Limnol. Oceanogr.: Methods 6, 2008
© 2008, by the American Society of Limnology and Oceanography, Inc.

Characterizing dissolved organic matter fluorescence with parallel factor analysis: a tutorial

Colin A. Stedmon[1]* *and Rasmus Bro*[2]
[1]Department of Marine Ecology, National Environmental Research Institute, Aarhus University, Frederiksborgvej 399, Roskilde, Denmark
[2]Dept. Food Science, Faculty of Life Sciences, University Copenhagen, Rolighedsvej 30, DK-1958, Frederiksberg, Denmark
*Corresponding author e-mail: cst@dmu.dk, Tel: +45 46301805.

Appendix 1: Tutorial

Characterizing DOM fluorescence using the DOMFluor Toolbox for MATLAB®

In this section a brief tutorial in applying PARAFAC to DOM fluorescence EEMs will be given with recommendations on a suitable series of stages to follow in order to facilitate the process based on the authors' own experience. The tutorial is based on MATLAB software and requires that this is installed on your computer. The DOMFluor toolbox for MATLAB provided is in the compressed file "DOMFluorv1_4.zip" (Appendix 1) and works independent of your computer's operating system. This is also freely downloadable from the Chemometrics site at University of Copenhagen (www.models.life.ku.dk). The toolbox contains within it the N-Way toolbox v.3.1 (Anderson and Bro, 2000) and additional functions which facilitate running PARAFAC on DOM fluorescence data. The DOMFluor toolbox is written and tested in MATLAB release 2007b (v7.5) and is free software that can be used and modified under the terms of the GNU General Public License.

Within the toolbox a test dataset is provided for the tutorial, consisting of 65 EEMs which have been spectrally corrected for instrument biases, corrected for inner filter effects and Raman calibrated. As a result, all of the Rayleigh scatter and most of the Raman scatter has been removed. The remaining scatter will be removed in the pre-processing phase. The data consists of samples collected from a cruise on RV Gunnar Thorson in the Kattegat and Belt Sea region (at the entrance to the Baltic Sea) in August 2001. Three "outlier" samples were created and are present in the data for the benefit of the tutorial.

The tutorial is composed of phases; i) data pre-processing, ii) initial explorative data analysis, iii) modeling and model validation, iv) interpreting results. The more thoroughly the first two phases are carried out, the easier the latter two will be. Part of the pre-processing on the example data set has already been carried out (importing the data to MATLAB, spectral corrections, calibrations, etc.). This is very instrument specific and therefore not appropriate for this tutorial, which focus on the modeling of the data.

During this tutorial any statements to be written in MATLAB will be written in green with `Courier New font`. When writing in the Command Window, note that MATLAB is case sensitive.

A. Setting up MATLAB to run with N-way toolbox and DOMFluor toolbox.

1. Decide on a sensible place to locate the DOMFluor toolbox on your computer. (This could, for example, be C:\Program Files\MATLAB\R2007b\toolbox but can also be elsewhere). Create a folder and name it DOMFluor. Extract the contents of the zip file "DOMFluor.zip" into the DOMFluor folder.
2. Start MATLAB. Select [File] menu then click [Set path…]. Click [Add with sub folders] and locate the DOMFluor folder you created in Step 1 above. If the N-Way toolbox is currently installed, remove it from the Path list to avoid conflicts. The N-Way toolbox is included as part of the DOMFluor toolbox. Click [Save] then [Close]. You have now associated the DOMFluor toolbox with MATLAB and this means that the functions in these folders can be recognized.
3. To check that the toolboxes are correctly associated you can type

```
DOMFluor
```

   and press enter in the Command Window in MATLAB. If "Yes" is printed to the Command Window, the toolbox is installed. If red text appears the toolbox has not been correctly installed. Redo Step 2. Note that if you have other toolboxes with similarly named functions, only the function upmost in the folder-list will work properly.
4. Section A will have to be repeated if a new version of MATLAB is installed.

B. Loading the tutorial data and plotting the EEMs.

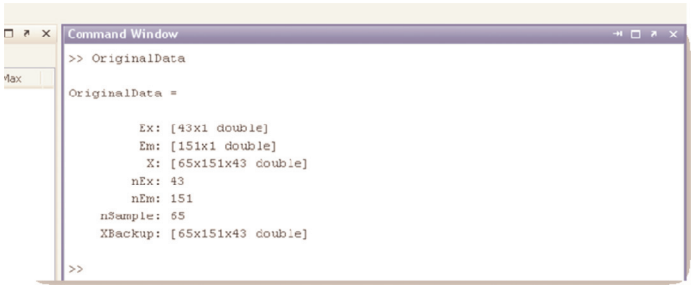
1. Type

```
load PARAFACexample.mat
```

   in the Command Window. This will load the tutorial data into the MATLAB Workspace. This consists of a data structure called OriginalData. Try typing

```
OriginalData
```

in the command window. This will output the following:



This details the data that is contained within the data structure.

i. Ex- is a list of the excitation wavelengths measured (in nm). In this case 240 to 450 every 5 nm.
ii. Em- is a list of the emission wavelengths measured (in nm). In this case 300 to 600 every 2 nm.
iii. X- is the fluorescence data as a three dimensional array (65 samples x 151 emission wavelengths x 43 excitation wavelengths).
iv. nSample, nEx and nEm : state the number of samples, excitation and emission wavelengths.
v. XBackup-is a backup copy of the data and not used in the tutorial.

2. Now we will plot the EEMs to check that the data is correctly loaded. In the toolbox there are several functions that can be used to plot EEMs.
   a. Typing

```
PlotEEMby1(1:5,OriginalData,'R.U.')
```

will plot the first 5 EEMs in the dataset one at a time as contour plots. The next graph can be viewed by pressing any key on the keyboard. If `1:5` is replaced with `1:65` all 65 samples will be plotted one at a time. The plotting process can be halted at any time by entering [Ctrl+C] on the keyboard and then closing the Figure window.

   b. Typing

```
PlotEEMby4(1,OriginalData,'R.U.')
```

will plot the data four at a time. This is quicker when dealing with large datasets.

   c. Typing

```
PlotEEMby4FixZ(1,OriginalData,'R.U.')
```

does the same as above but plots the data with a fixed z-axis (color bar scale) automatically derived from the min and max measured data.

3. Try using the three functions. Also try typing `help PlotEEMby1`, `help PlotEEMby4`, and finally `help PlotEEMby4FixZ`. Some instructions on how to use the functions are printed to the Command Window. This is true for all functions in MATLAB (e.g. try typing `help load`).
4. Surface plots can also be plotted in a similar way using the `PlotSurfby1`, and `PlotSurfby4` functions.

C. Cutting the region of the spectra influenced by scatter peaks.

1. This step creates a new copy of the data where the wavelengths influenced by scatter peaks have been cut and replaced with missing values or zeros. Type

```
[CutData]=EEMCut(OriginalData,20,20,NaN,NaN,
'No')
```

the data will be cut and then plotted so that the EEMs before and after the cut can be compared. The graphs will plot automatically from the first sample to the last. The function deletes the data in the region of no fluorescence (where emission wavelength is less than excitation wavelength) and the regions greatly influenced by first order scatter (where Rayleigh and Raman peaks dominate the signal) and replaces them with missing values (“NaN” (Not A Number) in MATLAB). Additionally a region of zeros is inserted to assist the PARAFAC modeling.

2. Type

```
help EEMCut
```

to read an explanation of what this function does.

3. Experiment with changing the input values for `(20,20,NaN,NaN,'No')` and observe how the data is cut differently.
4. Before proceeding with the tutorial type

```
[CutData]=EEMCut(OriginalData,20,20,NaN,NaN,'')
```

so that the data is processed appropriately for the rest of the tutorial. This will cut the data, but not plot the results.

D. Initial explorative data analysis and outlier identification.

In this step, a series of PARAFAC models are run in order to explore the data for outlier samples, noisy wavelengths, or other potential problems with the data that are not easily identified by visual analysis of the EEM plots. The step is structured into two tests: one on the original (complete) data and a second on a modified data set where outliers have been removed. The dataset can be reduced in size by selecting every other Emission wavelength, (i.e. 300, 304, 306 ....nm, instead of the measured 300, 302, 304, 306. This will speed the mod-

eling process considerably, without necessarily notably influencing the outcome (except the spectral resolution of the loadings). This is not done here but try it out later. The questions to be addressed here are;

- Are there specific samples or wavelengths that are influencing the model fit much more than the others?
- What seems to be the appropriate "ball park" number of PARAFAC components (i.e. the number of distinct fluorescent phenomena)? 3-5 components, 8-10 ? etc. When do the spectra of the components found begin not to look like organic fluorophores but more like scatter peaks or noise?

1. Type

```
help OutlierTest
```

   into the Command Window and read the explanation of the function.
2. Type

```
[Test1]=OutlierTest(CutData,2,1,7,'No','No')
```

   to perform the first test. After pressing any key this will run a series of models from 2 components to 7 components. The results from these initial five models (2, 3, 4, 5, 6 and 7 component models) can be evaluated in many ways but for this tutorial we will use two types of plots; loadings and leverages. These tests will take approx 3 minutes for your computer to run.
3. Type

```
help PlotLoadings
```

   and read about the function then type

```
PlotLoadings(Test1,2)
```

   and a figure will be created showing the scores and loadings of the model (*a*, *b* and *c* in Equation 1).
4. Plots *b* and *c* show the emission and excitation loadings of the two components. Plot *a* shows how the concentration of the two components varies between samples. Check that the loadings look reasonable, i.e. that they are smooth and spectral appearance that corresponds well with your understanding of the data.
5. Another useful plot is that of leverages. Type

```
help PlotLeverage
```

   and read about the function and then type

```
PlotLeverage(Test1,2)
```

   and a figure will be created showing the leverages. In the plot, look for samples that have extreme leverages indicating extreme and potentially outlying samples.
6. Tip: Using the function `PlotLL` creates a combination of both the leverage and loading plots. Try typing

```
PlotLL(Test1,2)
```

7. Now try creating the same plots for the other models (3, 4, 5, 6 and 7 components). After examining the loading plots it should be apparent that we need to begin to constrain the model. Some of the loadings are negative. For example in the 4-component model one of the components has negative values of a and appears to cancel out one of the other components. These "wrong" loadings could appear because a wrong number of components is used but pursuing the analysis under the tentative assumption that four components are valid, it is reasonable to try to circumvent the problem by forcing the parameter to be non-negative. In a more final model it is useful to focus on why such an added constraint is needed for the model to provide chemically meaningful results but this is not necessary at this stage.
8. Rerun the test but with non-negativity constraints by typing

```
[Test2]=OutlierTest(CutData,2,1,7,'Yes','No')
```

   Now plot the loadings and leverages. (You can compare the effect of applying non-negativity constraints by plotting the results from both tests, e.g. for a four component model type `PlotLL(Test1,4)` then `PlotLL(Test2,4)`). Afterwards plot the loadings and leverages from the other models.
9. The leverage plots seem to suggest that samples 5, 21, 30, 40 and 53 may be problematic. So we should now try plotting the EEMs of these samples and compare them to the others to see if they contain some measurement error. (e.g. use `PlotEEMby4(1,CutData,'R.U.')` and see if these sample differ from the others). Only samples 5 and 30 are clear outliers with measurement errors.
10. Now we can resample the data, removing samples 5 and 30. This is done using the `RemoveOutliers` function. (type help `RemoveOutliers` for a description). Next type

```
[Test3]=RemoveOutliers(CutData,[5 30],[],[])
```

   into the Command Window. This creates a new copy of the data called Test3 without samples 5 and 30.
11. Now re-run the models but on the new data

```
[Test3]=OutlierTest(Test3,1,1,7,'No','No')
```

and afterwards plot the leverage and loadings data for each model, e.g. `PlotLL(Test2,3)`. The leverages appear to have improved slightly. However it is still clear that we have problems with negative concentrations as mentioned earlier. This can be addressed by applying non-negativity constraints to the model. We can rerun the model and overwrite Test3 with non-negativity constraints applied. Type

```
[Test3]=OutlierTest(Test3,1,1,7,'Yes','No')
```

to do this. After the models have been fitted (this will take a little longer than earlier) try plotting the loadings and leverages for all the models again (e.g. `PlotLL(Test3,3)`). Now the loadings appear to be more logical (i.e. they do not exhibit negative fluorescence). The leverage plots suggest that we should check sample number 20 (originally sample 21 before we removed sample 5 earlier). Type

```
PlotEEMby1(18:22,Test3,'R.U.')
```

to plot samples 18 to 22 and see if it is apparent why sample 20 has a higher leverage. It seems to have a high amino-acid-like fluorescence and does not appear to contain any measurement error. For now we will leave it in the data set, however later you can try testing this by removing this sample and seeing what effect it has on the model outcome.[1]

12. The next function we will use is `EvalModel` which creates a series of graphs which we can use to evaluate the model fit by looking at the residuals (measured minus modeled data). Type

```
help EvalModel
```

and read its description. (If you prefer surface plots use the `EvalModelSurf` function.) To evaluate the model type

```
EvalModel(Test3,4)
```

and examine the plots made for the 4-component model. In particular examine the residual EEM. Ideally this should not contain any systematic signals and consist mostly of instrument noise. One can see that the four component model is generally adequate for the majority of the samples. There are however a handful of samples with an unexplained fluorescence in the region of 325 nm excitation and 400 emission. Next evaluate the other models (2, 3, 5, and 6 components) using the same function. Try also plotting the results from the models fitted before the outliers were removed (e.g. Test2) and examine the residuals for the identified outlier samples.

13. Next we can compare the results of two different models using the `Compare2Models` (or `Compare2ModelsSurf`) and `CompareSpecSSE` functions. Type

```
help Compare2Models
```

and

```
help CompareSpecSSE
```

for a description of each function. Then enter

```
Compare2Models(Test3,3,4)
```

to compare the 3 and 4 component models. Likewise compare the 4 and 5 component models and 5 and 6, and 6 and 7 component models. From these plots it appears that at least four components are required.

14. Next compare the loadings from the 4, 5, 6 and 7 component models by typing

```
PlotLoadings(Test3,4)

PlotLoadings(Test3,5)

PlotLoadings(Test3,6)

PlotLoadings(Test3,7)
```

Here one can see how the components are evolving as the model increases in complexity. The components from each model are generally very similar and with each step one component is split into two in order to model the data better and remove systematic variability left in the residuals. This a positive result and indicates consistency between the models rather than random local minima results.

15. Next the spectral sum of squared error can be examined using the `CompareSpecSSE` function. Try entering

```
CompareSpecSSE(Test3,3,4,5)
```

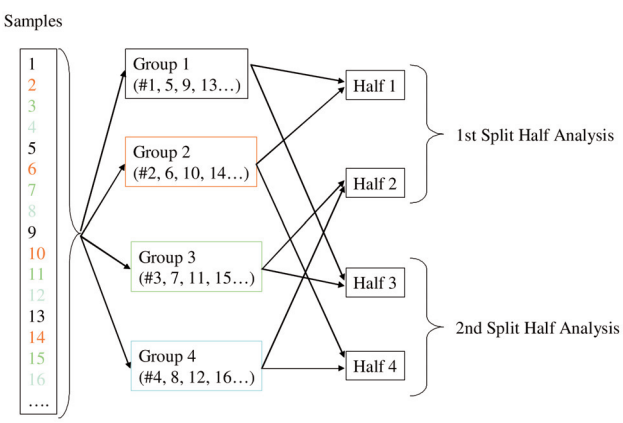
The sum of squared residuals in the excitation and emission directions for three different models are plotted. Try also comparing the 4, 5 and 6 component models and finally the 5, 6, and 7 models. From these results it seems clear that the step from the 6 to 7 component model offers little improvement of fit suggesting that 6 or fewer components are adequate for this data.

[1] Create new sub set of the data: `[Test4]=RemoveOutliers(CutData,[5 21 30],[],[])`
Run models on new data: `[Test4]=OutlierTest(Test4,1,1,6,'Yes','No')`
Plot and compare loadings: `PlotLoadings(Test2,4)` `PlotLoadings(Test4,4)`

16. After this stage of the analysis we have identified and removed the suspected outlier samples and found that the correct number of components required to model this data lies between 4 and 6 components. Now it is time to attempt to validate the models.

E. Split Half analysis and validation

Now the data will be split (divided) into halves. Instead of just making one type of split, two different splits are made, each dividing the data in a different way. This allows us to run two split half analyses and to use the one that performs best. Sometimes a split can be sub-optimal with an uneven distribution of types of samples between the two halves. Ideally the split should be as "random" as possible. The diagram below shows how the data is split using the `SplitData` function in the toolbox.



1. Type

```
help SplitData
```

to read an explanation of what the function does. Then enter

```
[AnalysisData]=SplitData(Test3)
```

into the Command Window. Four graphs are plotted which show the sum of squared values for each half of the data in the excitation and emission directions. The curves for each pair of halves should be similar (1-2, 3-4).

2. Next perform the split half analysis using the `SplitHalfAnalysis` function. There are several options to choose from. For a description type

```
help SplitHalfAnalysis
```

Now enter,

```
[AnalysisData]=SplitHalfAnalysis(AnalysisData,
(3:7),'MyData.mat')
```

to carry out an analysis. This will fit models with 3 to 7 components to the data. This step will take 10-20 minutes, depending on the speed of your computer.

3. Now the results of the analysis can be examined using `SplitHalfValidation` function (enter `help SplitHalfValidation` for an explanation). This function mathematically compares the excitation and emission loadings of the models run on separate splits of the data using Tucker Congruence Coefficients as described by (Lorenzo-Seva and Berge, 2006) and states in the Command Window whether the model is validated or not. Additionally the emission and excitation loadings from each half are plotted so you can compare them visually. Type

```
SplitHalfValidation(AnalysisData,'1-2',3)
```

to see the results for the 3 component model on the first split (1-2). You will find that it is validated. Now try for the other component models (3 to 7) and for the other split (3-4). You will find that four components can be split half validated.

F: Analysis using random initialization

Next a series of models will be fitted to the whole data using a random initialization of the model. We need to ensure that the model we derive is in fact the least squares result and not a local minimum. The PARAFAC model is fitted using an alternating least squares algorithm and as with other iterative fitting procedures (e.g. a simple non-linear regression) it can be influenced by its starting estimates. In this step a random initialization procedure is used to fit many models with the same number of components, so that we find the true least squares result (best fit).

1. To read about the function type

```
help RandInitAnal.
```

2. Next type

```
[AnalysisData]=RandInitAnal(AnalysisData,4,10)
```

to run 10 four component models using random initialisation. This will take 15-20 minutes.

3. Once the modeling is finished a plot showing the sum of squared error from each model is plotted. The model with the least squares result is highlighted with a green circle. (The plot can be re-plotted without rerunning the analysis using the `RandInitResult` function).
4. Examine the loadings and fit of the model using the `PlotLL` and `EvalModel` functions you used earlier (e.g. `PlotLoadings(AnalysisData,4)`, `EvalModel (AnalysisData,4)`).

5. Finally check to see that the emission and excitation loadings are the same as those found during the split half validation earlier. This should always be the case, however it is advisable to check. This can be done both visually by re-plotting the split half analysis results

```
SplitHalfValidation(AnalysisData,'1-2',4)
```

6. This can also be done mathematically by using the Tucker Congruence Coefficients described earlier (type `help TCC` for an explanation then

```
TCC(AnalysisData.Model4,AnalysisData.Split(1)
.Fac_4)
```

7. Evaluate the model fit one last time using the `EvalModel` function. We have now arrived at a robust four component model for the data. Try to test and see if the five and six component models can be validated in this way.

G:Create plots of components

To create surface or contour plots of each component, the `ComponentEEM` or `ComponentSurf` functions can be used. Each component will be labeled plotted in a separate window. Type `help ComponentEEM` or `ComponentSurf` for assistance.

H: Export data out of MATLAB

Now that a six component model has been fitted and validated to the data, the results can be exported out of MATLAB if required using the `ModelOut` function. This will create an excel file with the fluorescence intensities of each component in each sample and the emission and excitation loadings of each component. For example type

```
[FMax,B,C]=ModelOut(AnalysisData,4,'C:\MyPara
facResults.xls')
```

These data are also available in the MATLAB workspace as FMax, B and C respectively, should one want to carry on working in MATLAB.

References,

Lorenzo-Seva, U., Berge, J.M.F.T., 2006. Tucker's congruence coefficient as a meaningful index of factor similarity. Methodology 2 (2): 57–64.