EIGENVECTOR RESEARCH INCORPORATED
Research, Training, and Software

196 Hyacinth Road
Manson, WA 98831
www.Eigenvector.com

Fitting Smooth Curves Part IV: Baselining with Asymmetric Least-Squares and Basis Functions

Neal B. Gallagher

Key words: Data fitting, smoothing, robust fitting, basis functions

Introduction: The `datafit_engine` function in PLS_Toolbox[1] is a flexible tool for fitting smooth curves to data[2-5]. Imposing constraints by fitting to basis functions allows more control over the fit. For example, in the previous white paper the baselines fit to the OES spectra were very flexible and strongly non-linear with respect to wavelength.[5] This white paper starts with the results from that flexible fit and shows how the `datafit_engine` function can be used to baseline spectra using an asymmetric least-squares algorithm with basis functions via penalty functions. As with the previous example, the baseline is a smooth curve but less flexible in selected regions of the OES spectra.

DATAFIT_Engine Objective: The objective function used by `datafit_engine` has been presented previously[3-5] and, in the interest of brevity, it is left to the reader to review that material.

For a vector of measured data, $\mathbf{y}$, the corresponding smooth curve fit to the data is $\mathbf{z}$. The outputs from `datafit_engine` include **z**, the smoothed estimate, and **yb** = **y** – **z**, the difference between the measured signal and the smoothed signal. The strength of the smoothing if given by a scalar penalty, λ_s, and the diagonal elements of $\mathbf{W}_s$ (with elements between 0 and 1) can be used to relax smoothing in selected regions of a spectrum. Similarly, the strength of the fit to a set of basis functions is given by the scalar penalty, λ_b. The diagonal elements of $\mathbf{W}_b$ (with elements between 0 and 1) can be used to relax fitting in selected regions of a spectrum. See [4] for a more complete description. The example shown below can be reproduced in PLS_Toolbox: run `datafit_engine demo` with option 5.

Baseline Estimation with DATAFIT_Engine: As in [4], the example here uses two calls to `datafit_engine`. The first call is used to fit a smooth curve to the bottom of a set of the OES spectra – this is output **z** and is an estimate of a spectral baselines. The penalties and weights are input using an options structure and for the example in [4] an asymmetric least-squares fit was obtained by setting 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 λ_s to 10^5. (`options.lambdas = 1e5`). Figure 1 shows the estimated baseline spectra, **z**). The region 245 to 300 nm shows significant non-linearity that might not be desired. New options will be included to minimize the flexibility in this region.

To reduce the non-linear behavior in the 245 to 300 nm region, the following changes are made to `options`. To fit an offset, the polynomial order is set to 0 (`options.orderp = 0`) and the fitting penalty λ_b is set to 10^2 (`options.lambdab = 1e2`). The weights (diagonal elements of $\mathbf{W}_b$) are set to a smoothly varying function given by[1]

```
[options.wb = (1+ peaksigmoid([1 270 -
      0.2], oes1.axisscale{2}))/2]
```

Figure 2 shows the resulting baseline fits, **z**, using asymmetric least-squares with fitting to a basis function (0^{th} order polynomial). The baselines in the 245 to 300 nm region have been 'flattened' while the region > 300 nm, where the elements of $\mathbf{W}_b$ are very small or zero is unaffected. The baselined signal is given by **yb** = **y** – **z**.

As in the previous example [4], the baselined signal, **yb** = **y** – **z**, is smoothed using a second call to `datafit_engine` with **yb** from the first call used as the input for measured signal. The changes to the options input are as follows. First initializing $\mathbf{W}_s$ to be the identity matrix **I** [`options.ws = ones(1,size(oes1,2))`] and then change the elements with peaks to have a small smoothing penalty [`opts.ws(yb.data(1,:)>4) = 1e-5`]. Next, trbflag is set to 'none' to use traditional least-squares fitting [`options.trbflag = 'none'`] and the smoothing penalty is reduced to a more moderate level [`options.lambdas = 1e3`]. Finally, the basis fitting penalty, λ_b, is set to zero (`opts.lambdab = 0`). The results are shown in Figure 3 where the input was the baselined signal from the first call to `datafit_engine` (blue), the smoothed baselined signal (i.e., the signal of interest) is **z** (yellow) and the noise is now captured in **yb** (red).

196 Hyacinth Road
Manson, WA 98831
www.Eigenvector.com

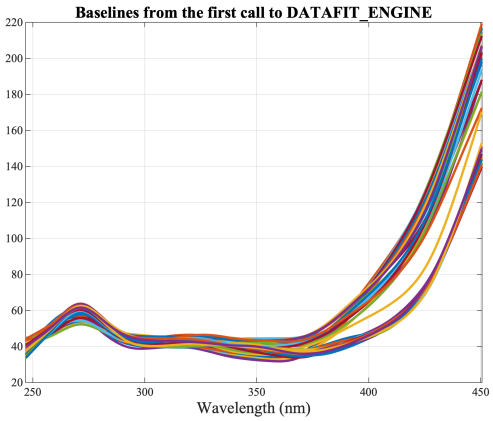

Figure 1: Estimated baselines for all 46 OES spectra for the first call to `datafit_engine` using asymmetric least-squares fitting. (This is Figure 4 from [4]).

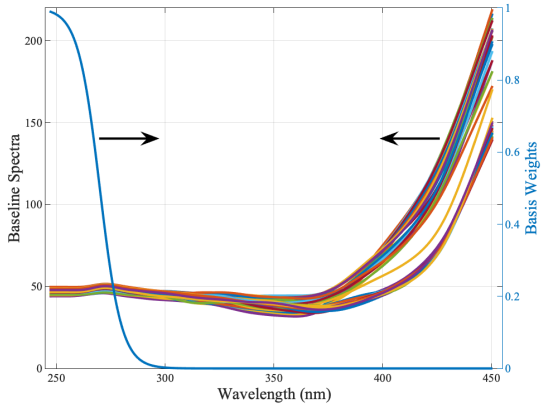

Figure 2: Estimated baselines for all 46 OES spectra for the first call to `datafit_engine` using asymmetric least-squares fitting plus a fit to an offset (left y-axis). The diagonal elements of Wb are shown as having non-zero elements ≤1 and ≥0 in the region 244 – 300 nm (right y-axis).

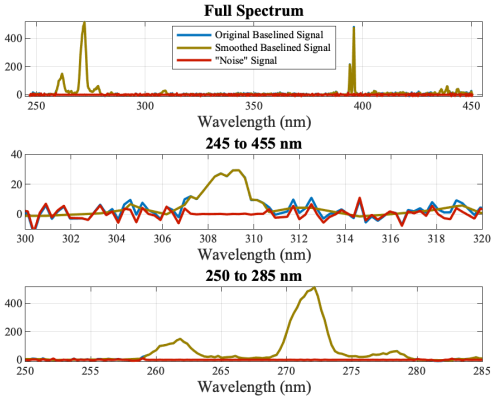

Figure 3: Baselined and smoothed signal from `datafit_engine`. (yellow), baselined signal w/o smoothing (blue), and noise (red) Asymmetric fitting was not used: trbflag = 'none'.

Conclusions: The `datafit_engine` function can be used to fit flexible non-linear baseline spectra using an asymmetric least-squares algorithm as seen in Figures 1 and 2. Adding fits to basis functions provides additional flexibility to design a baseline fitting algorithm for a wide variety of signals.

Likely the biggest potential confusion is being clear about the desired signal of interest. Recall that the input to `datafit_engine` is **y** and outputs are **yb** and **z** where **yb** = **y** - **z** and **z** is the smoothed fit to the data. In the first call to `datafit_engine` in this example, the desired output was the baselined signal $\mathbf{yb}_1 = \mathbf{y}_1 - \mathbf{z}_1$ where the baselines were given by $\mathbf{z}_1$. (The subscript is given to indicate the call to `datafit_engine`.) In the second call to `datafit_engine`, the input $\mathbf{y}_2 = \mathbf{yb}_1$ and the signal of interest was the resulting smoothed baselined data given by $\mathbf{z}_2$. The output $\mathbf{yb}_2$ corresponded to the noise removed from $\mathbf{y}_2$.

References:

[1] PLS_Toolbox and Solo. Eigenvector Research, Inc., Manson, WA USA 98831; software available at www.eigenvector.com.

[2] Gallagher, NB, "Whittaker Smoother," white paper Eigenvector Research, Inc., www.eigenvector.com.

[3] Gallagher, NB, O'Sullivan, D, "Fitting Smooth Curves Part I: Fitting with Equality Constraints and Basis Functions," white paper Eigenvector Research, Inc., www.eigenvector.com.

[4] Gallagher, NB, "Fitting Smooth Curves Part II: Fitting with a Robust Algorithm," white paper Eigenvector Research, Inc., www.eigenvector.com.

[5] Gallagher, NB, "Fitting Smooth Curves Part III: Baselining with an Asymmetric Least-Squares Algorithm," white paper Eigenvector Research, Inc., www.eigenvector.com.