Fitting Smooth Curves Part III: Baselining with an Asymmetric Least-Squares Algorithm
Neal B. Gallagher

Key words: Data fitting, smoothing, robust fitting

Introduction: The datafit_engine function in PLS_Toolbox[1] provides a flexible tool for fitting smooth curves to data[2-4] and can be used to avoid end-effects observed in the Savitzky-Golay algorithm.[5] This white paper shows how the datafit_engine function can be used to baseline spectra using an asymmetric least-squares algorithm where the baseline is the smooth function.

DATAFIT_Engine Objective: The following objective function is used in the datafit_engine function

\[ O(z,a) = (y-z)^\top W_0 (y-z) + \lambda_s z^\top D_s^\top W D_s z \\
+ \lambda_s (z_c - z)^\top W_c (z_c - z) + \lambda_e (Pa - z)^\top W_b (Pa - z) \]

where \( y \) is a \( N \times 1 \) vector of measured data, \( z \) is a smooth curve to be fit to the data, \( W_0 \) is a diagonal matrix of weights (typically \( 0 \leq W_{0,n} \leq 1 \) for \( n = 1, ..., N \)), \( D_s \) is a second derivative operator (e.g., \( D_s z \) is the second derivative of \( z \)) \( \lambda_s \) is a scalar penalty on the smoothing term and \( W_s \) is a diagonal matrix of weights with entries \( 0 \leq W_{s,i} \leq 1 \) used to relax smoothing on selected regions of the signal. The last two terms can be used to impose equality constraints and fitting to basis functions. These two terms are not used in this white paper but are discussed in [3] & [6]. The present example shown here can be reproduced in PLS_Toolbox: run datafit_engine demo with option 4.

Baseline Estimation with DATAFIT_Engine: The input to datafit_engine is a vector of measured data \( y \). The outputs include \( z \), the smoothed estimate, and \( yb = y - z \), the difference between the measured signal and the smoothed signal. If \( z \) can be found by fitting a smooth curve to the bottom of a spectrum then \( z \) is an estimate of a spectral baseline and \( yb \) corresponds to the spectrum with the baseline removed. For example, an optical emission (OES) spectrum is plotted in Figure 1 (blue) (The OES are available at www.eigenvector.com.[1]) The spectrum shows a significant offset that is ~flat in the region 245 to 375 nm and then increasing non-linearly with wavelength in the region 380 to 450 nm. Instead of using polynomial fits (e.g., see [3]) an asymmetric least-squares fit can be used by changing the trbflag to fit to the bottom of the spectrum (options.trbflag = ‘bottom’), setting the tolerance to 4 (options.tol = 4) and setting the smoothing penalty \( \lambda_s \) to \( 10^5 \). (options.lambdas = 1e5) [options is an input to datafit_engine].

The asymmetric least-squares algorithm iteratively de-weights terms with high residuals relative to the tolerance.[3] When trbflag is set to ‘bottom’, terms with high positive residuals are de-weighted by setting the corresponding element of \( W_0 \) to a small number (significantly <1) and the high smoothing penalty ensures that \( z \) will be a smooth curve representing the baseline [note that \( z \) (yellow) passes through the noise and not at the bottom of the noise]. Figure 2 shows the fit results where \( yb \) (red) is the baselined spectrum.

The baseline \( z \) was forced to be a smooth curve. However, that means that the noise from the original spectrum \( y \) is retained in the baselined spectrum \( yb \) and it may be desirable to smooth the noise from this signal. This means that \( yb \) is now the input to a second call to datafit_engine and the output of interest now is the smoothed signal \( z \): The signal of interest...
depends on the specific objective. In the first call, the signal of interest was \( y_b \) where the smoothed signal \( z \) was the baseline to be removed. In the second call, the signal of interest is the smooth signal however, it is not desired to smooth the peaks – only the noise is to be smoothed. To achieve this objective, the default options structure is changed by first initializing \( W_s \) to be the identity matrix \( I \) \[ \text{options.ws} = \text{ones(1,size(oes1,2))} \] and then changing the elements with peaks to have a small smoothing penalty \[ \text{opts.ws}(y_b.data(1,:)>4) = 1e-5 \]. Finally, trbflag is set to ‘none’ to use traditional least-squares fitting \[ \text{options.trbflag} = 'none' \] and the smoothing penalty is reduced to a more moderate level \[ \text{options.lambdas} = 1e3 \]. The results are shown in Figure 3 where the smoothed signal of interest is \( z \) (yellow) and the noise is captured in \( y_b \) (red).

Figure 2: Baselined signal from \textit{datafit_engine}. Asymmetric fitting was used: trbflag = ‘bottom’.

Conclusions: The \textit{datafit_engine} function can be used to fit flexible non-linear baseline spectra using an asymmetric least-squares algorithm as seen in Figure 4. It is clear that the estimated baselines are more complicated than simple polynomial functions. Basis functions and equality constraints can also be imposed to reduce the flexibility if desired.[3,6] To obtain the desired results from \textit{datafit_engine}, the data analyst must be clear if the desired signal is the smoothed signal \( z \) or the output \( y_b = y - z \).

Figure 3: Baselined and smoothed signal from \textit{datafit_engine}. Asymmetric fitting was not used: trbflag = ‘none’.

Figure 4: Estimated baselines for all 46 OES spectra for the first call to \textit{datafit_engine} using asymmetric least-squares fitting.

References:


