathbb{R}^q$ is a vector whose $k$-th element is $\partial^2 \ell_{\mathrm{M}}(\boldsymbol{\lambda}^\dagger_{(j)}; \boldsymbol{y})/\partial \lambda_j \partial \lambda_k$, $\boldsymbol{\lambda}^\dagger_{(j)} = \boldsymbol{\lambda} + t_j(\boldsymbol{\lambda} - \boldsymbol{\lambda}^*)$ for $t_j \in [0, 1]$, and $\cdot$ is the scalar product. If we put these equations together in matrix form we can write

$$\boldsymbol{U}_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y}) - \boldsymbol{U}_{\mathrm{M}}(\boldsymbol{\lambda}^*; \boldsymbol{y}) = \boldsymbol{J}(\boldsymbol{\lambda}^\dagger_{(1)}, \dots, \boldsymbol{\lambda}^\dagger_{(q)})(\boldsymbol{\lambda} - \boldsymbol{\lambda}^*),$$

where $\boldsymbol{U}_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y}) \in \mathbb{R}^q$ is the vector whose $j$-th component is $\partial \ell_{\mathrm{M}}(\boldsymbol{\lambda}; \boldsymbol{y})/\partial \lambda_j$, and $\boldsymbol{J}(\boldsymbol{\lambda}^\dagger_{(1)}, \dots, \boldsymbol{\lambda}^\dagger_{(q)}) \in \mathbb{R}^{q \times q}$ is a matrix whose $j$-th row contains the vector $\mathrm{D}\{\partial \ell_{\mathrm{M}}(\boldsymbol{\lambda}^\dagger_{(j)}; \boldsymbol{y})/\partial \lambda_j\}$ and is therefore of order $O(n)$ since each element is a sum of $n$ log-likelihood derivatives.

The maximum likelihood estimate $\hat{\boldsymbol{\lambda}}$ and its counterpart $\hat{\boldsymbol{\lambda}}^*$ based on the Laplace approximation satisfy

$$\boldsymbol{U}_{\mathrm{M}}(\hat{\boldsymbol{\lambda}}; \boldsymbol{y}) = \boldsymbol{0}_q, \quad \boldsymbol{U}^*_{\mathrm{M}}(\hat{\boldsymbol{\lambda}}^*; \boldsymbol{y}) = \boldsymbol{0}_q.$$

On writing $\dot{\boldsymbol{A}}(\boldsymbol{\lambda}; \boldsymbol{y}) \in \mathbb{R}^q$ for the vector of components $\partial A(\boldsymbol{\lambda}; \boldsymbol{y})/\partial \lambda_j$, and denoting the vector of error terms of order $O(n^{-a-1})$ by $\boldsymbol{O}(n^{-a-1}) \in \mathbb{R}^q$, we deduce from (27) that

$$\boldsymbol{0}_q = \boldsymbol{U}^*_{\mathrm{M}}(\hat{\boldsymbol{\lambda}}^*; \boldsymbol{y}) = \boldsymbol{U}_{\mathrm{M}}(\hat{\boldsymbol{\lambda}}^*; \boldsymbol{y}) - \dot{\boldsymbol{A}}(\hat{\boldsymbol{\lambda}}^*; \boldsymbol{y})/n^a + \boldsymbol{O}(n^{-a-1}),$$

which yields

$$\begin{aligned} \boldsymbol{J}(\hat{\boldsymbol{\lambda}}^\dagger_{(1)}, \dots, \hat{\boldsymbol{\lambda}}^\dagger_{(q)})\,(\hat{\boldsymbol{\lambda}} - \hat{\boldsymbol{\lambda}}^*) &= \boldsymbol{U}_{\mathrm{M}}(\hat{\boldsymbol{\lambda}}; \boldsymbol{y}) - \boldsymbol{U}_{\mathrm{M}}(\hat{\boldsymbol{\lambda}}^*; \boldsymbol{y}) \\ &= -\dot{\boldsymbol{A}}(\hat{\boldsymbol{\lambda}}^*; \boldsymbol{y})/n^a + \boldsymbol{O}(n^{-a-1}). \end{aligned}$$

Since $\boldsymbol{J}(\hat{\boldsymbol{\lambda}}^{\dagger}_{(1)}, \ldots, \hat{\boldsymbol{\lambda}}^{\dagger}_{(q)})$ is $O(n)$ and $\dot{\boldsymbol{A}}(\boldsymbol{\lambda}; \boldsymbol{y})$ is $O(1)$, we must have $\hat{\boldsymbol{\lambda}} - \hat{\boldsymbol{\lambda}}^{*} = \boldsymbol{O}(n^{-a-1})$.

In a conventional Laplace approximation, we would have $a = 1$, and then $\hat{\boldsymbol{\lambda}} - \hat{\boldsymbol{\lambda}}^{*} = \boldsymbol{O}(n^{-2})$, but with the double Laplace approximation used in the proposed method we have $a = 2$, so $\hat{\boldsymbol{\lambda}} - \hat{\boldsymbol{\lambda}}^{*} = \boldsymbol{O}(n^{-3})$. Thus the maximizers of the true log-marginal likelihood and its Laplace approximation differ by much less than the 'statistical variation' of $\hat{\boldsymbol{\lambda}}$, which is $O_p(n^{-1/2})$ away from the 'true' $\boldsymbol{\lambda}$. Hence the difference $\hat{\boldsymbol{\lambda}} - \hat{\boldsymbol{\lambda}}^{*}$ is statistically negligible, and the same will apply to the corresponding values of $\hat{\boldsymbol{\beta}}$, which are smooth functions of $\boldsymbol{\lambda}$.

## References

A. Ba, M. Sinn, Y. Goude, and P. Pompey. Adaptive learning of smoothing functions: application to electricity load forecasting. In *Advances in Neural Information Processing Systems 25*, pages 2510–2518. USA, 2012.

D. Bates and D. Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013.

L. Breiman and J. H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80(391):580–598, 1985.

J. C. Burkner. brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1):1–28, 2017.

B. Carpenter, D. Lee, M. A. Brubaker, A. Riddell, A. Gelman, B. Goodrich, J. Guo, M. Hoffman, M. Betancourt, and P. Li. Stan: a probabilistic programming language, 2017.

V. Chavez-Demoulin and A. C. Davison. Generalized additive modelling of sample extremes. *Journal of the Royal Statistical Society, Series C*, 54(1):207–222, 2005.

V. Chavez-Demoulin and A. C. Davison. Modelling time series extremes. *Revstat-Statistical Journal*, 10:109–133, 2012.

W. S. Cleveland, E. Grosse, and W. M. Shyu. *Local Regression Models.* Chapman & Hall, New York, 1993.

T. J. Cole and P. J. Green. Smoothing reference centile curves: the LMS method and penalized likelihood. *Statistics in Medicine*, 11(10):1305–1319, 1992.

A. C. Davison and N. I. Ramesh. Local likelihood smoothing of sample extremes. *Journal of the Royal Statistical Society, Series B*, 62:191–208, 2000.

L. de Haan and A. Ferreira. *Extreme Value Theory.* Springer-Verlag New York, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

D. K. Duvenaud, H. Nickisch, and C. E. Rasmussen. Additive Gaussian processes. In *Advances in Neural Information Processing Systems 24*, pages 226–234. USA, 2011.

A. C. Faul and M. E. Tipping. Analysis of sparse Bayesian learning. In *Advances in Neural Information Processing Systems 14*, pages 383–389. Cambridge, USA, 2001.

R. A. Fisher and L. H. C. Tippett. Limiting forms of the frequency distributions of the largest or smallest member of a sample. *Proceedings of the Cambridge Philosophical Society*, 24:180–190, 1928.

G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, Maryland, 4th edition, 2013.

C. Gu. Cross-validating non-Gaussian data. *Journal of Computational and Graphical Statistics*, 1(2):169–179, 1992.

G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2018.

T. J. Hastie and R. J. Tibshirani. Generalized additive models (with discussion). *Statistical Science*, 1:297–310, 1986.

T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models.* Chapman & Hall, 1990.

T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, 2nd edition, 2009.

A. F. Jenkinson. The frequency distribution of the annual maximum (or minimum) values of meteorological elements. *Journal of the Royal Meteorological Society*, 81:158–171, 1955.

G. S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2): 495–502, 1970.

D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

D. J. C. MacKay. Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11(5):1035–1068, 1999.

A. McHutchon and C. E. Rasmussen. Gaussian process training with input noise. In *Advances in Neural Information Processing Systems 24*, pages 1341–1349. USA, 2011.

G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions (Wiley Series in Probability and Statistics).* Wiley-Interscience, 2nd edition, 2008.

M. Mutny and A. Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. In *Advances in Neural Information Processing Systems 31*, pages 9005–9016. USA, 2018.

R. M. Neal. *Bayesian Learning for Neural Networks.* Springer-Verlag, Berlin, Heidelberg, 1996.

J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society, Series A*, 135(3):370–384, 1972.

D. Nychka. Bayesian confidence intervals for smoothing splines. *Journal of the American Statistical Association*, 83(404):1134–1143, 1988.

D. Oakes. Direct calculation of the information matrix via the EM. *Journal of the Royal Statistical Society, Series B*, 61(2):479–482, 1999.

F. O'Sullivan, B. S. Yandell, and W. J. Raynor. Automatic smoothing of regression functions in generalized linear models. *Journal of the American Statistical Association*, 81(393): 96–103, 1986.

Y. Qi, T. P. Minka, R. W. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. In *International Conference on Machine Learning*, page 85, New York, USA, 2004.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2019.

P. T. Reiss and R. T. Ogden. Smoothing parameter selection for a class of semiparametric linear models. *Journal of the Royal Statistical Society, Series B*, 71(2):505–523, 2009.

R. A. Rigby and D. M. Stasinopoulos. A semi-parametric additive model for variance heterogeneity. *Statistics and Computing*, 6(1):57–65, 1996.

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape (with discussion). *Journal of the Royal Statistical Society, Series C*, 54(3):507–554, 2005.

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society, Series B*, 71(2):319–392, 2009.

B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society, Series B*, 47(1):1–52, 1985.

B. M. Steele. A modified EM algorithm for estimation in generalized mixed models. *Biometrics*, 52(4):1295–1310, 1996.

L. Tierney, R. E. Kass, and J. B. Kadane. Fully exponential Laplace approximations to expectations and variances of nonpositive functions. *Journal of the American Statistical Association*, 84(407):710–716, 1989.

M. E. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems 12*, pages 652–658, Cambridge, USA, 1999.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

M. Tsang, H. Liu, S. Purushotham, P. Murali, and Y. Liu. Neural interaction transparency: disentangling learned interactions for improved interpretability. In *Advances in Neural Information Processing Systems 31*, pages 5804–5813. USA, 2018.

E. F. Vonesh, H. Wang, L. Nie, and D. Majumdar. Conditional second-order generalized estimating equations for generalized linear and nonlinear mixed-effects models. *Journal of the American Statistical Association*, 97(457):271–283, 2002.

S. N. Wood. Thin plate regression splines. *Journal of the Royal Statistical Society, Series B*, 65(1):95–114, 2003.

S. N. Wood. Fast stable direct fitting and smoothness selection for generalized additive models. *Journal of the Royal Statistical Society, Series B*, 70(3):495–518, 2008.

S. N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society, Series B*, 73(1):3–36, 2011.

S. N. Wood, Y. Goude, and S. Shaw. Generalized additive models for large data sets. *Journal of the Royal Statistical Society, Series C*, 64(1):139–155, 2015.

S. N. Wood, N. Pya, and B. Safken. Smoothing parameter and model selection for general smooth models. *Journal of the American Statistical Association*, 111(516):1548–1563, 2016.

S. N. Wood, Z. Li, G. Shaddick, and N. H. Augustin. Generalized additive models for gigadata: modeling the UK black smoke network daily data. *Journal of the American Statistical Association*, 112(519):1199–1210, 2017.

T. W. Yee and C. J. Wild. Vector generalized additive models. *Journal of the Royal Statistical Society, Series B*, 58(3):481–493, 1996.