



Robust smoothing of gridded data in one and higher dimensions with missing values

Damien Garcia*

CRCHUM – Research Centre, University of Montreal Hospital, Montreal, Canada

ARTICLE INFO

Article history:

Received 28 March 2009

Received in revised form 15 September 2009

Accepted 16 September 2009

Available online 30 September 2009

ABSTRACT

A fully automated smoothing procedure for uniformly sampled datasets is described. The algorithm, based on a penalized least squares method, allows fast smoothing of data in one and higher dimensions by means of the discrete cosine transform. Automatic choice of the amount of smoothing is carried out by minimizing the generalized cross-validation score. An iteratively weighted robust version of the algorithm is proposed to deal with occurrences of missing and outlying values. Simplified Matlab codes with typical examples in one to three dimensions are provided. A complete user-friendly Matlab program is also supplied. The proposed algorithm, which is very fast, automatic, robust and requiring low storage, provides an efficient smoother for numerous applications in the area of data analysis.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

In this paper, a fast robust version of a discretized smoothing spline is introduced. The proposed method, based on the discrete cosine transform (DCT), allows robust smoothing of equally spaced data in one and higher dimensions. The following paragraphs of the present section briefly describe the underlying penalized least squares approach, in the particular case of one-dimensional data. Section 2 provides a review of the generalized cross-validation which is used to estimate the smoothing parameter. When data are evenly spaced, it is shown (Section 3) that the linear system can be greatly simplified and solved by means of the DCT. An iterative robust scheme of this DCT-based penalized regression is then proposed in order to deal with weighted, missing (Section 4) and outlying (Section 5) values. It is finally explained, in Section 6, how the smoothing technique proposed in this paper can be applied to multidimensional data. To illustrate the effectiveness of the smoother, several examples related to one-, two- and three-dimensional data are presented (Section 7). Two simplified Matlab functions and one complete optimized Matlab code are also provided.

In statistics and data analysis, smoothing is used to reduce experimental noise or small-scale information while keeping the most important imprints of a dataset. Consider the following model for the one-dimensional noisy signal y :

$$y = \hat{y} + \varepsilon, \quad (1)$$

where ε represents a Gaussian noise with mean zero and unknown variance, and \hat{y} is supposed to be smooth, i.e. has continuous derivatives up to some order (typically ≥ 2) over the whole domain. Smoothing y relies upon finding the best estimate of \hat{y} . Data smoothing is generally carried out by means of parametric or nonparametric regression. Parametric regression requires some *a priori* knowledge of the so-called regression equation that can represent the data well. The majority of observed values, however, cannot be parameterized in terms of predetermined analytical functions, so nonparametric regression is usually the best option for smoothing of data (Takezawa, 2005). To cite a few instances, some of the most common approaches to nonparametric regression used in data processing include kernel regression (Hastie

* Corresponding address: LBUM, CRCHUM, Pavillon J.A. de Sève (room Y-1619), 2099 Alexandre de Sève, Montreal, QC, H2L 2W5, Canada. Tel.: +1 514 890 8000 (24705).

E-mail address: Damien.Garcia@crchum.qc.ca.

and Loader, 1993) like moving average filtering, local polynomial regression (Watson, 1964) and the Savitzky–Golay filter (Savitzky and Golay, 1964). Another classical approach to smoothing is the penalized least squares regression. This method was first introduced in the early 1920's by Whittaker (1923) and it has been extensively studied ever since (Wahba, 1990a). This technique consists in minimizing a criterion that balances the fidelity to the data, measured by the residual sum-of-squares (RSS), and a penalty term (P) that reflects the roughness of the smooth data. One thus seeks to minimize

$$F(\hat{y}) = \text{RSS} + s P(\hat{y}) = \|\hat{y} - y\|^2 + s P(\hat{y}), \quad (2)$$

where $\|\cdot\|$ denotes the Euclidean norm. The parameter s is a real positive scalar that controls the degree of smoothing: as the smoothing parameter increases, the smoothing of \hat{y} also increases. When the roughness penalty is written in terms of the square integral of the p th derivative of \hat{y} , the penalized regression is known as a smoothing spline (Schoenberg, 1964; Takezawa, 2005; Wahba, 1990a). Another simple and straightforward approach to express the roughness is by using a second-order divided difference (Weinert, 2007; Whittaker, 1923) which yields, for a one-dimensional data array,

$$P(\hat{y}) = \|D\hat{y}\|^2, \quad (3)$$

where D is a tridiagonal square matrix defined by

$$D_{i,i-1} = \frac{2}{h_{i-1}(h_{i-1} + h_i)}, \quad D_{i,i} = \frac{-2}{h_{i-1}h_i}, \quad D_{i,i+1} = \frac{2}{h_i(h_{i-1} + h_i)},$$

for $2 \leq i \leq n-1$, where n is the number of elements in \hat{y} , and h_i represents the step between \hat{y}_i and \hat{y}_{i+1} . Assuming repeating border elements ($y_0 = y_1$ and $y_{n+1} = y_n$) gives

$$-D_{1,1} = D_{1,2} = \frac{1}{h_1^2} \quad \text{and} \quad D_{n,n-1} = -D_{n,n} = \frac{1}{h_{n-1}^2}.$$

This procedure is similar to the Whittaker smoothing (Eilers, 2003; Weinert, 2007; Whittaker, 1923), apart from the repeating boundaries. Such boundary conditions lead to further numerical simplifications with evenly spaced data (Buckley, 1994), as explained further in Section 3.

Now, using Eqs. (2) and (3), minimization of $F(\hat{y})$ gives the following linear system that allows the determination of the smoothed data:

$$(I_n + sD^T D)\hat{y} = y, \quad (4)$$

where I_n is the $n \times n$ identity matrix and D^T stands for the transpose of D . Because $(I_n + sD^T D)$ is a symmetric pentadiagonal matrix, the latter equation can be numerically solved very efficiently with technical computing software (Eilers, 2003; Weinert, 2007). Computation of \hat{y} using Matlab (The MathWorks, Natick, MA, USA) and Eq. (4) has recently been introduced by Eilers (2003) and Weinert (2007). In the case of evenly spaced data, Weinert has proposed a relatively fast algorithm based on Cholesky decomposition (Weinert, 2007). This algorithm, however, loses efficiency when an estimation of the smoothing parameter is required, as explained in Section 2.

2. Estimation of the smoothing parameter

As indicated by Eq. (4), the output \hat{y} is strongly influenced by the smoothing parameter s . Assuming the model given by Eq. (1), it is appropriate to use the smoothing parameter that yields the best estimate of the original data and thus avoids over- or under-smoothing as much as possible. Such a correct value can be estimated by the method of generalized cross-validation (GCV).

The GCV method was introduced by Craven and Wahba (1978) and Wahba (1990b) in the context of smoothing splines. Assuming that one wants to solve the smoothing linear system

$$\hat{y} = H(s)y,$$

where H is the so-called hat matrix, the GCV method chooses the parameter s that minimizes the GCV score (Craven and Wahba, 1978; Golub et al., 1979):

$$s = \text{argmin}(GCV) \quad \text{with} \quad GCV(s) \equiv \frac{\text{RSS}/n}{(1 - \text{Tr}(H)/n)^2},$$

where Tr denotes the matrix trace. Finding s which minimizes the GCV score thus requires $\text{Tr}(H)$ to be known. Here, according to Eq. (4), H is given by

$$H = (I_n + sD^T D)^{-1}. \quad (5)$$

To calculate the GCV score, Eilers (2003) and Weinert (2007) have both proposed algorithms that involve the determination of H at each iterative step of the minimization. Such a process is very time-consuming and can be avoided. Indeed, $\text{Tr}(H)$ can be simply reduced to

$$\text{Tr}(H) = \sum_{i=1}^n \frac{1}{1 + s\lambda_i^2}, \quad (6)$$

where $(\lambda_i^2)_{i=1 \dots n}$ are the eigenvalues of $D^T D$. The GCV score thus reduces to

$$\text{GCV}(s) = n \sum_{i=1}^n (\hat{y}_i - y_i)^2 / \left(n - \sum_{i=1}^n (1 + s\lambda_i^2)^{-1} \right)^2. \quad (7)$$

Finding the s value that minimizes the GCV score yielded by Eq. (7) makes the smoothing algorithm fully automated. Because the components of \hat{y} appear in the expression of the GCV score, \hat{y} has to be calculated at each step of the minimization process. This can be avoided for the particular case of equally spaced data, as demonstrated in Section 2.

3. Smoothing of evenly spaced data

Eq. (4) can be efficiently solved with a Matlab code using the so-called left matrix division (see Matlab documentation) applied to sparse matrices. Solving such a linear system, however, may rapidly become time-consuming with an increasing number of data. This algorithm can be greatly simplified and sped up if the data are evenly spaced. Indeed, in numerous conditions, experimental acquisition and data measurement lead to equally spaced or evenly gridded datasets. An efficient algorithm for smoothing of uniformly sampled data is described in the present section. A generalization for multidimensional data will be described in Section 6. Assuming now that the data are equally spaced with $h_i = 1, \forall i$, the divided difference matrix D (in Eq. (4)) can be rewritten as the simple difference matrix

$$D = \begin{pmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{pmatrix}.$$

An eigendecomposition of D yields

$$D = U \Lambda U^{-1},$$

where Λ is the diagonal matrix containing the eigenvalues of D defined by Yueh (2005):

$$\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n) \quad \text{with } \lambda_i = -2 + 2 \cos((i-1)\pi/n). \quad (8)$$

Because U is a unitary matrix (i.e. $U^{-1} = U^T$ and $UU^T = I_n$), Eq. (4) leads to

$$\hat{y} = U (I_n + s\Lambda^2)^{-1} U^T y \equiv U \Gamma U^T y, \quad (9)$$

where the components of the diagonal matrix Γ , according to Eq. (8), are given by

$$\Gamma_{i,i} = [1 + s(2 - 2 \cos((i-1)\pi/n))^2]^{-1} \quad \text{and} \quad \Gamma_{i,j} = 0 \quad \text{if } i \neq j. \quad (10)$$

It is worth noting that U^T and U are actually n -by- n type-2 discrete cosine transform (DCT) and inverse DCT matrices, respectively (Strang, 1999). Thus, the smooth output \hat{y} can also be expressed as

$$\hat{y} = U \Gamma \text{DCT}(y) = \text{IDCT}(\Gamma \text{DCT}(y)), \quad (11)$$

where DCT and IDCT refer to the discrete cosine transform and the inverse discrete cosine transform, respectively.

In the case of equally spaced data, the GCV score can also be simplified considerably. The trace of the hat matrix [see Eqs. (6) and (8)] indeed reduces to

$$\text{Tr}(H) = \sum_{i=1}^n [1 + s(2 - 2 \cos((i-1)\pi/n))^2]^{-1}.$$

Note that, when n becomes large, one has

$$\frac{\text{Tr}(H)}{n} \approx \frac{1}{\pi} \int_0^\pi [1 + s(2 - 2 \cos x)^2]^{-1} dx = \frac{\sqrt{1 + \sqrt{1 + 16s}}}{\sqrt{2}\sqrt{1 + 16s}}. \quad (12)$$

Moreover, using Eq. (9), the residual sum of squares (RSS) can be written as

$$\begin{aligned} \text{RSS} &= \|\hat{y} - y\|^2 \\ &= \left\| \left((I_n + s\Lambda^2)^{-1} - I_n \right) \text{DCT}(y) \right\|^2 \\ &= \sum_{i=1}^n \left(\frac{1}{1 + s\lambda_i^2} - 1 \right)^2 \text{DCT}_i^2(y), \end{aligned}$$

where DCT_i refers to the i th component of the discrete cosine transform. The GCV score given by Eq. (7) thus becomes

$$\text{GCV}(s) = \frac{n \sum_{i=1}^n \left(\frac{1}{1+s\lambda_i^2} - 1 \right)^2 \text{DCT}_i^2(y)}{\left(n - \sum_{i=1}^n \frac{1}{1+s\lambda_i^2} \right)^2}, \quad (13)$$

with λ_i given by Eq. (8).

It is much more convenient to use an algorithm based on Eq. (11) rather than Eq. (4) when the data are equispaced. The DCT indeed has a computational complexity of $O(n \log(n))$, whereas Eq. (4) requires a Cholesky factorization whose computational complexity is $O(n^3)$. Note also that the computation of the GCV score from Eq. (13) is straightforward and does not require any matrix operation and manipulation, which makes the automated smoothing very fast. A simplified Matlab code for fully automated smoothing is given in Appendix A. The above-mentioned smoothing procedure requiring the DCT is similar to that introduced by Buckley (1994). It is shown in Section 4 how this DCT-based smoother can be adapted to weighted data and missing values.

4. Dealing with weighted data and occurrence of missing values

The occurrence of missing data due to measurement infeasibility or instrumentation failure is frequent in practice. It can also be convenient to give outliers a low weight or, on the contrary, allocate a relatively high weight to high-quality data. Let W be the diagonal matrix $\text{diag}(w_i)$ that contains the weights $w_i \in [0, 1]$ corresponding to the data y_i . In the presence of weighted data, the RSS becomes

$$\text{wRSS} = \|W^{1/2}(\hat{y} - y)\|^2.$$

The minimization of $F(\hat{y})$ (Eq. (2)) thus yields

$$(W + sD^T D) \hat{y} = Wy,$$

which can be rewritten as

$$(I_n + sD^T D) \hat{y} = (I_n - W) \hat{y} + Wy.$$

Letting $A \equiv sD^T D + W$ and using the above-mentioned hat matrix H , this equation can also be expressed as

$$H^{-1} \hat{y} = (H^{-1} - A) \hat{y} + Wy.$$

This implicit formula can be solved using an iterative procedure:

$$H^{-1} \hat{y}_{[k+1]} = (H^{-1} - A) \hat{y}_{[k]} + Wy \quad \text{with an arbitrary } \hat{y}_{[0]}, \quad (14)$$

where $\hat{y}_{[k]}$ refers to \hat{y} calculated at the k th iteration step. It can be shown that this equation converges, for any $\hat{y}_{[0]}$, since the linear system is positive definite. Indeed, the matrix D is real nonsingular; then $sD^T D$ is positive definite if $s > 0$. In addition, W is positive semidefinite (because $w_i \geq 0$); A is thus positive definite and, according to the Theorem 3 of Keller (1965), convergence of $\hat{y}_{[k]}$ towards the desired solution \hat{y} is ensured for any $s > 0$.

Note that, in the presence of missing values, W is simply defined by $w_i = 0$ if y_i is missing, while an arbitrary finite value is assigned to y_i . In that case, the algorithm performs both smoothing and interpolation. Contrarily to classical methods which usually operate with linear or cubic local interpolations, missing data are assigned to values that are estimated using the entire dataset.

In the case of evenly spaced data, one can take advantage of the DCT by rewriting Eq. (14) as

$$\hat{y}_{[k+1]} = (I_n + sD^T D)^{-1} (W(y - \hat{y}_{[k]}) + \hat{y}_{[k]}),$$

which, similarly to Eq. (11), becomes

$$\hat{y}_{[k+1]} = \text{IDCT}(\Gamma \text{DCT}(W(y - \hat{y}_{[k]}) + \hat{y}_{[k]})), \quad (15)$$

with Γ given by Eq. (10). The iterative convergent process defined by Eq. (15), with the additional determination of the GCV score, allows automatic smoothing with weighted and/or missing data. When weighted and/or missing data occur, Eq. (7) describing the GCV score, however, becomes invalid since the weighted residuals (wRSS) must be accounted for. Moreover, the number of missing data must be taken into account in the expression of the average square error. In the presence of weighted data, the GCV score is therefore given by

$$\text{GCV}(s) = \frac{\text{wRSS}/(n - n_{\text{miss}})}{(1 - \text{Tr}(H)/n)^2} = \frac{\|W^{1/2}(\hat{y} - y)\|^2/(n - n_{\text{miss}})}{(1 - \text{Tr}(H)/n)^2},$$

where n_{miss} represents the number of missing data and n is the number of elements in y . Note that the GCV score given by the latter equation, in comparison with Eq. (13), requires \hat{y} to be known, so \hat{y} must be calculated at each iteration of the GCV minimization. Smoothing of weighted data will thus involve additional computation time.

5. Robust smoothing

In regression analysis, it is habitually assumed that the residuals follow a normal distribution with mean zero and constant variance, usually unknown. In spite of that, faulty data, erroneous measurements or instrumentation malfunction may lead to observations that lie abnormally far from the others. The main drawback of the penalized least squares methods is their sensitivity to these outliers. Besides outlying points, high leverage points also adversely affect the outcome of regression models. In statistics, the leverage denotes a measure of the influence (between 0 and 1) of a given point on a fitting model due to its location in the space of the inputs. More precisely, the points which are far removed from the main body of points will have high leverage (Chatterjee and Hadi, 1986). To minimize or cancel the side effects of high leverage points and outliers, one can assign a low weight to them using an iteratively reweighted process, as often used in robust local regression. This method consists in constructing weights with a specified weighting function by using the current residuals and updating them, from iteration to iteration, until the residuals remain unchanged (Rousseeuw and Leroy, 1987). In practice, five iterative steps are sufficient.

Several weighting functions are available for robust regression. The most common one, the so-called bisquare weight function, will be used hereinafter, although other weight functions would be suitable as well (Heiberger and Becker, 1992). The bisquare weights are given by

$$w_i = \begin{cases} \left(1 - \left(\frac{u_i}{4.685}\right)^2\right)^2 & \text{if } \left|\frac{u_i}{4.685}\right| < 1 \\ 0 & \text{if } \left|\frac{u_i}{4.685}\right| \geq 1, \end{cases}$$

where u_i is the Studentized residual which is adjusted for standard deviation and leverage and is defined as (Rousseeuw and Leroy, 1987)

$$u_i = \frac{r_i}{\hat{\sigma} \sqrt{1 - h_i}}. \quad (16)$$

In Eq. (16), $r_i = y_i - \hat{y}_i$ is the residual of the i th observation, h_i is its corresponding leverage and $\hat{\sigma}$ is a robust estimate for the standard deviation of the residuals given by 1.4826 MAD, where MAD denotes the median absolute deviation (Rousseeuw and Croux, 1993). The leverage values h_i are all given by the diagonal elements of the hat matrix H (Hoaglin and Welsch, 1978). However, a faster and economical alternative for robust smoothing can be obtained using an average leverage:

$$\forall i, \quad h_i = h = \frac{1}{n} \sum_i H_{ii} = \text{Tr}(H)/n,$$

where an approximated value for $\text{Tr}(H)/n$ is given by Eq. (12). The approximated Studentized residuals finally reduce to

$$u_i = r_i \left[1.4826 \text{ MAD}(r) \sqrt{1 - \frac{\sqrt{1 + \sqrt{1 + 16s}}}{\sqrt{2}\sqrt{1 + 16s}}} \right]^{-1}.$$

The use of the bisquare weights in combination with these approximated Studentized residuals provides a robust version of the above-mentioned smoother. Note that if y_i represents a missing data, r_i is not involved in the calculation of the $\text{MAD}(r)$. Appendix B contains a simplified Matlab code for robust smoothing using the method described in this paper.

6. Multidimensional smoothing of evenly gridded data

Because a multidimensional DCT is basically a composition of one-dimensional DCTs along each dimension (Strang, 1999), Eq. (11) can be immediately extended to two-dimensional or N -dimensional regularly gridded data. A detailed description of the extension of Eq. (11) to higher dimensions is given in Buckley (1994). In the case of N -dimensional data, Eq. (11) becomes

$$\hat{y} = \text{IDCTN}(\Gamma^N \circ \text{DCTN}(y)), \quad (17)$$

where DCTN and IDCTN refer to the N -dimensional discrete cosine transform and inverse discrete cosine transform, respectively, and \circ stands for the Schur (elementwise) product. As an extension of Eq. (10), Γ^N is a tensor of rank N defined by

$$\Gamma^N = 1^N \div (1^N + s \Lambda^N \circ \Lambda^N). \quad (18)$$

Here, the operator \div symbolizes the element-by-element division and 1^N is an N -rank tensor of ones. Λ^N , similarly to Eq. (8), is the following N -rank tensor (Buckley, 1994):

$$\Lambda_{i_1, \dots, i_N}^N = \sum_{j=1}^N \left(-2 + 2 \cos \frac{(i_j - 1)\pi}{n_j} \right), \quad (19)$$

```

1. Initialize:  $\hat{y} = \mathbf{0}$ 
2. Do until  $\hat{y}$  converges
    (a) Calculate  $DCT\hat{y} = \text{dctn}(W \circ (y - \hat{y}) + \hat{y})$  according to (20)
    (b) Find the smoothing parameter  $s$  that minimizes the GCV score, and
        determine the corresponding tensor  $\Gamma^N$  defined by (18) and (19)
    (c) Update  $\hat{y}$ :  $\hat{y} = \text{idctn}(\Gamma^N \circ DCT\hat{y})$  according to (20)
3. Go to step 5 if no robust option is required
4. For k from 1 to 5 (i.e. do five times)
    (a) Calculate the residuals  $(y - \hat{y})$ 
    (b) Calculate the bisquare weights  $Wbs$ 
    (c) Do until  $\hat{y}$  converges:
        i. Calculate  $DCT\hat{y} = \text{dctn}(W \circ Wbs \circ (y - \hat{y}) + \hat{y})$  according to (20)
        ii. If  $k = 1$ , find the smoothing parameter  $s$  that minimizes the GCV score,
            and determine the corresponding tensor  $\Gamma^N$  defined by (18) and (19)
        iii. Update  $\hat{y}$ :  $\hat{y} = \text{idctn}(\Gamma^N \circ DCT\hat{y})$  according to (20)
5. Return  $\hat{y}$ 

```

Fig. 1. Pseudocode of the automatic robust smoothing algorithm. The variable names and acronyms are described in the text. The complete Matlab code is supplied in the supplemental material (*smoothn.m*). When robust option is required, minimization of the GCV score is performed at the first robust iterative step only (see 4.c.ii) and the last estimated smoothness parameter (s) is used during the successive steps. This makes the algorithm faster without altering the final results significantly.

where n_j denotes the size of Λ^N along the j th dimension. Automated determination of s requires the GCV score which, in the case of multidimensional data, can be written as

$$GCVs = \frac{\|\hat{y} - y\|_F^2 / n}{(1 - \|\Gamma^N\|_1 / n)^2},$$

where $\|\cdot\|_F$ denotes the Frobenius norm, $\|\cdot\|_1$ is the 1-norm, and $n = \prod_{j=1}^N n_j$ is the number of elements in y . Similarly to Eq. (7), the GCV score with no weighted and no missing data can be rewritten as follows:

$$GCVs = \frac{n \left\| (\Gamma^N - \mathbf{1}^N) \circ \text{DCTN}(y) \right\|_F^2}{(n - \|\Gamma^N\|_1)^2}.$$

The same procedure as described in Section 5 can also be directly applied to weighted or missing data. The iterative process illustrated by Eq. (15) thus becomes

$$\hat{y}_{\{k+1\}} = \text{IDCTN}(\Gamma^N \circ \text{DCTN}(W \circ (y - \hat{y}_{\{k\}}) + \hat{y}_{\{k\}})). \quad (20)$$

A robust version can also be immediately obtained using the Studentized residuals given by (16) with the following average leverage:

$$h = \left(\frac{\sqrt{1 + \sqrt{1 + 16s}}}{\sqrt{2}\sqrt{1 + 16s}} \right)^N.$$

7. Matlab codes and examples

Eqs. (11) and (15) with optimal smoothing by means of the GCV method are relatively easy to implement in Matlab. Simplified Matlab codes for automatic smoothing (*smooth*) and robust smoothing (*rsmooth*) of one-dimensional and two-dimensional datasets are given in [Appendices A and B](#). These two programs have a similar syntax and both require the two-dimensional (inverse) discrete cosine transform (*dct2* and *idct2*) provided by the Matlab image processing toolbox. The input represents the dataset to be smoothed, while the output represents the smoothed data. A complete documented Matlab function for smoothing of one-dimensional to N -dimensional data (*smoothn*), with automated and robust options, and which can deal with weighted and/or missing data, is also supplied in the supplemental material. The pseudocode for *smoothn* is given in [Fig. 1](#). No Matlab toolbox is needed for the use of *smoothn*. Updated versions of this function are downloadable from the author's personal website ([Garcia, 2009](#)). The first four examples illustrated hereinafter can be run in Matlab using the function *smoothdemo*, also available in the supplemental material.

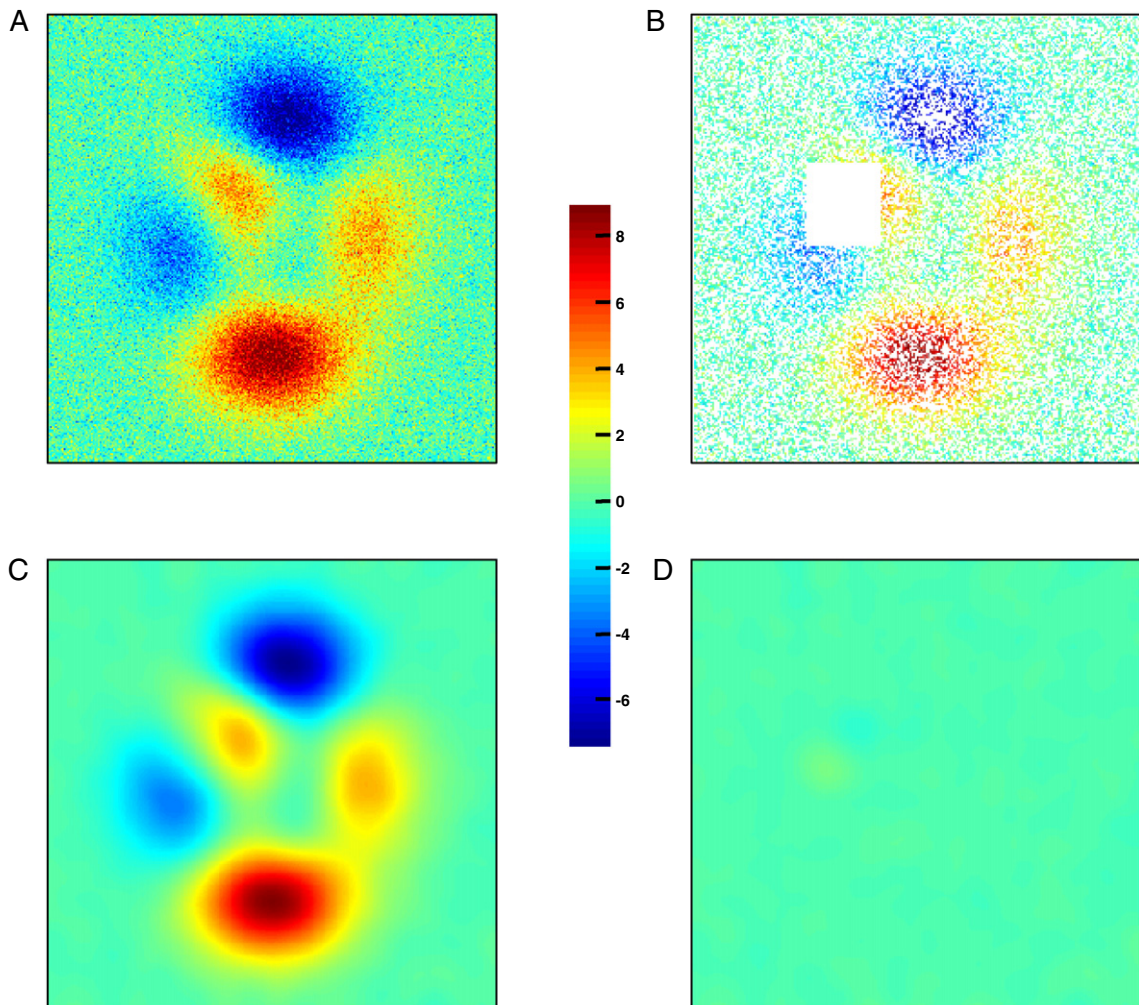


Fig. 2. Automatic smoothing of two-dimensional data with missing values. (A). Noisy data. (B). Corrupted data with missing values. (C). Smoothed data restored from B. (D). Absolute errors between the restored and original data.

Two-dimensional smoothing with missing values. To illustrate the effectiveness of the DCT-based smoothing algorithm, highly corrupted two-dimensional data have been smoothed and the result has been compared with the original data. The original matrix data (300×300) consisted of a function of two variables obtained by translating and scaling Gaussian distributions created by the Matlab “peaks” function. The corrupted data were obtained by first adding a Gaussian noise with mean zero (Fig. 2A), then randomly removing half of the data, and finally creating a 50×50 square of missing data (Fig. 2B). The automatic smoother was able satisfactorily to recover the original surface (Fig. 2C) with an average relative error (defined by $\|\text{smoothed} - \text{original}\| / \|\text{original}\|$) smaller than 5% (see also the absolute error mapping on Fig. 2D).

Robust smoothing. Penalized least squares methods are very sensitive to outlying data. Fig. 3A demonstrates how outliers may adversely bend the smoothed curve (see also the one-dimensional example in Appendix B). Clearly, the smoothed values do not reflect the behavior of the bulk of the data points. To overcome the distortion due to a small fraction of outliers, the data can be smoothed using the robust procedure described in Section 5. Fig. 3B shows that the robust smoothing is resistant to outliers. An example for two-dimensional robust smoothing is also given in Appendix B.

Three-dimensional smoothing. Fig. 4 now illustrates three-dimensional automated smoothing by means of the *smoothn* program supplied in the supplementary material. Gaussian noise with mean zero and standard deviation of 0.06 has been added to a three-dimensional function defined by $f(x, y, z) = x \exp(-x^2 - y^2 - z^2)$ in the $[-2; 2]^3$ domain (Fig. 4A). Fig. 4B shows that *smoothn* significantly reduced the additive noise. Note that *smoothn* could also be applied to higher dimensions.

Smoothing of parametric curves. The discrete cosine transform (DCT) and inverse DCT are linear operators and thus work with complex numbers. This property can be used to smooth parametric curves defined by $(x(t), y(t))$, where t is equispaced. By way of example, Fig. 5 depicts a noisy cardioid (dots) whose parametric equations are $x(t) = 2 \cos(t)[1 - \cos(t)] + \mathcal{N}(0, 0.2)$ and $y(t) = 2 \sin(t)[1 - \cos(t)] + \mathcal{N}(0, 0.2)$, where \mathcal{N} refers to the normal distribution and t is equally spaced between 0 and 2π . Let us now define the complex function $z(t) = x(t) + iy(t)$, where i is the unit complex number, and its smoothed

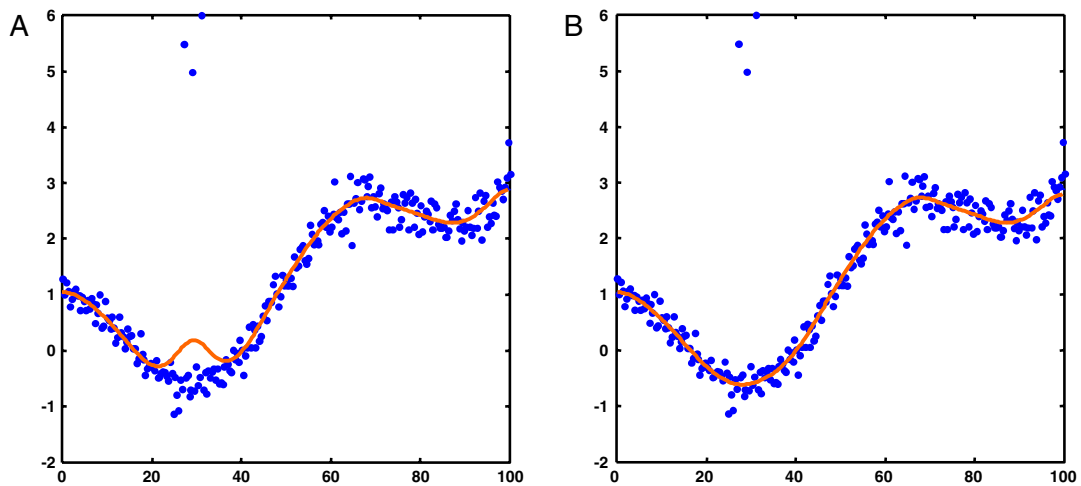


Fig. 3. Non-robust versus robust smoothing. Outliers may bend the smoothed curve (A). A robust smoothing may get rid of this drawback (B).

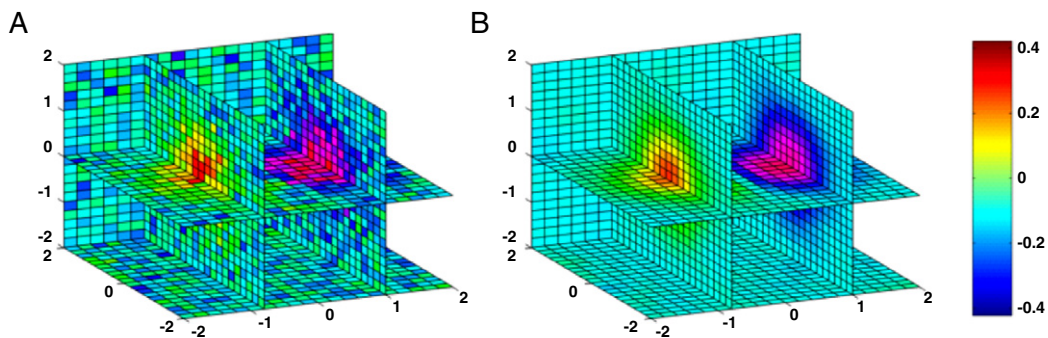


Fig. 4. Automatic smoothing of three-dimensional data.

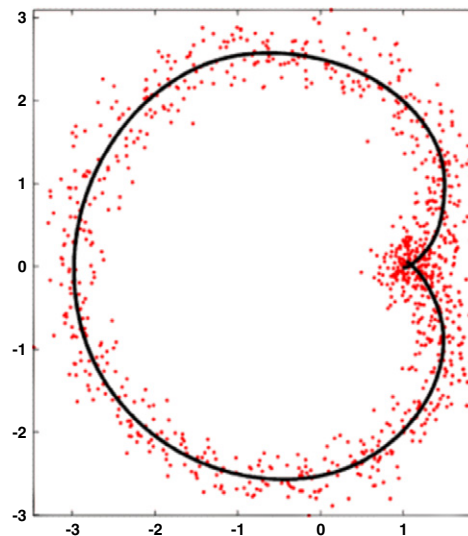


Fig. 5. Automatic smoothing of a noisy cardioid.

output $\hat{z}(t)$. The parametric curve defined by $(\text{Re}(\hat{z}(t)), \text{Im}(\hat{z}(t)))$, where Re and Im refer to the real and imaginary parts, respectively, provides a smooth cardioid (solid line; see Fig. 5).

Applications to surface temperature anomalies. Automatic smoothing with the minimization of the GCV score has finally been tested on temperature data available in the Met Office Hadley Centre's website (Brohan et al., 2006; Kennedy, 2007). Fig. 6 illustrates the evolution of global average land temperature anomaly (in °C) with respect to 1961–1990. The smooth

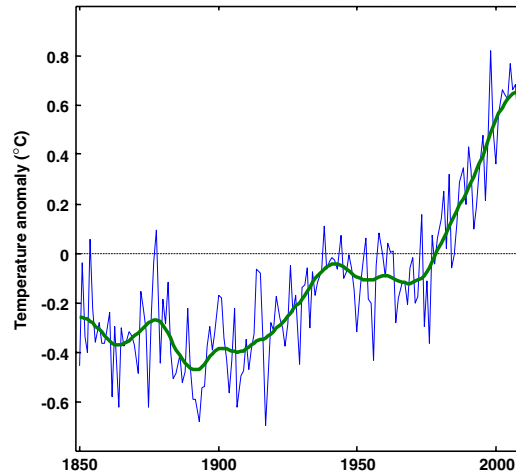


Fig. 6. Global average land temperature anomaly (°C) with respect to 1961–1990: smoothed versus original year-averaged data. Year-averaged data are available in <http://hadobs.metoffice.com/crutem3/diagnostics/global/nh+sh/annual>.

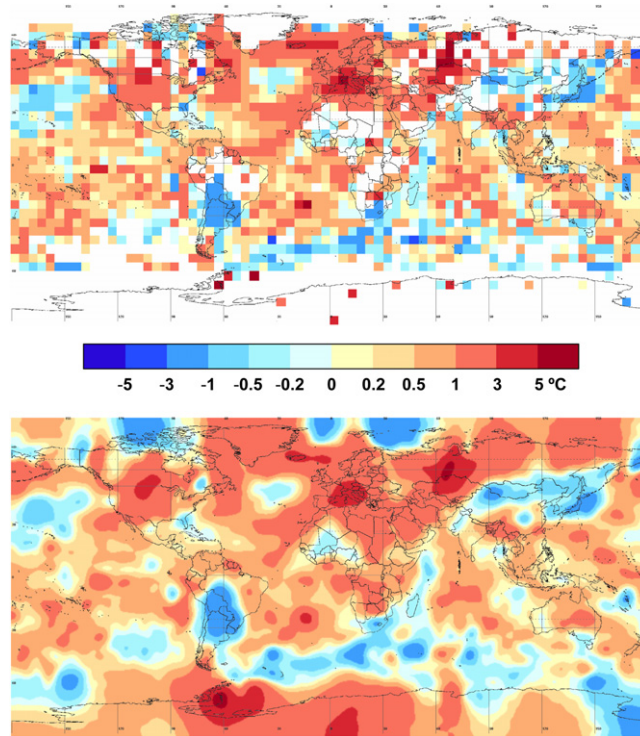


Fig. 7. August 2003 surface temperature difference (°C) with respect to 1961–1990: smoothed (bottom panel) versus original year-averaged (top panel) data. The year-averaged dataset is available in <http://hadobs.metoffice.com/hadcrut3/index.html>. The smoothed output has been upsampled for a better visualization of the surface temperatures.

curve clearly depicts the continuing rise of global temperature that has occurred since the 1970's. Fig. 7 shows the Earth's surface temperature difference of August 2003 with respect to 1961–1990. The top panel represents a mapping of the best estimates of temperature anomalies as provided by the Met Office Hadley Centre. Note that numerous temperature data are missing due to nonexistent station measurements. A smooth map of Earth temperature anomalies with filled gaps can be automatically obtained using the DCT-based algorithm described in the present paper (Fig. 7, bottom panel).

The resolution of Fig. 7, bottom panel, has been increased to obtain a better visualization of surface temperatures. A basic upsampling of the smoothed data was performed by padding zeros after the last element of DCTN (\hat{y}) along each dimension before using the inverse DCT operator, as follows:

$$\hat{y}_{\text{upsampled}} = \sqrt{\frac{n_{\text{upsampled}}}{n}} \text{IDCTN} \left(\text{ZeroPad} \left(\text{DCTN} \left(\hat{y} \right) \right) \right),$$

where $n_{\text{upsampled}}$ represents the number of elements after upsampling. It is worth recalling that the DCT assumes repeating boundary conditions whereas periodic boundaries would have been more appropriate for the east–west edges. Such conditions caused east–west discordances at the level of New Zealand and Antarctica.

8. Discussion

The best choice of the smoothing algorithm to be used in data analysis depends specifically upon the original data and on the properties of the additive noise. As a consequence, the search for a perfect universal smoother remains illusory (Eilers, 2003). In this paper, a robust, fast and fully automated smoothing procedure has been proposed, and its efficacy has been illustrated on a few cases in the previous section. The user, however, should be aware of some limitations of the algorithm. Better results will be obtained if \hat{y} itself is sufficiently smooth, i.e. if it has continuous derivatives up to some desired order over the whole domain of interest. Moreover, because the algorithm process involves a discrete cosine transform, the inherent repeating conditions may slightly distort the smoothed output on the boundaries, especially when the total number of data is small. The GCV criterion is used to make the smoothing procedure fully automated and good results are expected if the additive noise ε in Eq. (1) follows a Gaussian distribution with nil mean and constant variance. It has been reported, however, that the GCV remains fairly adapted even with nonhomogenous variances and non-Gaussian errors (Wahba, 1990b). Alternatively, if the measurement errors in the data present variable standard deviations and if the variances are known (σ_i^2), the response variances can be transformed to a constant value using weights given by $w_i = 1/\sigma_i^2$. In addition, if outlying data occur, it could be convenient to perform a robust smoothing. Finally, the GCV criterion may cause problems when the sample size is small, which could make the automated version unsatisfactory with a small number of data. If this occurs, the smoothing parameter s must be tuned manually until a visually acceptable result is obtained.

Appendix A

A simplified Matlab code (*smooth*) for one-dimensional (1-D) and two-dimensional (2-D) smoothing of equally gridded data, and two examples are given below. The input represents the dataset to be smoothed and the output represents the smoothed data. This code has been written with Matlab R2007b.

```
function z = smooth(y)
[n1,n2] = size(y); n = n1*n2;
Lambda = bsxfun(@plus, repmat(-2+2*cos((0:n2-1)*pi/n2), n1, 1), ...
    -2+2*cos((0:n1-1).*pi/n1));
DCTy = dct2(y);
fminbnd(@GCVscore, -15, 38);
z = idct2(Gamma.*DCTy);
function GCVs = GCVscore(p)
    s = 10^p;
    Gamma = 1./(1+s*Lambda.^2);
    RSS = norm((DCTy(:).*(Gamma(:)-1)))^2;
    TrH = sum(Gamma(:));
    GCVs = RSS/n/(1-TrH/n)^2;
end
end
```

1-D Example:

```
x = linspace(0,100,2^8);
y = cos(x/10)+(x/50).^2 + randn(size(x))/10;
z = smooth(y);
plot(x,y,'.',x,z)
```

2-D Example:

```
xp = linspace(0,1,2^6);
[x,y] = meshgrid(xp);
f = exp(x+y)+sin((x-2*y)*3) + randn(size(x))/2;
g = smooth(f);
figure, subplot(121), surf(xp,xp,f), zlim([0 8])
subplot(122), surf(xp,xp,g), zlim([0 8])
```

Appendix B

A simplified Matlab code (*rsmooth*) for robust smoothing of one-dimensional (1-D) and two-dimensional (2-D) equally sampled data, and two examples are given below. The syntax is similar to that of *smooth* (see Appendix A). A complete

optimized code (*smoothn*) allowing smoothing in one or more dimensions is also available in the supplemental material and in the author's personal website (Garcia, 2009).

```
function z = rsmooth(y)
[n1,n2] = size(y); n = n1*n2;
N = sum([n1,n2]~=1);
Lambda = bsxfun(@plus, repmat(-2+2*cos((0:n2-1)*pi/n2), n1, 1), ...
    -2+2*cos((0:n1-1).'*pi/n1));
W = ones(n1,n2);
zz = y;
for k = 1:6
    tol = Inf;
    while tol>1e-5
        DCTy = dct2(W.*(y-zz)+zz);
        fminbnd(@GCVscore, -15, 38);
        tol = norm(zz(:)-z(:))/norm(z(:));
        zz = z;
    end
    tmp = sqrt(1+16*s);
    h = (sqrt(1+tmp)/sqrt(2)/tmp)^N;
    W = bisquare(y-z, h);
end
function GCVs = GCVscore(p)
    s = 10^p;
    Gamma = 1./(1+s*Lambda.^2);
    z = idct2(Gamma.*DCTy);
    RSS = norm(sqrt(W(:)).*(y(:)-z(:)))^2;
    TrH = sum(Gamma(:));
    GCVs = RSS/n/(1-TrH/n)^2;
end
end
function W = bisquare(r, h)
    MAD = median(abs(r(:)-median(r(:)))));
    u = abs(r/(1.4826*MAD)/sqrt(1-h));
    W = (1-(u/4.685).^2).^2.*(u/4.685<1);
end
```

1-D Example:

```
x = linspace(0,100,2^8);
y = cos(x/10)+(x/50).^2 + randn(size(x))/10;
y([70 75 80]) = [5.5 5 6];
z = smooth(y);
zr = rsmooth(y);
subplot(121), plot(x,y, '.', x, z), title('Non robust')
subplot(122), plot(x,y, '.', x, zr), title('Robust')
```

2-D Example:

```
xp = linspace(0,1,2^6);
[x,y] = meshgrid(xp);
f = exp(x+y)+sin((x-2*y)*3) + randn(size(x))/2;
f(20:3:35,20:3:35) = randn(6,6)*10+20;
g = smooth(f);
gr = rsmooth(f);
subplot(121), surf(xp,xp,g), zlim([0 8]), title('Non robust')
subplot(122), surf(xp,xp,gr), zlim([0 8]), title('Robust')
```

Appendix C. Supplemental data

The supplemental material contains four Matlab programs (*smoothn*, *smoothndemo*, *dctn* and *idctn*). The function *smoothn* includes all the properties described in this paper. It carries out manual or automatic smoothing of one-dimensional

to N -dimensional uniformly sampled data, and can deal with weighted and/or missing values. A robust option is also offered. Enter “*help smoothn*” in the Matlab command window to obtain a detailed description and the syntax for *smoothn*. Updated versions of *smoothn* are also downloadable from the author's personal website (Garcia, 2009). The second program *smoothndemo* illustrates the first four examples from Section 7. Enter “*smoothndemo*” in the Matlab command window to create four Matlab figures corresponding to the Figs. 2–5 of this paper. The script of *smoothndemo* can be of help for a better understanding of *smoothn*. Finally, *dctn* and *idctn* allow the computation of the discrete cosine transform (DCT) and inverse DCT of N -dimensional arrays. These two functions are necessary for the use of *smoothn*. The functions *smoothn*, *smoothndemo*, *dctn* and *idctn* have been written with Matlab R2007b.

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.csda.2009.09.020.

References

- Brohan, P., Kennedy, J.J., Harris, I., Tett, S.F.B., Jones, P.D., 2006. Uncertainty estimates in regional and global observed temperature changes: A new data set from 1850. *Journal of Geophysical Research* 111, D12106. doi:10.1029/2005JD006548.
- Buckley, M.J., 1994. Fast computation of a discretized thin-plate smoothing spline for image data. *Biometrika* 81, 247–258.
- Chatterjee, S., Hadi, A.S., 1986. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science* 1, 379–393.
- Craven, P., Wahba, G., 1978. Smoothing noisy data with spline functions. Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik* 31, 377–403.
- Eilers, P.H., 2003. A perfect smoother. *Anal. Chem.* 75, 3631–3636.
- Garcia, D., Matlab functions in BioméCardio. <http://www.biomecardio.com/matlab>, 2009. Ref Type: Electronic Citation.
- Golub, G., Heath, M., Wahba, G., 1979. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics* 21, 215–223.
- Hastie, T., Loader, C., 1993. Local regression: Automatic kernel carpentry. *Statistical Science* 8, 120–129.
- Heiberger, R.M., Becker, R.A., 1992. Design of an S function for robust regression using iteratively reweighted least squares. *Journal of Computational and Graphical Statistics* 1, 181–196.
- Hoaglin, D.C., Welsch, R.E., 1978. The hat matrix in regression and ANOVA. *The American Statistician* 32, 17–22.
- Keller, H.B., 1965. On the solution of singular and semidefinite linear systems by iteration. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* 2, 281–290.
- Kennedy, J., Met Office Hadley Centre observations datasets. <http://hadobs.metoffice.com/crutem3>, 2007. Ref Type: Electronic Citation.
- Rousseeuw, P.J., Leroy, A.M., 1987. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., New York, NY, USA.
- Rousseeuw, P.J., Croux, C., 1993. Alternatives to the median absolute deviation. *Journal of the American Statistical Association* 88, 1273–1283.
- Savitzky, A., Golay, M.J.E., 1964. Smoothing and differentiation of data by simplified least squares procedures. *Anal. Chem.* 36, 1627–1639.
- Schoenberg, I.J., 1964. Spline functions and the problem of graduation. *Proceedings of the National Academy of Sciences of the United States of America* 52, 947–950.
- Strang, G., 1999. The discrete cosine transform. *SIAM Review* 41, 135–147.
- Takezawa, K., 2005. *Introduction to Nonparametric Regression*. John Wiley & Sons, Inc., Hoboken, NJ.
- Wahba, G., 1990a. *Spline Models for Observational Data*. Society for Industrial Mathematics, Philadelphia.
- Wahba, G., 1990b. Estimating the smoothing parameter. In: *Spline Models for Observational Data*. Society for Industrial Mathematics, Philadelphia, pp. 45–65.
- Watson, G.S., 1964. Smooth regression analysis. *The Indian Journal of Statistics, Series A* 26, 359–372.
- Weinert, H.L., 2007. Efficient computation for Whittaker–Henderson smoothing. *Computational Statistics & Data Analysis* 52, 959–974.
- Whittaker, E.T., 1923. On a new method of graduation. *Proceedings of the Edinburgh Mathematical Society* 41, 62–75.
- Yueh, W.C., 2005. Eigenvalues of several tridiagonal matrices. *Applied Mathematics E-Notes* 5, 66–74.