Supporting Material to: A Perfect Smoother

Paul H. C. Eilers
Department of Medical Statistics
Leiden University Medical Centre
P.O. Box 9604, 2300 RC Leiden, The Netherlands
e:mail: p.eilers@lumc.nl

1 Introduction

This document presents supporting material to my paper "A Perfect Smoother", that appeared in *Analytical Chemistry*, Volume 75, 2003. It contains the following:

- A section on how to handle arbitrarily spaced (in x) data.
- Additional comparisons of the Savitzky-Golay smoother and the Whittaker smoother.

2 Arbitrarily spaced data

Here we relax the (implicit) condition that the samples that make up the observed series y have equal distances. We will consider two situations. In the first, which might be called the scatterplot problem, we observe m pairs (x_i, y_i), the xs have no order and duplicate values of x may occur. In the second situation the xs are ordered, increase monotonically and hence no duplicate x occur.

2.1 Scatterplot smoothing

In the scatterplot case we divide the domain of x into a reasonable number, n, of intervals, say 50 or 100. We are going to compute a vector u, of length n, that gives the smooth trend curve to the scatterplot of pairs (x, y). Let the left boundary be of the x-domain be x_0 and and the interval width be b. Compute an interval index k for each x: $k_i = \lfloor (x_i - x_0)/b \rfloor + 1$. Construct an indicator matrix G with m rows and n columns. Each row of G has $n - 1$ zeros and one 1. In row i, the 1 will be in column k_i. Now we can write $\hat{y} = Gu$. We put a roughness penalty on u and minimize the following goal function

$$Q = |y - Gu|^2 + \lambda |Du|^2, \tag{1}$$

leading to the system of equations, of size n,

$$(G'G + \lambda D'D)u = G'y. \tag{2}$$

The essential part of the Matlab code is

```
k = floor((x - x0) / b);
G = sparse(1:m, k, 1);
D = diff(speye(n), d);
z = (G' * G + lambda * D' * D) \ (G' * y);
```

The extreme sparseness of G makes this very efficient, even for large m and n. The relatively small size of n makes it unnecessary to use the Cholesky decomposition when solving the equations.

Alternatively, we can use the fact that $G'G$ is diagonal, with g_{ii} being the number of observations in bin i, and $G'y$ a vector with element i being the sum of the elements of y having their x in bin i. This leads to:

```
k = floor((x - x0) / b);
w = sparse(k, 1, 1);
s = sparse(k, 1, y);
D = diff(speye(n), d);
z = (diag(w) + lambda * D' * D) \ s;
```

2.2 Non-uniform sampling

In the case of a series of observations (x_i, y_i) with monotonically increasing x, the scatterplot approach can be used too, especially if the goal is to estimate a rather smooth trend. If however we want an estimate $z_i = \hat{y}_i$ at each i, we need something else: divided differences [1]. First-order divided differences are defined as

$$g_{i1} = \frac{\Delta y_i}{\Delta x_i} = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{3}$$

If we let V be a $m-1$ by $m-1$ matrix with $1/\Delta x$ on its diagonal, the divided differences can be written as VDy. A penalized likelihood function can then be constructed as

$$Q = (y-z)'W(y-z) + \lambda |VDz|^2, \tag{4}$$

and minimizing Q leads to the system of equations

$$(W + D'V^2D)z = Wy. \tag{5}$$

The Matlab code is

```
V = spdiags(1 ./ diff(x), 0, m - 1, m - 1);
W = spdiags(w, 0, m, m);
D = V * diff(speye(m));
C = chol(W + lambda * D' * D);
z = C \ (C' \ (w .* y ));
```

Unfortunately we can not repeat this process mechanically (dividing differences of y by those of x) for higher orders. The correct formula for second order divided differences is

$$g_{i2} = \frac{g_{i1} - g_{i-1,1}}{x_i - x_{i-2}}. \tag{6}$$

In general we have the recursion

$$g_{i,d} = \frac{g_{i,d-1} - g_{i-1,d-1}}{x_i - x_{i-d}}. \tag{7}$$

This forms the basis of the following recursive Matlab function to compute a matrix D_d that constructs divided differences of order d of y by the multiplication Dy:

```
function D = ddmat(x, d)
m = length(x);
if d == 0
  D = speye(m);
else
  dx = x((d + 1):m) - x(1:(m - d));
  V = spdiags(1 ./ dx, 0, m - d, m - d);
  D = V * diff(ddmat(x, d - 1));
end
```

It might seem that weights are not necessary here, as we could simply leave out the missing pairs (x, y). But weights may come in handy when interpolating to a uniform grid, say u. First we attach u to x, to give a vector x^* and attach zeros to y, to give y^*. We also form a vector w which contains a 1 corresponding each element of x and a 0 corresponding to each element of u. Then we reorder x^*, y^* and w^* simultaneously such that x is increasing. After smoothing, to give z^* we select the elements of z^* that correspond to zero weights. These are the required smoothly interpolated values. To prevent accidental equality of some elements of x and u, it is advisable to "jitter" x by adding small random numbers to it.

3 Detailed comparison of SGS and WS

A referee suggested to contrast in more detail the Whittaker smoother (WS) and the Savitzky-Golay smoother (SGS). This will be done in this section. A straightforward implementation of the SGS can be found in the file `savgol.m`.

Speed On a 1000 MHz Pentium III computer, it takes less than 0.1 second to smooth the NMR spectrum (1024 channels) with the WS, for any value of λ. The SGS takes about 2 seconds. Surprisingly, this is hardly influenced by the size of the window. Apparently, the function `polyfit()` in Matlab is so fast that the explicit `for` loop over the data is the critical factor.

Smoothness The SGS gives only discrete control over smoothness, by varying the window width, an odd integer. This is not a problem with a relatively large window. When only light smoothing, and hence a small window, is needed, as

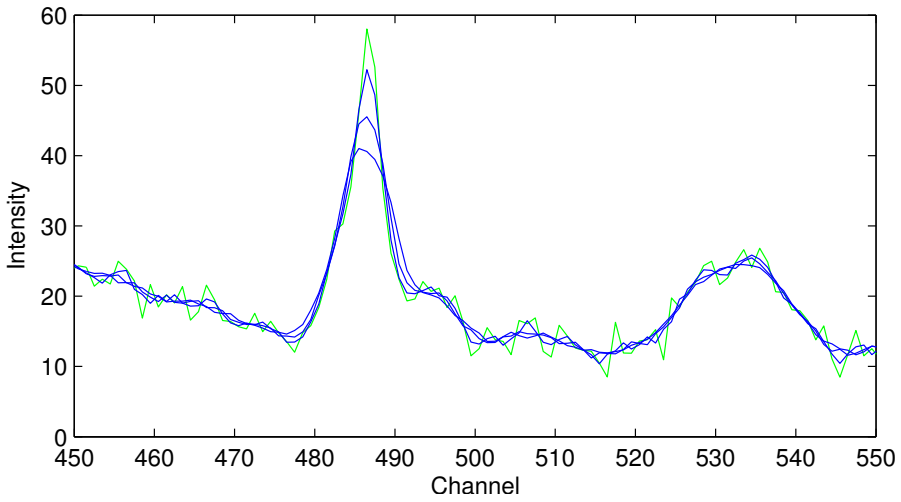

Figure 1: Smoothing of the largest peaks in the NMR spectrum with a Savitzky-Golay smoother based on a running linear curve fit with window size 3, 5 and 7. The larger the window, the smoother the curve.

when the height of a peak has not te be reduced too much, this can be a little problematic. Figure 1 shows results for the SGS with a running linear curve fit, for the three smallest possible window sizes: 3, 5 and 7, Figure 2 does the same for a running quadratic curve fit, with window sizes 5, 7, and 9. If one is not happy with these discrete choices, a weighting scheme could be introduced, at the cost of more complicated programming and a slight increase in computation. In contrast, the WS gives continuous control over smoothness with the parameter λ.

Cross-validation Leave-one-out cross-validation, to find the optimal amount of smoothing, is extremely expensive with the SGS. To my knowledge, there is no efficient way to derive a hat matrix.

Interpolation The WS will always interpolate stretches of missing data, whatever their width. The SGS can interpolate only if missing data stretches are small enough to be covered by the window.

Edge effects At the boundaries of the data special care has to be taken. Straightforward use of the SGS gives only a smoothed signal up to one half window width from both ends of the data. An ad hoc solution is to do just a curve fit to the data at the end and keep the left part (right part) of the fitted curve at the left (right) boundary.

Real smoothness The jump from one window to the next introduces small jumps in the smoothed curve. In many cases they will do no harm, but when the goal is to estimate a derivative, an unpleasant surprise may occur. This is illustrated by Figure 3. The WS performs much better, as Figure 4 shows. Also note that the SGS needs relatively large window for noisy data, with consequences at the edges.

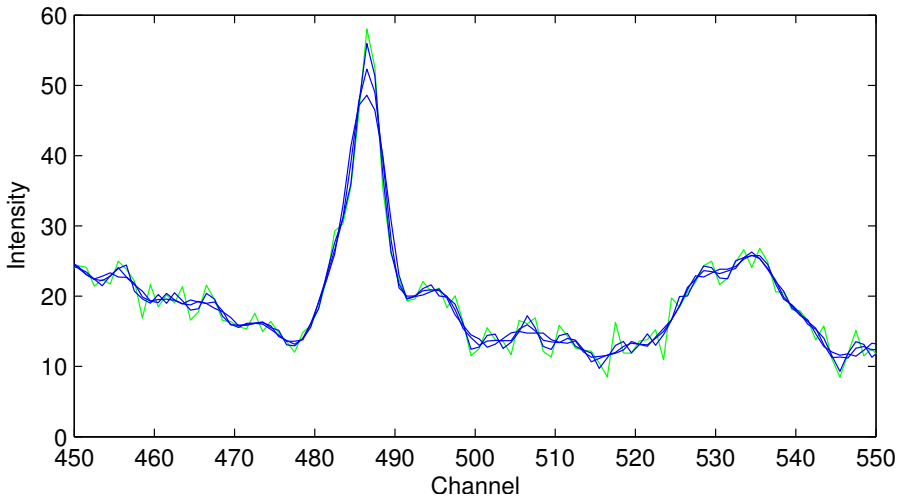

Figure 2: Smoothing of the largest peaks in the NMR spectrum with a Savitzky-Golay smoother based on a running quadratic curve fit with window size 5, 7 and 9. The larger the window, the smoother the curve.

One can improve the SGS result by computing the derivative of the fitted local polynomial, as shown in Figure 5, but is

References

[1] F. Scheid. *Numerical Analysis*. McGraw-Hill, 1968.

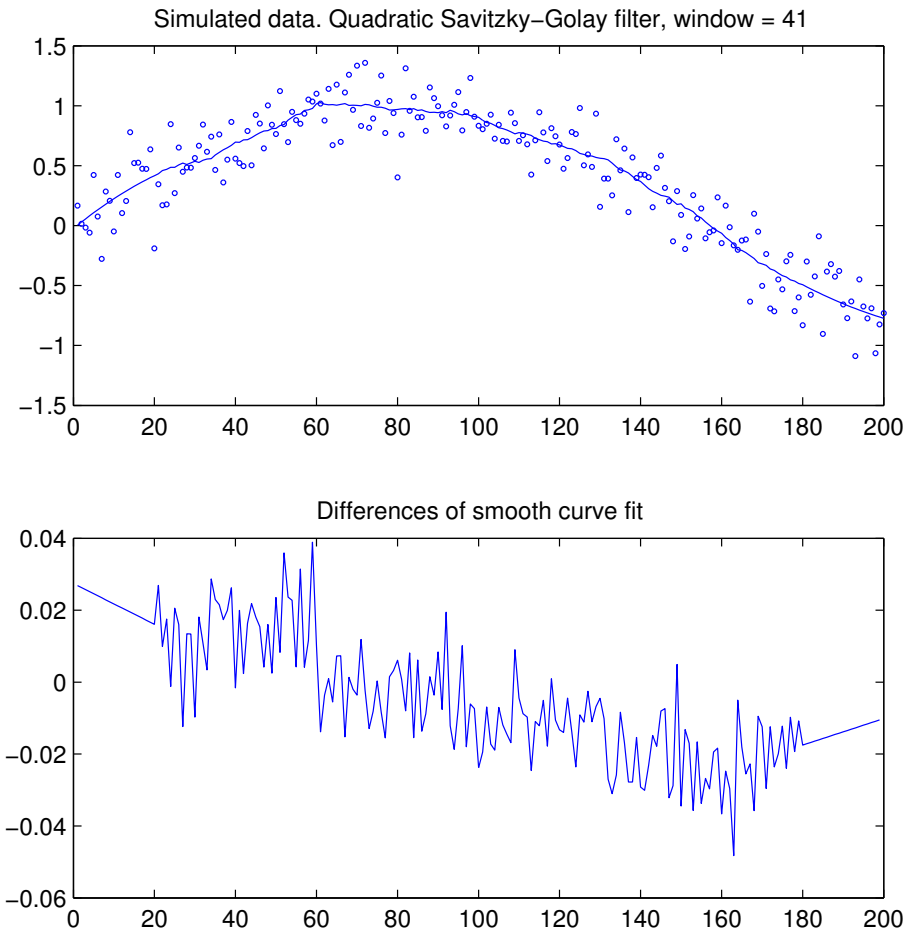

Figure 3: Smoothing of simulated noisy data with the Savitzky-Golay smoother (upper panel) and differences of the estimated trend (lower panel).

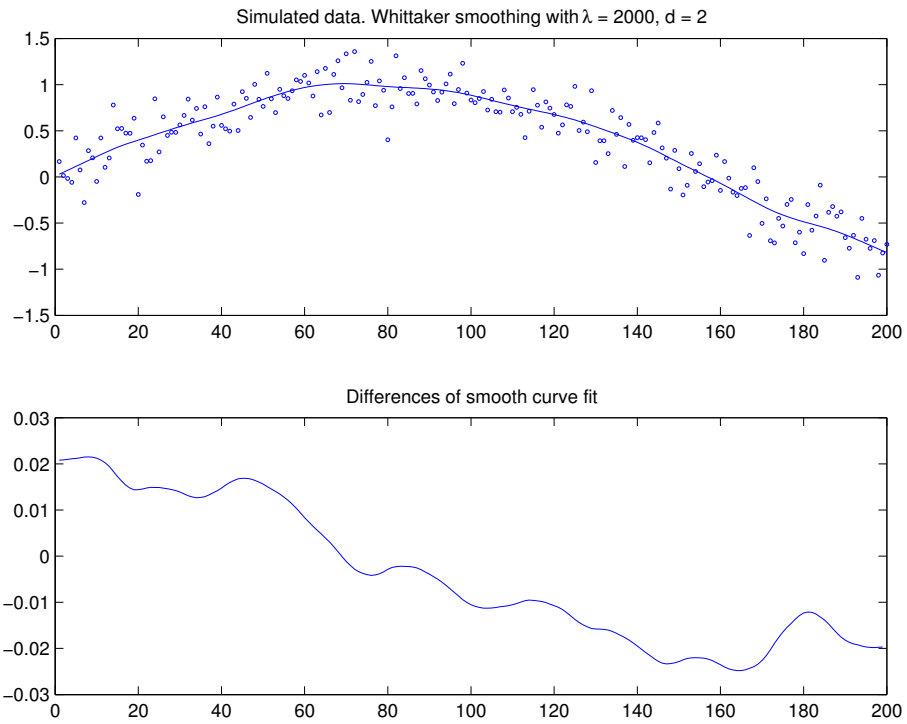

Figure 4: Smoothing of simulated noisy data with the Whittaker smoother smoother (upper panel) and differences of the estimated trend (lower panel).

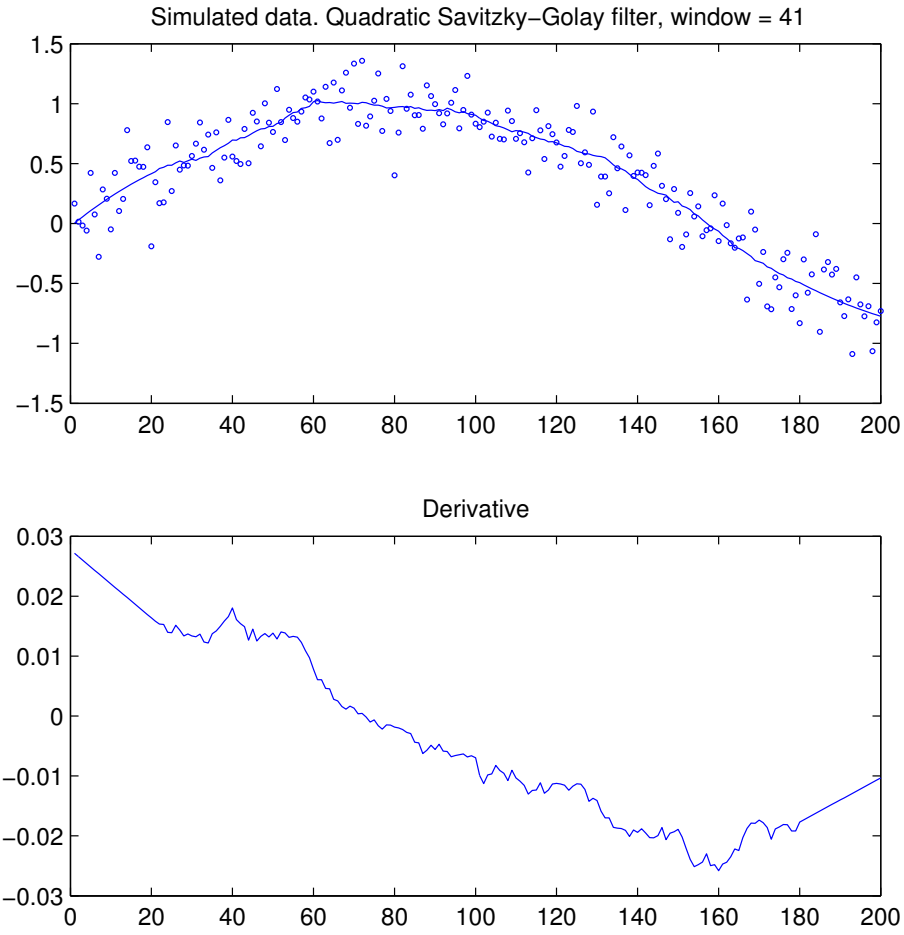

Figure 5: Smoothing of simulated noisy data with the Savitzky-Golay smoother (upper panel) and derivative of the estimated trend as computed from the fitted local polynomials (lower panel).