Comput Stat (2013) 28:1989–2005
DOI 10.1007/s00180-012-0390-5

ORIGINAL PAPER

Revisiting fitting monotone polynomials to data

Kevin Murray · Samuel Müller ·
Berwin A. Turlach

Received: 26 April 2012 / Accepted: 27 November 2012 / Published online: 23 December 2012

Abstract We revisit Hawkins' (Comput Stat 9(3):233–247, 1994) algorithm for fitting monotonic polynomials and discuss some practical issues that we encountered using this algorithm, for example when fitting high degree polynomials or situations with a sparse design matrix but multiple observations per x-value. As an alternative, we describe a new approach to fitting monotone polynomials to data, based on different characterisations of monotone polynomials and using a Levenberg–Marquardt type algorithm. We consider different parameterisations, examine effective starting values for the non-linear algorithms, and discuss some limitations. We illustrate our methodology with examples of simulated and real world data. All algorithms discussed in this paper are available in the R Development Core Team (A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, 2011) package `MonoPoly`.

Keywords Monotone polynomial · Monotone regression

K. Murray · S. Müller
School of Mathematics and Statistics, University of Sydney,
Carslaw Building (F07), Sydney, NSW 2006, Australia

S. Müller
e-mail: samuel.mueller@sydney.edu.au

K. Murray · B. A. Turlach
School of Mathematics and Statistics (M019), University of Western Australia,
35 Stirling Highway, Crawley, WA 6009, Australia

B. A. Turlach
e-mail: berwin.turlach@gmail.com

K. Murray (✉) · B. A. Turlach
Centre for Applied Statistics (M019), University of Western Australia,
35 Stirling Highway, Crawley, WA 6009, Australia
e-mail: kevin.murray@uwa.edu.au

1 Introduction

In many regression settings it is known, e.g. from some underlying physical or economic theory, that the regression curve is monotone. Further examples include, but are not limited to, calibration problems, estimation of monotone transformations (e.g. to transform a variable to normality), growth curves, and dose-response curves. While many fully parametric monotone regression models exist, in particular in the growth curve literature, such models may miss important features such as the number and location of roots, extrema and inflection points of (higher order) derivatives of a regression function.

Typical approaches for combining shape constraints with flexible regression techniques are the use of either spline smoothing techniques or kernel smoothing techniques. While incorporating shape constraints into spline smoothing has been well studied (see, among others, the introductory sections of Turlach (2005) or Hazelton and Turlach (2011) for reviews of the existing literature), kernel smoothing techniques are somewhat less often used for more general shape constraints (Marron et al. 1997; Mammen et al. 2001), but there is some notable work on monotone kernel smoothers (Friedman and Tibshirani 1984; Mammen 1991; Hall and Huang 2001; Dette et al. 2006; Dette and Pilz 2006).

Most approaches that impose monotonicity on nonparametric smoothing techniques can be problematic due to the estimated regression curve having flat stretches, and monotone regression "has been criticized because practitioners do not believe in all those flat spots" (Dette et al. 2006). Having to contend with estimated regression functions with many (spurious) flat spots is highly undesirable, especially in situations where the estimation of derivatives (and features thereof) is important, as these would translate to increased variability of such estimates. While the approaches of Ramsay (1998), Dette et al. (2006) and Hazelton and Turlach (2011) lead to estimated regression functions that are strictly monotone, the functional form of the estimated regression function does not lend itself readily to a subsequent analysis to aspects of its derivatives.

Given the many useful properties of monotone polynomials e.g. that they are naturally strictly monotone and have easily identifiable derivatives, we revisit the fitting of monotone polynomials to data. Our initial motivation was to improve upon the methodology used in Firmin et al. (2011, 2012) who fitted a monotone decreasing function into the measured motor evoked potentials (TST amplitude) as a function of stimulation delay on more than 40 different data sets, one from each participating patient in their study. Firmin et al. (2011, 2012) used smoothing splines via the smooth.monotone function which is part of the R-package fda (Ramsay et al. 2012) and extracted those inflection points (roots of the second derivative) from the fitted curve that relate to local maxima of the first derivative. These estimated modes are of relevance in Neuroscience. The purpose of this article is not to demonstrate that using monotone polynomials in this context outperforms "full-blown" nonparametric smoothing techniques, which it does and which will be published in a separate article, but to present an improved fitting algorithm for monotone polynomials.

Recently, fitting monotone polynomials to data has been considered in a Bayesian framework by Curtis and Ghosh (2011), and previously in a frequentist setting by Hawkins (1994), whose algorithm we revisit. While Hawkins' algorithm is fast and effective, we identify situations in which it cannot be used. For these situations, and for use in general, we consider several parameterisations of monotone polynomials, and show how the best fitting monotone polynomial, using any of these parameterisations, can be fitted to data using a Levenberg–Marquardt modified Newton–Raphson algorithm. Our experience with Hawkins' (1994) algorithm, the various parameterisations of monotone polynomials and our proposed optimisation algorithms are described in Sect. 2. In Sect. 3 we describe the results from our numerical experiments, and provide conclusions in Sect. 4.

2 Methodology

The usual parameterisation for a polynomial regression function is

$$p(x) = p(x; \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_q x^q \tag{1}$$

where q is the degree of the polynomial and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_q)^T$, with the β_js being the regression parameters in the linear regression model

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_q x^q + \epsilon. \tag{2}$$

Note that such a polynomial can be monotone only if its degree $q = 2K + 1$ is odd. To fit a polynomial regression curve to given data (x_i, y_i), $i = 1, \ldots, n$, one usually minimises the residual sum of squares (RSS), which under the assumption of Gaussian errors in (2) achieves the same as maximising the likelihood. That is, the fit is determined by minimising

$$\mathrm{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - p(x_i))^2. \tag{3}$$

Note that this objective function is strictly convex in $\boldsymbol{\beta}$ if the number of distinct x-values exceeds q. Moreover, the set of monotone increasing polynomials, $\mathcal{I} = \{p(x) : \forall x, p'(x) \geq 0\}$, and the set of monotone decreasing polynomials, $\mathcal{D} = \{p(x) : \forall x, p'(x) \leq 0\}$, are both convex, closed sets. Thus, minimising (3) over either $\mathcal{I}$ or $\mathcal{D}$ is a convex optimisation problem, with a unique solution, which should be easy to solve. However, while it would be relatively easy to find starting values in $\mathcal{I}$ or $\mathcal{D}$, e.g. by fitting a simple linear regression and setting $\beta_j = 0$ for $j \geq 2$ or fitting the function $\beta_0 + \beta_q x^q$ to the data, finding descent directions is extremely difficult at best.

2.1 Hawkins' approach

Hawkins (1994) shows that the solution to minimising $\mathrm{RSS}(\boldsymbol{\beta})$ subject to $p \in \mathcal{I}$ must be a polynomial with $p'(x; \boldsymbol{\beta}) > 0$ for all but a finite number of points, say, x_m^* with $p'(x_m^*; \boldsymbol{\beta}) = p''(x_m^*; \boldsymbol{\beta}) = 0, m = 1, \ldots M$ (with $M = 0$ a possibility), which he calls "hips" (short for "horizontal inflection points"). This insight shows that the optimal monotonic (increasing) polynomial is the saddle point of the Lagrangian

$$\mathrm{RSS}(\boldsymbol{\beta}) - \sum_{m=1}^{M} \lambda_m p'(x_m^*; \boldsymbol{\beta}).$$

Thus, if M and the location of the hips were known, then the optimal monotonic polynomial is given by the solution to the quadratic program

$$\text{minimise}_{\boldsymbol{\beta}}\ \mathrm{RSS}(\boldsymbol{\beta}) \quad \text{subject to} \quad p'(x_m^*; \boldsymbol{\beta}) = 0, \quad m = 1, \ldots, M. \tag{4}$$

This leads Hawkins (1994) to propose the following algorithm:

1. Set $M = 0$ and solve the unconstrained problem minimise$_{\boldsymbol{\beta}}$ RSS$(\boldsymbol{\beta})$.
2. Check whether the first derivative $p'(x, \boldsymbol{\beta})$ of the fitted polynomial is non-negative everywhere. If it is, then the fitted polynomial is monotonic already, and therefore solves the problem.
   If the resulting polynomial is not monotonic however, then it will be necessary to find an additional candidate hip at which to force the polynomial's slope to zero. To do this,
3. Increase M by 1, adding an additional candidate hip and forcing a zero derivative there. Solve the resulting equality-constrained problem.
   The additional constraint at the new candidate hip may make one of the existing constraints redundant, a fact that will be detected by its Lagrange multiplier λ_m becoming negative. If this occurs, remove that candidate hip from the active set. Repeat the algorithm from step 2.

To complete the description of his algorithm, Hawkins (1994) suggests (with slight modifications to adapt to the notation used here):

> The test for monotonicity can be made quickly and efficiently, finding the minimum of $p'(x)$ by solving $p''(x) = 0$ [...] which is easily done using standard packages for manipulating polynomials. The test for monotonicity is made by evaluating $p'(x)$ at each real root of $[p''(x)]$—by definition the x values at which $p'(x)$ takes on its most extreme values. If any of these derivatives is negative the polynomial is not monotonic. The converse is not true; $p'(x)$ may have positive extreme values but be negative for all sufficiently large $|x|$; for this reason while the primary test for monotonicity is based on the values of $p'(x)$ at the roots of $p''(x) = 0$, we supplement this test with a direct evaluation of $p'(x)$ at some relative extreme value of x.
> Where the current polynomial is not monotonic, the algorithm needs to select an additional candidate hip. A good choice is the x value at which the largest negative slope occurs, a value which is the byproduct of the suggested method for checking monotonicity. This value is then used as the candidate additional hip in step 3.

and describes how (4) can be solved for a given set of hips.

In attempting to implement this algorithm, we frequently experienced two or more Lagrange multipliers becoming negative in step 3. It appears that one should still drop only one constraint during this step, which may require a judicious selection. More problematic were situations in which two, or more, of the Lagrange multipliers became close to zero, i.e. having an absolute value smaller than typically employed tolerance levels for numerical calculations. In such situations it is difficult to determine whether the corresponding constraints are inactive and should be dropped, and if so, in which order. However, not dropping such constraints leads to an increasing set of candidate hips and (4) being solved with an increasing number of equality constraints which may create numerical problems.

Thus, we choose to implement step 3 by using the R package `quadprog` (Turlach and Weingessel 2011) which implements the quadratic programming algorithm of Goldfarb and Idnani (1982, 1983). In doing so, we replaced the equality constraints in (4) by inequality constraints ($p'(x^*_m; \boldsymbol{\beta}) \geq 0$, $m = 1, \ldots, M$) and removed all candidate hips for which the Lagrangian parameter was non-positive at the solution. Finally, to cope with different levels of numerical precision for the polynomial root finder (function `polyroot()` in R), the evaluation of polynomials and the quadratic programming solver, we found it necessary to declare a polynomial to be monotone if $p'(x) > -\varepsilon$ at the roots of $p''(x)$, where ε can be chosen to be in the order of 10^{-7} to 10^{-12}, and $p'(x) > 0$ at some relative extreme value of x. This correlates with Hawkins' (1994) warning that "Care is needed [...] since [...] it is not always trivial to decide whether a polynomial is really non-monotonic, or just seems so because of roundoff noise". The choice of ε obviously influences the number of iterations of the algorithm, but we found the algorithm to be fast and reliable for values of ε in the indicated range.

We note that for numerical stability a QR factorisation of the design matrix $D = \left(x_i^j\right)_{\substack{i=1,\ldots,n \\ j=0,\ldots,q}}$ should be used as recommended by Hawkins (1994). However, it becomes increasingly difficult to calculate a QR factorisation of the design matrix as q increases, irrespective of the x-design. The reason is that the approach of Hawkins uses unorthogonalised polynomials which pose severe numerical challenges at higher degrees and the calculations may fail with default settings of numerical routines. Consequently judicious choice of numerical tolerances becomes necessary.

While Hawkins (1994) surmises that "Polynomials of degree 1, 3, 5, 7 and 9 were fitted—[...] to test [...] the method when using a polynomial of degree far above what most people would normally consider using for data modeling", the application that motivated our research, namely the search for inflection points, may necessitate the use of polynomials of (much) higher degrees. With the methods described in the subsequent sections we are able to fit much higher degree monotone polynomials to data than can be achieved using Hawkins' approach.

Furthermore, in situations with a sparse design but multiple observations per x-value, such as our example data $W2$ below in Sect. 3, one might want to fit a monotone polynomial of order q in a situation where one has only q, or even slightly less, unique x values. In such circumstances the algorithm of Hawkins (1994) will fail as it starts off at the unconstrained solution.

These two drawbacks, which limits the usefulness of Hawkins' approach in our application of interest, leads to the consideration of other approaches for fitting monotone polynomials which are based on isotonic parameterisations.

2.2 Isotonic parameterisations

As an alternative approach to overcome the problem that (3) cannot be easily minimised, over $\mathcal{I}$ or $\mathcal{D}$, when the parameterisation (1) is used, we consider here several parameterisations of isotonic polynomials. The first parameterisation is due to Elphinstone (1983) who realised that a polynomial of degree $q = 2K + 1$ is isotonic if, and only if, all real roots of its derivative have an even multiplicity. This insight leads to the following parameterisation

$$p_1(x) = d + a \int_0^x \prod_{j=1}^{K} \left\{ \left(c_j^2 + b_j^2 \right) + 2b_j t + t^2 \right\} dt \tag{5}$$

where a, d, b_j and c_j, $j = 1, \ldots, K$, are arbitrary real values; the sign of a determines whether the polynomial is monotone increasing or monotone decreasing. In this parameterisation $a/(2K + 1)$ is the coefficient of the x^q term, and the roots of the derivative of $p_1(x)$ are given by $-b_j \pm c_j\,\iota$, where $\iota = \sqrt{-1}$.

Elphinstone (1983) also proposed the following parameterisation, subsequently mentioned by Hawkins (1994) and used by Heinzmann (2008),

$$p_2(x) = d + a \int_0^x \prod_{j=1}^{K} \left\{ 1 + 2b_j t + \left(b_j^2 + c_j^2 \right) t^2 \right\} dt \tag{6}$$

and motivated it by observing that "when stepping from $K - 1$ to K the starting values [...] would be the solution which [minimise] the criterion for $K - 1$, and the two new parameters [...] would be set to zero". We shall comment on this observation in Sect. 2.2.2, but note for now that unlike parameterisation (5) this parameterisation cannot be used for all isotonic polynomials, for example $p(x) = x^q$ cannot be represented in the form (6). The reason being that the roots of the derivative of $p_2(x)$ are given by $-b_j/(b_j^2 + c_j^2) \pm c_j/(b_j^2 + c_j^2)\,\iota$. As in (5) the sign of a determines whether $p_2(x)$ lies in $\mathcal{I}$ or $\mathcal{D}$.

The final parameterisation that we consider, proposed by Penttila (2006), is a modification of (6) and, to the best of our knowledge, is published here for the first time:

$$p_3(x) = d + a \int_0^x \prod_{j=1}^{K} \left\{ b_j^2 + 2b_j t + \left(1 + c_j^2 \right) t^2 \right\} dt. \tag{7}$$

The roots of the derivative of $p_3(x)$ are given by $-b_j/(1 + c_j^2) \pm b_j c_j/(1 + c_j^2)\,\iota$, which shows that this parameterisation can also not represent all isotonic polynomials;

the derivative of $p_3(x)$ cannot have roots of the form $0 \pm e\,\iota$ with $e \neq 0$. Again the sign of a determines whether $p_3(x)$ is monotonic increasing or decreasing. It should be noted that with parameterisations (5) and (7) the fitted polynomial will either be constant or of degree q, while for (6) and for Hawkins' (1994) method, described in Sect. 2.1, the fitted polynomial can potentially have degree less than q.

For all three parameterisations we denote the vector of parameters generically by $\boldsymbol{\theta} = (d, a, b_1, c_1, \ldots, b_K, c_K)^T = \left(d, a, \tilde{\boldsymbol{\theta}}^T\right)^T$. Note that all these parameterisations have the form

$$p(x) = d + a\tilde{p}(x) = d + a\tilde{p}(x; \tilde{\boldsymbol{\theta}}) \tag{8}$$

where $\tilde{p}(x)$ depends only on the b_js and c_js. Finally, for the remainder of the paper, it will be clear from the context whether we regard the residual sum of squares as a function of $\boldsymbol{\beta}$ or as a function of $\boldsymbol{\theta}$, and we avoid the subscript of p.

2.2.1 Evaluating the objective function and its derivatives

To evaluate the RSS using any of the parameterisations in Sect. 2.2 we first calculate the $\boldsymbol{\beta}$ which corresponds to the given $\boldsymbol{\theta}$. This allows an easy evaluation of $p(x)$ for arbitrary values of x, using for example the Horner scheme for numerical stability (Fausett 2003), and to use (3) for the calculation of the RSS.

We illustrate the necessary calculations using parameterisation (5), the other two parameterisations can be handled analogously. To calculate $\boldsymbol{\beta}$ for a given $\boldsymbol{\theta}$, we first build triples $(c_j^2 + b_j^2, 2b_j, 1)$ which are the coefficients for the quadratic functions appearing in (5). By convoluting these triplets, we can calculate the coefficients $\boldsymbol{\gamma} = (\gamma_0, \ldots, \gamma_{2K})^T$ of the polynomial

$$\gamma_0 + \gamma_1 t + \cdots + \gamma_{2K} t^{2K} = \prod_{j=1}^{K} \left\{ \left(c_j^2 + b_j^2\right) + 2b_j t + t^2 \right\}.$$

From $\boldsymbol{\gamma}$ we can readily calculate $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta} = \left(d, a\gamma_0, a\frac{\gamma_1}{2}, \ldots, a\frac{\gamma_{2K}}{2K+1}\right)^T.$$

Note that $\left(0, \gamma_0, \frac{\gamma_1}{2}, \ldots, \frac{\gamma_{2K}}{2K+1}\right)^T$ is the vector that contains the coefficient of the polynomial $\tilde{p}(x)$ in (8).

To minimise RSS numerically we also need first and second derivatives for a derivative based optimisation algorithm. These derivatives can easily be calculated in a similar manner. From (8) we have

$$\frac{\partial}{\partial d}\text{RSS} = -2\sum_{i=1}^{n}(y - p(x_i)) \tag{9a}$$

$$\frac{\partial}{\partial a}\text{RSS} = -2\sum_{i=1}^{n}(y - p(x_i))\tilde{p}(x_i) \tag{9b}$$

$$\frac{\partial}{\partial \tilde{\theta}_k}\text{RSS} = -2a\sum_{i=1}^{n}(y - p(x_i))\frac{\partial}{\partial \tilde{\theta}_k}\tilde{p}(x_i) \tag{9c}$$

where $\tilde{\theta}_k$ is a component of the vector $\tilde{\boldsymbol{\theta}}$, i.e. one of the b_js or c_js. Previously we have discussed how $\tilde{p}(x_i)$ can be evaluated once $\boldsymbol{\gamma}$ is determined. The partial derivatives $\frac{\partial}{\partial \tilde{\theta}_k}\tilde{p}(x_i)$ can be evaluated similarly. For example, if $\tilde{\theta}_k$ is b_{j_0}, then we build the triples $(c_j^2 + b_j^2, 2b_j, 1)$ for $j \neq j_0$ and the triple $(2b_{j_0}, 2, 0)$. After convoluting these K triples, the coefficients of the polynomial $\frac{\partial}{\partial \tilde{\theta}_k}\tilde{p}(\cdot)$ can be readily calculated. Similarly, if $\tilde{\theta}_k$ is c_{j_0}, then we build the triples $(c_j^2 + b_j^2, 2b_j, 1)$ for $j \neq j_0$ and the triple $(2c_{j_0}, 0, 0)$. After convoluting these K triples, the coefficients of the polynomial $\frac{\partial}{\partial \tilde{\theta}_k}\tilde{p}(\cdot)$ can be readily calculated.

Tedious but straightforward calculus allows the use of (9) to find formulae for the second partial derivatives of RSS, i.e. of $\frac{\partial^2}{\partial \theta_k\, \partial \theta_l}\text{RSS}$. Some of these second derivatives involve second derivatives of the polynomial $\tilde{p}(x)$. Again, the coefficients of $\frac{\partial^2}{\partial \theta_k\, \partial \theta_l}\tilde{p}(x; \tilde{\boldsymbol{\theta}})$ can be determined by convoluting appropriately constructed triples. Though if either θ_l or θ_k is either a or d, then, trivially, $\frac{\partial^2}{\partial \theta_k\, \partial \theta_l}\tilde{p}(x; \tilde{\boldsymbol{\theta}}) \equiv 0$. Also note that $\frac{\partial^2}{\partial b_j \partial c_j}\tilde{p}(x; \tilde{\boldsymbol{\theta}}) \equiv 0$, $j = 1, \ldots, K$.

2.2.2 Some remarks on the isotonic parameterisations

Earlier we noted that the RSS is strictly convex in $\boldsymbol{\beta}$ if the regressor variable has sufficiently many distinct values. However, RSS as a function of $\boldsymbol{\theta}$ is no longer convex. It is easy to see that RSS is a quartic polynomial in the b_js and c_js and this leads to potential non-convexity and the possibility of local extrema in RSS$(\boldsymbol{\theta})$.

In fact, note that for each of the parameterisations considered in Sect. 2.2 the polynomial $\tilde{p}(x; \tilde{\boldsymbol{\theta}})$ depends on the c_js only through c_j^2. Generalising an observation by Heinzmann (2008), we could replace in all the parameterisations considered here c_j^2 by $\tau(c_j)$, where $\tau(z)$ is a function with $\tau(0) = 0$ and whose range is the non-negative reals. If additionally $\tau(z)$ is differentiable then $\tau'(0) = 0$ since zero is a global minimum of $\tau(z)$. With this generalisation, (9c) implies

$$\frac{\partial}{\partial c_j}\text{RSS} = -2a\sum_{i=1}^{n}(y - p(x_i))\left(\frac{\partial}{\partial c_j}\tilde{p}(x_i)\right)\tau'(c_j). \tag{10}$$

Moreover, if θ_k is a component of $\boldsymbol{\theta}$ different from c_j then

$$\frac{\partial^2}{\partial c_j\,\partial\theta_k}\text{RSS} = -2a\sum_{i=1}^{n}\left\{(y - p(x_i))\left(\frac{\partial^2}{\partial c_j\,\partial\theta_k}\tilde{p}(x_i)\right)\right.$$
$$\left. - \left(\frac{\partial}{\partial\theta_k}p(x_i)\right)\left(\frac{\partial}{\partial c_j}\tilde{p}(x_i)\right)\right\}\tau'(c_j). \tag{11}$$

These results have undesirable consequences for derivative based optimisation algorithms such as the Newton–Raphson algorithm and its variants. If during the iterative optimisation a c_j becomes zero, (10) implies that the corresponding entry in the gradient of RSS will be zero. More seriously, (11) implies that in the Hessian matrix all elements in the row and column corresponding to c_j equal zero, with the possible exception of the diagonal element. This, in turn, implies that the step direction calculated from this Hessian matrix for the next iteration will have a zero in the component corresponding to c_j. In other words, if during the iterations a c_j becomes zero, it will remain so.

Similar results hold for smooth loss functions other than the RSS. Thus, we discourage following the proposition in Elphinstone (1983), i.e. to change from (5) to (6) so that when stepping from $K-1$ to K the values which minimise the criterion for $K-1$ can be used as starting values with the two new parameters set to zero. One of these new parameters would be c_K. Setting c_K to zero when a derivative based optimisation routine is used ensures that c_K remains at zero. Admittedly, Elphinstone (1983) does not discuss the optimisation algorithm that he uses. By way of contrast, Heinzmann (2008) appears to use a Levenberg–Marquardt modification of the Newton–Raphson algorithm, i.e. a second-derivative based algorithm, but does not discuss the starting values that he uses. We note that neither of these authors discussed the problem with local extrema in the parameterisation of isotonic polynomials and that c_js can get trapped at zero.

In this paper we also propose to use a Levenberg–Marquardt modification of the Newton–Raphson algorithm, described in the next section. Our numerical experiments showed that all three parameterisations of isotonic polynomials indeed suffer under the problem of local extrema when one, or more, of the c_js become fixed at zero. Thus, we also investigate replacing c_j^2 in each of the parameterisations by c_j and optimising the RSS under the constraints $c_j \geq 0$, $j = 1, \ldots, K$.

2.2.3 Algorithms for optimising the objective function

We considered various approaches for minimising RSS($\boldsymbol{\theta}$). Initial experiments using derivative free methods (Nelder and Mead 1965; Powell 2009; Bates et al. 2011) showed that such algorithms struggle with minimising RSS($\boldsymbol{\theta}$). Coordinate descent algorithms that were recently applied quite successfully to other computational statistics problems (see, among others, Friedman et al. 2007, 2010) proved to be somewhat more successful but required a large number of iterations, and long run-times, to achieve convergence to the optimiser of RSS($\boldsymbol{\theta}$). Here we describe a Levenberg–Marquardt modification to the Newton–Raphson algorithm, which differs from the one described by Heinzmann (2008) and is more in the spirit of Osborne (1976), that proved to be effective for minimising RSS($\boldsymbol{\theta}$).

Iterations In the following description ΔRSS($\boldsymbol{\theta}$) denotes the gradient vector of RSS and $H(\boldsymbol{\theta})$ the Hessian matrix, calculated as outlined in Sect. 2.2.1. The algorithm that we propose to use for optimising RSS($\boldsymbol{\theta}$) is the following:

```
 1: Set $\lambda = 0.1$, $t = 0$ and initialise $\boldsymbol{\theta}^{(t)}$
 2: **repeat**
 3:   Calculate $\text{RSS}_t = \text{RSS}\left(\boldsymbol{\theta}^{(t)}\right)$, $\boldsymbol{g}_t = \Delta\text{RSS}\left(\boldsymbol{\theta}^{(t)}\right)$ and $\boldsymbol{H}_t = H\left(\boldsymbol{\theta}^{(t)}\right)$
 4:   Set $\boldsymbol{H}_D$ to be a diagonal matrix with $i^{\text{th}}$ diagonal entry being $h_{ii}$, if
      $h_{ii} > 0$, and 1 otherwise; where $h_{ii}$ is the $i^{\text{th}}$ diagonal entry of $\boldsymbol{H}_t$
 5:   Set $l = 1$
 6:   **loop**
 7:     Calculate $\boldsymbol{\theta}^c = \boldsymbol{\theta}^{(t)} - (\boldsymbol{H}_t + \lambda\boldsymbol{H}_D)^{-1}\boldsymbol{g}_t$
 8:     Calculate $\text{RSS}_c = \text{RSS}(\boldsymbol{\theta}^c)$
 9:     **if** $\text{RSS}_c \leq \text{RSS}_t$ **then**
10:       **if** $l = 1$ **then**
11:         Set $\lambda = \lambda/10$
12:       **end if**
13:       Set $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^c$
14:       Set $t = t + 1$ and exit from loop
15:     **else**
16:       Set $\lambda = 10 * \lambda$ and $l = l + 1$
17:       **if** $\lambda > 10^{10}$ **then**
18:         Terminate algorithm with failure to converge
19:       **end if**
20:     **end if**
21:   **end loop**
22: **until** convergence criteria are met
```

Note that $\boldsymbol{H}_D$ in step 4 is always positive definite and will typically contain information on the curvature of RSS with respect to each component of $\boldsymbol{\theta}$. Thus, for large λ, the algorithm essentially takes steps according to a steepest descent algorithm where the components of the gradient vector are rescaled by the curvature. Such steps are typically taken when the current value of $\boldsymbol{\theta}$ is far away from the optimal value and, hence, $H(\boldsymbol{\theta})$ may be neither positive definite nor useful for determining a descent direction. The typical scenario, as the iterates approach a (local) minimum, is that each proposal $\boldsymbol{\theta}^c$ is immediately accepted ($l = 1$), λ becomes very small, and the algorithm turns into a Newton–Raphson algorithm.

In the variation of the parameterisations that use c_j instead of c_j^2, the algorithm is much the same. Except in step 7 the proposal is calculated as $\boldsymbol{\theta}^c = \boldsymbol{\theta}^{(t)} + \boldsymbol{\delta}^{(t)}$ where $\boldsymbol{\delta}^{(t)}$ is determined by solving the following quadratic program:

$$\text{minimise}_{\delta}\ \frac{1}{2}\boldsymbol{\delta}^T(\boldsymbol{H}_t + \lambda\boldsymbol{H}_D)\boldsymbol{\delta} + \boldsymbol{g}_t^T\boldsymbol{\delta}$$

where the imposed constraints are such that all entries in $\boldsymbol{\theta}^c$ that correspond to any of the c_js will be non-negative.

To complete the description of the algorithm we discuss in the following sections how $\boldsymbol{\theta}^{(0)}$ is initialised and the stopping criteria.

Starting values During our numerical experiments, discussed in more detail in Sect. 3, the following procedure seemed to yield a reliable and satisfactory initialisation of $\boldsymbol{\theta}^{(0)}$. For any parameterisation, we begin by setting the b_js to zero. Next, for (5) and (6) the c_js are initialised to $c_j = 0.1 + (j-1)\frac{0.9}{K-1}$, $j = 1, \ldots, K$, and for (7) to $c_j = 1 + (j-1)\frac{1}{K-1}$, $j = 1, \ldots, K$. Finally, based on (8), a and d are determined by regressing the responses y_i on the predictor $\tilde{p}(x_i)$, which completes the initialisation of $\boldsymbol{\theta}$.

Stopping criteria We considered several criteria for stopping the algorithm. Namely monitoring the change in $\boldsymbol{\beta}$, monitoring the change in RSS, and monitoring the entries of the gradient vector ΔRSS. Our numerical work showed that RSS as a function of $\boldsymbol{\theta}$ is typically relatively flat in a neighbourhood of the optimal solution. Hence monitoring changes in $\boldsymbol{\beta}$ or RSS, and basing a stopping criteria on either absolute or relative changes in either of these quantities being small, leads in many situations to a premature termination of the optimisation algorithm. Based on our numerical experiments, the only reliable way to determine convergence is to monitor the gradient vector ΔRSS, with a suitable default stopping criteria being that the absolute value of each entry in ΔRSS is smaller than 10^{-5}. Of course, in the parameterisations that employ c_j instead of c_j^2 and optimise RSS under the constraints that $c_j \geq 0$, $j = 1, \ldots, K$, the entries in ΔRSS that corresponds to c_js that are zero are not required to be smaller than 10^{-5}.

3 Numerical experiments and results

To assess the effectiveness of our methodology, and to determine which combinations of parameterisations and formulations provide stable results, we carried out numerical experiments using four datasets. We used the simulated data ($n = 50$) published in Hawkins (1994), growth data with $n = 31$ measurements for the first male individual from the Berkeley Growth Study described in Ramsay and Silverman (2002), and two simulated data sets from the regression models (12) and (13), respectively:

$$W0: \quad Y_i = p(x_i) + \epsilon_i = 0.1x_i^3 + \epsilon_i \tag{12}$$

where $x_i = -1 + \frac{i-1}{10}$ and $\epsilon_i \overset{i.i.d}{\sim} N(0, 0.01^2)$, $i = 1, \ldots, 21$; and

$$W2: \quad Y_{ij} = 4\pi - x_i + \cos(x_i - \pi/2) + \epsilon_{ij} \tag{13}$$

where $x_i = i - 1$, $i = 1, \ldots, 13$, and $\epsilon_{ij} \overset{i.i.d.}{\sim} N(0, 0.5^2)$ with $j = 1, \ldots, n_i$ and all $n_i = 3$ except for $n_1 = 5$, for a sample size of $n = 41$. We denote these four data sets

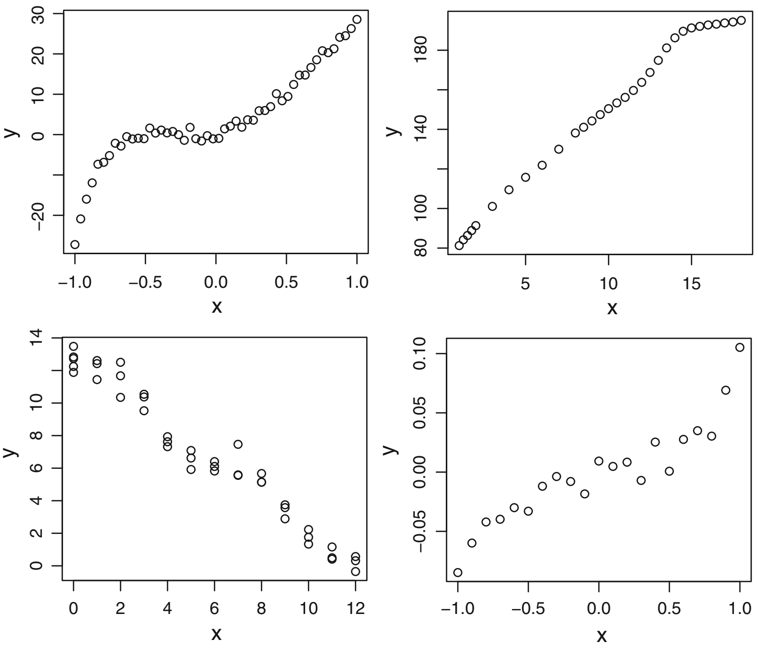

Fig. 1 Data sets used for numerical experiments. Clockwise from *top left*: Hawkins, RS, $W0$ and $W2$

for the remainder of this paper as $Hawkins$, RS, $W0$ and $W2$, respectively, and show scatterplots in Fig. 1. Note the latter of these was specifically generated to mimic our motivating problem described in the introduction, and is typical of any of the datasets in Firmin et al. (2011, 2012).

Figure 1 shows that the ranges for the x and y values are quite different for the various examples. Our numerical experiments indicate that it is beneficial, for numerical stability, to re-scale all x and y values to have minimum and maximum of -1 and 1, respectively, before starting the iterative optimisation process described in Sect. 2.2.3.

Using the methodology described in the previous section, we fitted monotone polynomials up to degree 9 to each data set, for the three different parameterisations and both the constrained and unconstrained formulation. These results are described in Table 1, which also includes results using Hawkins' (1994) approach.

During our numerical experiments we noticed that when using the c^2 formulations, the RSS indeed appears to have local extrema. This is consistent with our comments in the previous section regarding second derivative based algorithms. For example, when fitting a degree 9 polynomial to the RS data, using (6) with c_j^2, we note that all c_j estimates become zero early in the iterative process and are subsequently trapped at that value. For the same data and degree polynomial with the constrained formulation, the same phenomenon again occurs early in the iterative process, and all c_js become zero. However, subsequently they depart from zero and the process continues giving a much lower objective function value on termination of the algorithm.

Table 1 Objective function value RSS/n × 10, 000, by parameterisation for the four data sets

	Using (5)		Using (6)		Using (7)		Hawkins' approach
	c^2	$c \geq 0$	c^2	$c \geq 0$	c^2	$c \geq 0$	
Data 1 (Hawkins)							
K=1	83.16	83.16	83.16	83.16	83.16	83.16	83.16
K=2	16.34	16.34	17.44	16.34	16.34	16.34	16.34
K=3	11.55	10.79	11.55	10.79	11.55	10.79	10.79
K=4	10.77	10.77	11.15	10.77	10.77	10.77	10.77
Data 2 (RS)							
K=1	34.75	34.75	34.75	34.75	34.75	34.75	34.75
K=2	13.83	13.83	34.98	13.83	34.98	13.83	13.83
K=3	5.86	4.05	35.06	4.05	4.05	4.05	4.05
K=4	5.88[a]	4.04	26.18	4.04	4.04	4.04[a]	4.04
Data 3 ($W0$)							
K=1	105.52	105.52	105.52	105.52	105.52	105.52	105.52
K=2	77.52	77.52	77.52	77.52	77.52	77.52	77.52
K=3	73.82	73.82	77.14	73.82	75.05[a]	73.82	73.82
K=4	73.76	73.75	73.75	73.75	73.83[a]	73.75	73.75
Data 4 ($W2$)							
K=1	130.93	130.93	130.93	130.93	130.93	130.93	130.93
K=2	129.57	129.57	129.57	129.57	129.57	129.57	129.57
K=3	58.37	57.35	58.37	57.35	58.37	57.35	57.35
K=4	56.51	56.51	56.53	56.51	58.41	57.35[a]	56.51

[a] Algorithm did not converge

For all parameterisations we have $d = \beta_0$ and for (5) and (6), $a \propto \beta_q$ and $a \propto \beta_1$ respectively. Thus, for these two parameterisations a is identifiable, and only the remaining β_js are functions of a, the b_js and the c_js. By way of contrast, for the parameterisation (7) all β_js ($j \neq 0$) are functions of a, the b_js and the c_js which makes none of the latter parameters identifiable. This can lead to additional problems when (7) is used as we observed in our numerical experiments. In these instances we observed that $a \to 0$ and at least one of the $(b_j, c_j) \to \infty$, hence at least partially cancelling each other's effect on $\boldsymbol{\beta}$. Such problems were not as apparent for the other two parameterisations.

Specifically we also note the following: The performance (i.e. lower objective function values in smaller number of iterations) when using (7) is inferior to that of (5) and (6). When comparing the performance of the latter two parameterisations there is little difference when using the constrained formulation. However, (5) seems to perform better when the unconstrained formulation is used. Given the demonstrated problems with local minima with the c^2 parameterisation, we would ordinarily not recommend its use. However, if one does use it we have found it beneficial to perturb any c_j that was zero at the solution and restart the algorithm with these perturbed

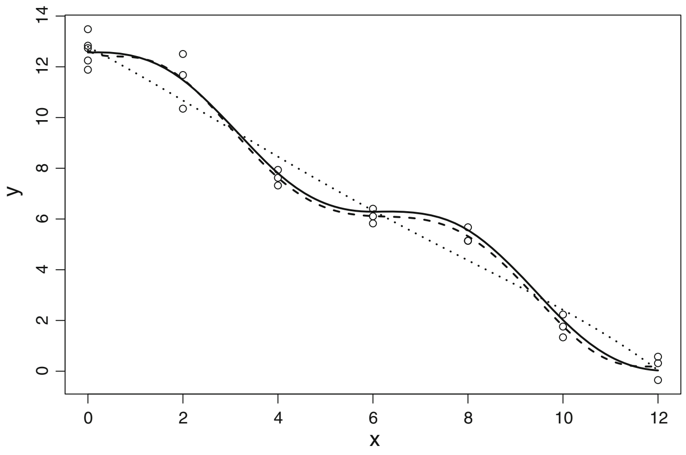

Fig. 2 Monotone fits to subset of $W2$ data. *Solid line* is the underlying regression function while the *dotted* and *dashed lines* are the best fitting monotone polynomials of degree 5 and 7, respectively

estimates as starting values. This usually ensured that the objective function value after the restart was comparable with other parameterisations' performances.

3.1 Random starting values

As in other situations when an objective function with local extrema is optimised using a derivative based algorithm, it is recommended to start the optimisation process from several randomly chosen starting values. In our numerical experiments, for constrained parameterisations (5) and (6) we took 100 simulations using random starting values for the b_js and c_js by simulating from a Normal(0, 1) distribution, taking the absolute value for the constrained c_js. In all simulations the algorithm converged to the value shown in Table 1. However, the same amount of success was not seen when the y values were not rescaled initially.

3.2 Rank deficient designs

To further illustrate our methods, and to illustrate the point mentioned in 2.1 regarding situations with a sparse design but multiple observations per x-value, we subset our $W2$ data to retain only the even values of x. Using Hawkins' (1994) approach and the isotonic parameterisations described previously, we consider fitting monotone polynomials of degrees 5 and 7. The results are shown in Fig. 2. Here, the solid line shows the underlying regression function and the dotted line is the monotone polynomial of degree 5, which all isotonic parameterisations and Hawkins' method agree upon. The dashed line is the monotone polynomial of degree 7, which is consisently produced using our three isotonic parameterisations. However, due to the Hawkins' method starting at the unconstrained solution, fitting a degree 7 polynomial to this data is not achievable using his methodology.

We observe that Hawkins' approach starts at the unconstrained solution as he proposes to use the Goldfarb-Idnani (Goldfarb and Idnani 1982, 1983) quadratic programming algorithm iteratively to solve the semi-infinite quadratic programming problem

$$\text{minimise}_{\boldsymbol{\beta}}\, \text{RSS}(\boldsymbol{\beta}) \quad \text{subject to} \quad p'(x; \boldsymbol{\beta}) \geq 0, \forall x.$$

If his approach were substantially modified to use another quadratic programming problem solver it could possibly work for rank deficient designs.

4 Conclusions

In this paper, we have revisited Hawkins' (1994) algorithm for fitting monotone polynomials and discussed its implementation. We have identified two situations in which this algorithm becomes problematic, namely when q is large causing problems with the QR factorisation of the design matrix, or when q is large relative to the number of unique points in the design matrix. Both these situations are important for using monotone polynomials to detect the location and number of inflection points, our major motivation for this work. Consequently we have provided a new algorithm for fitting monotone polynomials to data using various formulations for isotonic polynomials, which overcomes these issues.

We have also identified some of the drawbacks of previous methodologies, that are not based on monotone polynomials, and conclude that our method, which has the advantage of being parametric, is an alternative to many of the existing smoothing spline based techniques.

Four numerical examples for fitting monotone polynomials up to degree nine with our algorithm were presented. However, in other investigations—not reported here—comparing our methodology to smoothing spline based approaches, we have fitted models up to degree 21, which indicates that in principle our methods are not degree constrained.

In comparing the parameterisations and formulations, we have shown that using the c^2 formulation as originally described in Elphinstone (1983), and subsequently used by other authors, produces converged optimisations to local extrema when a derivative based approach such as Newton–Raphson is used. This problem is observed in particular when the 'step up' approach is used but is not limited to this. It can also be problematic when one (or more) c_j parameter estimate becomes zero.

We have provided recommendations on the choice of sensible starting values and which of the formulations provide more consistent results. Implementations of all discussed algorithms are provided by the R package `MonoPoly`.

In summary, using our methodology, our results suggest that the best results for fitting monotone polynomials to data can be achieved by using any of the two parameterisations originally described in Elphinstone (1983) with a constrained formulation on the c_js and scaling of the bivariate data to the $[-1, 1] \times [-1, 1]$ square. Current versions of R-code (R Development Core Team 2011) to reproduce our results are available on request from the corresponding author.

Acknowledgments We would like to thank the referees and editors for their helpful comments and suggestions. Samuel Müller was supported by a grant from the Australian Research Council (DP110101998).

References

Bates D, Mullen KM, Nash JC, Varadhan R (2011) minqa: Derivative-free optimization algorithms by quadratic approximation. R package version 1.1.18/r530. URL http://R-Forge.R-project.org/projects/optimizer/

Curtis SM, Ghosh SK (2011) A variable selection approach to monotonic regression with bernstein polynomials. J Appl Stat 38:961–976

Dette H, Neumeyer N, Pilz KF (2006) A simple nonparametric estimator of a strictly monotone regression function. Bernoulli 12(3):469–490

Dette H, Pilz KF (2006) A comparative study of monotone nonparametric kernel estimates. J Stat Comput Simul 76(1):41–56

Elphinstone CD (1983) A target distribution model for non-parametric density estimation. Commun Stat Theory Methods 12(2):161–198

Fausett LV (2003) Numerical methods: algorithms and applications. Prentice Hall, Upper Saddle River

Firmin L, Müller S, Rösler KM (2011) A method to measure the distribution of latencies of motor evoked potentials in man. Clin Neurophysiol 122:176–182

Firmin L, Müller S, Rösler KM (2012) The latency distribution of motor evoked potentials in patients with multiple sclerosis. Clin Neurophysiol 123:2414–2421. doi:10.1016/j.clinph.2012.05.008

Friedman J, Hastie T, Höfling H, Tibshirani R (2007) Pathwise coordinate optimization. Ann Appl Stat 1(2):302–332

Friedman J, Hastie T, Tibshirani R (2010) Regularisation paths for generalized linear models via coordinate descent. J Stat Softw 33(1). URL http://www.jstatsoft.org/v33/i01/paper

Friedman JH, Tibshirani R (1984) The monotone smoothing of scatterplots. Technometrics 26(3):243–250

Goldfarb D, Idnani A (1982) Dual and Primal-Dual Methods for Solving Strictly Convex Quadratic Programs. In: Hennart JP (ed) Numerical analysis, Proceedings, Cocoyoc, Mexico 1981, vol 909., of lecture notes in mathematics. Springer, Berlin, pp 226–239

Goldfarb D, Idnani A (1983) A numerically stable dual method for solving strictly convex quadratic programs. Math Program 27:1–33

Hall P, Huang LS (2001) Nonparametric kernel regression subject to monotonicity constraints. Ann Stat 29(3):624–647

Hawkins DM (1994) Fitting monotonic polynomials to data. Comput Stat 9(3):233–247

Hazelton ML, Turlach BA (2011) Semiparametric regression with shape constrained penalized splines. Comput Stat Data Anal 55(10):2871–2879. doi:10.1016/j.csda.2011.04.018

Heinzmann D (2008) A filtered polynomial approach to density estimation. Comput Stat 23(3):343–360. doi:10.1007/s00180-007-0070-z

Mammen E (1991) Estimating a smooth monotone regression function. Ann Stat 19(2):724–740

Mammen E, Marron JS, Turlach BA, Wand MP (2001) A general projection framework for constrained smoothing. Stat Sci 16(3):232–248. doi:10.1214/ss/1009213727

Marron JS, Turlach BA, Wand MP (1997) Local polynomial smoothing under qualitative constraints. In: Billard L, Fisher NI (eds) Graph-image-vision, vol 28 of computing science and statistics. Interface Foundation of North America, Inc., Fairfax Station, 22039–7460, pp 647–652. URL http://www.maths.uwa.edu.au/berwin/psfiles/interface96.pdf

Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7(4):308–313. doi:10.1093/comjnl/7.4.308

Osborne MR (1976) Nonlinear least squares – the Levenberg algorithm revisited. J Aust Math Soc Ser B 19(3):343–357. doi:10.1017/S033427000000120X

Penttila TJ (2006) Personal communication.

Powell MJD (2009) The BOBYQA algorithm for bound constrained optimization without derivatives. Report No. DAMTP 2009/NA06, Centre for Mathematical Sciences, University of Cambridge, Cambridge. http://www.damtp.cam.ac.uk/user/na/NA_papers/NA2009_06.pdf

R Development Core Team (2011) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, ISBN 3-900051-07-0. URL http://www.R-project.org/

Ramsay JO (1998) Estimating smooth monotone functions. J R Stat Soc Ser B 60(2):365–375

Ramsay JO, Silverman BW (2002) Applied functional data analysis: methods and case studies. Springer, New York

Ramsay JO, Wickham H, Graves S, Hooker G (2012) FDA: functional data analysis. R package version 2.2.8. URL http://CRAN.R-project.org/package=fda

Turlach BA (2005) Shape constrained smoothing using smoothing splines. Comput Stat 20(1):81–103. doi:10.1007/BF02736124

Turlach BA, Weingessel A (2011) quadprog: Functions to solve quadratic programming problems. S original by Turlach BA, R port by Weingessel A; R package version 1.5–4