Statistics and Computing (2023) 33:1
https://doi.org/10.1007/s11222-022-10178-z

ORIGINAL PAPER

Automatic search intervals for the smoothing parameter in penalized splines

Zheyuan Li[1] · Jiguo Cao[2]

Received: 30 May 2022 / Accepted: 29 October 2022 / Published online: 18 November 2022

Abstract
The selection of smoothing parameter is central to the estimation of penalized splines. The best value of the smoothing parameter is often the one that optimizes a smoothness selection criterion, such as generalized cross-validation error (GCV) and restricted likelihood (REML). To correctly identify the global optimum rather than being trapped in an undesired local optimum, grid search is recommended for optimization. Unfortunately, the grid search method requires a pre-specified search interval that contains the unknown global optimum, yet no guideline is available for providing this interval. As a result, practitioners have to find it by trial and error. To overcome such difficulty, we develop novel algorithms to automatically find this interval. Our automatic search interval has four advantages. (i) It specifies a smoothing parameter range where the associated penalized least squares problem is numerically solvable. (ii) It is criterion-independent so that different criteria, such as GCV and REML, can be explored on the same parameter range. (iii) It is sufficiently wide to contain the global optimum of any criterion, so that for example, the global minimum of GCV and the global maximum of REML can both be identified. (iv) It is computationally cheap compared with the grid search itself, carrying no extra computational burden in practice. Our method is ready to use through our recently developed **R** package **gps** ($\geq$ version 1.1). It may be embedded in more advanced statistical modeling methods that rely on penalized splines.

Keywords Grid search · O-splines · Penalized B-splines · P-splines

1 Introduction

Penalized splines are flexible and popular tools for estimating unknown smooth functions. They have been applied in many statistical modeling frameworks, including generalized additive models (Spiegel et al. 2019; Cao 2012), single-index models (Wang et al. 2018), functional single-index models (Jiang et al. 2020), generalized partially linear single-index models (Yu et al. 2017), functional mixed-effects models (Chen et al. 2018; Cao and Ramsay 2010; Liu et al. 2017), survival models (Orbe and Virto 2021; Bremhorst and Lambert 2016), trajectory modeling for longitudinal data (Koehler et al. 2017; Andrinopoulou et al. 2018; Nie et al. 2022), additive quantile regression models (Muggeo et al. 2021; Sang and Cao 2020), varying coefficient models (Hendrickx et al. 2018), quantile varying coefficient models (Gijbels et al. 2018), spatiotemporal models (Minguez et al. 2020; Goicoa et al. 2019), spatiotemporal quantile and expectile regression models (Franco-Villoria et al. 2019; Spiegel et al. 2020).

To explain the fundamental idea of a penalized spline, consider the following smoothing model for observations (x_i, y_i), $i = 1, \ldots, n$:

$$y_i = f(x_i) + e_i, \quad e_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2).$$

After expressing $f(x) = \sum_{j=1}^{p} \mathcal{B}_j(x)\beta_j$ with some spline basis $\mathcal{B}_j(x)$, $j = 1, \ldots, p$, we estimate basis coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)^{\mathrm{T}}$ by minimizing the penalized least squares (PLS) objective:

$$\|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta}\|^2 + \mathrm{e}^{\rho}\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{\beta}. \tag{1}$$

✉ Jiguo Cao
jiguo_cao@sfu.ca

Zheyuan Li
zheyuan.li@vip.henu.edu.cn

[1] School of Mathematics and Statistics, Henan University, Kaifeng, Henan, China

[2] Department of Statistics and Actuarial Science, Simon Fraser University, Burnaby, BC, Canada

The first term is the least squares, where $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^{\mathrm{T}}$ and $\boldsymbol{B}$ is a design matrix whose $(i, j)^{\text{th}}$ entry is $\mathcal{B}_j(x_i)$. The second term is the penalty, where $\boldsymbol{S}$ is a penalty matrix (such that $\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{\beta}$ is a wiggliness measure for f) and $\rho \in (-\infty, +\infty)$ is a smoothing parameter controlling the strength of the penalization. The choice of ρ is critical, as it trades off f's closeness to data for f's smoothness. The best ρ value often optimizes a smoothness selection criterion, like generalized cross-validation error (GCV) (Wahba 1990) and restricted likelihood (REML) (Wood 2017). Many strategies can be applied to this optimization task. However, to correctly identify the global optimum rather than being trapped in a local optimum, grid search is recommended. Specifically, we attempt an equidistant grid of ρ values in a search interval $[\rho_{\min}, \rho_{\max}]$ and pick the one that minimizes GCV or maximizes REML.

To illustrate that a criterion can have multiple local optima, and mistaking a local optimum for the global optimum may give undesirable result, consider smoothing daily new deaths attributed to COVID-19 in Finland from 2020-09-01 to 2022-03-01 and daily new confirmed cases of COVID-19 in Netherlands from 2021-09-01 to 2022-03-01 (data source: Our World in Data (Ritchie et al. 2020)). Figure 1 shows that the GCV (against ρ) in each example has a local minimum and a global minimum. The fitted spline corresponding to the local minimum is very wiggly, especially for Finland. Instead, the fit corresponding to the global minimum is smoother and more plausible; for example, it reasonably depicts the single peak during the first Omicron wave in Netherlands. In both cases, minimizing GCV via gradient descent or Newton's method can be trapped in the local minimum, if the initial guess for ρ is in its neighborhood. By contrast, doing a grid search in $[-6, 5]$ guarantees that the global minimum can be found.

In general, to successfully identify the global optimum of a criterion function in grid search, the search interval $[\rho_{\min}, \rho_{\max}]$ must be sufficiently wide so that the criterion can be fully explored. Unfortunately, no guideline is available for pre-specifying this interval, so practitioners have to find it by trial and error. This inevitably causes two problems. First of all, the PLS problem (1) is not numerically solvable when ρ is too large, because the limiting linear system for $\boldsymbol{\beta}$ is rank-deficient. A ρ value too small triggers the same problem if $p > n$, because the limiting unpenalized regression spline problem has no unique solution. As a result, practitioners do not even know what ρ range is numerically safe to attempt. Secondly, a PLS solver is often a computational kernel for more advanced smoothing problems, like robust smoothing with M-estimators (Dreassi et al. 2014; Osorio 2016) and backfitting-based generalized additive models (Eilers and Marx 2002; Oliveira and Paula 2021; Hernando Vanegas and Paula 2016). In these problems, a PLS needs be solved at each iteration for reweighted data, and it is more difficult to pre-

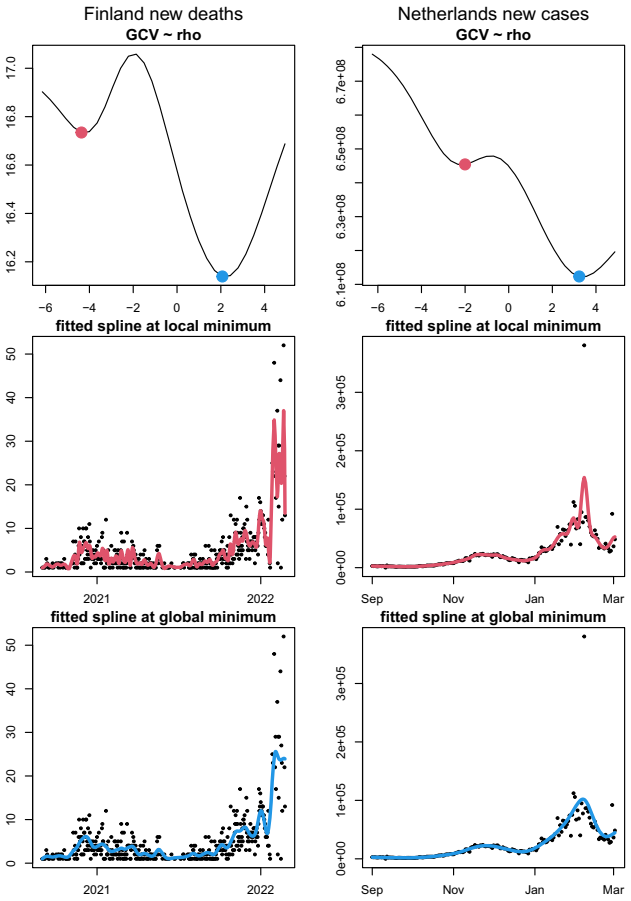

Fig. 1 Daily COVID-19 data smoothing. Left column: new deaths in Finland from 2020-09-01 to 2022-03-01; Right column: new cases in Netherlands from 2021-09-01 to 2022-03-01. The first row shows that GCV has a local minimum (red) and a global minimum (blue). The second row shows that the fitted spline corresponding to the local minimum is wiggly. The last row shows that the fit corresponding to the global minimum is smoother and more plausible. (Color figure online)

specify a search interval because it changes from iteration to iteration.

To address these issues, we develop algorithms that automatically find a suitable $[\rho_{\min}, \rho_{\max}]$ for the grid search. Our automatic search interval has four advantages. (i) It gives a safe ρ range where the PLS problem (1) is numerically solvable. (ii) It is criterion-independent so that different smoothness selection criteria, including GCV and REML, can be explored on the same ρ range. (iii) It is wide enough to contain the global optimum of any selection criterion, such as the global minimum of GCV and the global maximum of REML. (iv) It is computationally cheap compared with the grid search itself, carrying no extra cost in practice.

We will introduce our method using penalized B-splines, because it is particularly challenging to develop algorithms for this class of penalized splines to achieve (iv). We will describe how to adapt our method for other types of penalized splines at the end of this paper. Our method is ready to use

via our **R** package **gps** ($\geq$ version 1.1) (Li and Cao 2022b). It may be embedded in more advanced statistical modeling methods that rely on penalized splines.

The rest of this paper is organized as follows: Sect. 2 introduces penalized B-splines and their estimation for any trial ρ value. Section 3 derives, discusses and illustrates several search intervals for ρ. Section 4 conducts simulations for these intervals. Section 5 applies our search interval to practical smoothing by revisiting the COVID-19 example, with a comparison between GCV and REML in smoothness selection. The last section summarizes our method and addresses reviewers' concerns. Our **R** code is available on the internet at https://github.com/ZheyuanLi/gps-vignettes/blob/main/gps2.pdf.

2 Penalized B-splines

There exists a wide variety of penalized splines, depending on the setup of basis and penalty. If we choose the basis functions $\mathcal{B}_j(x)$, $j = 1, \ldots, p$ to be B-splines (de Boor 2001) and accompany them by either a difference penalty or a derivative penalty, we arrive at a particular class of penalized splines: the penalized B-splines family. This family has three members: O-splines (O'Sullivan 1986), standard P-splines (Eilers and Marx 1996) and general P-splines (Li and Cao 2022a); see the last reference for a good overview of this family. For this family, the matrix $\boldsymbol{S}$ in (1) can be induced by $\boldsymbol{S} = \boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{D}_m$. Therefore, the PLS objective can be expressed as:

$$\|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta}\|^2 + \mathrm{e}^{\rho}\|\boldsymbol{D}_m\boldsymbol{\beta}\|^2. \tag{2}$$

In this context, $\boldsymbol{B}$ is an $n \times p$ design matrix for B-splines of order $d \geq 2$, and $\boldsymbol{D}_m$ is a $(p-m) \times p$ penalty matrix of order $1 \leq m \leq d-1$. Basically, the three family members only differ in (i) the type of knots for constructing B-splines; (ii) the penalty matrix $\boldsymbol{D}_m$ applied to B-spline coefficients $\boldsymbol{\beta}$. See Appendix A for concrete examples.

When estimating penalized B-splines, we can pre-compute some ρ-independent quantities that are reused over ρ values on a search grid. These include $\|\boldsymbol{y}\|^2$, $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}$, $\boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{D}_m$, $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{y}$ and the lower triangular factor $\boldsymbol{L}$ in the Cholesky factorization $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$. In the following, we will consider them to be known quantities so that they do not count toward the overall computational cost of the estimation procedure.

Now, for any ρ value on a search grid, the PLS solution to (2) is $\hat{\boldsymbol{\beta}} = \boldsymbol{C}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{y}$, where $\boldsymbol{C} = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} + \mathrm{e}^{\rho}\boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{D}_m$. The vector of fitted values is $\hat{\boldsymbol{y}} = \boldsymbol{B}\hat{\boldsymbol{\beta}} = \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{y}$. By definition, the sum of squared residuals is $\mathrm{RSS}(\rho) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2$, but it can also be computed without using $\hat{\boldsymbol{y}}$:

$$\begin{aligned}\mathrm{RSS}(\rho) &= \|\boldsymbol{y} - \boldsymbol{B}\hat{\boldsymbol{\beta}}\|^2 = (\boldsymbol{y} - \boldsymbol{B}\hat{\boldsymbol{\beta}})^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{B}\hat{\boldsymbol{\beta}}) \\ &= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - 2\hat{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{y} + \hat{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}\hat{\boldsymbol{\beta}} \\ &= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - 2\hat{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{y} + \hat{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}\hat{\boldsymbol{\beta}} \\ &= \|\boldsymbol{y}\|^2 - 2\hat{\boldsymbol{\beta}}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{y} + \|\boldsymbol{L}^{\mathrm{T}}\hat{\boldsymbol{\beta}}\|^2.\end{aligned}$$

The complexity of the fitted spline is measured by its effective degree of freedom (edf), defined as the trace of the hat matrix mapping $\boldsymbol{y}$ to $\hat{\boldsymbol{y}}$:

$$\mathrm{edf}(\rho) = \mathrm{tr}\left(\boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{B}^{\mathrm{T}}\right) = \mathrm{tr}\left(\boldsymbol{C}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}\right).$$

The GCV error can then be calculated using RSS and edf:

$$\mathrm{GCV}(\rho) = \frac{n \cdot \mathrm{RSS}}{(n - \mathrm{edf})^2}.$$

The restricted likelihood $\mathrm{REML}(\rho)$ has a more complicated from and is detailed in Appendix B. All these quantities vary with ρ, explaining why the choice of ρ affects the resulting fit. It should also be pointed out that although edf depends on the basis $\boldsymbol{B}$ and the penalty $\boldsymbol{D}_m$, it depends on neither the response data $\boldsymbol{y}$ nor the choice of the smoothness selection criterion.

The actual computations of $\hat{\boldsymbol{\beta}}$ and edf do not require explicitly forming $\boldsymbol{C}^{-1}$. We can compute the Cholesky factorization $\boldsymbol{C} = \boldsymbol{K}\boldsymbol{K}^{\mathrm{T}}$ for the lower triangular factor $\boldsymbol{K}$, then solve triangular systems $\boldsymbol{K}\boldsymbol{x} = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{y}$ and $\boldsymbol{K}^{\mathrm{T}}\hat{\boldsymbol{\beta}} = \boldsymbol{x}$ for $\hat{\boldsymbol{\beta}}$. The edf can also be computed using the Cholesky factors:

$$\begin{aligned}\mathrm{edf}(\rho) &= \mathrm{tr}\left(\boldsymbol{C}^{-1}\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}\right) = \mathrm{tr}\left(\boldsymbol{L}^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{L}\right) \\ &= \mathrm{tr}\big((\boldsymbol{K}^{-1}\boldsymbol{L})^{\mathrm{T}}\boldsymbol{K}^{-1}\boldsymbol{L}\big) = \|\boldsymbol{K}^{-1}\boldsymbol{L}\|_F^2,\end{aligned} \tag{3}$$

where we solve the triangular system $\boldsymbol{K}\boldsymbol{X} = \boldsymbol{L}$ for $\boldsymbol{X} = \boldsymbol{K}^{-1}\boldsymbol{L}$, and calculate the squared Frobenius norm $\|\boldsymbol{X}\|_F^2$ (the sum of squared elements in $\boldsymbol{X}$).

Penalized B-splines are sparse and computationally efficient. Notably, $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}$, $\boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{D}_m$, $\boldsymbol{C}$, $\boldsymbol{L}$ and $\boldsymbol{K}$ are band matrices, so the aforementioned Cholesky factorizations and triangular systems can be computed and solved in $O(p^2)$ floating point operations. That is, both $\hat{\boldsymbol{\beta}}$ and edf come at $O(p^2)$ cost. The computation of RSS (without using $\hat{\boldsymbol{y}}$) has $O(p)$ cost. Once RSS and edf are known, GCV is known. In addition, Appendix B shows that the REML score can be obtained at $O(p)$ cost. Thus, computations for a given ρ value have $O(p^2)$ cost. When ρ is selected over N trial values on a search grid to optimize GCV or REML, the overall computational cost is $O(Np^2)$.

3 Automatic search intervals

To start with, we formulate edf in a different way that better reveals its mathematical property. Let $q = p - m$. We form

the $p \times q$ matrix (at $O(p^2)$ cost):

$$\boldsymbol{E} = \boldsymbol{L}^{-1}\boldsymbol{D}_m^{\mathrm{T}}, \tag{4}$$

and express $\boldsymbol{C}$ as:

$$\boldsymbol{C} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}} + \mathrm{e}^{\rho}\boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{D}_m = \boldsymbol{L}[\boldsymbol{I} + \mathrm{e}^{\rho}\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}}]\boldsymbol{L}^{\mathrm{T}}$$

Plugging this into $\mathsf{edf}(\rho) = \mathrm{tr}\left(\boldsymbol{L}^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{L}\right)$ (see (3)), we get:

$$\mathsf{edf}(\rho) = \mathrm{tr}\big([\boldsymbol{I} + \mathrm{e}^{\rho}\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}}]^{-1}\big),$$

where $\boldsymbol{I}$ is an identity matrix. The $p \times p$ symmetric matrix $\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}}$ is positive semi-definite with rank q. Thus, it has q positive eigenvalues $\lambda_1 > \lambda_2 > \ldots > \lambda_q$, followed by m zero eigenvalues. Let its eigendecomposition be $\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}$, where $\boldsymbol{U}$ is an orthonormal matrix with eigenvectors and $\boldsymbol{\Lambda}$ is a $p \times p$ diagonal matrix with eigenvalues. Then,

$$\begin{aligned}
\mathsf{edf}(\rho) &= \mathrm{tr}\big([\boldsymbol{U}\boldsymbol{U}^{-1} + \mathrm{e}^{\rho}\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}]^{-1}\big) \\
&= \mathrm{tr}\big([\boldsymbol{U}(\boldsymbol{I} + \mathrm{e}^{\rho}\boldsymbol{\Lambda})\boldsymbol{U}^{-1}]^{-1}\big) \\
&= \mathrm{tr}\big(\boldsymbol{U}(\boldsymbol{I} + \mathrm{e}^{\rho}\boldsymbol{\Lambda})^{-1}\boldsymbol{U}^{-1}\big) \\
&= \mathrm{tr}\big((\boldsymbol{I} + \mathrm{e}^{\rho}\boldsymbol{\Lambda})^{-1}\big) \\
&= m + \textstyle\sum_{j=1}^{q}(1 + \mathrm{e}^{\rho}\lambda_j)^{-1}.
\end{aligned}$$

We call $(\lambda_j)_1^q$ the Demmler–Reinsch eigenvalues, in tribute to Demmler and Reinsch (1975) who first studied the eigenvalue problem of smoothing splines (Reinsch 1967, 1971; Wang 2011). We also define the restricted edf as:

$$\mathsf{redf}(\rho) = \mathsf{edf}(\rho) - m = \sum_{j=1}^{q}\frac{1}{1 + \mathrm{e}^{\rho}\lambda_j}, \tag{5}$$

which is a monotonically decreasing function of ρ. As $\rho \to -\infty$, it increases to q; as $\rho \to +\infty$, it decreases to 0. Given this one-to-one correspondence between ρ and redf, it is clear that choosing the optimal ρ is equivalent to choosing the optimal redf.

3.1 An exact search interval

The relation between ρ and redf is enlightening. Although it is difficult to see what $[\rho_{\min}, \rho_{\max}]$ is wide enough for searching for ρ, it is easy to see what $[\mathsf{redf}_{\min}, \mathsf{redf}_{\max}]$ is adequate for searching for redf. For example,

$$\begin{aligned}
\mathsf{redf}_{\min} &= q\kappa, \\
\mathsf{redf}_{\max} &= q(1-\kappa),
\end{aligned} \tag{6}$$

with a small κ are reasonable. We interpret κ as a coverage parameter, as $[\mathsf{redf}_{\min}, \mathsf{redf}_{\max}]$ covers $100(1-2\kappa)\%$ of $[0, q]$. In practice, $\kappa = 0.01$ is good enough, in which case $[\mathsf{redf}_{\min}, \mathsf{redf}_{\max}]$ covers 98% of $[0, q]$. It then follows that $\rho_{\min}$ and $\rho_{\max}$ satisfy

$$\begin{aligned}
\mathsf{redf}(\rho_{\min}) &= \mathsf{redf}_{\max}, \\
\mathsf{redf}(\rho_{\max}) &= \mathsf{redf}_{\min},
\end{aligned} \tag{7}$$

and can be solved for via root-finding. As $\mathsf{redf}(\rho)$ is differentiable, we can use Newton's method. See Algorithm 1 for an implementation of this method for a general root-finding problem $g(x) = 0$.

Algorithm 1 Newton's method for finding the root of $g(x) = 0$. Inputs: (i) x, an initial value, (ii) $\delta_{\max}$, maximum size of a Newton step.

```
g = g(x)
loop
    g' = g'(x)
    δ = −g'/g
    if |δ| < |g|10^−6 then
        break
    end if
    δ = sign(δ) · min(|δ|, δ_max)
    loop
        x̃ = x + δ_k
        g̃ = g(x̃)
        if |g̃| < |g| then
            break
        end if
        δ = δ/2
    end loop
    x = x̃
    g = g̃
end loop
return x
```

It should be pointed out that Algorithm 1 is not fully automatic. To make it work, we need an initial ρ value, as well as a suitable $\delta_{\max}$ for bounding the size of a Newton step (which helps prevent overshooting). This is not easy, though, since we don't know a sensible range for ρ (in fact, we are trying to find such a range). We will come back to this issue in the next section. For now, let's discuss the strength and weakness of this approach.

The interval $[\rho_{\min}, \rho_{\max}]$ is back-transformed from redf. As is stressed in Sect. 2, edf (and thus redf) does not depend on y values or the choice of the smoothness selection criterion; neither does the interval. This is a nice property. It is also an advantage of our method, for there is no need to find a new interval when y values or the criterion change. We call this idea of back-transformation the redf-oriented thinking.

Nevertheless, the interval is computationally expensive. To get $(\lambda_j)_1^q$, an eigendecomposition at $O(p^3)$ cost is needed. Section 2 shows that solving PLS along with grid search only has $O(Np^2)$ cost. Thus, when p is big, finding the interval

is even more costly than the subsequent grid search. This is unacceptable and we need a better strategy.

3.2 A wider search interval

In fact, we don't have to find the exact $[\rho_{\min}, \rho_{\max}]$ that satisfies (6) and (7). It suffices to find a wider interval:

$$[\rho^*_{\min}, \rho^*_{\max}] \supseteq [\rho_{\min}, \rho_{\max}].$$

That is, find $\rho^*_{\min}$ and $\rho^*_{\max}$ such that $\rho_{\min} \geq \rho^*_{\min}$ and $\rho_{\max} \leq \rho^*_{\max}$. Surprisingly, in this way, we only need the maximum and the minimum eigenvalues (λ_1 and λ_q) instead of all the eigenvalues.

We now derive this wider interval. From $\lambda_1 \geq \lambda_j \geq \lambda_q$, we have:

$$\frac{1}{1+\mathrm{e}^{\rho}\lambda_1} \leq \frac{1}{1+\mathrm{e}^{\rho}\lambda_j} \leq \frac{1}{1+\mathrm{e}^{\rho}\lambda_q}.$$

Then, applying $\sum_{j=1}^{q}$ to these terms, we get:

$$\frac{q}{1+\mathrm{e}^{\rho}\lambda_1} \leq \mathsf{redf}(\rho) \leq \frac{q}{1+\mathrm{e}^{\rho}\lambda_q}.$$

Since the result holds for any ρ, including $\rho_{\min}$ and $\rho_{\max}$, there are:

$$\begin{aligned}\mathsf{redf}(\rho_{\min}) &\geq \frac{q}{1+\mathrm{e}^{\rho_{\min}}\lambda_1},\\ \mathsf{redf}(\rho_{\max}) &\leq \frac{q}{1+\mathrm{e}^{\rho_{\max}}\lambda_q}.\end{aligned}$$

Together with (6) and (7), we see:

$$\begin{aligned}q(1-\kappa) &\geq \frac{q}{1+\mathrm{e}^{\rho_{\min}}\lambda_1},\\ q\kappa &\leq \frac{q}{1+\mathrm{e}^{\rho_{\max}}\lambda_q}.\end{aligned}$$

These imply:

$$\begin{aligned}\rho_{\min} &\geq \log\left(\frac{\kappa}{(1-\kappa)\lambda_1}\right) = \rho^*_{\min},\\ \rho_{\max} &\leq \log\left(\frac{1-\kappa}{\kappa\lambda_q}\right) = \rho^*_{\max}.\end{aligned}$$

Better yet, we can replace the maximum eigenvalue λ_1 by the mean eigenvalue $\bar{\lambda} = \sum_{j=1}^{q}\lambda_j/q$ for a tighter lower bound. In general, the harmonic mean of positive numbers $(a_j)_1^q$ is no larger than their arithmetic mean:

$$\frac{q}{\sum_{j=1}^{q}\frac{1}{a_j}} \leq \bar{a} \quad \Rightarrow \quad \frac{q}{\bar{a}} \leq \sum_{j=1}^{q}\frac{1}{a_j}.$$

If we set $a_j = 1+\mathrm{e}^{\rho}\lambda_j$, we get:

$$\frac{q}{1+\mathrm{e}^{\rho}\bar{\lambda}} \leq \mathsf{redf}(\rho).$$

This allows us to update $\rho^*_{\min}$. In the end, we have:

$$\begin{aligned}\rho^*_{\min} &= \log\left(\frac{\kappa}{(1-\kappa)\bar{\lambda}}\right),\\ \rho^*_{\max} &= \log\left(\frac{1-\kappa}{\kappa\lambda_q}\right).\end{aligned} \tag{8}$$

At first glance, it appears that we still need all the eigenvalues in order to compute their mean $\bar{\lambda}$. But the trick is that $(\lambda_j)_1^q$ add up to:

$$\sum_{j=1}^{q}\lambda_j = \mathrm{tr}\,(\boldsymbol{\Lambda}) = \mathrm{tr}\left(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}\right) = \mathrm{tr}\left(\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}}\right) = \|\boldsymbol{E}\|_F^2,$$

so we can easily compute $\bar{\lambda}$ (at $O(p^2)$ cost):

$$\bar{\lambda} = \frac{1}{q}\sum_{j=1}^{q}\lambda_j = \frac{\|\boldsymbol{E}\|_F^2}{q}. \tag{9}$$

This leaves the minimum eigenvalue λ_q the only nontrivial quantity to compute. As we will see in the next section, the computation of λ_q merely involves $O(p^2)$ complexity. Therefore, the wider search interval $[\rho^*_{\min}, \rho^*_{\max}]$ is available at $O(p^2)$ cost, much cheaper than the exact search interval $[\rho_{\min}, \rho_{\max}]$ that comes at $O(p^3)$ cost.

Another advantage of the wider interval is its closed-form formula. The exact interval is defined implicitly by root-finding, and its computation is not automatic as Algorithm 1 requires ρ-relevant inputs. However, the wider interval can be directly computed using eigenvalues that are irrelevant to ρ, so its computation is fully automatic.

In practice, we can utilize the wider interval to automate the computation of the exact interval. For example, when applying Algorithm 1 to find $\rho_{\min}$ or $\rho_{\max}$, we may use $(\rho^*_{\min}+\rho^*_{\max})/2$ for its initial value, and bound the size of a Newton step by $\delta_{\max} = (\rho^*_{\max}-\rho^*_{\min})/4$. This improvement does not by any means make the computation of the exact interval less expensive, though.

3.3 Computational details

We now describe $O(p^2)$ algorithms for computing the maximum and the minimum eigenvalues, i.e., λ_1 and λ_q, to aid the fast computation of the wider interval $[\rho^*_{\min}, \rho^*_{\max}]$. Note that although λ_1 does not show up in (8), it is useful for assessing the credibility of the computed λ_q, which will soon be explained.

Recall that $(\lambda_j)_1^q$ are the positive eigenvalues of the $p \times p$ positive semi-definite matrix $\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}}$, and the matrix also has m zero eigenvalues. To get rid of these nuisance eigenvalues, we can work with the $q \times q$ positive-definite matrix $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$ instead. Here, the trick is that $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$ and $\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}}$ have the same positive eigenvalues that equal the squared nonzero singular values of $\boldsymbol{E}$. This can be proved using the singular value decomposition of $\boldsymbol{E}$.

In general, given a positive definite matrix $\boldsymbol{A}$, we compute its maximum eigenvalue using power iteration and its minimum eigenvalue using inverse iteration. The two algorithms iteratively compute $\boldsymbol{A}\boldsymbol{v}$ and $\boldsymbol{A}^{-1}\boldsymbol{v}$, respectively. Unfortunately, when $\boldsymbol{A} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$, the latter operation is as expensive as $O(p^3)$ because $\boldsymbol{A}$ is fully dense. Therefore, naively applying inverse iteration to compute λ_q is not any cheaper than a full eigendecomposition.

To obtain λ_q at $O(p^2)$ cost, we are to exploit the following partitioning:

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{E_1} \\ \boldsymbol{E_2} \end{bmatrix},$$

where $\boldsymbol{E_1}$ is a $q \times q$ lower triangular matrix and $\boldsymbol{E_2}$ is an $m \times q$ rectangular matrix. Below is an illustration of such structure (with $p = 6, m = 2$ and $q = 4$), where '$\times$' and '$\circ$' denote the nonzero elements in $\boldsymbol{E_1}$ and $\boldsymbol{E_2}$, respectively.

$$\left[\begin{array}{cccc} \times & & & \\ \times & \times & & \\ \times & \times & \times & \\ \times & \times & \times & \times \\ \hline \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \end{array}\right]$$

Such "trapezoidal" structure exclusively holds for the penalized B-splines family, and it allows us to express $\boldsymbol{A}$ as an update to $\boldsymbol{E_1^{\mathrm{T}}}\boldsymbol{E_1}$:

$$\boldsymbol{A} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E} = \boldsymbol{E_1^{\mathrm{T}}}\boldsymbol{E_1} + \boldsymbol{E_2^{\mathrm{T}}}\boldsymbol{E_2}.$$

Using Woodbury identity (Woodbury 1950), we obtain an explicit inversion formula:

$$\begin{aligned} \boldsymbol{A}^{-1} &= (\boldsymbol{E_1^{\mathrm{T}}}\boldsymbol{E_1})^{-1} - \boldsymbol{F}(\boldsymbol{I} + \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R})^{-1}\boldsymbol{F}^{\mathrm{T}} \\ &= (\boldsymbol{E_1^{\mathrm{T}}}\boldsymbol{E_1})^{-1} - \boldsymbol{F}(\boldsymbol{G}\boldsymbol{G}^{\mathrm{T}})^{-1}\boldsymbol{F}^{\mathrm{T}}, \end{aligned}$$

where $\boldsymbol{R} = (\boldsymbol{E_1^{\mathrm{T}}})^{-1}\boldsymbol{E_2^{\mathrm{T}}}$ and $\boldsymbol{F} = \boldsymbol{E_1^{-1}}\boldsymbol{R}$ are both $q \times m$ matrices, and $\boldsymbol{G}$ is the lower triangular Cholesky factor of the $m \times m$ matrix $\boldsymbol{I} + \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}$. We thus convert the expensive computation of $\boldsymbol{A}^{-1}\boldsymbol{v}$ to solving triangular linear systems involving $\boldsymbol{E_1}$, $\boldsymbol{E_1^{\mathrm{T}}}$, $\boldsymbol{G}$ and $\boldsymbol{G}^{\mathrm{T}}$, which is much more efficient. See Algorithm 2 for implementation details as well as a breakdown of computational costs. The penalty order m is often very small (usually 1, 2 or 3) and the number of iterations till convergence is much smaller than q, so the overall cost remains $O(q^2)$ in practice. Since $q = p - m \approx p$, we report this complexity as $O(p^2)$.

Algorithm 2 Compute the minimum eigenvalue λ_q of $\boldsymbol{A} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$. Lines 1 to 5 compute matrices for applying Woodbury identity. Line 6 starts inverse iteration. Lines 10 to 16 compute $\boldsymbol{u} = \boldsymbol{A}^{-1}\boldsymbol{v}$.

1: $\boldsymbol{E} = \begin{bmatrix} \boldsymbol{E_1} \\ \boldsymbol{E_2} \end{bmatrix}$
2: solve $\boldsymbol{E_1^{\mathrm{T}}}\boldsymbol{R} = \boldsymbol{E_2^{\mathrm{T}}}$ for $\boldsymbol{R}$ ▷ $O(mq^2)$
3: solve $\boldsymbol{E_1}\boldsymbol{F} = \boldsymbol{R}$ for $\boldsymbol{F}$ ▷ $O(mq^2)$
4: $\boldsymbol{H} = \boldsymbol{I} + \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}$ ▷ $O(m^3)$
5: Cholesky factorization $\boldsymbol{H} = \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}}$ ▷ $O(m^3)$
6: initialize $\boldsymbol{u}$ as a random vector ▷ $O(q)$
7: $\lambda = 0$
8: **loop**
9: $\quad \boldsymbol{v} = \boldsymbol{u}/\|\boldsymbol{u}\|$ ▷ $O(q)$
10: $\quad$ solve $\boldsymbol{E_1^{\mathrm{T}}}\boldsymbol{a_1} = \boldsymbol{v}$ for $\boldsymbol{a_1}$ ▷ $O(q^2)$
11: $\quad$ solve $\boldsymbol{E_1}\boldsymbol{a_2} = \boldsymbol{a_1}$ for $\boldsymbol{a_2}$ ▷ $O(q^2)$
12: $\quad \boldsymbol{c_1} = \boldsymbol{F}^{\mathrm{T}}\boldsymbol{v}$ ▷ $O(qm)$
13: $\quad$ solve $\boldsymbol{G}\boldsymbol{b_1} = \boldsymbol{c_1}$ for $\boldsymbol{b_1}$ ▷ $O(m^2)$
14: $\quad$ solve $\boldsymbol{G}^{\mathrm{T}}\boldsymbol{b_2} = \boldsymbol{b_1}$ for $\boldsymbol{b_2}$ ▷ $O(m^2)$
15: $\quad \boldsymbol{c_2} = \boldsymbol{F}\boldsymbol{b_2}$ ▷ $O(qm)$
16: $\quad \boldsymbol{u} = \boldsymbol{a_2} - \boldsymbol{c_2}$ ▷ $O(q)$
17: $\quad \tilde{\lambda} = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{u}$ ▷ $O(q)$
18: $\quad$ **if** $\tilde{\lambda} < 0$ **then**
19: $\quad\quad$ Warning: $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$ is numerically singular!
20: $\quad\quad \lambda_q = 0$
21: $\quad\quad$ break
22: $\quad$ **end if**
23: $\quad$ **if** $|\tilde{\lambda} - \lambda| < \lambda 10^{-6}$ **then**
24: $\quad\quad$ break
25: $\quad$ **end if**
26: $\quad \lambda = \tilde{\lambda}$
27: **end loop**
28: $\lambda_q = 1/\lambda$
29: **if** $\lambda_q < \lambda_1 \varepsilon$ **then**
30: $\quad$ Warning: $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$ is numerically singular!
31: $\quad \lambda_q = \lambda_1 \varepsilon$
32: **end if**
33: **return** λ_q

An important technical detail in Algorithm 2 is that it checks the credibility of the computed λ_q. Although $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$ is positive definite, in finite precision arithmetic performed by our computers, it becomes numerically singular if λ_q/λ_1 is smaller than the machine precision ε (the largest positive number such that $1 + \varepsilon = 1$). On modern 64-bit CPUs, this precision is about 1.11×10^{-16}. In case of numerical singularity, λ_q can not be accurately computed for the loss of significant digits, and the output λ_q is fake. The best bet in this case, is to reset the computed λ_q to $\lambda_1 \varepsilon$ (see lines 29 to 31). Sometimes, the computed $\tilde{\lambda}$ at line 17 is negative. This is an early sign of singularity, and we can immediately stop the iteration (see lines 18 to 22). These procedures are necessary safety measures, without which the computed λ_q will be too

small and the derived $\rho^*_{\max}$ will be too big, making the PLS problem (2) unsolvable. In short, Algorithm 2 helps determine a search interval $[\rho^*_{\min}, \rho^*_{\max}]$ that is both sufficiently wide and numerically safe for grid search.

We need the maximum eigenvalue λ_1 before applying Algorithm 2. The computation of λ_1 is less challenging: even a direct application of power iteration by iteratively computing $\boldsymbol{E}^{\mathrm{T}}(\boldsymbol{E}\boldsymbol{v})$ is good enough at $O(p^2)$ cost. But we can make it more efficient by exploiting the band sparsity behind the "factor form" (4) of matrix $\boldsymbol{E}$. See Algorithm 3 for details. The overall complexity is $O(p)$ in practice.

Algorithm 3 Compute the maximum eigenvalue λ_1 of $\boldsymbol{A} = \boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$, exploiting the band sparsity of $\boldsymbol{L}$ and $\boldsymbol{D}_m^{\mathrm{T}}$ in the "factor form" (4) of matrix $\boldsymbol{E}$. Lines 5 to 8 compute $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{v}$.

1: initialize $\boldsymbol{u}$ as a random vector ▷ $O(q)$
2: $\lambda = 0$
3: **loop**
4: $\quad \boldsymbol{v} = \boldsymbol{u}/\|\boldsymbol{u}\|$ ▷ $O(q)$
5: $\quad \boldsymbol{b} = \boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{v}$ ▷ $O(p)$
6: $\quad$ solve $\boldsymbol{L}\boldsymbol{a}_1 = \boldsymbol{b}$ for $\boldsymbol{a}_1$ ▷ $O(p)$
7: $\quad$ solve $\boldsymbol{L}^{\mathrm{T}}\boldsymbol{a}_2 = \boldsymbol{a}_1$ for $\boldsymbol{a}_2$ ▷ $O(p)$
8: $\quad \boldsymbol{u} = \boldsymbol{D}_m\boldsymbol{a}_2$ ▷ $O(p)$
9: $\quad \tilde{\lambda} = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{u}$ ▷ $O(q)$
10: $\quad$ **if** $|\tilde{\lambda} - \lambda| < \lambda 10^{-6}$ **then**
11: $\quad\quad$ break
12: $\quad$ **end if**
13: $\quad \lambda = \tilde{\lambda}$
14: **end loop**
15: **return** λ

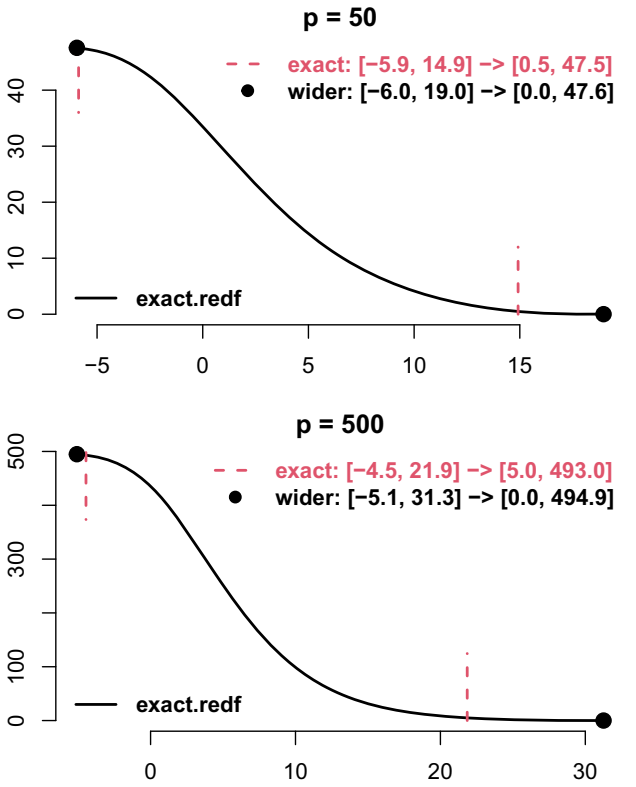

Fig. 2 An example of $\mathsf{redf}(\rho)$ curve for p cubic B-splines ($d = 4$) on unevenly spaced knots, penalized by a 2nd order ($m = 2$) difference penalty. The black dots ($\rho^*_{\min}$ and $\rho^*_{\max}$) lie beyond the red dashed lines ($\rho_{\min}$ and $\rho_{\max}$) on both ends, implying that $[\rho^*_{\min}, \rho^*_{\max}]$ is wider than $[\rho_{\min}, \rho_{\max}]$. The text (with numbers rounded to 1 decimal place) state the actual mapping from ρ range to redf range. (Color figure online)

3.4 An illustration of the intervals

We now illustrate search intervals $[\rho_{\min}, \rho_{\max}]$ and $[\rho^*_{\min}, \rho^*_{\max}]$ through a simple example. We set up p cubic B-splines ($d = 4$) on unevenly spaced knots and penalized them by a 2nd order ($m = 2$) difference penalty matrix $\boldsymbol{D}_2$. We then generated 10 uniformly distributed x values between every two adjacent knots and constructed the design matrix $\boldsymbol{B}$ at those locations. Figure 2 illustrates the resulting $\mathsf{redf}(\rho)$ for $p = 50$ and $p = 500$. The nominal mapping from ρ range to redf range is $(-\infty, +\infty) \to [0, q]$ and here we have $q = p - 2$. For the exact interval, the mapping is $[\rho_{\min}, \rho_{\max}] \to [0.01q, 0.99q]$. The wider interval $[\rho^*_{\min}, \rho^*_{\max}]$ is mapped to a wider range than $[0.01q, 0.99q]$.

Our search intervals do not depend on y values or the choice of the smoothness selection criterion. To illustrate this, we simulated two sets of y values for the $p = 50$ case, and chose the optimal ρ by minimizing GCV or maximizing REML. Figure 3 shows that while the two datasets yield different GCV or REML curves, they share the same search interval for ρ. Incidentally, both GCV and REML choose similar optimal ρ values in each example, leading to indistinguishable fit. This is not always true, though, and we will discuss more about this in Sect. 5.

3.5 Heuristic improvementk

Let's take a second look at Fig. 2. As the black circle on the right end is far off the red dashed line, $\rho^*_{\max}$ is a loose upper bound. Moreover, the bigger p is, the looser it is. We now use some heuristics to find a tighter upper bound. Specifically, we seek approximated eigenvalues $(\hat{\lambda}_j)_1^q$ such that $\hat{\lambda}_1 = \lambda_1$, $\hat{\lambda}_q = \lambda_q$ and $\sum_{j=1}^q \hat{\lambda}_j = q\bar{\lambda}$. Then, replacing λ_j by $\hat{\lambda}_j$ in (5) gives an approximated redf, with which we can solve (6) and (7) for an "exact" interval $[\hat{\rho}_{\min}, \hat{\rho}_{\max}]$. We are most interested in $\hat{\rho}_{\max}$. If it is tighter than $\rho^*_{\max}$, even if just empirically, we can narrow the search interval $[\rho^*_{\min}, \rho^*_{\max}]$ to $[\rho^*_{\min}, \hat{\rho}_{\max}]$. We hereafter call $\hat{\rho}_{\max}$ the heuristic upper bound.

For reasonable approximation, knowledge on λ_j's decay pattern is useful. Recently, Xiao (2019,Lemma 5.1) established an asymptotic decay rate at about $O((1 - \frac{j}{q+m})^{2m}) \sim O((1 - \frac{j+m}{q+m})^{2m})$ (Xiao arranged $(\lambda_j)_1^q$ in ascending order

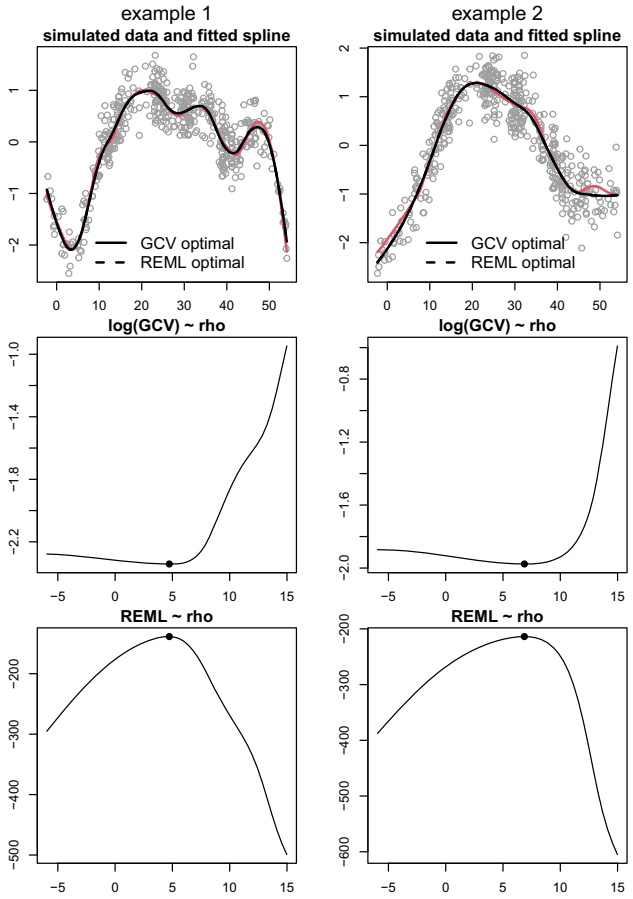

Fig. 3 Two sets of y values are simulated and smoothed for the $p = 50$ case in Fig. 2. They yield different GCV or REML curves, but share the same search interval for ρ. In the first row, red curves are the true functions and black curves are the optimal spline fit. (Color figure online)

and his original result is $O((\frac{j}{q+m})^{2m}) \sim O((\frac{j+m}{q+m})^{2m}))$. So, $\log(\lambda_j)$ roughly decays like $z_j = \log(1 - t_j)$, where $t_j = j/(q + 1)$. In practice, however, we observed that the first few eigenvalues often drop faster, while the decay of the remaining eigenvalues well follows the theory. Empirically, the actual decay resembles an "S"-shaped curve:

$$z_j = z(t_j, \gamma) = \log(1 - t_j) + \gamma \log(1/t_j),$$

where $\gamma \in [0, 1]$ is a shape parameter. For example, Fig. 4 illustrates the decay of $\log(\lambda_j)$ against t_j for the examples in the previous section. The fast decay of the first few eigenvalues and the resulting "S"-shaped curve are most noticeable for $p = 500$. The figure also plots $z(t_j, \gamma)$ for various γ values. When $\gamma = 0$, it is the asymptotic decay; when $\gamma = 1$, it is a symmetric "S"-shaped curve similar to the quantile function of the logistic distribution. Moreover, it appears that we can tweak γ value to make z_j similar to $\log(\lambda_j)$.

As it is generally not possible to tweak γ value to make z_j identical to $\log(\lambda_j)$, we chose to model $\log(\lambda_j)$ as a

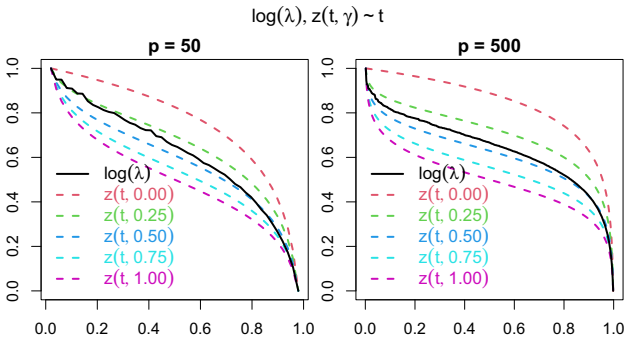

Fig. 4 The asymptotic decay established by Xiao (2019) (red dashed) often does not reflect the actual decay (black solid) of $\log(\lambda_j)$, because the first few eigenvalues may drop faster. Empirically, the actual decay of $\log(\lambda_j)$ resembles an "S"-shaped curve $z(t_j, \gamma)$, and we can tweak the value of the shape parameter γ to make this curve similar (but not identical) to $\log(\lambda_j)$. (Note: the ranges of $\log(\lambda_j)$ and z_j have been transformed to [0, 1] to aid shape comparison.) (Color figure online)

function of z_j, i.e., $\log(\hat{\lambda}_j) = Q(z_j)$. For convenience, let $a = \log(\lambda_q)$ and $b = \log(\lambda_1)$. We also transform the range of z_j to [0, 1] by $z_j \leftarrow (z_j - z_q)/(z_1 - z_q)$. As we require $\hat{\lambda}_q = \lambda_q$ and $\hat{\lambda}_1 = \lambda_1$, the function must satisfy $Q(0) = a$ and $Q(1) = b$. To be able to determine $Q(z_j)$ with the last constraint $\sum_{j=1}^{q} \hat{\lambda}_j = q\bar{\lambda}$, we can only parameterize the function with one unknown. We now denote this function by $Q(z_j, \alpha)$ and suggest two parametrizations that prove to work well in practice. The first one is a quadratic polynomial:

$$Q_1(z_j, \alpha) = a + (b - a)z_j + \alpha(z_j^2 - z_j).$$

It is convex and monotonically increasing when $\alpha \in [0, b - a]$. The second one is a cubic polynomial:

$$\begin{aligned} Q_2(z_j, \alpha) = &[c_0(z_j) + c_2(z_j)]a + [c_2(z_j) + c_3(z_j)]b \\ &+ [c_1(z_j) - c_2(z_j)]\alpha, \end{aligned}$$

represented using cubic Bernstein polynomials:

$$\begin{aligned} c_0(z_j) &= (1 - z_j)^3, & c_1(z_j) &= 3z_j(1 - z_j)^2, \\ c_2(z_j) &= 3z_j^2(1 - z_j), & c_3(z_j) &= z_j^3. \end{aligned}$$

It is an "S"-shaped curve and if $\alpha \in [a, (2a + b)/3]$, it is monotonically increasing, concave on [0, 0.5] and convex on [0.5, 1]. See Fig. 5 for what the two functions look like as α varies. To facilitate computation, we rewrite both cases as:

$$\log(\hat{\lambda}_j) = Q(z_j, \alpha) = \theta_j + h_j\alpha, \quad \alpha \in [\alpha_l, \alpha_r].$$

Finding α such that $\sum_{j=1}^{q} \hat{\lambda}_j = q\bar{\lambda}$ is equivalent to finding the root of $g(\alpha) = \sum_{j=1}^{q} \exp(\theta_j + h_j\alpha) - q\bar{\lambda}$. As $g(\alpha)$ is differentiable, we use Newton's method (see Algorithm 1) for this task. Obviously, a root exists in $[\alpha_l, \alpha_r]$ if and

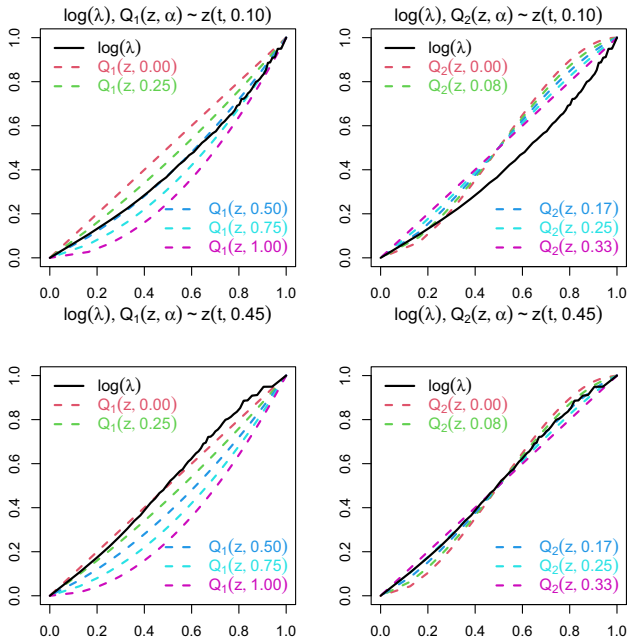

Fig. 5 We can approximate $\log(\lambda)$ by $\log(\hat{\lambda}_j) = Q(z_j, \alpha)$, if γ value is properly chosen for $z_j = z(t_j, \gamma)$ such that $\log(\lambda_j)$ (sketched against z_j) largely lies on the "paths" of $Q(z_j, \alpha)$ as α varies. In this example (the $p = 50$ case in Fig. 4), $\gamma = 0.1$ is good for $Q_1(z_j, \alpha)$, but not for $Q_2(z_j, \alpha)$; whereas $\gamma = 0.45$ is good for $Q_2(z_j, \alpha)$, but not for $Q_1(z_j, \alpha)$. (Note: the ranges of $\log(\lambda_j)$ and z_j have been transformed to [0, 1] to aid shape comparison.) (Color figure online)

only if $g(\alpha_l) \cdot g(\alpha_r) <= 0$. Therefore, we are not able to obtain approximated eigenvalues $(\hat{\lambda}_j)_1^q$ if this condition does not hold. Intuitively, the condition is met if $\log(\lambda_j)$, when sketched against z_j, largely lies on the "paths" of $Q(z_j, \alpha)$ as α varies (see Fig. 5). This in turn implies that γ value should be properly chosen for $z_j = z(t_j, \gamma)$. For example, Fig. 5 shows that $\gamma = 0.1$ is good for $Q_1(z_j, \alpha)$, but not for $Q_2(z_j, \alpha)$; whereas $\gamma = 0.45$ is good for $Q_2(z_j, \alpha)$, but not for $Q_1(z_j, \alpha)$.

In reality, we don't know what γ value is good in advance, so we do a grid search in [0, 1], and for each trial γ value, we attempt both $Q_1(z_j, \alpha)$ and $Q_2(z_j, \alpha)$ (see Algorithm 4). We might have several successful approximations, so we take their average for the final $(\hat{\lambda}_j)_1^q$. Figure 6 shows $\log(\lambda_j)$ and $\log(\hat{\lambda}_j)$ (both transformed to range between 0 and 1) for the examples demonstrated through Figs. 2, 3, 4 and 5. The approximation is perfect for small p (which is not surprising because a smaller p means fewer eigenvalues to guess). As p grows, the approximation starts to deviate from the truth, but still remains accurate on both ends (which is important, since the quality of the heuristic upper bound $\hat{\rho}_{\max}$ mainly relies on the approximation to extreme eigenvalues). Figure 7 displays (on top of Fig. 2) the approximated redf and the resulting $\hat{\rho}_{\max}$. Clearly, $\hat{\rho}_{\max}$ is closer to $\rho_{\max}$ than $\rho^*_{\max}$ is. Thus, it is a tighter bound. (More simulations of $\rho_{\max}$, $\rho^*_{\max}$ and $\hat{\rho}_{\max}$ are conducted in Sect. 4.)

Algorithm 4 Approximate eigenvalues of $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E}$. Inputs: (i) q, number of eigenvalues, (ii) λ_1, λ_q and $\bar{\lambda}$, max/min/mean eigenvalues. Quantities with subscript j are calculated for $j = 1, 2, \ldots, q$. For the inputs of Algorithm 1, use $(\alpha_l + \alpha_r)/2$ for the initial value and bound the size of a Newton step by $\delta_{\max} = (\alpha_r - \alpha_l)/4$.

$a = \log(\lambda_q), b = \log(\lambda_1)$
$t_j = j/(q+1)$
$\hat{\lambda}_j = 0$ ▷ initialize approximation
$N = 0$ ▷ number of successes
for $\gamma = 0,\ 0.05,\ 0.10,\ \ldots,\ 1$ **do**
  $z'_j = \log(1 - t_j) - \gamma \log(t_j)$ ▷ try $z(t_j, \gamma)$
  $z_j = (z'_j - z'_q)/(z'_1 - z'_q)$
  $\theta_j = a + (b - a)z_j$ ▷ try $Q_1(z_j, \alpha)$
  $h_j = z_j^2 - z_j$
  $\alpha_l = 0, \alpha_r = b - a$
  if $g(\alpha_l)g(\alpha_r) <= 0$ **then**
    find $g(\alpha)$'s root α using Algorithm 1
    $N = N + 1$
    $\hat{\lambda}_j = \hat{\lambda}_j + \exp(\theta_j + \alpha h_j)$
  end if
  $c_{0j} = (1 - z_j)^3$ ▷ try $Q_2(z_j, \alpha)$
  $c_{1j} = 3z_j(1 - z_j)^2$
  $c_{2j} = 3z_j^2(1 - z_j)$
  $c_{3j} = z_j^3$
  $\theta_j = a(c_{0j} + c_{2j}) + b(c_{2j} + c_{3j})$
  $h_j = c_{1j} - c_{2j}$
  $\alpha_l = a, \alpha_r = (2a + b)/3$
  if $g(\alpha_l)g(\alpha_r) <= 0$ **then**
    find $g(\alpha)$'s root α using Algorithm 1
    $N = N + 1$
    $\hat{\lambda}_j = \hat{\lambda}_j + \exp(\theta_j + \alpha h_j)$
  end if
end for
if $N > 0$ **then** ▷ average over successes
  $\hat{\lambda}_j = \hat{\lambda}_j / N$
else
  Warning: unable to approximate λ_j
end if
return λ_j

In summary, to get the heuristic upper bound $\hat{\rho}_{\max}$, we first apply Algorithm 4 to compute $(\hat{\lambda}_j)_1^q$ for approximating redf (5), then apply Algorithm 1 to solve the approximated redf for $\hat{\rho}_{\max}$. Both steps are computationally efficient at $O(p)$ cost, as they do not involve matrix computations. Thus, the overall computational cost of the heuristically improved interval $[\rho^*_{\min}, \hat{\rho}_{\max}]$ remains $O(p^2)$. The computation is also fully automatic. In the first step, all variable inputs required by Algorithm 4, namely q, λ_1, λ_q and $\bar{\lambda}$, can be obtained during the computation of the wider interval $[\rho^*_{\min}, \rho^*_{\max}]$. In the second step, we can automate Algorithm 1 by using $(\rho^*_{\min} + \rho^*_{\max})/2$ for the initial ρ value, and bounding the size of a Newton step by $\delta_{\max} = (\rho^*_{\max} - \rho^*_{\min})/4$.

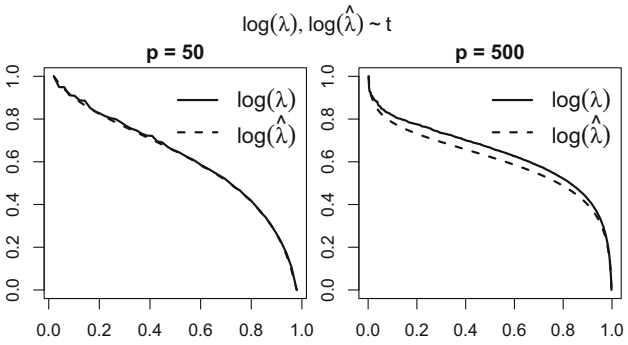

Fig. 6 Log eigenvalues $\log(\lambda_j)$ (solid) and their heuristic approximation $\log(\hat{\lambda}_j)$ (dashed) against t_j

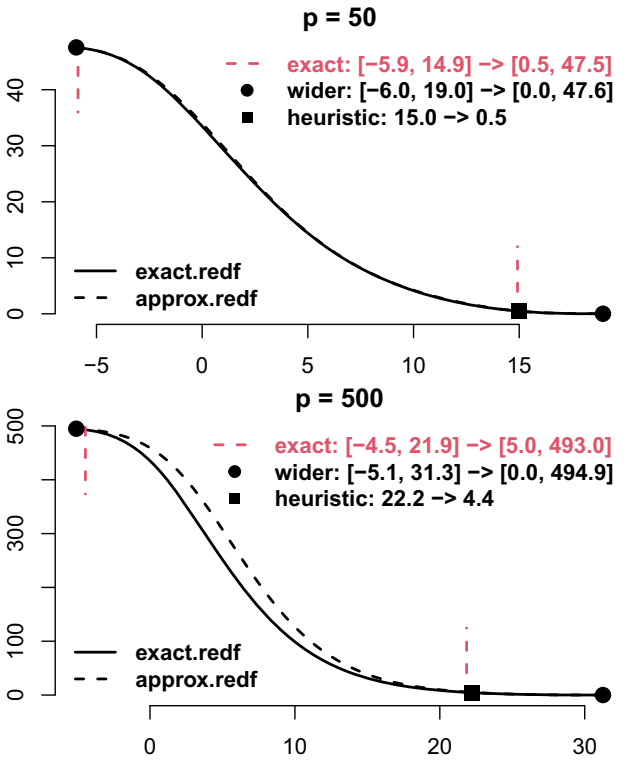

Fig. 7 Approximated $\mathsf{redf}(\rho)$ (black dashed) and heuristic upper bound $\hat{\rho}_{\max}$ (black square). This figure enhances Fig. 2. As $\hat{\rho}_{\max}$ is closer to $\rho_{\max}$ (red dashed) than $\rho^*_{\max}$ (black circle) is, it is a tighter upper bound. (Color figure online)

In rare cases, Algorithm 4 fails to approximate eigenvalues. On this occasion, $\hat{\rho}_{\max}$ is not available and we have to stick to $\rho^*_{\max}$. In principle, the chance of failure can be reduced by enhancing the heuristic. For example, we may introduce a shape parameter $\nu \geq 1$ and model the "S"-shaped decay as $z_j = z(t_j, \gamma, \nu) = \log(1 - t_j) + \gamma[\log(1/t_j)]^\nu$, which allows the first few eigenvalues to decrease even faster. We may also propose $Q(z_j, \alpha)$ of new parametric forms. In short, Algorithm 4 can be easily extended for improvement.

Table 1 Runtime (in seconds (s) or milliseconds (1 ms = 0.001 s) of different computational tasks for growing p, on an Intel i5-8250U CPU @ 1.60 GHz

p	500 (ms)	1000 (s)	1500 (s)	2000 (s)
$\rho^*_{\min}, \rho^*_{\max}, \hat{\rho}_{\max}$	4.74	0.02	0.07	0.15
$\rho_{\min}, \rho_{\max}$	156.50	1.85	7.48	17.37
PLS ($N = 20$)	58.30	0.25	0.58	1.00

3.6 Summary

We propose two search intervals for ρ: the exact one $[\rho_{\min}, \rho_{\max}]$ and the wider one $[\rho^*_{\min}, \rho^*_{\max}]$. We prefer the wider one to the exact one for three reasons. Firstly, the wider one has a closed-form formula and its computation is fully automatic, whereas the exact one is implicitly defined by root-finding and its computation is not automatic by itself. Secondly, the wider one is computationally efficient. Its $O(p^2)$ cost is no greater than the $O(Np^2)$ cost for solving PLS over N trial ρ values, whereas the $O(p^3)$ cost for computing the exact one is too expensive. Thirdly, the wider one, or rather, its upper bound $\rho^*_{\max}$, can be tightened using simple heuristics, and the heuristic upper bound $\hat{\rho}_{\max}$ can be computed at only $O(p)$ cost. To reflect the practical impact of computational costs, we report in Table 1 the actual runtime of different computational tasks for growing p.

4 Simulations

The exact interval $[\rho_{\min}, \rho_{\max}]$ and the wider one $[\rho^*_{\min}, \rho^*_{\max}]$ have known theoretical property. By construction, we have (for $\kappa = 0.01$):

$$\rho^*_{\min} \leq \rho_{\min}, \qquad \mathsf{redf}(\rho^*_{\min}) \geq \mathsf{redf}(\rho_{\min}) = 0.99q$$
$$\rho^*_{\max} \geq \rho_{\max}, \qquad \mathsf{redf}(\rho^*_{\max}) \leq \mathsf{redf}(\rho_{\max}) = 0.01q.$$

However, the heuristic bound $\hat{\rho}_{\max}$ has unknown property. We expect it to be tighter than $\rho^*_{\max}$, in the sense that it is closer to $\rho_{\max}$. Ideally, it should satisfy $\rho_{\max} \leq \hat{\rho}_{\max} \leq \rho^*_{\max}$. This is supported by Fig. 7, but we still need extensive simulations to be confident about this in general.

For comprehensiveness, let's experiment every possible setup for penalized B-splines that affects $\mathsf{redf}(\rho)$. The placement of equidistant or unevenly spaced knots produces different $\boldsymbol{B}$. The choice of difference or derivative penalty gives different $\boldsymbol{D}_m$. We also consider weighted data where (x_i, y_i) has weight w_i. This leads to a penalized weighted least squares problem (PWLS) that can be transformed to a PLS problem by absorbing weights into $\boldsymbol{B}$: $\boldsymbol{B} \leftarrow \boldsymbol{W}^{1/2}\boldsymbol{B}$, where $\boldsymbol{W}$ is a diagonal matrix with element $W_{ii} = w_i$. Altogether, we have 8 scenarios (see Table 2).

Table 2 The 8 scenarios for simulations

	Derivative penalty	Equidistant knots	Weighted data
1	FALSE	FALSE	FALSE
2	TRUE	FALSE	FALSE
3	FALSE	TRUE	FALSE
4	TRUE	TRUE	FALSE
5	FALSE	FALSE	TRUE
6	TRUE	FALSE	TRUE
7	FALSE	TRUE	TRUE
8	TRUE	TRUE	TRUE

Here are more low-level details for setting up $\boldsymbol{B}$, $\boldsymbol{D}_m$ and $\boldsymbol{W}$. In general, to construct order-d B-splines $\mathcal{B}_j(x)$, $j = 1, \ldots, p$, we need to place $p + d$ knots $(\xi_k)_1^{p+d}$. For our simulations, we take $\xi_k = k$ for equidistant knots and $\xi_k \sim \mathrm{N}(k, [(p+d)/10]^2)$ for unevenly spaced knots. We then generate 10 uniformly distributed x values between every two nearby knots for constructing the design matrix $\boldsymbol{B}$ and the penalty matrix $\boldsymbol{D}_m$. For the weights, we use random samples from Beta(3, 3) distribution.

To finish our setup, we still need to specify the B-spline order d, the penalty order m and basis dimension p. By design, each combination of these parameters has 8 scenarios. For our simulations, we are to test different (d, m) pairs for growing p. We call each combination of d, m, p and scenario ID an experiment.

In principle, we want to run an experiment, say 200 times, and see how $\hat{\rho}_{\max}$ is distributed relative to $\rho_{\max}$ and $\rho^*_{\max}$. But as none of these quantities stays fixed between runs, visualization is not easy. Back to our redf-oriented thinking, let's compare the proportion of $[0, q]$ covered by $[\mathsf{redf}(\rho_{\max}), q]$, $[\mathsf{redf}(\hat{\rho}_{\max}), q]$ and $[\mathsf{redf}(\rho^*_{\max}), q]$:

$$\mathrm{P}(\rho_{\max}) = 1 - \mathsf{redf}(\rho_{\max})/q = 0.99,$$
$$\mathrm{P}(\hat{\rho}_{\max}) = 1 - \mathsf{redf}(\hat{\rho}_{\max})/q,$$
$$\mathrm{P}(\rho^*_{\max}) = 1 - \mathsf{redf}(\rho^*_{\max})/q.$$

Since $\mathsf{redf}(\rho)$ is decreasing, we shall observe $0.99 \le \mathrm{P}(\hat{\rho}_{\max}) \le \mathrm{P}(\rho^*_{\max})$ if we expect $\rho_{\max} \le \hat{\rho}_{\max} \le \rho^*_{\max}$. Interestingly, our simulations show that the variance of $\mathrm{P}(\rho^*_{\max})$ is so low that its probability density function almost degenerates to a vertical line through its mean. This implies that we need to check if the density curve of $\mathrm{P}(\hat{\rho}_{\max})$ lies between two vertical lines. Figure 8 shows our simulation results for cubic splines with a 2nd order penalty and quadratic splines with a 1st order penalty as p grows. They look satisfying except for scenarios 1 and 5. In these cases, a non-negligible proportion of the density curve breaches 0.99, so we will get $\hat{\rho}_{\max} < \rho_{\max}$ occasionally. Therefore, the resulting interval $[\rho^*_{\min}, \hat{\rho}_{\max}]$ is narrower than $[\rho_{\min}, \rho_{\max}]$. However, it is

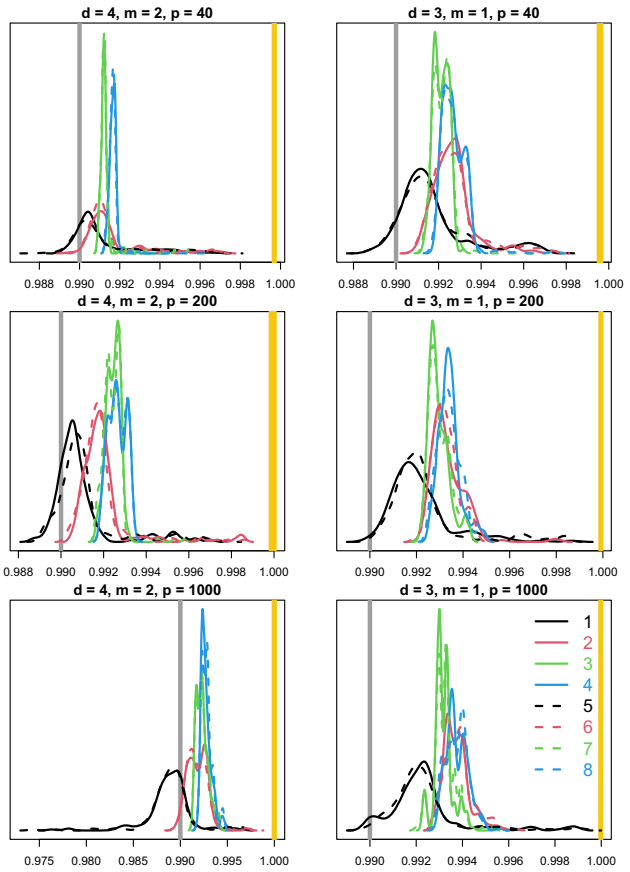

Fig. 8 Estimated probability density function of $\mathrm{P}(\hat{\rho}_{\max})$, i.e., the proportion of $[0, q]$ covered by $[\mathsf{redf}(\hat{\rho}_{\max}), q]$. If $\rho_{\max} \le \hat{\rho}_{\max} \le \rho^*_{\max}$, the density curve should lie between 0.99 (gray line) and $\mathrm{P}(\rho^*_{\max})$ (yellow line). Numbers 1 to 8 refer to the scenarios in Table 2. (Color figure online)

wide enough for grid search, as the corresponding redf range still covers a significant proportion of $[0, q]$ (check out the numbers labeled along the x-axis of each graph in the figure). So, empirically, $\hat{\rho}_{\max}$ is a fair tight upper bound.

One anonymous reviewer asked if lowering the value of κ would improve the performance of $\hat{\rho}_{\max}$ for scenarios 1 and 5. The answer is no, because κ has no effect on the heuristic approximation to eigenvalues (see Algorithm 4). To really improve the performance of $\hat{\rho}_{\max}$, we need to enhance our heuristics rather than alter the value of κ. Figure S1 in the supplementary material shows further simulation results with $\kappa = 0.005$ and 0.001. The results are very similar to Fig. 8, confirming that the value of κ is not critical.

5 Application

As an application of our automatic search interval to practical smoothing, we revisit the COVID-19 data example in the Introduction. To stress that our interval is criterion-

independent, we illustrate smoothness selection using both GCV and REML.

The Finland dataset reports new deaths on 408 out of 542 days from 2020-09-01 to 2022-03-01. The Netherlands dataset reports new cases on 181 out of 182 days from 2021-09-01 to 2022-03-01. Let n be the number of data. For smoothing we set up cubic B-splines on $n/4$ knots placed at equal quantiles of the reporting days, and penalize them by a 2nd order difference penalty matrix. For the Finland example, our computed search interval is $[\rho^*_{\min}, \hat{\rho}_{\max}] = [-6.15, 14.93]$ (with $\rho^*_{\max} = 20.25$). For the Netherlands example, the search interval is $[\rho^*_{\min}, \hat{\rho}_{\max}] = [-6.26, 13.05]$ (with $\rho^*_{\max} = 16.97$). Figure 9 sketches $\mathsf{edf}(\rho)$, $\mathrm{GCV}(\rho)$ and $\mathrm{REML}(\rho)$ on the search intervals. For both examples, GCV has a local minimum and a global minimum (see Fig. 1 for a zoomed-in display), whereas REML has a single maximum. In addition, the optimal ρ value chosen by REML is bigger than that selected by GCV, yielding a smoother yet more plausible fit. In fact, theoretical properties of both criteria have been well studied by Reiss and Ogden (2009). In short, GCV is more likely to have multiple local optima. It is also more likely to underestimate the optimal ρ and cause overfitting. Thus, REML is superior to GCV for smoothness selection. But anyway, the focus here is not to discuss the choice of the selection criterion, but to demonstrate that our automatic search interval for ρ is wide enough for exploring any criterion.

The edf curves in Fig. 9 show that more than half of the B-spline coefficients are suppressed by the penalty in the optimal fit. In the Finland case, the maximum possible edf in Fig. 9 is 106, but the optimal edf (either 43.1 or 12.6) is less than half of that. To avoid such waste, we can halve the number of knots, i.e., place $n/8$ knots. This alters the design matrix, the penalty matrix and hence $\mathsf{edf}(\rho)$, so the search interval needs be recomputed. The new interval is $[\rho^*_{\min}, \hat{\rho}_{\max}] = [-6.16, 13.30]$ (with $\rho^*_{\max} = 17.47$) for the Finland example and $[\rho^*_{\min}, \hat{\rho}_{\max}] = [-6.43, 11.51]$ (with $\rho^*_{\max} = 14.66$) for the Netherlands example. Figure 10 sketches $\mathsf{edf}(\rho)$, $\mathrm{GCV}(\rho)$ and $\mathrm{REML}(\rho)$ on their new range. Interestingly, $\mathrm{GCV}(\rho)$ no longer has a second local minimum, yet it still underestimates the optimal ρ value when compared with REML. The fitted splines are almost identical to their counterparts in Fig. 9, and thus not shown in the figure.

6 Discussion

We have developed algorithms to automatically produce a search interval for the grid search of the smoothing parameter ρ in penalized splines (1). Our search interval has four properties. (i) It gives a safe ρ range where the PLS problem is numerically solvable. (ii) It does not depend on the

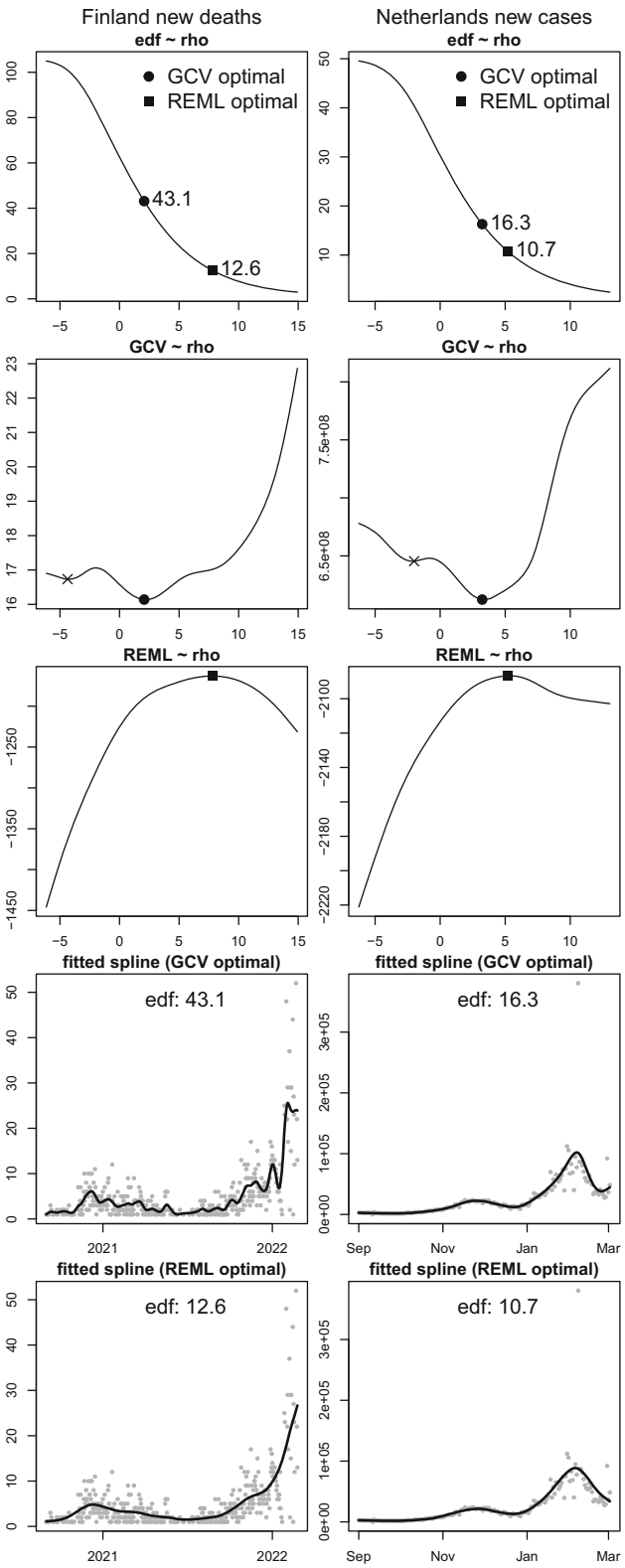

Fig. 9 Smoothing daily COVID-19 data in Finland ($n = 408$ data; displayed in column 1) and Netherlands ($n = 182$ data; displayed in column 2). A cubic general P-splines with $n/4$ knots and a 2nd order difference penalty is used for smoothing. For both examples, the optimal ρ value chosen by REML is bigger than that selected by GCV, resulting in a smoother yet more plausible fit

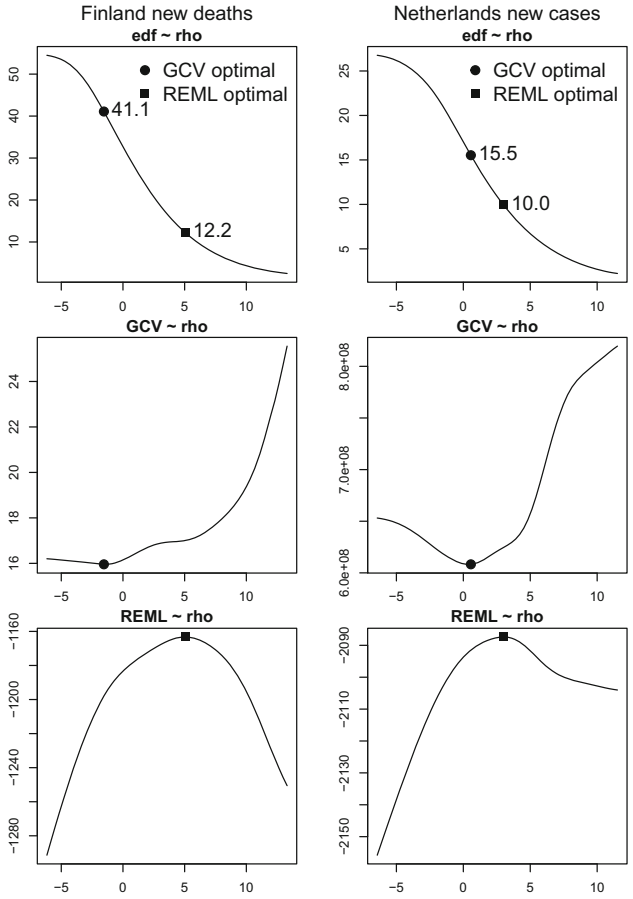

Fig. 10 The altered edf(ρ), GCV(ρ) and REML(ρ) after we halve the number of knots in the COVID-19 smoothing example. Interestingly, GCV no longer has a second local minimum, yet it still underestimates the optimal ρ value when compared with REML

choice of the smoothness selection criterion. (iii) It is wide enough to contain the global optimum of any criterion. (iv) It is computationally cheap compared with the grid search itself.

The Demmler–Reinsch eigenvalues $(\lambda_j)_1^q$ play a pivotal role in our methodology development. They reveal a one-to-one correspondence between ρ (varying in $(-\infty, +\infty)$) and redf (5) (varying in $[0, q]$), which motivates us to back-transform a target redf range $[q\kappa,\ q(1-\kappa)]$ for a suitable ρ range $[\rho_{\min}, \rho_{\max}]$, where κ is a coverage parameter such that the target redf range covers $100(1-2\kappa)\%$ of $[0, q]$. As such, the search interval satisfies (i)–(iii) naturally. To achieve (iv), the computational strategy for the interval needs to meet different goals, depending on whether the PLS problem (1) is dense or sparse.

- A dense PLS problem has $O(p^3)$ computational cost. This applies to penalized splines with a dense design matrix $\boldsymbol{B}$ and/or a dense penalty matrix $\boldsymbol{S}$. Examples are truncated power basis splines (Ruppert et al. 2003), natural cubic splines (Wood 2017, Sect. 5.3.1) and thin-plate splines (Wood 2017, Sect. 5.5.1);
- A sparse PLS problem has $O(p^2)$ computational cost. This applies to penalized B-splines (for which the PLS objective is (2)) with a sparse design matrix $\boldsymbol{B}$ and a sparse penalty matrix $\boldsymbol{D}_m$. Examples are O-splines (O'Sullivan 1986), standard P-splines (Eilers and Marx 1996) and general P-splines (Li and Cao 2022a).

Therefore, the computational costs of our search intervals need be no greater than $O(p^3)$ and $O(p^2)$, respectively.

We have made great efforts to optimize our computational strategy for the sparse penalized B-splines. The key is to give up computing all the eigenvalues (as an eigendecomposition has $O(p^3)$ cost) and compute only the mean, the maximum and the minimum eigenvalues. This allows us to obtain a wider interval $[\rho^*_{\min}, \rho^*_{\max}] \supseteq [\rho_{\min}, \rho_{\max}]$ at $O(p^2)$ cost. We can further tighten this interval using some heuristics. Precisely, step by step, we compute:

1. $\bar{\lambda}$, using (9);
2. λ_1 and λ_q, using Algorithms 3 and 2;
3. $[\rho^*_{\min}, \rho^*_{\max}]$, using (8);
4. $(\hat{\lambda}_j)_1^q$, using Algorithm 4;
5. $\hat{\rho}_{\max}$, by applying Algorithm 1 to (5) (with λ_j replaced by $\hat{\lambda}_j$), (6) and (7). (To automatically start the algorithm, use $(\rho^*_{\min} + \rho^*_{\max})/2$ for the initial value and set $\delta_{\max} = (\rho^*_{\max} - \rho^*_{\min})/4$.)

Steps 1-2 each have $O(p^2)$ cost; step 3 is simple arithmetic; steps 4-5 each have $O(p)$ cost. They are implemented by function `gps2GS` in our **R** package **gps** ($\geq$ version 1.1). The function solves the PLS problem (2) for a grid of ρ values in $[\rho^*_{\min}, \hat{\rho}_{\max}]$. It can be embedded in more advanced statistical modeling methods that rely on penalized splines, such as robust smoothing and generalized additive models.

One anonymous reviewer commented that it appears possible to bypass the use of eigenvalues. It was noticed that the initial edf formula (3) is free of eigenvalues. So, it was suggested that we could work with $\text{redf} = \text{edf} - m = \|\boldsymbol{K}^{-1}\boldsymbol{L}\|_F^2 - m$ instead of (5), when back-transforming a target redf range for $[\rho_{\min}, \rho_{\max}]$ via root-finding. Since the computational complexity of (3) is $O(p^2)$, and a root-finding algorithm only takes a few iterations to converge, this method should have $O(p^2)$ cost, too. While this is true, it is not practicable unless we automate the root-finding. If we use Newton's method (see Algorithm 1) for this task, we need an initial value and a maximum stepsize for ρ; if we use bisection or Brent's method, we need a search interval for ρ. Either way seems to be a deadlock, as we have to supply something relevant to what we hope to find. In fact, whether we express redf using eigenvalues or not, we will face this

difficulty as long as the search interval is implicitly defined by root-finding. This difficulty is not eliminated, until we have a wider interval $[\rho^*_{\min}, \rho^*_{\max}]$ in closed form. This interval can be directly computed by (8) using eigenvalues. It can further be exploited to automate the root-finding (for example, see step 5 of our method). In summary, the wider interval and hence the Demmler–Reinsch eigenvalues are key to the automaticity of our method. There is no way to bypass the use of those eigenvalues. The best we can do, is to only compute the maximum, the minimum and the mean eigenvalues required for computing the wider interval.

Since we have to use eigenvalues anyway, we express $\mathsf{redf}(\rho)$ as (5) (the eigen-form) instead of $\|\boldsymbol{K}^{-1}\boldsymbol{L}\|_F^2 - m$ (the PLS-form) when presenting our method. In addition, we prefer the eigen-form to the PLS-form for its simplicity. Given $(\lambda_j)_1^q$, it is easy to compute redf and its derivative if using the eigen-form, and Newton's method has $O(p)$ cost. By contrast, working with the PLS-form requires matrix computations and matrix calculus so that Newton's method has $O(p^2)$ cost. The difficulty with the eigen-form is the $O(p^3)$ computational cost behind $(\lambda_j)_1^q$. However, once we replace them by their heuristic approximation $(\hat{\lambda}_j)_1^q$ that can be obtained at $O(p)$ cost, the simplicity of the eigen-form becomes real computational efficiency.

It may still be asked why we would rather do some heuristic approximation to tighten our wider interval (as steps 4-5 of our method show), than compute the exact interval by applying Algorithm 1 to the PLS-form, (6) and (7) (where we can use the wider interval to automate the root-finding). After all, the $O(p^2)$ cost behind the PLS-form, while greater the $O(p)$ cost behind the eigen-form, is no greater than the $O(p^2)$ cost for computing the wider interval. Thus, the overall computational cost of our method would still be $O(p^2)$. While this is true, in our view, it is awkward to compute the exact interval; or at least, it is not worth it. The PLS-form is coupled with PLS solving. If we apply this idea, we would have to do PLS solving on three different sets of ρ grids: one for computing $\rho_{\min}$, one for $\rho_{\max}$ and one for the grid search on $[\rho_{\min}, \rho_{\max}]$. The first two rounds of PLS solving are a waste, given that we are only interested in the final grid search for smoothness selection. We want to separate the computation of our search interval from PLS solving and grid search, and thus deprecate this idea.

Another comment from the same reviewer, is that we have restricted the search for the optimal ρ in our bounded search interval, so that $\rho = -\infty$ or $+\infty$ can not be chosen. This is a good point. In terms of our redf-oriented thinking, it means that $\mathsf{redf} = 0$ and $\mathsf{redf} = q$ have been excluded. This is deliberately done. By construction, the minimum and the maximum redf we can reach are $q\kappa$ and $q(1-\kappa)$, respectively. Apparently, if we want to approximately cover these two endpoints, we can choose a very small κ value. But this is not a good strategy (see the next paragraph for a discussion on the choice of κ). Instead, we compute the GCV error and the REML score for these limiting cases separately, then include them in our grid search. In either case, the PLS problem (2) degenerates to an ordinary least squares (OLS) problem. Thus, we can work with these OLS problems instead for these quantities. See Appendix C for details.

In this paper, the coverage parameter κ is fixed at 0.01. Although reducing this value would widen the target redf range, we don't recommend it. In reality, $\mathsf{redf}(\rho)$ quickly plateaus as it gets close to either endpoint (for example, see Fig. 2). Setting κ too small will result in long flat "tails" on both sides. Moreover, ρ values on each "tail" give similar redf values, fitted splines, GCV errors and REML scores. This is undesirable for grid search. When a fixed number of equidistant grid points are positioned, the smaller κ is, the more trial ρ values fall on those "tails", and thus, the fewer trial ρ values are available for exploring the "ramp" where $\mathsf{redf}(\rho)$ varies fast. The κ is to our method as the convergence tolerance is to an iterative algorithm. It needs to be small, but not unnecessarily small.

Our method has a limitation: it does not apply to a penalized spline whose design matrix $\boldsymbol{B}$ does not have full column rank. This is because we need matrix $\boldsymbol{L}$, the Cholesky factor of $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}$, to derive the eigen-form of redf. While it is common to have a full rank design matrix in practical smoothing, a rank-deficient design matrix is not problematic at all. A typical example is when $p > n$, i.e., there are more basis functions than (unique) x values. As Eilers and Marx (2021) put it, "It is impossible to have too many B-splines" (p. 15). Therefore, how to automatically produce a search interval for ρ in this case remains an interesting yet challenging question.

For dense penalized splines, it is acceptable to compute a full eigendecomposition for all the eigenvalues, because solving the PLS problem has $O(p^3)$ cost anyway. It is not the decomposition of $\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}}$, though, as the PLS objective reverts to (1) and neither $\boldsymbol{D}_m$ nor $\boldsymbol{E}$ is defined. So, we need to eliminate $\boldsymbol{D}_m$ and $\boldsymbol{E}$, and express the matrix using $\boldsymbol{S}$. First, (4) implies $\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}} = \boldsymbol{L}^{-1}\boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{D}_m(\boldsymbol{L}^{-1})^{\mathrm{T}}$. Then, replacing $\boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{D}_m$ by $\boldsymbol{S}$ (which is the link between (1) and (2)) gives $\boldsymbol{E}\boldsymbol{E}^{\mathrm{T}} = \boldsymbol{L}^{-1}\boldsymbol{S}(\boldsymbol{L}^{-1})^{\mathrm{T}}$. Therefore, $(\lambda_j)_1^q$ are the positive eigenvalues of $\boldsymbol{L}^{-1}\boldsymbol{S}(\boldsymbol{L}^{-1})^{\mathrm{T}}$. Our goal is to solve redf (5) for the exact interval, but to automate the root-finding, we need the wider interval first. Precisely, step by step, we compute:

1. $(\lambda_j)_1^q$, by computing a full eigendecomposition;
2. $[\rho^*_{\min}, \rho^*_{\max}]$, using (8);
3. $[\rho_{\min}, \rho_{\max}]$, by applying Algorithm 1 to (5), (6) and (7). (To automatically start the algorithm, use $(\rho^*_{\min}+\rho^*_{\max})/2$ for the initial value and set $\delta_{\max} = (\rho^*_{\max} - \rho^*_{\min})/4$.)

We didn't seek to optimize our method for dense penalize splines, so there is likely plenty of room for improvement.

Table 3 Penalized B-splines

	$\boldsymbol{D}_m^{\text{dif}}$	$\boldsymbol{D}_m^{\text{der}}$
UBS	SPS, GPS	OS
NUBS	GPS	OS

ESM1[[Supplementary information.: The supplementary file contains simulation results when $\kappa = 0.005$ and 0.001.]]

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11222-022-10178-z.

Acknowledgements We thank the Editor, the Associate Editor and two anonymous reviewers for their careful review and their valuable comments. These comments are very helpful for us to improve our work.

Funding Zheyuan Li was supported by the National Natural Science Foundation of China under the Young Scientists Fund (No. 12001166). Jiguo Cao was supported by the Natural Sciences and Engineering Research Council of Canada under the Discovery Grant (RGPIN-2018-06008).

Code availability **R** code is on the internet at https://github.com/ZheyuanLi/gps-vignettes/blob/main/gps2.pdf.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Appendix A Penalty matrix

The penalized B-splines family includes O-splines (OS), standard P-splines (SPS) and general P-splines (GPS). They differ in (i) the type of knots for constructing B-splines; (ii) the penalty matrix $\boldsymbol{D}_m$ applied to B-spline coefficients. See Table 3 for an overview.

Knots can be evenly or unevenly spaced. B-splines on equidistant knots are uniform B-splines (UBS); they have identical shapes. B-splines on unevenly spaced knots are non-uniform B-splines (NUBS); they have different shapes. See Fig. 11 for an illustration.

The matrix $\boldsymbol{D}_m$ may come from a difference penalty or a derivative penalty. The exact values of its elements also depend on the knot spacing. For details about its derivation and computation, see Li and Cao (2022a). Here, we simply write out some example $\boldsymbol{D}_m$ matrices (calculated using function `SparseD` from our **R** package **gps**). For the cubic UBS in Fig. 11, the 2nd order penalty matrices are:

$$\boldsymbol{D}_2^{\text{dif}} = \begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \quad 1 \end{bmatrix},$$

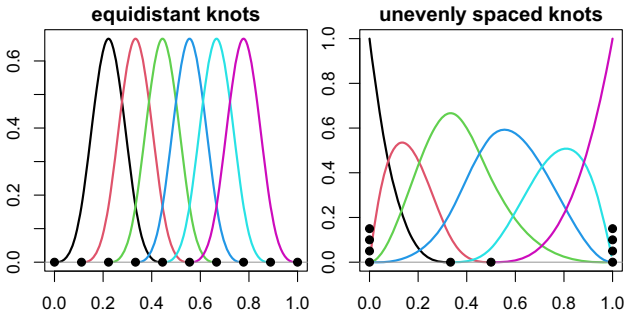

Fig. 11 6 Cubic B-splines $\mathcal{B}_1(x), \mathcal{B}_2(x), \ldots \mathcal{B}_6(x)$ on 10 knots in [0, 1]. Left: UBS on equidistant knots 0, 1/9, 2/9, ..., 1; Right: NUBS on unevenly spaced knots 0, 0, 0, 0, 1/3, 1/2, 1, 1, 1, 1. (Color figure online)

$$\boldsymbol{D}_2^{\text{der}} = \begin{bmatrix} 0.19 & -0.29 & 0.00 & 0.10 & \\ & 0.25 & -0.44 & 0.11 & 0.07 \\ & & 0.26 & -0.45 & 0.12 \quad 0.07 \\ & & & 0.18 & -0.36 \quad 0.18 \end{bmatrix}.$$

For the cubic NUBS in Fig. 11, the 2nd order penalty matrices are:

$$\boldsymbol{D}_2^{\text{dif}} = \begin{bmatrix} 54 & -90 & 36 & & \\ & 24 & -36 & 12.0 & \\ & & 9 & -22.5 & 13.5 \\ & & & 18.0 & -42.0 \quad 24 \end{bmatrix},$$

$$\boldsymbol{D}_2^{\text{der}} = \begin{bmatrix} 18 & -26.00 & 6.00 & 2.00 & \\ & 8.94 & -12.75 & 2.80 & 1.01 \\ & & 4.19 & -7.25 & -1.24 \quad 4.30 \\ & & & 6.60 & -15.41 \quad 8.81 \end{bmatrix}.$$

Numbers are rounded to 2 decimal places in these matrices.

Appendix B REML score

The estimation of penalized splines falls within the empirical Bayes framework. Hence, the restricted maximum likelihood (REML) can be used to select the smoothing parameter ρ in penalized splines. This section gives details about the derivation and the computation of REML. For convenience, let's denote the PLS objective (2) by PLS($\boldsymbol{\beta}$).

In the Bayesian view, the least squares term in PLS($\boldsymbol{\beta}$) corresponds to Gaussian likelihood:

$$\Pr(\boldsymbol{y}|\boldsymbol{\beta}) = c_1 \cdot \exp\left\{-\frac{\|\boldsymbol{y}-\boldsymbol{B}\boldsymbol{\beta}\|^2}{2\sigma^2}\right\},$$

where $c_1 = (2\pi\sigma^2)^{-n/2}$. The wiggliness penalty corresponds to a Gaussian prior:

$$\Pr(\boldsymbol{\beta}) = c_2 \cdot \exp\left\{-\frac{e^{\rho}\|\boldsymbol{D}_m\boldsymbol{\beta}\|^2}{2\sigma^2}\right\},$$

where $c_2 = (2\pi\sigma^2)^{-(p-m)/2} \cdot |\mathrm{e}^{\rho} \boldsymbol{D}_m \boldsymbol{D}_m^{\mathrm{T}}|^{1/2}$ and $|\boldsymbol{X}|$ is the determinant of $\boldsymbol{X}$. The unnormalized posterior is then:

$$\pi(\boldsymbol{\beta}|\boldsymbol{y}) = \Pr(\boldsymbol{y}|\boldsymbol{\beta}) \cdot \Pr(\boldsymbol{\beta}) = c_1 \cdot c_2 \cdot \exp\left\{-\frac{\mathrm{PLS}(\boldsymbol{\beta})}{2\sigma^2}\right\}.$$

Clearly, the PLS solution that minimizes $\mathrm{PLS}(\boldsymbol{\beta})$ is also the posterior mode that maximizes $\pi(\boldsymbol{\beta}|\boldsymbol{y})$. Section 2 has shown that the PLS solution is $\hat{\boldsymbol{\beta}} = \boldsymbol{C}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{y}$, where $\boldsymbol{C} = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{B} + \mathrm{e}^{\rho}\boldsymbol{D}_m^{\mathrm{T}}\boldsymbol{D}_m$. In fact, the Taylor expansion of $\mathrm{PLS}(\boldsymbol{\beta})$ at $\hat{\boldsymbol{\beta}}$ is exactly:

$$\mathrm{PLS}(\boldsymbol{\beta}) = \mathrm{PLS}(\hat{\boldsymbol{\beta}}) + (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^{\mathrm{T}}\boldsymbol{C}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}).$$

Plugging this into $\pi(\boldsymbol{\beta}|\boldsymbol{y})$ gives:

$$\pi(\boldsymbol{\beta}|\boldsymbol{y}) = c_1 \cdot c_2 \cdot c_3 \cdot \exp\left\{-\frac{(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})^{\mathrm{T}}\boldsymbol{C}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})}{2\sigma^2}\right\},$$

where

$$c_3 = \exp\left\{-\frac{\mathrm{PLS}(\hat{\boldsymbol{\beta}})}{2\sigma^2}\right\} = \exp\left\{-\frac{\mathrm{RSS}+\mathrm{e}^{\rho}\|\boldsymbol{D}_m\hat{\boldsymbol{\beta}}\|^2}{2\hat{\sigma}^2}\right\}.$$

The restricted likelihood of ρ and σ^2 is defined by integrating out $\boldsymbol{\beta}$ in $\pi(\boldsymbol{\beta}|\boldsymbol{y})$:

$$L_r(\rho, \sigma^2) = \int \pi(\boldsymbol{\beta}|\boldsymbol{y})\, \mathrm{d}\boldsymbol{\beta} = c_1 \cdot c_2 \cdot c_3 \cdot c_4,$$

where the key integral is:

$$\begin{aligned} c_4 &= \int \exp\left\{-\frac{(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})^{\mathrm{T}}\boldsymbol{C}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})}{2\sigma^2}\right\} \mathrm{d}\boldsymbol{\beta} \\ &= (2\pi\sigma^2)^{p/2} \cdot |\boldsymbol{C}|^{-1/2}. \end{aligned}$$

Thus, the restricted log-likelihood, or the REML criterion function, is:

$$\begin{aligned} l_r(\rho, \sigma^2) &= \log[L_r(\rho, \sigma^2)] = \log(c_1 \cdot c_2 \cdot c_3 \cdot c_4) \\ &= \tfrac{1}{2}\log|\mathrm{e}^{\rho}\boldsymbol{D}_m\boldsymbol{D}_m^{\mathrm{T}}| - \tfrac{1}{2}\log|\boldsymbol{C}| \\ &\quad - \tfrac{n-m}{2}\log(2\pi\sigma^2) - \tfrac{\mathrm{RSS}}{2\sigma^2} - \tfrac{\mathrm{e}^{\rho}\|\boldsymbol{D}_m\hat{\boldsymbol{\beta}}\|^2}{2\hat{\sigma}^2}. \end{aligned}$$

For practical convenience, we can replace σ^2 by its Pearson estimate $\hat{\sigma}^2 = \frac{\mathrm{RSS}}{n-\mathsf{edf}}$. This simplifies the restricted log-likelihood to a function of ρ only:

$$\begin{aligned} \mathrm{REML}(\rho) &= \tfrac{1}{2}\log|\mathrm{e}^{\rho}\boldsymbol{D}_m\boldsymbol{D}_m^{\mathrm{T}}| - \tfrac{1}{2}\log|\boldsymbol{C}| \\ &\quad - \tfrac{n-m}{2}\log(2\pi\hat{\sigma}^2) - \tfrac{n-\mathsf{edf}}{2} - \tfrac{\mathrm{e}^{\rho}\|\boldsymbol{D}_m\hat{\boldsymbol{\beta}}\|^2}{2\hat{\sigma}^2}. \end{aligned}$$

This is the REML criterion used in this paper and our implementation.

The log-determinants in the REML score can be computed using Cholesky factors. During the computation of $\hat{\boldsymbol{\beta}}$ (see Sect. 2 for details), we already obtained the Cholesky factorization $\boldsymbol{C} = \boldsymbol{K}\boldsymbol{K}^{\mathrm{T}}$. Therefore,

$$\begin{aligned} &|\boldsymbol{C}| = |\boldsymbol{K}|^2 = \textstyle\prod_{j=1}^{p} K_{jj}^2, \\ &\log|\boldsymbol{C}| = 2\textstyle\sum_{j=1}^{p}\log(K_{jj}), \end{aligned}$$

where K_{jj} is the j^{th} diagonal element of $\boldsymbol{K}$. For the other log-determinant, we first pre-compute the lower triangular Cholesky factor of $\boldsymbol{D}_m\boldsymbol{D}_m^{\mathrm{T}}$, denoted by $\boldsymbol{F}$. (Thanks to the band sparsity of $\boldsymbol{D}_m$, both $\boldsymbol{D}_m\boldsymbol{D}_m^{\mathrm{T}}$ and its Cholesky factorization can be computed at $O(p^2)$ cost.) Then for any trial ρ value, we have:

$$\begin{aligned} &|\mathrm{e}^{\rho}\boldsymbol{D}_m\boldsymbol{D}_m^{\mathrm{T}}| = \mathrm{e}^{(p-m)\rho}|\boldsymbol{D}_m\boldsymbol{D}_m^{\mathrm{T}}| = \mathrm{e}^{(p-m)\rho}|\boldsymbol{F}|^2, \\ &\log|\mathrm{e}^{\rho}\boldsymbol{D}_m\boldsymbol{D}_m^{\mathrm{T}}| = (p-m)\rho + 2\textstyle\sum_{j=1}^{p-m}\log(F_{jj}), \end{aligned}$$

where F_{jj} is the j^{th} diagonal element of $\boldsymbol{F}$.

In summary, quantities in $\mathrm{REML}(\rho)$ can be either pre-computed or obtained while computing $\hat{\boldsymbol{\beta}}$, $\mathrm{RSS}(\rho)$, $\mathsf{edf}(\rho)$ and $\mathrm{GCV}(\rho)$. As a result, for any trial ρ value on a search grid, its REML score can be efficiently computed at $O(p)$ cost.

Appendix C When $\rho = \pm\infty$

When $\rho = -\infty$ or $\rho = +\infty$, the PLS problem (2) degenerates to an ordinary least squares (OLS) problem.

- When $\rho = -\infty$, the penalty term in the PLS objective (2) vanishes, leaving only $\|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta}\|^2$.
- When $\rho = +\infty$, the PLS objective (2) is $+\infty$ anywhere except when $\boldsymbol{D}_m\boldsymbol{\beta} = \boldsymbol{0}$, i.e., $\boldsymbol{\beta}$ is in $\boldsymbol{D}_m$'s null space. Let $\boldsymbol{N}$ be a $p \times m$ matrix whose columns form an orthonormal basis of this null space. Then, we can write $\boldsymbol{\beta} = \boldsymbol{N}\boldsymbol{\alpha}$ in terms of a new coefficient vector $\boldsymbol{\alpha}$. Thus, the objective becomes $\|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{N}\boldsymbol{\alpha}\|^2$.

Without loss of generality, let's express an OLS objective as $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|^2$. Clearly, to match the objective for $\rho = -\infty$, we let $\boldsymbol{X} = \boldsymbol{B}$ and $\boldsymbol{b} = \boldsymbol{\beta}$; to match the objective for $\rho = +\infty$, we let $\boldsymbol{X} = \boldsymbol{B}\boldsymbol{N}$ and $\boldsymbol{b} = \boldsymbol{\alpha}$.

A limiting PLS problem and its corresponding OLS problem have the same edf, residuals, fitted values, GCV errors and REML scores. Since the number of coefficients in an OLS problem defines the edf of the problem, we have $\mathsf{edf} = p$ for $\rho = -\infty$ and $\mathsf{edf} = m$ for $\rho = +\infty$. The GCV error can still be computed by $n \cdot \mathrm{RSS}/(n - \mathsf{edf})^2$, where $\mathrm{RSS} = \|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{b}}\|^2$ and $\hat{\boldsymbol{b}} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$ is the OLS solution. To derive the REML score, we start with the likelihood for the OLS prob-

lem:

$$\Pr(\boldsymbol{y}|\boldsymbol{b}) = c_1 \cdot \exp\left\{-\frac{\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{b}\|^2}{2\sigma^2}\right\},$$

where $c_1 = (2\pi\sigma^2)^{-n/2}$. The Taylor expansion of the least squares at $\hat{\boldsymbol{b}}$ is exactly:

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{b}\|^2 = \text{RSS} + (\boldsymbol{b} - \hat{\boldsymbol{b}})^{\mathrm{T}}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})(\boldsymbol{b} - \hat{\boldsymbol{b}}).$$

Plugging this into $\Pr(\boldsymbol{y}|\boldsymbol{b})$ gives:

$$\Pr(\boldsymbol{y}|\boldsymbol{b}) = c_1 \cdot c_2 \cdot \exp\left\{-\frac{(\boldsymbol{b}-\hat{\boldsymbol{b}})^{\mathrm{T}}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})(\boldsymbol{b}-\hat{\boldsymbol{b}})}{2\sigma^2}\right\},$$

where $c_2 = \exp\{-\frac{\text{RSS}}{2\sigma^2}\}$. The restricted likelihood of σ^2 is defined by integrating out $\boldsymbol{b}$ in $\Pr(\boldsymbol{y}|\boldsymbol{b})$:

$$L_r(\sigma^2) = \int \Pr(\boldsymbol{y}|\boldsymbol{b})\, \mathrm{d}\boldsymbol{b} = c_1 \cdot c_2 \cdot c_3,$$

where the key integral is:

$$\begin{aligned} c_3 &= \int \exp\left\{-\frac{(\boldsymbol{b}-\hat{\boldsymbol{b}})^{\mathrm{T}}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})(\boldsymbol{b}-\hat{\boldsymbol{b}})}{2\sigma^2}\right\} \mathrm{d}\boldsymbol{b} \\ &= (2\pi\sigma^2)^{\mathsf{edf}/2} \cdot |\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}|^{-1/2}. \end{aligned}$$

Thus, the restricted log-likelihood is:

$$\begin{aligned} l_r(\sigma^2) &= \log[L_r(\sigma^2)] = \log(c_1 \cdot c_2 \cdot c_3) \\ &= -\tfrac{n-\mathsf{edf}}{2}\log(2\pi\sigma^2) - \tfrac{1}{2}\log|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}| - \tfrac{\text{RSS}}{2\sigma^2}. \end{aligned}$$

Replacing σ^2 by its Pearson estimate $\hat{\sigma}^2 = \frac{\text{RSS}}{n-\mathsf{edf}}$ gives the REML score:

$$-\tfrac{n-\mathsf{edf}}{2}[1 + \log(2\pi\hat{\sigma}^2)] - \tfrac{1}{2}\log|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}|.$$

References

Andrinopoulou, E.R., Eilers, P.H.C., Takkenberg, J.J.M., et al.: Improved dynamic predictions from joint models of longitudinal and survival data with time-varying effects using P-splines. Biometrics **74**(2), 685–693 (2018). https://doi.org/10.1111/biom.12814

Bremhorst, V., Lambert, P.: Flexible estimation in cure survival models using Bayesian P-splines. Comput. Stat. Data Anal. **93**(SI), 270–284 (2016). https://doi.org/10.1016/j.csda.2014.05.009

Cao, J.: Estimating generalized semiparametric additive models using parameter cascading. Stat. Comput. **22**(4), 857–865 (2012)

Cao, J., Ramsay, J.O.: Linear mixed-effects modeling by parameter cascading. J. Am. Stat. Assoc. **105**(489), 365–374 (2010)

Chen, J., Ohlssen, D., Zhou, Y.: Functional mixed effects model for the analysis of dose-titration studies. Stat. Biopharm. Res. **10**(3), 176–184 (2018). https://doi.org/10.1080/19466315.2018.1458649

de Boor, C.: A Practical Guide to Splines (Revised Edition), Applied Mathematical Sciences, vol. 27. Springer, New York (2001)

Demmler, A., Reinsch, C.: Oscillation matrices with spline smoothing. Numer. Math. **24**(5), 375–382 (1975). https://doi.org/10.1007/BF01437406

Dreassi, E., Ranalli, M.G., Salvati, N.: Semiparametric M-quantile regression for count data. Stat. Methods Med. Res. **23**(6), 591–610 (2014). https://doi.org/10.1177/0962280214536636

Eilers, P.H.C., Marx, B.D.: Flexible smoothing with B-splines and penalties. Stat. Sci. **11**(2), 89–102 (1996). https://doi.org/10.1214/ss/1038425655

Eilers, P., Marx, B.: Generalized linear additive smooth structures. J. Comput. Graph. Stat. **11**(4), 758–783 (2002). https://doi.org/10.1198/106186002844

Eilers, P.H., Marx, B.D.: Practical Smoothing: The Joys of P-Splines. Cambridge University Press, Cambridge (2021)

Franco-Villoria, M., Scott, M., Hoey, T.: Spatiotemporal modeling of hydrological return levels: a quantile regression approach. Environmetrics **30**(2, SI), e2522 (2019). https://doi.org/10.1002/env.2522

Gijbels, I., Ibrahim, M.A., Verhasselt, A.: Testing the heteroscedastic error structure in quantile varying coefficient models. Can. J. Stat. **46**(2), 246–264 (2018). https://doi.org/10.1002/cjs.11346

Goicoa, T., Adin, A., Etxeberria, J., et al.: Flexible Bayesian P-splines for smoothing age-specific spatio-temporal mortality patterns. Stat. Methods Med. Res. **28**(2), 384–403 (2019). https://doi.org/10.1177/0962280217726802

Hendrickx, K., Janssen, P., Verhasselt, A.: Penalized spline estimation in varying coefficient models with censored data. TEST **27**(4), 871–895 (2018). https://doi.org/10.1007/s11749-017-0574-y

Hernando Vanegas, L., Paula, G.A.: An extension of log-symmetric regression models: R codes and applications. J. Stat. Comput. Simul. **86**(9), 1709–1735 (2016). https://doi.org/10.1080/00949655.2015.1081689

Jiang, F., Baek, S., Cao, J., et al.: A functional single-index model. Stat. Sin. **30**(1), 303–324 (2020)

Koehler, M., Umlauf, N., Beyerlein, A., et al.: Flexible Bayesian additive joint models with an application to type 1 diabetes research. Biom. J. **59**(6, SI), 1144–1165 (2017). https://doi.org/10.1002/bimj.201600224

Li, Z., Cao, J.: General P-splines for non-uniform B-splines. Preprint at https://arxiv.org/abs/2201.06808 (2022a)

Li, Z., Cao, J.: gps: general P-splines. R package version 1.1. https://CRAN.R-project.org/package=gps (2022b)

Liu, B., Wang, L., Cao, J.: Estimating functional linear mixed-effects regression models. Comput. Stat. Data Anal. **106**, 153–164 (2017)

Minguez, R., Basile, R., Durban, M.: An alternative semiparametric model for spatial panel data. Stat. Methods Appl. **29**(4), 669–708 (2020). https://doi.org/10.1007/s10260-019-00492-8

Muggeo, V.M.R., Torretta, F., Eilers, P.H.C., et al.: Multiple smoothing parameters selection in additive regression quantiles. Stat. Model. **21**(5), 428–448 (2021). https://doi.org/10.1177/1471082X20929802

Nie, Y., Yang, Y., Wang, L., et al.: Recovering the underlying trajectory from sparse and irregular longitudinal data. Can. J. Stat. **50**(1), 122–141 (2022)

Oliveira, R.A., Paula, G.A.: Additive models with autoregressive symmetric errors based on penalized regression splines. Comput. Stat. **36**(4), 2435–2466 (2021). https://doi.org/10.1007/s00180-021-01106-2

Orbe, J., Virto, J.: Selecting the smoothing parameter and knots for an extension of penalized splines to censored data. J. Stat. Comput. Simul. **91**(14), 2953–2985 (2021). https://doi.org/10.1080/00949655.2021.1913737

Osorio, F.: Influence diagnostics for robust P-splines using scale mixture of normal distributions. Ann. Inst. Stat. Math. **68**(3), 589–619 (2016). https://doi.org/10.1007/s10463-015-0506-0

O'Sullivan, F.: A statistical perspective on ill-posed inverse problems. Stat. Sci. **1**(4), 502–518 (1986). https://doi.org/10.1214/ss/1177013525

Reinsch, C.H.: Smoothing by spline functions. Numer. Math. **10**(3), 177–183 (1967). https://doi.org/10.1007/BF02162161

Reinsch, C.H.: Smoothing by spline functions. II. Numer. Math. **16**(5), 451–454 (1971). https://doi.org/10.1007/BF02169154

Reiss, P.T., Ogden, R.T.: Smoothing parameter selection for a class of semiparametric linear models. J. R. Stat. Soc.: Ser. B (Stat. Methodol.) **71**(2), 505–523 (2009). https://doi.org/10.1111/j.1467-9868.2008.00695.x

Ritchie, H., Mathieu, E., Rodés-Guirao, L., et al.: Coronavirus Pandemic (COVID-19). Our World in Data https://ourworldindata.org/coronavirus (2020)

Ruppert, D., Wand, M.P., Carroll, R.J.: Semiparametric Regression. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge (2003)

Sang, P., Cao, J.: Functional single-index quantile regression models. Stat. Comput. **30**(4), 771–781 (2020)

Spiegel, E., Kneib, T., Otto-Sobotka, F.: Generalized additive models with flexible response functions. Stat. Comput. **29**(1), 123–138 (2019). https://doi.org/10.1007/s11222-017-9799-6

Spiegel, E., Kneib, T., Otto-Sobotka, F.: Spatio-temporal expectile regression models. Stat. Model. **20**(4), 386–409 (2020). https://doi.org/10.1177/1471082X19829945

Wahba, G.: Spline Models for Observational Data. Society for Industrial and Applied Mathematics, Philadelphia (1990)

Wang, Y.: Smoothing Splines: Methods and Applications. Chapman & Hall, London (2011)

Wang, X., Roy, V., Zhu, Z.: A new algorithm to estimate monotone nonparametric link functions and a comparison with parametric approach. Stat. Comput. **28**(5), 1083–1094 (2018). https://doi.org/10.1007/s11222-017-9781-3

Wood, S.N.: Generalized Additive Models: An Introduction with R, 2nd edn. Chapman & Hall (2017)

Woodbury, M.A.: Inverting modified matrices. Technical Report 42, Princeton University (1950)

Xiao, L.: Asymptotic theory of penalized splines. Electron. J. Stat. **13**(1), 747–794 (2019). https://doi.org/10.1214/19-EJS1541

Yu, Y., Wu, C., Zhang, Y.: Penalised spline estimation for generalised partially linear single-index models. Stat. Comput. **27**(2), 571–582 (2017). https://doi.org/10.1007/s11222-016-9639-0