Available online at www.sciencedirect.com

ScienceDirect

Computational Statistics & Data Analysis 52 (2007) 959–974

COMPUTATIONAL STATISTICS & DATA ANALYSIS

www.elsevier.com/locate/csda

Efficient computation for Whittaker–Henderson smoothing

Howard L. Weinert*

Johns Hopkins University, 3400 N. Charles St., 105 Barton Hall, Baltimore, MD 21218, USA

Available online 22 December 2006

Abstract

Efficient algorithms that compute both the estimates and the generalized cross-validation score for the problem of Whittaker–Henderson smoothing are presented. Algorithm efficiency results from carefully exploiting the problem's rich structure to reduce execution time and memory use. The algorithms are much faster than existing ones, and use significantly less memory. MATLAB M-files are included.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Smoothing; Graduation; Cross-validation; Hodrick–Prescott filter

1. Introduction

In the nineteenth century, actuaries began to develop smoothing methods to adjust, or graduate, raw mortality data in order to set life insurance premiums. These early methods involved the application of a moving weighted average filter to the data. The filter coefficients and length were determined by a variety of criteria, and each filter smoothed the data to a different degree. Although simple to implement, these filters have two major drawbacks. They cannot smooth near the ends of the data record, and the degree of smoothing for any particular filter is fixed. See Seal (1981) for a review of this early work. Henderson (1938), Miller (1946), and London (1985) are also useful.

Bohlmann (1899) made the first attempt to rectify both deficiencies of moving weighted average filters. He proposed solving a regularized least-squares problem in which a scalar parameter determines the tradeoff between fidelity to the data and smoothness of the filtered sequence. Whittaker (1923), unaware of Bohlmann's work, proposed the same idea two decades later, and is commonly credited with its invention.

Here is the idea. Given a sequence of n measurements $\{y_1, y_2, \ldots, y_n\}$, a positive real number λ, and a positive integer $p < n$, find the sequence $\{x_1, x_2, \ldots, x_n\}$ that minimizes

$$\lambda \sum_{j=1}^{n} (y_j - x_j)^2 + \sum_{j=1}^{n-p} (\Delta^p x_j)^2, \tag{1.1}$$

where Δ is the forward difference operator:

$$\Delta x_j = x_{j+1} - x_j,$$

$$\Delta^2 x_j = \Delta(\Delta x_j) = x_{j+2} - 2x_{j+1} + x_j,$$

* Tel.: +1 443 310 4332; fax: +1 410 516 5566.
E-mail address: howard@jhu.edu.

0167-9473/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2006.11.038

and so on. The first sum in (1.1) measures fidelity to the data, and the second measures smoothness, where a polynomial of degree $p-1$ is considered maximally smooth. The parameter λ controls the tradeoff between fidelity and smoothness: as $\lambda \to 0$ the solution converges to the polynomial of degree $p-1$ that is the best least-squares fit to the data, and as $\lambda \to \infty$ the solution converges to the measurement sequence. Furthermore, the first sum is a monotonically decreasing function of λ, while the second is a monotonically increasing function of λ.

If

$$y^{\mathrm{T}} = [y_1 \;\; y_2 \;\; \cdots \;\; y_n],$$

$$x^{\mathrm{T}} = [x_1 \;\; x_2 \;\; \cdots \;\; x_n],$$

the cost functional (1.1) can be written as

$$\lambda(y-x)^{\mathrm{T}}(y-x) + x^{\mathrm{T}}M^{\mathrm{T}}Mx, \tag{1.2}$$

where M is a $(n-p) \times n$ differencing matrix. For example, when $p=2$ and $n=6$,

$$M = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix}.$$

The minimizer of (1.2) is the solution of the normal equations

$$A\hat{x} = \lambda y, \tag{1.3}$$

where

$$A = \lambda I + M^{\mathrm{T}}M. \tag{1.4}$$

The A matrix is symmetric, persymmetric (hence centrosymmetric), positive definite, banded (bandwidth $= p$), and quasi-Toeplitz (Toeplitz except for upper left and lower right $p \times p$ blocks). For example, when $p=2$ and $n=6$,

$$A = \begin{bmatrix} 1+\lambda & -2 & 1 & 0 & 0 & 0 \\ -2 & 5+\lambda & -4 & 1 & 0 & 0 \\ 1 & -4 & 6+\lambda & -4 & 1 & 0 \\ 0 & 1 & -4 & 6+\lambda & -4 & 1 \\ 0 & 0 & 1 & -4 & 5+\lambda & -2 \\ 0 & 0 & 0 & 1 & -2 & 1+\lambda \end{bmatrix}. \tag{1.5}$$

The solution can also be obtained via

$$(I + \lambda^{-1}MM^{\mathrm{T}})M\hat{x} = My, \tag{1.6}$$

$$\hat{x} = y - \lambda^{-1}M^{\mathrm{T}}M\hat{x}. \tag{1.7}$$

This smoothing problem has a number of desirable features. Reversing the order of the measurements simply reverses the order of the estimates. Also, the first p moments of the data are preserved: when $p=2$,

$$\sum_{j=1}^{n} \hat{x}_j = \sum_{j=1}^{n} y_j, \quad \sum_{j=1}^{n} j\hat{x}_j = \sum_{j=1}^{n} jy_j.$$

Polynomials of degree $p-1$ are unaffected by the smoothing operation:

$$\hat{x} = y \quad \Leftrightarrow \quad My = 0.$$

Furthermore, since the eigenvalues of MM^{T} are all positive, the eigenvalues of $(I+\lambda^{-1}MM^{\mathrm{T}})^{-1}$ are all less than one, and therefore (see (1.6)), as long as $My \neq 0$,

$$\hat{x}^{\mathrm{T}}M^{\mathrm{T}}M\hat{x} < y^{\mathrm{T}}M^{\mathrm{T}}My,$$

which means that the estimates are actually smoother than the data. See Greville (1957) for a different proof of this fact. Finally, if the smoothing operation is applied iteratively,

$$M\hat{x}^{(k)} = (I+\lambda^{-1}MM^{\mathrm{T}})^{-1}M\hat{x}^{(k-1)} = (I+\lambda^{-1}MM^{\mathrm{T}})^{-k}My \to 0,$$

as $k \to \infty$. Since each $\hat{x}^{(k)}$ will have the same first p moments as the data, $\hat{x}^{(k)}$ will converge to the polynomial of degree $p-1$ that is the best least-squares fit to the data. In other words, iterated smoothing produces the same result as letting $\lambda \to 0$.

Both Bohlmann (1899) and Whittaker (1923) treated the normal equations as a difference equation of order $2p$ with p boundary conditions at each end. Bohlmann (1899) gave a complete solution for $p=1$, while for $p=3$, Whittaker (1923) provided an approximate solution valid for small λ. Later, Whittaker (1924) expressed the solution as an infinite series. For $p=1,2,3$, Henderson (1924) solved the difference equation by first ignoring the boundary conditions, then approximately compensating for them. Aitken (1925) derived an exact solution to the difference equation and boundary conditions. Spoerl (1937) has an in-depth treatment of the difference equation approach. Henderson (1925) was the first to use matrix methods to solve the normal equations with what appears to be the earliest use of LDL^{T} matrix factorization. His implementation is very efficient and is in fact identical to that of Martin et al. (1965).

Another way to solve the problem is to formulate it as a stochastic estimation problem, in which the smoothing parameter λ is the signal-to-noise ratio, and then apply a fixed interval smoothing algorithm. Kitagawa and Gersch (1984) and Verrall (1993) did so, but their state space algorithms are relatively inefficient.

None of the early researchers proposed an automatic way of choosing the smoothing parameter λ. However, Brooks et al. (1988) showed that the measurement-based generalized cross-validation (GCV) method, introduced by Craven and Wahba (1979) for continuous spline smoothing, can also be used for Whittaker–Henderson smoothing. With this adaptive method, λ is chosen to minimize the GCV score

$$n^{-1}\sum_{j=1}^{n}\left(\frac{y_j-\hat{x}_j}{1-n^{-1}\,\mathrm{trace}(\lambda A^{-1})}\right)^2. \tag{1.8}$$

Consequently, the estimates and the trace of λA^{-1} (often called the "hat" matrix) must be computed for many trial values of λ. Fortunately, by using a trick generally attributed to Takahashi et al. (1973) but first employed by Spoerl (1943), the trace can be computed with only $O(n)$ flops. Hutchinson and de Hoog (1985) took this approach for continuous spline smoothing. Alternatively, Kohn and Ansley (1989), using a result from Wahba (1983), computed the trace with $O(n)$ flops as part of a state space algorithm for continuous spline smoothing. For our problem, the work load can be further cut in half by fully exploiting the structure of A. Additionally, Eilers (2003) developed a scaling method to approximately compute the GCV score with $O(n)$ flops.

The issue of algorithm efficiency is critical with large data sets, which occur, for example, in communications or surveillance applications involving either a long observation interval T_d or a high sampling rate F_s. In general, $n=F_sT_d$ so even a 10 s observation interval coupled with a 100 kHz sampling rate produces one million measurements.

In the remainder of this paper we restrict attention to the $p=2$ case used in most applications. See, for example, Leser (1961), Phillips (1962), Unser et al. (1991), King and Rebelo (1993), Hodrick and Prescott (1997), Baxter and King (1999), Ravn and Uhlig (2002), Eilers (2003). In Section 2 we present a LDL^{T} factorization algorithm that computes both the estimates and the GCV score. In Section 3 we show that the estimates and GCV score can alternatively be obtained by solving an equivalent stochastic estimation problem, for which we give a simple derivation and a stable state space algorithm. In Section 4 we exploit convergence properties to produce an even more efficient truncated version of our factorization algorithm. In Section 5 we examine the performance of our full and truncated algorithms. In Section 6 we consider the frequency response in the steady-state case, and comment on the Hodrick–Prescott filter. Section 7 has conclusions and Section 8 contains MATLAB M-files of our algorithms.

2. Factorization algorithm

To solve (1.3) we factor the coefficient matrix as

$$A = LDL^{\mathrm{T}}, \tag{2.1}$$

where L is a banded (bandwidth $=2$) unit lower triangular matrix and D is a diagonal matrix. Denote the elements on the first subdiagonal of L as $\{-e_1, -e_2, \dots, -e_{n-1}\}$ and those on the second subdiagonal as $\{f_1, f_2, \dots, f_{n-2}\}$. Denote the elements on the diagonal of D as $\{d_1, d_2, \dots, d_n\}$. In (2.1), equating corresponding entries, row by row, on the diagonal and first and second superdiagonals leads to

$$d_1 = 1+\lambda, \quad f_1 = 1/d_1, \quad \mu_1 = 2, \quad e_1 = \mu_1 f_1, \tag{2.2}$$

$$d_2 = 5+\lambda-\mu_1 e_1, \quad f_2 = 1/d_2, \quad \mu_2 = 4-e_1, \quad e_2 = \mu_2 f_2, \tag{2.3}$$

$$d_j = 6+\lambda-\mu_{j-1}e_{j-1}-f_{j-2}, \quad f_j = 1/d_j, \quad \mu_j = 4-e_{j-1}, \quad e_j = \mu_j f_j, \tag{2.4}$$

$$d_{n-1} = 5+\lambda-\mu_{n-2}e_{n-2}-f_{n-3}, \quad f_{n-1} = 1/d_{n-1}, \quad \mu_{n-1} = 2-e_{n-2}, \quad e_{n-1} = \mu_{n-1} f_{n-1}, \tag{2.5}$$

$$d_n = 1+\lambda-\mu_{n-1}e_{n-1}-f_{n-2}, \quad f_n = 1/d_n. \tag{2.6}$$

In this way we do not need to form the A matrix in the MATLAB M-file, thus greatly reducing memory use and array access time. As L and D are being obtained, we solve the triangular system

$$LDb = \lambda y,$$

using

$$b_1 = f_1\lambda y_1, \quad b_2 = f_2(\lambda y_2 + \mu_1 b_1), \tag{2.7}$$

$$b_j = f_j(\lambda y_j + \mu_{j-1}b_{j-1} - b_{j-2}). \tag{2.8}$$

Finally, we solve the triangular system

$$L^{\mathrm{T}}\hat{x} = b,$$

using

$$\hat{x}_n = b_n, \quad \hat{x}_{n-1} = b_{n-1} + e_{n-1}\hat{x}_n, \tag{2.9}$$

$$\hat{x}_j = b_j + e_j\hat{x}_{j+1} - f_j\hat{x}_{j+2}. \tag{2.10}$$

Note that we could just as easily have used a $U\bar{D}U^{\mathrm{T}}$ factorization of A, where U is unit *upper* triangular, but this produces the same d, e, f sequences only in reverse order. In other words, U and $\overline{D}$ are 180° rotations of L and D.

The GCV score depends on the diagonal entries of A^{-1}. Since A^{-1} is centrosymmetric, only about half of its diagonal entries are unique. From (2.1),

$$A^{-1} = L^{-\mathrm{T}}D^{-1}L^{-1},$$

and thus,

$$L^{\mathrm{T}}A^{-1} = D^{-1}L^{-1}.$$

Consequently,

$$A^{-1} = A^{-1} + D^{-1}L^{-1} - L^{\mathrm{T}}A^{-1} = D^{-1}L^{-1} + (I - L^{\mathrm{T}})A^{-1}. \tag{2.11}$$

Since L^{-1} is unit lower triangular, $D^{-1}L^{-1}$ is lower triangular with jth diagonal entry f_j. Furthermore, $I - L^{\mathrm{T}}$ is upper triangular and banded (bandwidth $=2$) with zeros on its diagonal and with $\{e_1, e_2, \dots, e_{n-1}\}$ and $\{-f_1, -f_2, \dots, -f_{n-2}\}$ on its first and second superdiagonals, respectively. The unique diagonal entries of A^{-1} can be obtained from

(2.11) by equating corresponding entries on the lower parts of the diagonal and first and second superdiagonals. If g_j, h_j, and q_j, respectively, denote entries on the diagonal and first and second superdiagonals of A^{-1}, then the resulting recursions are

$$g_1 = f_n, \quad h_1 = e_{n-1}g_1, \quad g_2 = f_{n-1} + e_{n-1}h_1, \tag{2.12}$$

$$q_{j-2} = e_{n-j+1}h_{j-2} - f_{n-j+1}g_{j-2}, \quad h_{j-1} = e_{n-j+1}g_{j-1} - f_{n-j+1}h_{j-2}, \tag{2.13}$$

$$g_j = f_{n-j+1} + e_{n-j+1}h_{j-1} - f_{n-j+1}q_{j-2}, \tag{2.14}$$

where j runs from 3 to ceil$(n/2)$. Once the unique diagonal entries have been computed, the trace and the GCV score can be evaluated.

3. State space algorithm

Consider the stochastic model

$$Mx = u, \tag{3.1}$$

$$y = x + v, \tag{3.2}$$

where u and v are mutually uncorrelated with zero means and covariance matrices I and $\lambda^{-1}I$, respectively. Also let

$$\theta_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{3.3}$$

where θ_1 is uncorrelated with u and v, and has zero mean and covariance matrix βI with $\beta > 0$. We can solve (3.1), (3.3) as

$$x = W \begin{bmatrix} \theta_1 \\ u \end{bmatrix}, \tag{3.4}$$

where W is a $n \times n$ unit lower triangular matrix satisfying

$$\begin{aligned} &[I_2 \;\; 0]W = [I_2 \;\; 0], \\ &MW = [0 \;\; I_{n-2}]. \end{aligned} \tag{3.5}$$

If R_x denotes the covariance matrix of x, then from (3.4),

$$R_x = W \begin{bmatrix} \beta I_2 & 0 \\ 0 & I_{n-2} \end{bmatrix} W^{\mathrm{T}}.$$

If $\hat{x}$ is the linear least-squares estimate of x given the measurements y in (3.2), and R_e is the associated error covariance matrix, then

$$\hat{x} = R_x(\lambda^{-1}I + R_x)^{-1}y = (\lambda I + R_x^{-1})^{-1}\lambda y,$$

$$R_e = R_x - R_x(\lambda^{-1}I + R_x)^{-1}R_x = (\lambda I + R_x^{-1})^{-1}.$$

Since (3.5) implies

$$M^{\mathrm{T}}M = W^{-\mathrm{T}} \begin{bmatrix} 0 & 0 \\ 0 & I_{n-2} \end{bmatrix} W^{-1},$$

then for $\beta^{-1} = 0$ (diffuse prior),

$$\lambda I + R_x^{-1} = \lambda I + M^{\mathrm{T}}M = A.$$

Therefore,

$$\hat{x} = A^{-1}\lambda y,$$

which is the minimizer of (1.2), and

$$A^{-1} = R_e. \tag{3.6}$$

Therefore, we can determine the Whittaker–Henderson estimates and GCV score by solving the signal-plus-noise estimation problem modeled by (3.1)–(3.2) with a diffuse prior, and evaluating the trace of R_e.

This estimation problem can be solved by writing (3.1)–(3.3) in state space form. If

$$\theta_k = \begin{bmatrix} x_k \\ x_{k+1} \end{bmatrix}, \tag{3.7}$$

then

$$\theta_{k+1} = F\theta_k + Gu_k,$$

$$y_k = H\theta_k + v_k,$$

where the system matrices are

$$F = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad H = [1 \ \ 0]. \tag{3.8}$$

The state estimate $\hat{\theta}_k$ can be obtained by applying a recursive fixed interval smoothing algorithm, of which there are four basic types (Weinert, 2001). Only the backward–forward algorithm of Mayne (1966) and Desai et al. (1983) and the forward–backward algorithm of Watanabe and Tzafestas (1989) can seamlessly accommodate a diffuse prior without any modifications or complications. Since both algorithms are equally efficient, we will examine only the backward–forward one.

The backward recursions of this algorithm are

$$S_n = \lambda H^{\mathrm{T}}H, \quad r_n = \lambda H^{\mathrm{T}}y_n, \tag{3.9}$$

$$K_k = (1 + G^{\mathrm{T}}S_kG)^{-1}G^{\mathrm{T}}S_kF, \tag{3.10}$$

$$S_{k-1} = (F - GK_k)^{\mathrm{T}}S_k(F - GK_k) + K_k^{\mathrm{T}}K_k + \lambda H^{\mathrm{T}}H, \tag{3.11}$$

$$r_{k-1} = (F - GK_k)^{\mathrm{T}}r_k + \lambda H^{\mathrm{T}}y_{k-1}. \tag{3.12}$$

The forward recursion is

$$\hat{\theta}_1 = S_1^{-1}r_1, \tag{3.13}$$

$$\hat{\theta}_k = (F - GK_k)\hat{\theta}_{k-1} + (1 + G^{\mathrm{T}}S_kG)^{-1}GG^{\mathrm{T}}r_k, \tag{3.14}$$

$$\hat{x}_k = H\hat{\theta}_k. \tag{3.15}$$

The matrix S_1 is nonsingular if $n > 1$.

If P_k denotes the covariance matrix of $(\theta_k - \hat{\theta}_k)$, then from (3.6)–(3.7), (3.15),

$$g_k = (A^{-1})_{kk} = (R_e)_{kk} = HP_kH^{\mathrm{T}}. \tag{3.16}$$

P_k can be computed from the forward recursion

$$P_1 = S_1^{-1}, \tag{3.17}$$

$$P_k = (F - GK_k)P_{k-1}(F - GK_k)^{\mathrm{T}} + (1 + G^{\mathrm{T}}S_kG)^{-1}GG^{\mathrm{T}}, \tag{3.18}$$

where, since A^{-1} is centrosymmetric, k runs from 2 to ceil $(n/2)$. All the above recursions are stable because, for our system matrices (3.8), the spectral radius of $(F - GK_k)$ is less than one for all $k < n$. Note that any off-diagonal entry in A^{-1} can be expressed in terms of P_k via

$$(A^{-1})_{jk} = (R_e)_{jk} = H(F - GK_j) \cdots (F - GK_{k+1}) P_k H^{\mathrm{T}}, \quad j > k. \tag{3.19}$$

See Weinert (2001) and, for related results, Koopman and Harvey (2003). Furthermore, one can verify that

$$P_k = \begin{bmatrix} g_k & h_k \\ h_k & g_{k+1} \end{bmatrix}, \quad 1 \leqslant k \leqslant n-1, \tag{3.20}$$

so that (3.17)–(3.18) is just a version of (2.12)–(2.14).

It has long been known that there is a close connection between triangular factorization and the solution of Riccati equations (Kailath et al., 2000). For our particular problem, one can obtain the following explicit relations between the Riccati equation solution and closed-loop system matrix, and the entries in L:

$$S_j = \begin{bmatrix} \lambda + 1 - f_{n-j-1} & e_{n-j-1} - 2 \\ e_{n-j-1} - 2 & f_{n-j}^{-1} - 1 \end{bmatrix}, \quad 2 \leqslant j \leqslant n-2, \tag{3.21}$$

$$F - GK_j = \begin{bmatrix} 0 & 1 \\ -f_{n-j} & e_{n-j} \end{bmatrix}, \quad 2 \leqslant j \leqslant n-1. \tag{3.22}$$

Hence the Riccati equation (3.11) is a version of (2.2)–(2.6). Therefore, we would expect a MATLAB implementation of the state space algorithm to have nearly the same execution time and memory use as that of the factorization algorithm, and indeed that is the case. Consequently, we will not include its MATLAB M-file in Section 8.

4. Truncated factorization algorithm

Both Weaver (1943) and Spoerl (1943) observed that the sequences in (2.4) converge. Bauer (1954, 1955, 1956) proved convergence and identified the limits and the rate of convergence. See also Malcolm and Palmer (1974) and Hafner (1995). In particular, Bauer showed that as $j, n \to \infty$,

$$e_j \to e, \quad f_j \to f, \tag{4.1}$$

where e and f satisfy the polynomial equation (recall (1.5))

$$z^4 - 4z^3 + (6+\lambda)z^2 - 4z + 1 = \frac{1}{f}(z^2 - ez + f)(fz^2 - ez + 1). \tag{4.2}$$

He also showed that

$$|f_j - f| \leqslant \gamma \rho^{2j}, \tag{4.3}$$

for $\gamma > 0$, where ρ is the magnitude of the roots of the polynomial (4.2) that lie inside the unit circle. (Since this polynomial has real, symmetric coefficients, its roots occur in conjugate and reciprocal pairs.) The e_j sequence converges at the same rate. In light of (3.21)–(3.22), one can also deduce (4.1) and (4.3) from known facts about the convergence of the solution of the Riccati equation (3.11) (Kailath et al., 2000).

Earlier, Henderson (1924) had studied the factorization (4.2) and had shown that

$$e = \frac{2\alpha}{\alpha+1}, \quad f = \frac{\alpha}{\alpha+2}, \tag{4.4}$$

where $\alpha > 0$ is related to the original smoothing parameter λ by

$$\lambda = \frac{4}{\alpha(\alpha+1)^2(\alpha+2)}. \tag{4.5}$$

Table 1
Number of iterations in (2.4) and (2.13)–(2.14)

σ	N	$\hat{N}$
$J=6$		
0.1	70	70
0.3	24	24
0.5	14	14
0.7	10	9
$J=9$		
0.1	104	105
0.3	35	35
0.5	20	20
0.7	14	13

However, it will prove more convenient to use $\sigma \in (0, 1)$ as the basic smoothing parameter, where

$$\sigma = \frac{1}{\alpha+1}, \tag{4.6}$$

in which case,

$$e = 2(1-\sigma), \quad f = \frac{1-\sigma}{1+\sigma}, \quad \lambda = \frac{4\sigma^4}{1-\sigma^2}. \tag{4.7}$$

Note that as $\sigma \to 0$, $\lambda \to 0$, and as $\sigma \to 1$, $\lambda \to \infty$. It turns out (see (6.3)) that $\sigma = \sin\varphi$, where φ is the angle of the first quadrant roots of (4.2), and

$$\rho^2 = f. \tag{4.8}$$

Spoerl (1937, 1943), expanding on the work of Aitken (1925), showed that the sequences in (2.12)–(2.14) also converge. In particular, as $j, n \to \infty$, $g_j \to g$, where, in terms of our smoothing parameter σ,

$$g = \frac{1-\sigma^2}{4\sigma^3(2-\sigma^2)}. \tag{4.9}$$

The rate of convergence of the g_j sequence is identical to that of the f_j and e_j sequences.

With the above facts, we can greatly reduce execution time and memory use for large data sets by computing the sequences in (2.4) and (2.13)–(2.14) only until they are sufficiently close to their limiting values. Ideally, we want to find the smallest integer N such that

$$\max\left\{\left|\frac{f_j - f}{f}\right|, \left|\frac{e_j - e}{e}\right|, \left|\frac{g_j - g}{g}\right|\right\} < 10^{-J}, \quad j \geqslant N, \tag{4.10}$$

where the error exponent J is chosen by the user. For programming purposes, however, it is more efficient to estimate the number of required iterations ahead of time, in terms of σ and J. Recalling (4.3) and (4.8), we seek the smallest integer $\hat{N}$ such that

$$\max\left\{f^{j-1}, \frac{f^j}{e}, \frac{f^j}{g}\right\} < 10^{-J}, \quad j \geqslant \hat{N}. \tag{4.11}$$

Since $f < e$ and $f < g$, the first quantity in the braces is the largest, so we will take

$$\hat{N} = \text{ceil}\left(1 - \frac{J}{\log_{10} f}\right), \tag{4.12}$$

where f is given by (4.7). With this estimate, we evaluate the quantities in (2.4) and (2.13)–(2.14) for $3 \leqslant j \leqslant \hat{N}$, then set

$$f_j = f, \quad e_j = e, \quad g_j = g, \quad j \geqslant \hat{N}+1. \tag{4.13}$$

Note that as $\sigma \to 0$, $f \to 1$ and $\hat{N}$ will eventually exceed $\text{ceil}(n/2)$, in which case one should use the full algorithm. Table 1 shows N and $\hat{N}$ for four values of σ and two values of the error exponent J.

We see that the estimate of N is extremely good, and that for large data sets, the number of necessary iterations will be relatively small.

5. Algorithm performance

Table 2 shows the execution time and memory use of our full and truncated factorization algorithms and that of Eilers (2003). All tests were run on a 1.73 GHz (Centrino) Windows laptop using MATLAB 7.0. Execution times are averages for 50 runs, all using the same simulated measurements and the same value for σ to compute the estimates and the GCV score. Results for the truncated algorithm were the same for all values of σ and J considered in Table 1. Execution time and memory use are proportional to the number of measurements for all three algorithms. The full algorithm runs about 30 times faster than Eilers' while using about one-fifth of the memory. The truncated algorithm requires half the memory and less than 60% of the execution time of the full algorithm.

To assess how much accuracy is lost when the truncated algorithm is used, measurements were simulated with the following MATLAB commands:

$$n = 1 : 100000,$$

$$y = n.*\exp(-.01*n) + \text{randn}(\text{size}(n)).$$

The estimates and GCV score were computed with the full and truncated algorithms. Table 3 shows the maximum relative error in the estimates and the relative error in the GCV score for four values of σ and two values of the error exponent J. Not surprisingly, the errors decreased as J increased. In general, a larger σ meant smaller errors. The entries changed only slightly with different realizations of the random number generator or with different n.

Of course, we do not want to smooth the data using an arbitrary value for σ. We want to use the value that minimizes the GCV score. So we should compare the estimates from the two algorithms when both use optimal σ values. As an

Table 2
Execution time/memory use

Number of measurements	Full algorithm	Truncated algorithm	Eilers' algorithm
10^5	21 ms/3.2 MB	12 ms/1.6 MB	555 ms/15 MB
10^6	210 ms/32 MB	123 ms/16 MB	6047 ms/148 MB

Table 3
Accuracy of truncated algorithm

σ	Maximum relative error in estimates	Relative error in GCV score
$J = 6$		
0.1	1.6×10^{-6}	1.9×10^{-10}
0.3	4.8×10^{-7}	1.1×10^{-10}
0.5	2.5×10^{-7}	2.2×10^{-11}
0.7	3.3×10^{-7}	3.4×10^{-12}
$J = 9$		
0.1	3.7×10^{-8}	8.7×10^{-13}
0.3	3.2×10^{-10}	5.0×10^{-13}
0.5	3.5×10^{-10}	1.2×10^{-13}
0.7	3.1×10^{-10}	1.3×10^{-12}

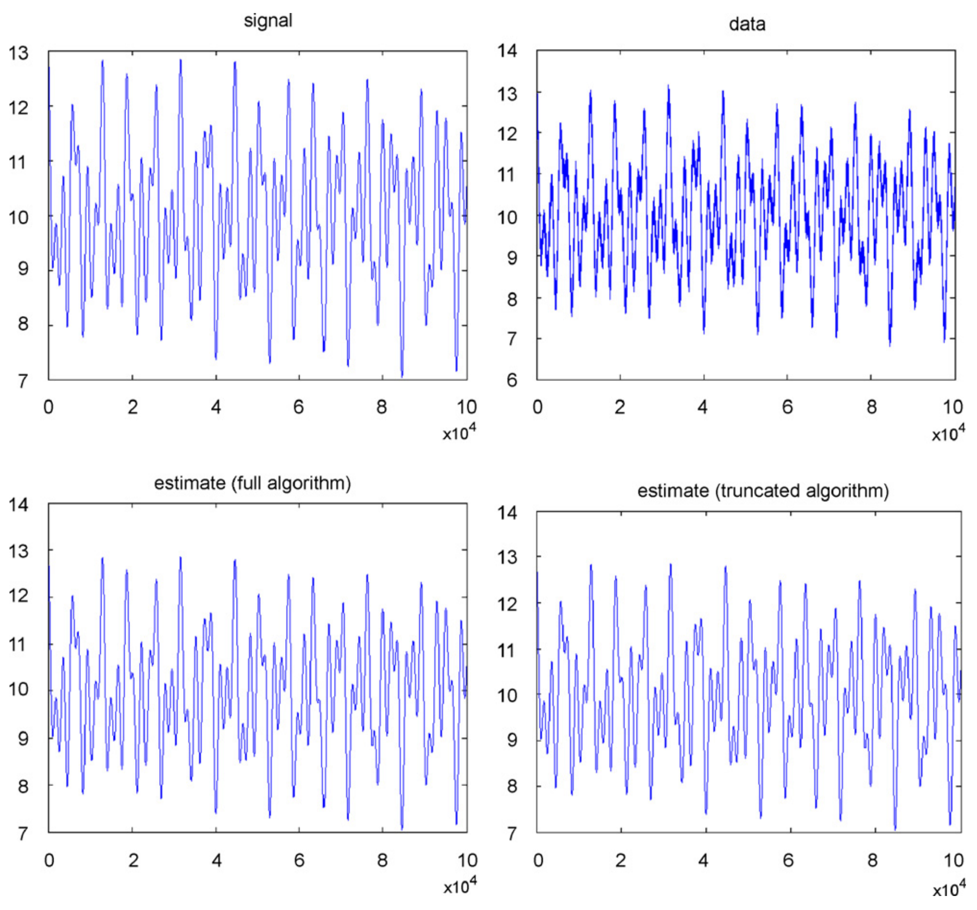

Fig. 1. Smoothing example.

example, measurements were simulated using the commands

$$n = 1 : 100000,$$

$$c = .00001,$$

$$s = 10 + \cos(100 * c * n) + \cos(197 * c * n) + \cos(338 * c * n),$$

$$y = s + 0.1 * \text{randn}(\text{size}(n)).$$

The optimal value for the smoothing parameter was found to be $\sigma = .010$ (corresponding to $\lambda = 4 \times 10^{-8}$) for both the full and truncated algorithms. With error exponent $J = 6$, the maximum relative error between the two estimates was 2.5×10^{-6}; for $J = 9$, the maximum relative error was 8.5×10^{-9}. The rms errors were much smaller: 7.9×10^{-15} and 3.9×10^{-17}, respectively. Plots of the signal s, the data y, and the estimates from the full and truncated ($J = 6$) algorithms are shown in Fig. 1. Simulations with other signals and other noise levels produced similar excellent results.

The choice $J = 6$ should be sufficient for most applications, although taking $J = 9$ entails only a negligible increase in execution time and memory use. Since the user can control the resulting error, the truncated algorithm should be the algorithm of choice.

6. Frequency response of the steady-state smoother

For finite n, the Whittaker–Henderson smoother is a time-varying linear filter and thus is not amenable to standard frequency domain analysis. However, for sufficiently large n, the smoother looks like a time-invariant linear filter except near the ends of the data record. These boundary effects can be eliminated by letting the number of measurements become infinite. The estimates will then satisfy the following fourth-order difference equation determined by the rows of A:

$$\hat{x}_i - 4\hat{x}_{i+1} + (\lambda+6)\hat{x}_{i+2} - 4\hat{x}_{i+3} + \hat{x}_{i+4} = \lambda y_{i+2}. \tag{6.1}$$

This particular difference equation was studied in detail by Spoerl (1937), whose work was based on the earlier research of Henderson (1924) and Aitken (1925). These results, some of which have been rederived by Unser et al. (1991), are summarized in the next paragraph.

Taking the (bilateral) z-transform of (6.1), we see that the transfer function of the steady-state smoother is

$$H(z) = \frac{\hat{X}(z)}{Y(z)} = \frac{\lambda z^2}{z^4 - 4z^3 + (\lambda+6)z^2 - 4z + 1} = \frac{\lambda z^2}{(z-1)^4 + \lambda z^2}. \tag{6.2}$$

The four poles occur in conjugate and reciprocal pairs, and are in the first and fourth quadrants of the complex plane. If ρ is the magnitude of the poles inside the unit circle, and if φ is the angle of the first quadrant poles, the following relations hold (see (4.7)–(4.8)):

$$\sin^2\varphi = \frac{2\sqrt{\lambda}}{\sqrt{\lambda+16}+\sqrt{\lambda}}, \quad \rho^2 = \frac{1-\sin\varphi}{1+\sin\varphi}. \tag{6.3}$$

Note that the region of convergence of $H(z)$ is $\rho < |z| < \rho^{-1}$. By expanding (6.2) in partial fractions and carrying out long division term by term, we can write the transfer function as

$$H(z) = k_0 + \sum_{i=1}^{\infty} k_i(z^{-i} + z^i), \tag{6.4}$$

where

$$k_i = k_0\rho^i(\cos(i\varphi) + \cos\varphi\sin(i\varphi)), \tag{6.5}$$

and

$$k_0 = \frac{\sin\varphi}{2-\sin^2\varphi} = \frac{\sigma}{2-\sigma^2}. \tag{6.6}$$

Consequently, the solution to the steady-state problem is

$$\hat{x}_j = k_0 y_j + \sum_{i=1}^{\infty} k_i(y_{j-i} + y_{j+i}). \tag{6.7}$$

Furthermore,

$$k_0 + 2\sum_{i=1}^{\infty} k_i = 1, \quad \lim_{i\to\infty} k_i = 0, \quad k_0 = \max_i k_i. \tag{6.8}$$

A comparison of (1.3) and (6.7) shows that

$$\lim_{n\to\infty} n^{-1}\mathrm{trace}(\lambda A^{-1}) = k_0, \tag{6.9}$$

and thus when n is very large, the GCV score for a given σ could be estimated as

$$n^{-1}\sum_{j=1}^{n}\left(\frac{y_j - \hat{x}_j}{1-k_0}\right)^2. \tag{6.10}$$

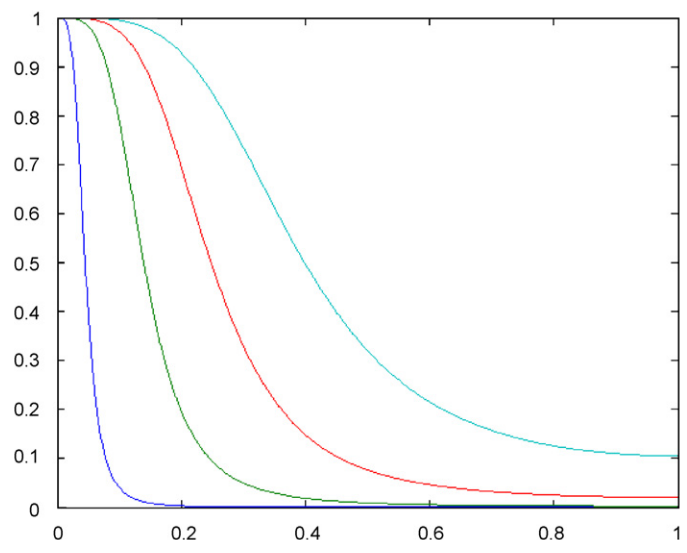

Fig. 2. Frequency response.

De Nicolao et al. (2000) obtained a result analogous to (6.9) for continuous cubic spline smoothing. Also, the fact that the k_i sequence converges exponentially to zero could be deduced from general properties of the inverses of band matrices (Demko, 1977).

Although neither Henderson, Aitken, nor Spoerl studied the frequency response, it can easily be obtained from the transfer function (6.2) by replacing z with $e^{j\omega}$, in which case

$$H(\omega) = \frac{\lambda}{\lambda + 4(1-\cos\omega)^2}. \tag{6.11}$$

Clearly,

$$\lim_{\lambda\to\infty} H(\omega) = 1, \quad \lim_{\lambda\to 0} H(\omega) = \begin{cases} 1, & \omega = 0, \\ 0, & \omega \neq 0. \end{cases} \tag{6.12}$$

See Fig. 2 for plots of the frequency response as a function of the normalized frequency ω/π. From left to right, the curves correspond to $\lambda = 4.0 \times 10^{-4}$ $(\sigma = 0.1)$, $\lambda = 3.6 \times 10^{-2}$ $(\sigma = 0.3)$, $\lambda = 0.33$ $(\sigma = 0.5)$, $\lambda = 1.9$ $(\sigma = 0.7)$.

As long as

$$\lambda \leqslant \frac{4}{\sqrt{2}-1} \cong 9.66, \tag{6.13}$$

this low-pass filter has a cutoff frequency ω_c given by

$$\omega_c = \cos^{-1}\left(1 - \tfrac{1}{2}\sqrt{(\sqrt{2}-1)\lambda}\right). \tag{6.14}$$

Note that

$$H(\omega) \cong \frac{\lambda}{\lambda + \omega^4} \quad \text{for small } \omega. \tag{6.15}$$

Also, since $H^{(2)}(0) = 0$, the steady-state smoother reproduces cubics (Schoenberg, 1946). For finite n, the Whittaker–Henderson smoother will reproduce lines, but not cubics.

Economists have used a modification of the Whittaker–Henderson smoother to study business cycles (King and Rebelo, 1993; Hodrick and Prescott, 1997; Baxter and King, 1999; Ravn and Uhlig, 2002). This so-called Hodrick–Prescott filter computes $(y - \hat{x})$ instead of $\hat{x}$. Its steady-state version is therefore a high-pass filter with frequency response:

$$H_{\text{hp}}(\omega) = 1 - H(\omega) = \frac{4(1-\cos\omega)^2}{\lambda + 4(1-\cos\omega)^2}. \tag{6.16}$$

As long as

$$\lambda \leqslant \frac{16(\sqrt{2}-1)}{4-\sqrt{2}} \cong 2.56, \tag{6.17}$$

the steady-state Hodrick–Prescott filter has a cutoff frequency $\bar{\omega}_{\text{c}}$ given by

$$\bar{\omega}_{\text{c}} = \cos^{-1}\left(1 - \frac{2\sqrt{\lambda}}{\sqrt{\sqrt{2}(\lambda+16) - 16}}\right). \tag{6.18}$$

Instead of letting the measurements speak for themselves by using generalized cross-validation to choose λ, economists arbitrarily set $\bar{\omega}_{\text{c}} = \pi/16$ when smoothing quarterly data, thus cutting off cyclical components with periods exceeding eight years, or 32 quarters. This choice implies $\lambda^{-1} = 1635$, which is rounded to 1600. There is less unanimity when data are acquired annually or at some other rate. However, if one accepts $\lambda^{-1} = 1635$ for quarterly data, then for other rates the filter should continue to cutoff periods above eight years. For annual data, therefore, we should set $\bar{\omega}_{\text{c}} = \pi/4$ which implies $\lambda^{-1} = 6.822$. This is approximately the conclusion of Ravn and Uhlig (2002), who reasoned along different lines. Of course, any rationale related to cutoff frequency requires that the number of measurements be large enough to make the steady-state approximation reasonable.

7. Conclusions

The key to efficient computation is special-purpose code that takes advantage of the mathematical structure of the problem and the characteristics of the programming language. We have produced full and truncated algorithms for the Whittaker–Henderson smoothing problem. The truncated algorithm involves a slight approximation, but the resulting error can be controlled by the user. By using our approach, the interested reader can produce similar code for other values of p. In forthcoming work, we will address the problem of smoothing with interpolation.

8. MATLAB M-files

The M-files for the full factorization algorithm *smooth* and the truncated algorithm *tsmooth* are given below. The inputs to *smooth* are the row vector of measurements and the smoothing parameter σ. The outputs are the row vector of estimates and the GCV score. The inputs to *tsmooth* are the row vector of measurements, the smoothing parameter σ, and the error exponent J. The outputs are the row vector of estimates and the GCV score.

```
function[x, score] = smooth(y, sig)
n = length(y); nc = ceil(n/2);
e = zeros(1, n - 1); f = zeros(1, n); x = zeros(1, n);
lam = 4 * sig^4/(1 - sig^2);
a1 = 1 + lam; a2 = 5 + lam; a3 = 6 + lam;
```

```
%Factor the coefficient matrix and solve the first triangular system

d = a1; f(1) = 1/d; x(1) = f(1) * lam * y(1); mu = 2; e(1) = mu * f(1);
d = a2 − mu * e(1); f(2) = 1/d; x(2) = f(2) * (lam * y(2) + mu * x(1)); mu = 4 − e(1); e(2) = mu * f(2);
for j = 3 : n − 2
   m1 = j − 1;
   m2 = j − 2;
   d = a3 − mu * e(m1) − f(m2);
   f(j) = 1/d;
   x(j) = f(j) * (lam * y(j) + mu * x(m1) − x(m2));
   mu = 4 − e(m1);
   e(j) = mu * f(j);
end
d = a2 − mu * e(n − 2) − f(n − 3); f(n − 1) = 1/d;
x(n − 1) = f(n − 1) * (lam * y(n − 1) + mu * x(n − 2) − x(n − 3));
mu = 2 − e(n − 2); e(n − 1) = mu * f(n − 1);
d = a1 − mu * e(n − 1) − f(n − 2); f(n) = 1/d;
x(n) = f(n) * (lam * y(n) + mu * x(n − 1) − x(n − 2));

%Solve the second triangular system and find avg squared error
sq = (y(n) − x(n))^2;
x(n − 1) = x(n − 1) + e(n − 1) * x(n);
sq = sq + (y(n − 1) − x(n − 1))^2;
for j = n − 2 : −1 : 1
   x(j) = x(j) + e(j) * x(j + 1) − f(j) * x(j + 2);
   sq = sq + (y(j) − x(j))^2;
end
sq = sq/n;

%Compute GCV score

g2 = f(n); tr = g2; h = e(n − 1) * g2;
g1 = f(n − 1) + e(n − 1) * h; tr = tr + g1;
for k = n − 2 : −1 : n − nc + 1
   q = e(k) * h − f(k) * g2;
   h = e(k) * g1 − f(k) * h; g2 = g1;
   g1 = f(k) + e(k) * h − f(k) * q;
   tr = tr + g1;
end
tr = (2 * tr − rem(n, 2) * g1) * lam/n;
score = sq/(1 − tr)^2;

function[x, score] = tsmooth(y, sig, J)
n = length(y); nc = ceil(n/2);
elim = 2 * (1 − sig); flim = (1 − sig)/(1 + sig); lam = 4 * sig^4/(1 − sig^2);
N = ceil(1 − J/ log 10(flim)); glim = (1 − sig^2)/(4 * sig^3 * (2 − sig^2));
e = zeros(1, N + 1); f = zeros(1, N + 2); x = zeros(1, n);
a1 = 1 + lam; a2 = 5 + lam; a3 = 6 + lam;
if N > nc
   error('sig too small, use smooth instead')
end
```

```
%Factor the coefficient matrix and solve the first triangular system

$d = a1$; $f(1) = 1/d$; $x(1) = f(1) * \mathrm{lam} * y(1)$; $\mathrm{mu} = 2$; $e(1) = \mathrm{mu} * f(1)$;
$d = a2 - \mathrm{mu} * e(1)$; $f(2) = 1/d$; $x(2) = f(2) * (\mathrm{lam} * y(2) + \mathrm{mu} * x(1))$; $\mathrm{mu} = 4 - e(1)$; $e(2) = mu * f(2)$;
for $j = 3 : N$
    $m1 = j - 1$; $m2 = j - 2$;
    $d = a3 - mu * e(m1) - f(m2)$;
    $f(j) = 1/d$;
    $x(j) = f(j) * (\mathrm{lam} * y(j) + \mathrm{mu} * x(m1) - x(m2))$;
    $\mathrm{mu} = 4 - e(m1)$;
    $e(j) = mu * f(j)$;
end
$\mathrm{mu} = 4 - \mathrm{elim}$;
for $j = N + 1 : n - 2$
    $x(j) = \mathrm{flim} * (\mathrm{lam} * y(j) + \mathrm{mu} * x(j - 1) - x(j - 2))$;
end
$d = a2 - \mathrm{mu} * \mathrm{elim} - \mathrm{flim}$; $f(N + 1) = 1/d$;
$x(n - 1) = f(N + 1) * (\mathrm{lam} * y(n - 1) + \mathrm{mu} * x(n - 2) - x(n - 3))$;
$\mathrm{mu} = 2 - \mathrm{elim}$; $e(N + 1) = \mathrm{mu} * f(N + 1)$;
$d = a1 - \mathrm{mu} * e(N + 1) - \mathrm{flim}$; $f(N + 2) = 1/d$;
$x(n) = f(N + 2) * (\mathrm{lam} * y(n) + \mathrm{mu} * x(n - 1) - x(n - 2))$;

%Solve the second triangular system and find avg squared error

$\mathrm{sq} = (y(n) - x(n))$^2;
$x(n - 1) = x(n - 1) + e(N + 1) * x(n)$;
$\mathrm{sq} = \mathrm{sq} + (y(n - 1) - x(n - 1))$^2;
for $j = n - 2 : -1 : N + 1$
    $x(j) = x(j) + \mathrm{elim} * x(j + 1) - \mathrm{flim} * x(j + 2)$;
    $\mathrm{sq} = \mathrm{sq} + (y(j) - x(j))$^2;
end
for $j = N : -1 : 1$
    $x(j) = x(j) + e(j) * x(j + 1) - f(j) * x(j + 2)$;
    $\mathrm{sq} = \mathrm{sq} + (y(j) - x(j))$^2;
end
$\mathrm{sq} = \mathrm{sq}/n$;

%Compute GCV score

$g2 = f(N + 2)$; $\mathrm{tr} = g2$; $h = e(N + 1) * g2$;
$g1 = f(N + 1) + e(N + 1) * h$; $\mathrm{tr} = \mathrm{tr} + g1$;
for $k = n - 2 : -1 : n - N + 1$
    $q = \mathrm{elim} * h - \mathrm{flim} * g2$;
    $h = \mathrm{elim} * g1 - \mathrm{flim} * h$; $g2 = g1$;
    $g1 = \mathrm{flim} + \mathrm{elim} * h - \mathrm{flim} * q$;
    $\mathrm{tr} = \mathrm{tr} + g1$;
end
$\mathrm{tr} = \mathrm{tr} + (nc - N) * \mathrm{glim}$;
$\mathrm{tr} = (2 * \mathrm{tr} - \mathrm{rem}(n, 2) * \mathrm{glim}) * \mathrm{lam}/n$;
$\mathrm{score} = \mathrm{sq}/(1 - \mathrm{tr})$^2;
```

References

Aitken, A.C., 1925. On the theory of graduation. Proc. Roy. Soc. Edinburgh 46, 36–45.

Bauer, F.L., 1954. Beitrage zur entwicklung numerischer verfahren fur programmgesteuerte rechenanlagen I. Sitzungsberichte Math.-Naturwissen. Klasse Bayerischen Akad. Wissen. Munchen 275–303.

Bauer, F.L., 1955. Ein direktes iterationsverfahren zur Hurwitz-zerlegung eines polynoms. Archiv Elektrischen Ubertragung 9, 285–290.

Bauer, F.L., 1956. Beitrage zur entwicklung numerischer verfahren fur programmgesteuerte rechenanlagen II. Sitzungsberichte Math.-Naturwissen. Klasse Bayerischen Akad. Wissen. Munchen 163–203.

Baxter, M., King, R.G., 1999. Measuring business cycles: approximate bandpass filters for economic time series. Rev. Econ. Statist. 81, 575–593.

Bohlmann, G., 1899. Ein ausgleichungsproblem. Nachrichten Gesellschaft Wissenschaften Gottingen, Math.-Phys. Klasse 260–271.

Brooks, R.J., Stone, M., Chan, F.Y., Chan, L.K., 1988. Cross-validatory graduation. Insurance: Math. Econ. 7, 59–66.

Craven, P., Wahba, G., 1979. Smoothing noisy data with spline functions. Numer. Math. 31, 377–403.

De Nicolao, G., Ferrari-Trecate, G., Sparacino, G., 2000. Fast spline smoothing via spectral factorization concepts. Automatica 36, 1733–1739.

Demko, S., 1977. Inverses of band matrices and local convergence of spine projections. SIAM J. Numer. Anal. 14, 616–619.

Desai, U.B., Weinert, H.L., Yusypchuk, G.J., 1983. Discrete-time complementary models and smoothing algorithms: the correlated noise case. IEEE Trans. Automat. Control 28, 536–539.

Eilers, P.H.C., 2003. A perfect smoother. Analytical Chem. 75, 3631–3636.

Greville, T.N.E., 1957. On smoothing a finite table: a matrix approach. J. SIAM 5, 137–154.

Hafner, J.L., 1995. Explicit and asymptotic formulas for LDM^T factorization of banded Toeplitz matrices. Linear Algebra Appl. 222, 97–126.

Henderson, R., 1924. A new method of graduation. Trans. Actuarial Soc. Amer. 25, 29–40.

Henderson, R., 1925. Further remarks on graduation. Trans. Actuarial Soc. Amer. 26, 52–57.

Henderson, R., 1938. Mathematical Theory of Graduation. Actuarial Society of America, New York.

Hodrick, R.J., Prescott, E.C., 1997. Postwar U.S. business cycles: an empirical investigation. J. Money Credit Banking 29, 1–16.

Hutchinson, M.F., de Hoog, F.R., 1985. Smoothing noisy data with spline functions. Numer. Math. 47, 99–106.

Kailath, T., Sayed, A.H., Hassibi, B., 2000. Linear Estimation. Prentice-Hall, Upper Saddle River.

King, R.G., Rebelo, S.T., 1993. Low frequency filtering and real business cycles. J. Econ. Dynamics Control 17, 207–231.

Kitagawa, G., Gersch, W., 1984. A smoothness priors-state space modeling of time series with trend and seasonality. J. Amer. Statist. Assoc. 79, 378–389.

Kohn, R., Ansley, C.F., 1989. A fast algorithm for signal extraction, influence and cross-validation in state space models. Biometrika 76, 65–79.

Koopman, S.J., Harvey, A., 2003. Computing observation weights for signal extraction and filtering. J. Econ. Dynamics Control 27, 1317–1333.

Leser, C.E.V., 1961. A simple method of trend construction. J. Roy. Statist. Soc. B 23, 91–107.

London, D., 1985. Graduation: The Revision of Estimates. ACTEX, Abington.

Malcolm, M.A., Palmer, J., 1974. A fast method for solving a class of tridiagonal linear systems. Comm. ACM 17, 14–17.

Martin, R.S., Peters, G., Wilkinson, J.H., 1965. Symmetric decomposition of a positive definite matrix. Numer. Math. 7, 362–383.

Mayne, D.Q., 1966. A solution of the smoothing problem for linear dynamic systems. Automatica 4, 73–92.

Miller, M.D., 1946. Elements of Graduation. Actuarial Society of America, New York.

Phillips, D.L., 1962. A technique for the numerical solution of certain integral equations of the first kind. J. ACM 9, 84–97.

Ravn, M.O., Uhlig, H., 2002. On adjusting the Hodrick–Prescott filter for the frequency of observations. Rev. Econ. Statist. 84, 371–376.

Schoenberg, I.J., 1946. Contributions to the problem of approximation of equidistant data by analytic functions. Part A—On the problem of smoothing or graduation. Quart. Appl. Math. 4, 45–99.

Seal, H.L., 1981. Graduation by piecewise cubic polynomials: a historical review. Deutsche Ges. Versicherungsmath. 15, 89–114.

Spoerl, C.A., 1937. The Whittaker–Henderson graduation formula A. Trans. Actuarial Soc. Amer. 38, 403–462.

Spoerl, C.A., 1943. A fundamental proposition in the solution of simultaneous linear equations. Trans. Actuarial Soc. Amer. 44, 276–288.

Takahashi, K., Fagan, J., Chen, M. S., 1973. Formation of a sparse bus impedance matrix and its application to short circuit study. In: Proceedings of the Power Industry Computer Applications Conference, pp. 63–69.

Unser, M., Aldroubi, A., Eden, M., 1991. Recursive regularization filters: design, properties and applications. IEEE Trans. Pattern Anal. Mach. Intell. 13, 272–277.

Verrall, R.J., 1993. A state space formulation of Whittaker graduation with extensions. Insurance: Math. Econ. 13, 7–14.

Wahba, G., 1983. Bayesian confidence intervals for the cross-validated smoothing spline. J. Roy. Statist. Soc. B 45, 133–150.

Watanabe, K., Tzafestas, S.G., 1989. New computationally efficient formula for backward-pass fixed-interval smoother and its UD factorisation algorithm. IEE Proc. D 136, 73–78.

Weaver, C.L., 1943. Modification of working formulas in Whittaker–Henderson A graduation method to produce the moments automatically. Trans. Actuarial Soc. Amer. 44, 27–30.

Weinert, H.L., 2001. Fixed Interval Smoothing for State Space Models. Springer, New York.

Whittaker, E.T., 1923. On a new method of graduation. Proc. Edinburgh Math. Soc. 41, 63–75.

Whittaker, E.T., 1924. On the theory of graduation. Proc. Roy. Soc. Edinburgh 44, 77–83.