Guide to using the MATLAB EELS background subtraction scripts

Table of Contents

This is a step-by-step guide to using the three MATLAB scripts – EELS_fitting.m, EELS_fit_analysis.m, and EELS_subtracted_spectrum.m. The scripts were written in MATLAB version R2019b but should be usable in most versions of MATLAB above R2015a. This guide is intended to be as comprehensive as possible for newer users of MATLAB or indeed any coding. In addition to this, extensive commenting (text beginning with '%' or '%%') has been included in the scripts themselves to explain the code to the user. For this guide, the carbon K-edge of single-walled carbon nanotubes (SWNTs) has been used as an example. This is the same example as the one discussed in the main publication (https://www.sciencedirect.com/science/article/pii/S0304399120302035).

EELS_fitting.m – finding the appropriate fit

The purpose of the first script, EELS_fitting.m, is to survey what the best window for fitting might be and what model should be used for the fit. The script starts off by clearing any previous output or data that might already be in MATLAB.

```
clc
close all
clear all
```

clc – Clear Command Window of any input or output from the Command Window display

close all – Close all figures that might already have been generated by MATLAB

clear all – Close all items (such as existing data/variables) from the Workspace

The first part of the code that will need to be changed by the user is the file name of the dataset that is to be fitted. Data should be in a .msa format, directly saved from the .dm3 spectrum or spectrum image file. The .msa file should be in the same folder as all three scripts so that it is on the same MATLAB Search Path as the scripts. The file name of the .msa can be typed into the code below, between the quotation marks, to allow MATLAB to import the data. The rest of the code tells MATLAB to skip the lines of text at the beginning of .msa files exported from Digital Micrograph.

```
filename = 'file_name.msa';
delimiterIn = ','; % This is the character that separates the two columns
of data.
headerlinesIn = 20; % This is the number of lines of text at the start of
```

```
the data that are skipped.
msadata = importdata(filename,delimiterIn,headerlinesIn);
data = msadata.data;
```

The variables are then assigned from the imported matrix as 'xdata' and 'ydata' from columns 1 and 2 of the imported data respectively.

```
xdata = data(:,1);
ydata = data(:,2);
```

There may be situations where the data of interest is a smaller section of the total spectrum, such as when there are overlapping edges. The next part of the code allows the user to define the start and end of the edge of interest. A larger background before the start of the edge gives MATLAB more data to work with when fitting a model to the background, and therefore tends to lead to better fitting and subtraction of the background. In the example of SWNTs, there are no other overlapping edges near the carbon K-edge, which starts at 284 eV, and therefore it would be reasonable to include all data below the edge for background fitting. The start of the edge is called 'startedge' and has been defined as 176 eV.

```
startedge = xdata > 176;
xdata1 = xdata(startedge);
ydata1 = ydata(startedge);
```

The end of the data is defined in much the same way, this time called 'endedge'. In the case of the carbon K-edge of SWNTs, the whole spectrum is of interest, so 'endedge' has been given the value of 381 eV.

```
endedge = xdata1 < 381;
xdata2 = xdata1(endedge);
ydata2 = ydata1(endedge);
```

The next part of the script defines the colours of the generated plots. Because this script is designed to allow the user to find the best fit from a range of background windows, the 'colour order' or 'co' was pre-defined to be a rainbow so that it might be easier for the eye to follow how the fits change depending on the window width chosen. The colour order can, of course, be changed to be more suitable for those who have different colour vision or for personal preference. MATLAB uses RGB triplets to define colour (https://uk.mathworks.com/help/matlab/creating_plots/specify-plot-colors.html). Note that the second entry in the matrix below is the first colour of the plot; for some reason the first colour value of [0.00 0.00 0.00] is not used for plotting.

```
co = [0.00 0.00 0.00
    1.00  0.40  0.40
    1.00  0.70  0.40
    1.00  1.00  0.40
    0.70  1.00  0.40
```

```
    0.40   1.00   0.40
    0.40   1.00   0.70
    0.40   1.00   1.00
    0.40   0.70   1.00
    0.40   0.40   1.00
    0.70   0.40   1.00
    1.00   0.40   1.00
    1.00   0.40   0.70];
set(groot,'defaultAxesColorOrder',co)
```

The 3 sets of numbers in each row of the colour order represents the RGB values of colours respectively. Normally RGB are numbers out of a total of 255 (pure red being 255 0 0 for example) but MATLAB uses numbers out of a total of 1. Changing any of the colours to the desired RGB value requires minor calculation to convert the numbers from a fraction of 255 to a fraction of 1.

There is some preamble enclosed in comments describing the various fitting types that might be suitable to use for background subtraction (this is further explained in the publication) before reaching the main part of EELS_fitting.m. The script now requires user input to define where data should be excluded for the fitting. This defines the window(s) for fitting the chosen model to the background. This is assigned the variable (i) where any data above (i) is excluded from the fitting. Several values of (i) can be assigned in the fitting 'for' loop and plotted on the same axes to help the user find the best fit visually. In the example of SWNTs, nine values of (i) have been assigned for fitting: 200, 210, 220, 230, 240, 250, 260, 270, and 280. This is written in the form of first value of (i): the increment of (i): and the final value of (i).

```
for i = 200:10:280
```

Then, all data above these values of (i) are excluded sequentially in the following 'for' loop.

```
exclude1 = xdata2 > i;
```

The model for fitting the remaining data after exclusion can then be user-defined between the quotation marks. In the example below for the carbon K-edge of SWNTs, a two-term power law ('power2') was used. Generally, some trial and error is required to find the right model. If the generated background subtracted spectra look completely wrong, this may be because the model has not been given enough data points for fitting. If this is the case, the first and/or final values of (i) can be changed to different numbers to exclude fewer data points.

```
f = fit(xdata2,ydata2,'power2','Exclude',exclude1);
```

The background fit is subtracted from the original spectrum to give the subtracted spectrum.

```
residuals = ydata2 - f(xdata2);
```

The residual y values are then plotted against the original x values to give the subtracted spectrum 'p1'.

```
p1 = plot(xdata2,residuals);
```

The current axis of the plot is defined as 'ax1' to allow the user to define the characteristics of the plot at the end of the script.

```
ax1 = gca;
```

The plot colour of 'p1' is defined using the colour order from above and the thickness of the plot line 'LineWidth' has been defined as 2. The plot label is then assigned the appropriate value of (i) using 'DisplayName' and num2str(i). Finally, all the plots are plotted onto the same axes using 'hold on'. The loop is then ended using 'end'.

```
colorOrder = get(gca, 'ColorOrder');
set(p1,'Color',colorOrder(mod(length(get(gca, 'Children')),
size(colorOrder, 1))+1, :),'LineWidth',2,'DisplayName',num2str(i))
hold on
end
```

The original EELS data is then plotted on the same axes and given a black colour 'k'.

```
plot(xdata2,ydata2,'Color','k','LineWidth',2,'DisplayName','Original EELS
data');
hold off
```

The characteristics of each axis are defined at the end of the script. After defining values of (i) and the model for fitting, the script can then be run (press Run in the window or the keyboard short-cut F5) to generate the figure with all the background subtracted spectra plotted against the original EELS data. As can be seen in Figure 1 below, the background fit of the carbon K-edge of SWNTs, where data above 200 eV was excluded, has a sloping baseline. The model has underestimated the background and this window for fitting is therefore not appropriate for background subtraction. Fitting a model to the largest dataset possible is statistically better in terms of lowering the error of the fit so the largest window of all data below 280 eV was chosen for analysis in the second script, EELS_fit_analysis.m.

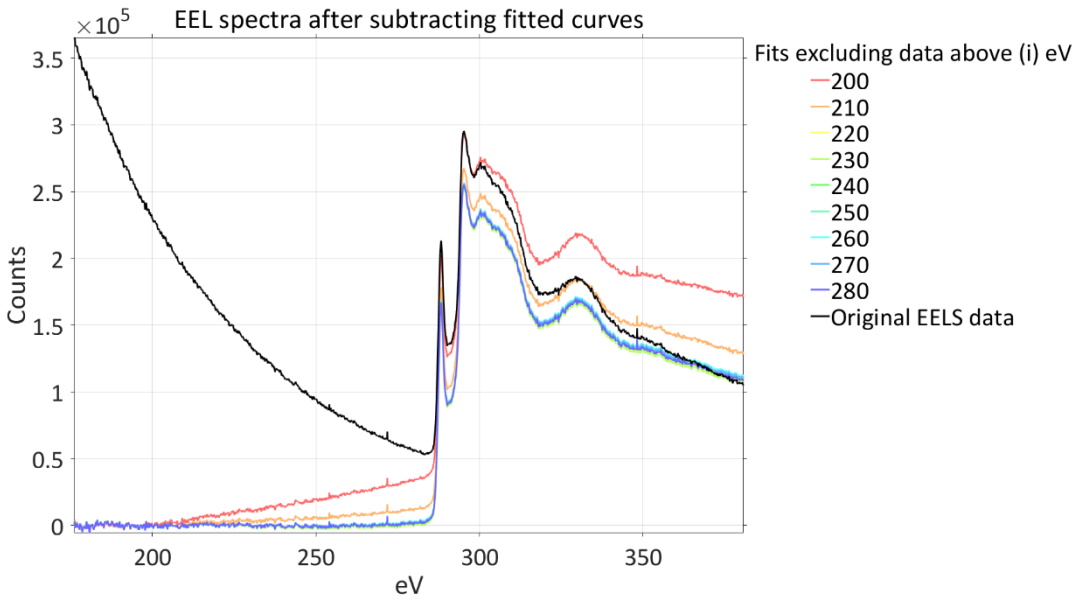

Figure 1. The pre-edge background of the carbon K-edge of single-walled carbon nanotubes (SWNTs) was fitted using a two-term power law of the form $f(x)=ax^b+c$ in the EELS_fitting.m MATLAB script. Several fitting windows were chosen below (i) eV, from 176-200 eV to 176-280 eV, and plotted in a rainbow of colours to visually aid finding the best fit.

EELS_fit_analysis.m – assessing the goodness-of-fit

Having estimated by eye the best fit, the goodness-of-fit for that value of (i) can then be assessed using EELS_fit_analysis.m. The data is imported from the same .msa file and the edge of interest extracted in the same manner as in EELS_fitting.m. A two-column table 'table1' of the extracted edge x and y data is then created for the purpose of data exclusion of both x and y variables for curve fitting.

```
table1 = table(xdata2,ydata2);
```

The desired value of (i) can be defined by the user. From EELS_fitting.m, it was determined that a value of 280 eV for (i) gave a good fit for the background of the carbon K-edge of SWNTs. All x values above 280 eV are excluded, along with the corresponding y values, in 'table1' to give a new dataset 'data2'.

```
i = 280
data2 = table1{table1.xdata2 < i,:};
```

New variables are created from the excluded data in 'data2'.

```
xdata3 = data2(:,1);
ydata3 = data2(:,2);
```

The model can then be defined by the user. In the case of the carbon K-edge of SWNTs, the most appropriate model for background fitting was a two-term power law 'power2'. The script also tells MATLAB to output information on the fit such as the equation used for the fit and the coefficients of

the equation with confidence bounds ('f'), the goodness-of-fit statistics such as the adjusted R-square value ('gof'), as well as information on the fitting algorithm ('output').

```
[f,gof,output] = fit(xdata3,ydata3,'power2')
```

The background fit is subtracted from the original spectrum to give the subtracted spectrum in the same way as in EELS_fitting.m.

```
residuals = ydata2 - f(xdata2);
```

The script then tells MATLAB to output the options for the fit algorithm ('fitoptions'), such as the method used for fitting, which is non-linear least squares.

```
fitoptions = fitoptions(f)
```

The script also gives information on the integral of the signal, calculated using the trapezoidal rule after background subtraction. The window for integration can be defined by the user. The start of integration 'startint' for the carbon K-edge of SWNTs was defined as 284 eV.

```
startint = xdata2 > 284;
xdataint1 = xdata2(startint);
residualsint1 = residuals(startint);
```

The end of integration 'endint' was defined as 300 eV.

```
endint = xdataint1 < 300;
xdataint2 = xdataint1(endint);
residualsint2 = residualsint1(endint);
```

This window is then integrated under the curve and given the name 'I_k'. This information can be useful for calculations of elemental composition as well as the h-parameter and the SNR.

```
ik = trapz(xdataint2,residualsint2)
```

The background i.e. the fit (‘fityvalues’) is also integrated using the trapezoidal rule. The start and end of integration (‘startint’ and ‘endint’) are the same as for integrating the signal. The window is then integrated under the curve and given the name ‘I_B’.

```
fityvalues = f(xdata2);
bkgint1 = fityvalues(startint);
bkgint2 = bkgint1(endint);
ib = trapz(xdataint2,bkgint2)
```

The variance of I_B (‘varib’) is calculated.

```
varib = var(bkgint2)
```

Using I_k, I_B, and the variance of I_B, the h-parameter is calculated.

```
h = (ib+varib)/ib
```

After this, the SNR value is calculated.

```
snr = ik/((ik+(h*ib))^0.5)
```

The original data and the fitted curve with observation prediction bounds are then plotted.

```
p1 = plot(f,'b',xdata2,ydata2,'r-','predobs');
```

The subtracted spectrum is plotted on the same axes.

```
p2 = plot(xdata2,residuals,'k');
```

Finally, the residuals from the fitted model are plotted on separate axes. The residuals are the differences between the original data and the fit to the original data. If the model is correct, the residuals approximate the random errors. Therefore, the model fits the data well if the residuals appear to behave randomly. If the residuals have a systematic pattern, the model likely does not fit the data very well.

```
p3 = plot(f,xdata3,ydata3,'Residuals');
```

The characteristics of each axis are defined at the end of the script. Running this script (press Run in the window or the keyboard short-cut F5) after inputting all the information generates both the figure as well as an output in the Command Window display which shows the fit used, the goodness-of-fit statistics, information on the fitting algorithm, as well as the options for the fit algorithm. In Figure 2, there is an approximately even distribution of the residuals, the small prediction bounds (the difference between the prediction bounds and the fitted curve) are so small that they are not visually resolved in the figure, and the adjusted R-square value is 0.9998 (see the Command Window output below). All of this indicates that the background fit is statistically good.

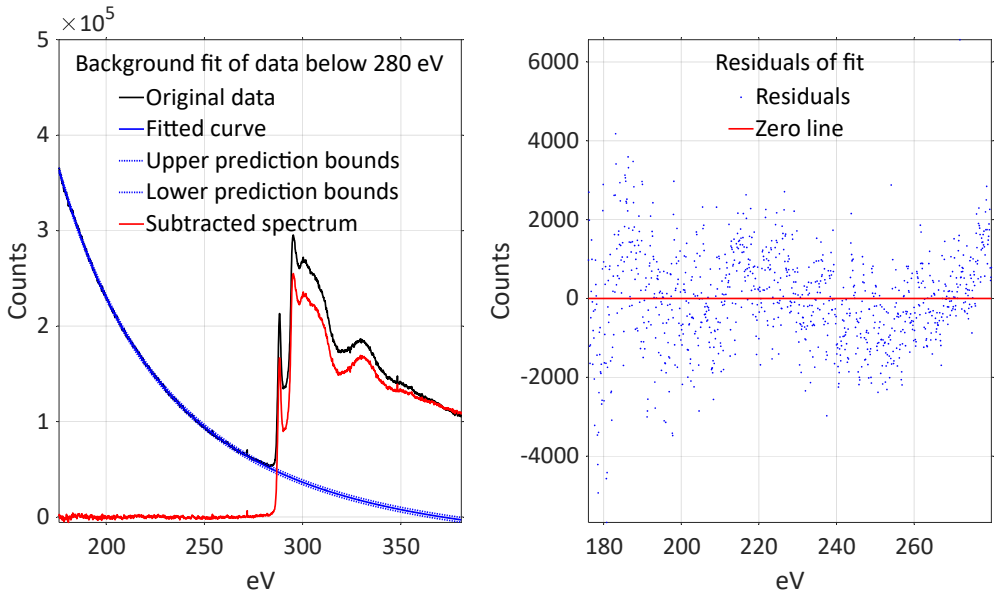

Figure 2. The best fit using the widest window of 176-280 eV was analysed using EELS_fit_analysis.m which produced two plots as well as statistical information on the fit. The left plot shows that the model closely matches the background (the prediction bounds are shown as dotted lines). The right plot shows that the residuals after subtracting the fitted curves from the original EELS data are randomly distributed which suggests that the fit is good.

Output of EELS_fit_analysis.m in the Command Window display:

```
f =

     General model Power2:
     f(x) = a*x^b+c
     Coefficients (with 95% confidence bounds):
       a =   5.086e+12  (4.692e+12, 5.481e+12)
       b =      -3.163  (-3.178, -3.147)
       c =    -3.8e+04  (-3.899e+04, -3.701e+04)

gof =

  struct with fields:

          sse: 1.6095e+09
      rsquare: 0.9998
          dfe: 1036
    adjrsquare: 0.9998
         rmse: 1.2464e+03


output =

  struct with fields:

          numobs: 1039
        numparam: 3
       residuals: [1039×1 double]
        Jacobian: [1039×3 double]
        exitflag: 3
   firstorderopt: 3.7057e+03
      iterations: 3
       funcCount: 8
    cgiterations: 0
       algorithm: 'trust-region-reflective'
        stepsize: 3.2854e-05
         message: 'Success, but fitting stopped because change in
residuals less than tolerance (TolFun).'


fitoptions =

        Normalize: 'off'
          Exclude: []
          Weights: []
           Method: 'NonlinearLeastSquares'
           Robust: 'Off'
      StartPoint: [1×0 double]
            Lower: [1×0 double]
            Upper: [1×0 double]
        Algorithm: 'Trust-Region'
   DiffMinChange: 1.0000e-08
```

```
    DiffMaxChange: 0.1000
          Display: 'Notify'
      MaxFunEvals: 600
          MaxIter: 400
           TolFun: 1.0000e-06
             TolX: 1.0000e-06


ik =

   2.2287e+06


ib =

   6.8310e+05


varib =

   1.6426e+07


h =

   25.0463


snr =

  506.8083

>>
```

It is important to look at all aspects of the fit to determine whether or not the fit is a good representation of the background signal. A good fit does not necessarily have to have the highest adjusted R-square value, for example.

After running EELS_fit_analysis.m, the Workspace items called 'xdata2' and 'residuals' can be renamed and saved separately as .mat variables (this can be done simply by right-clicking on the Workspace items and saved with an appropriate file name). These are the x and y values of the background-subtracted spectrum and can be loaded into the third script, EELS_subtracted_spectrum.m, for plotting.

EELS_subtracted_spectrum.m – plotting the background-subtracted spectra

The third and final script included was written to help newer users of MATLAB plot the background-subtracted spectrum separately as well as give all users the option to save the subtracted spectrum as a two-column .txt file for plotting in other software, such as Origin. The original EELS data can be imported in the same way as the other two scripts and the subtracted spectrum can be loaded from the x and y .mat files saved from the second script (see above).

```
load ('EELS_fit_xdata_file_name.mat')
load ('EELS_fit_ydata_residuals_file_name.mat')
```

After the variables are assigned from the imported data, the subtracted spectrum x and y variables are added to a table 't1'.

```
t1 = table(x2,y2);
```

This table is then saved to a two-column .txt format, delimited by commas, with the row names written at the top of the columns. When exporting data into a .txt, check that the correct variables of x and y are being saved, especially when plotting multiple spectra using this script. Every time the script is run, the .txt is saved over the previous version.

```
writetable(t1,'subtracted-spectrum.txt','WriteRowNames',true)
```

Finally, after running the script (press Run in the window or the keyboard short-cut F5), the original and the subtracted spectra are plotted on the same axes and the characteristics of the axes defined at the end of the script (Figure 3).

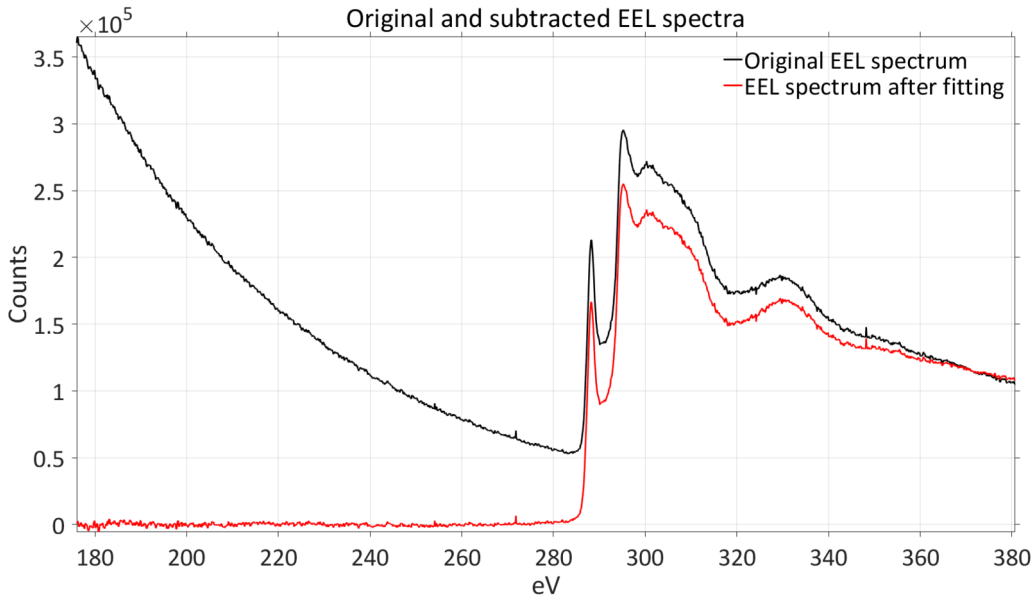

Figure 3. The original and background-subtracted spectra were plotted on the same axes for comparison using EELS_subtracted_spectrum.m.