FDOMcorr Toolbox Help Files

Kathleen R. Murphy

Revision 1.6 (2012-July-20)

Disclaimer: This product is offered freely without warranty of any kind. It was developed and tested using MATLAB vers. 2010a. It is possible that some earlier versions of MATLAB could have trouble executing some lines of code.

Report bugs and version incompatibilities to: krm@unsw.edu.au

Contents

1. STARTING OUT WITH THE TOOLBOX

To use the toolbox, extract the zipped M-files and save them together in a folder called FDOMcorr.

To add the toolbox to your MATLAB path, you can adjust your path via the MATLAB GUI or use the command "addpath".

For example:

```
> addpath('C:\EEMdata\FDOMcorr')
```

To confirm that you have access to the toolbox, try to locate it with the command "which", e.g.

```
> which FDOMcorrect
```

Instructions for using each M-file are obtained by typing "help" followed by the filename, e.g.:

```
> help FDOMcorrect
```

A template is included (FDOMcorr template.m) which lays out all of the steps for assembling and correcting your EEMs using the toolbox. If you are not already very competent at programming in MATLAB, it is **strongly recommended** that you use this template. The template lists the steps in the appropriate order and includes some tricks and hints that may make the process easier, including suggestions on organizing and utilizing information in your sample log and adjusting data to compensate for sample dilution. It is recommended that you save it with a new name and modify it as necessary to suit your data files.

When starting out with the template, open the m-file then select only a few lines of code to run at a time. This will 'walk you through' the EEM correction process and make it easier to troubleshoot when errors occur.

The FAQ section has further help for getting started, including suggestions regarding how to treat missing data files.

2. IMPORTING SCANS INTO MATLAB WITH READINSCANS.M

The MATLAB function "ReadInScans.m" loads individual scans and merges them into a 2D matrix in preparation for further processing. The program assumes that all files of the specified type (.xls, .xlsx, .txt or .csv) that are located in the current directory are part of the dataset and will attempt to (a) load them one by one in alphabetical order, then (b) merge them into a single 2D matrix (a large table).

Files that are not in the appropriate format will cause the program to abort by error. When reading in some file types (see 'help' documentation), if there is any text (rather than numbers) in the file, it will cause an error/abort.

Scans can be in rows or in columns, with one row/column of wavelengths followed by one row/column of data. The smallest size that can be imported is one wavelength and one data point (used, for example, to load quinine-sulfate slopes). If there are two data columns and more than two data rows, the data are assumed to be arranged in columns (wavelength, data). The output variables 'S' and 'W' list samples in rows, with corresponding sample names in 'filelist'. If 'W' is the same (within 0.5 nm rounding error) for each sample, then the variable 'wave' corresponds to all the scans contained in 'S'.

Examples

The three examples below show how to read in a range of scans containing Raman, absorbance or calibration data.

```
% [S,W,wave,filelist]=ReadInScans(type,format,range,display,outdat);
```

Example 1: A set of individual water Raman emission scans are contained in .csv files located in the directory 'RamanScans'. Each consists of two columns of numbers (wavelength, intensity) in cells A3 to B83. Plots are flashed on the screen for 0.5s as each file is imported. The output data include a matrix of merged scans (S_R), the wavelengths corresponding to each of them (W_R), and a list of imported files (filelist_R). If all the rows in W_R are the same, a single vector appropriate for all files is stored (wave_R). Specifying outdat=1 directs MATLAB to automatically export these data to an Excel file called Outdata_Raman350.xls .

```
cd 'C:\EEMdata\RamanScans'
type='Raman350', format = 'csv'; range='A3..B83';display=0.5;outdat=1;
[S_R,W_R,wave_R,filelist_R]=ReadInScans(type,format,range,display,outdat);
```

Example 2: Absorbance scans located in the directory 'AbsScans' each consist of two columns of numbers (wavelength, absorbance). No data range is specified when importing .txt files, the file must consist only of numbers, and all data in each file are imported (so each data file must contain only two rows or two columns of numbers and no text). Plots are flashed on the screen for 0.25s as each file is imported. Output data are written to a text file called Outdata_Abs.xls.

```
cd 'C:\EEMdata\Absscans'
type='Abs', format = 'txt'; range=[];display=0.25;outdat=1;
[S_abs,W_abs,wave_abs,filelist_abs]=ReadInScans(type,format,range,display,outdat);
```

Example 3: Calibration data consist of slope measurements from quinine-sulfate dilutions series. Data consist of a single emission wavelength in cell A3 and a single data point in B3 representing the slope of a linear dilution series at Ex/Em=350/450 nm. No plots are displayed. The output data are automatically written to a text file called Outdata_QS.xls.

```
cd 'C:\EEMdata\QS'
type='QS', format = 'csv'; range='A3..B3';display=0;outdat=1;
[S_qs,W_qs,wave_qs,filelist_qs]=ReadInScans(type,format,range,display,outdat);
```

3. IMPORTING EEMS INTO MATLAB WITH READINEEMS.M

The MATLAB function "ReadInEEMs.m" loads individual EEMs into a 3D matrix in preparation for further processing. The program assumes that all files of the specified type (xls, xlsx or csv) that are located in the current directory are EEMs and will attempt to (a) load them one by one in alphabetical order, then (b) merge them into a single 3D matrix.

Files that are not EEMs or that are not in the appropriate format will cause the program to abort by error. If there is text (rather than numbers) within the range you are uploading it will cause errors when reading in some file types (e.g. csv). If MATLAB has trouble reading in text data from your files, remove the text or exclude the affected cells from the imported range.

Before further processing with FDOMcorrect.m, check that each row of `Exmat` and column of `Emmat` are identical (within rounding error) before defining the global vectors `Em` and `Ex`, for example:

```
Ex=Exmat(1,:) %Ex defined as for the first sample
Em=round(Emmat(:,1)) %Em as for the first sample but rounded to the nearest 1 nm
```

Note that if either Em or Ex was not loaded automatically, then `Exmat` and/or `Emmat` will contain dummy vectors (identically numbered from 1:N wavelengths). In this case, refer to the original EEMs before defining appropriate vectors for `Em` and `Ex`.

Examples

Read in a dataset of 55 uncorrected EEMs from the directory "`c:/mydata/eems`". The EEMs were exported from a Varian Cary Eclipse fluorometer and as *.csv files with two rows of text headers followed by data that have emission wavelengths in every second column. Plots of each EEM are shown and raw data are automatically saved to the current directory. Emission wavelengths (Emmat) are read in as numbers, and excitation wavelengths (Exmat) are extracted from the first row of text.

```
>> cd 'c:/mydata/eems'
>> [X,Emmat,Exmat, filelist_eem,outdata]=ReadInEEMs(2,'csv','A3..CD113',[0 1],1,1);
```

Read in a dataset of 18 uncorrected EEMs from the current directory. The EEMs were exported from a FluoroMax 3 to *.xls files with excitation wavelengths in the first row and emission wavelengths in the first column. No plots or automatically saved data files. Then change to a directory containing 5 corresponding blank EEMs and read these in too:

```
>>[X,Emmat,Exmat,filelist_eem,outdata]= ReadInEEMs(1,'xls','A1..AT80',[1 1],0,0);
>> cd 'c:/mydata/blanks'
>>[X_b,Emmat_b,Exmat_b,filelist_b,outdata_b]= ReadInEEMs(1,'xls','A1..AT80',[1 1],0,0);
```

Read in a dataset of uncorrected EEMs from the current directory. The EEMs are *.csv files with no excitation or emission wavelengths. Don't show plots but do export the data:

```
>> [X,Emmat,Exmat,filelist_eem,outdata]= ReadInEEMs(1,'csv','A1..AS79',[0 0],0,1);
```

If you have selected the option to export data, but numerical Ex or Em headers were either missing or excluded, the program will prompt you to enter the wavelength range for Ex and Em. For example:

```
Specify the excitation wavelength range, e.g. 250:5:450:     >> 240:5:400
Specify the emission wavelength range, e.g. 300:2:600:       >> 300:2:550
```

Based on the data in Emmat and Exmat, specify Ex and Em before continuing with FDOMcorrect.m. For example:

```
>> Ex =240:5:400
>> Em =300:2:550
```

4. PAIRING SAMPLES WITH MATCHSAMPLES.M

The MATLAB function " MatchSamples.m" creates a new dataset X2m from input datasets X1 and X2 which pairs each sample in X1 with a sample in X2, according to information in a data log. Typically, X1 is a dataset with a large number of samples (e.g. sample EEMs) and X2 is a data set with fewer samples (e.g. daily blank EEMs) , such that a sample in X2 might be paired with more than one sample in X1.

Input variables include a 2 column data log ('PairedList') containing the name of each sample in X1 (first column) that is to be matched with one of the samples in X2. All the samples in X1 must be listed in the first column of PairedList and all the samples to be matched with X1 according to the second column of PairedList must be in X2. However, the PairedList can include samples that are not in either X1 or X2, since it will only attempt to match the samples that appear in the file list for X1.

The output variable `FL1` will be identical to the input variable filelist1, except that any leading rows of blank data that were present in filelist1 will have been deleted. If the two file lists FL1 and filelist1 are identical then typing: > isequal(F1, F2) will produce the number 1. If they are not identical, replace filelist1 with FL1 (check - it should have the same number of rows as X1). The same applies to filelist2 and FL2.

The output variable `PL` is the file list corresponding with X2m.

Example

First, read in the sample log for 33 samples from SampleLog.xls (sheet named 'Cruise1').

```
[LogNum,LogTXT]=xlsread('SampleLog.xls','Cruise1','A2:K34')
```

An example log file in excel might look like this (first few rows shown):

Sample	Raman	Absorbance	Quinine Sulfate	Blank	DF	La	Lf	flag	Notes
EEM1.xls	Rama1.csv	Abs_1.csv	QS_Oct_10.xls	BK_Oct30.xls	1	1	1		
EEM2.xls	Rama1.csv	Abs_2.csv	QS_Oct_10.xls	BK_Oct30.xls	10	1	1	1	Diluted
EEM36.xls	Rama6.csv	Abs_2.csv	QS_Oct_10.xls	BK_Oct30.xls	10	1	1	1	Diluted
EEM20.xls	Ramb4.csv	Abs_20.csv	QS_Nov_10.xls	DummyEEM.xls	1	1	1	2	Dummy blank

In this case, the columns of `LogTXT` contain sample notes in the last column and complete filenames for:
1. EEMs, 2. Raman scans, 3. Absorbance scans, 4. QS scans, 5. Milli-Q water blanks,

`LogNum` contains the numerical data in 4 columns:
1. DF: dilution factors, 2. La: Absorbance path length, 3. Lf: Fluorescence path length, 4. flag: coded error flags

Next, create matched datasets for use in FDOMcorrect.m

```
Pair_EEM_R=[LogTXT(:,1) LogTXT(:,2)]
Pair_EEM_abs=[LogTXT(:,1) LogTXT(:,3)]
Pair_EEM_qs=[LogTXT(:,1) LogTXT(:,4)]
Pair_EEM_bk=[LogTXT(:,1) LogTXT(:,5)]
Pair_BLK_R=[LogTXT(:,5) LogTXT(:,2)]
Pair_QS_R=[LogTXT(:,4) LogTXT(:,2)]
```

Match Sample EEMs and Raman (UV) scans

```
[Xr,FL1,FL2,PL]=MatchSamples(filelist_eem,filelist_R,Pair_EEM_R,X,S_R);
```

Match Sample EEMs and Absorbance scans

```
[Xabs,FL1,FL2,PL]=MatchSamples(filelist_eem,filelist_abs,Pair_EEM_abs,X,S_abs);
```

Match Sample EEMs and Blank EEMs

```
[Xbk,FL1,FL2,PL]=MatchSamples(filelist_eem,filelist_blk,Pair_EEM_bk,X,X_b);
```

Match Sample EEMs and QS slopes

```
 [Xqs,FL1,FL2,PL]=MatchSamples(filelist_eem,filelist_qs,Pair_EEM_qs,X,S_qs);
```

Match QS slopes and Raman (UV) scans

```
[Xqs_R,FL1,FL2,PL]=MatchSamples(filelist_qs,filelist_R,Pair_QS_R,S_qs,S_R);
```

Match Blank EEMs and Raman (UV) scans

```
[Xbk_R,FL1,FL2,PL]=MatchSamples(filelist_blk,filelist_R,Pair_BLK_R,X_b,S_R);
```

Spot check that the samples have been correctly matched before proceeding!

Troubleshooting error messages:

If the dataset uploaded with ReadInEEMs or ReadinScans that corresponds with Filelist1 contains samples that are not in your sample log, an error will be produced like this.

```
Error! The following filenames appearing in Filelist1
were not matched with any filenames in Paired List (1st column)
0786un.csv
726UN.csv
Press any key to continue...

To print the filenames in list 1 and 2 to Excel type "9", otherwise press enter:
??? Error using ==> MatchSamples at 118
some samples could not be matched
```

The message indicates that two of the files loaded with ReadInEEMs/ReadinScans are not mentioned in the sample log. To fix the error, either add the missing files to the sample log, or else remove the file(s) that can't be matched from their folder(s) and reload the data using ReadInEEMs or ReadInScans.

To print out the file lists showing which samples were matched and which not, enter 9 when directed by the earlier statement. The file will be located in the current folder (identify the name of the current folder by typing > pwd) .

Note that the matching is case-insensitive, so an EEM named `726UN.csv` but called `726un.csv` in the log will produce a match.

5. CORRECTING EEMS USING FDOMCORRECT.M

The MATLAB function " FDOMcorrect.m" implements fluorescence EEM correction algorithms as described in Murphy et al. (2010 *Environ. Sci. Technol.* **2010,** *44*, 9405–9412, doi: 10.1021/es102362t).

Type: help FDOMcorrect for full definitions of input and output variables and their formatting specifications. Briefly descriptions follow:

```
%% INPUT VARIABLES
Ex  -1D row vector of excitation wavelengths corresponding to EEMs.
Em  -1D row vector of emission wavelengths corresponding to EEMs.
X   -3D data cube of fluorescence intensities in primary samples.
Emcor - is either of (a) or (b):
      (a) Two-column 2D matrix (wavelength x correction factor)
      (b) Many-column 2D matrix (samples x Emcor) of correction factors
Excor - is either of (a) or (b):
      (a) Two-column 2D matrix (wavelength x correction factor)
      (b) Many-column 2D matrix (samples x Excor) of correction factors
W   - is either of (a) or (b):
      (a) a single water Raman emission scan
      (b) 2D matrix (samples x Em) of water Raman scans
RamOpt -  optional user-defined excitation and emission wavelengths for
     calculating Raman areas. Default values are [350 381 426].
landa - Excitation wavelength corresponding to all Raman scans, if not the
      default value of 350 nm.
A   - 2D matrix (samples x wavelengths) of Absorbances (decadal form)
B   - is either of (a) or (b):
      (a) single blank EEM (Em x Ex) to be used for all samples;
      (b) 3D data cube (samples x Em x Ex) of blank EEMs
T  - is either of (a) or (b):
      (a) T=[] indicating it is automatically extracted from B;
      (b) 2D matrix (samples x Em) of water Raman scans for blanks
Q  - is either of (a) or (b):
      (a) slope (uncorrected, unnormalized) of a linear QS dilution series
      (b) a vector of slopes of QS dilution series
Qw  - is either of (a) or (b):
      (a) a water Raman emission scan (Ex==landa) for the QS dilution series
      (b) 2D matrix (samples x Em) of water Raman scans for the QS series

%% OUTPUT VARIABLES

 XcRU - 3D matrix of corrected EEMs in Raman units, including inner filter
        correction and blank subtraction steps, if applied.
 Arp -  vector of Raman Areas in arbitrary units.
 IFCmat -3D matrix of inner filter correction factors.
 BcRU  -3D matrix of corrected blanks.
 XcQS - 3D matrix of corrected EEMs in QS units, including inner filter
        correction and blank subtraction steps, if applied.
 QS_RU - vector of conversion factors between QS and RU calibrated EEMs.
```

Examples

The simplest correction method requires 6 inputs and 2 outputs to produce spectrally corrected EEMs in RU units.

By default, Raman areas are calculated from the water Raman scan at λ_{ex} = 350, integrated over λ_{em} = 381-426 nm. This method excludes inner filter correction, blank subtraction and QS calibration:

```
>[Xc Arp]=FDOMcorrect(X,Ex,Em,Emcor,Excor,W);
```

If performing additional correction steps, all 12 input variables and 6 output variables must be specified and irrelevant variables appearing on the left hand side will be outputed as NaNs. For example:

Inner filter correction and QS calibration (`BcRU outputted as NaN`):

```
>[XcRU Arp IFCmat BcRU XcQS QS_RU]=FDOMcorrect(X,Ex,Em,Emcor,Excor,W,[],A,[],[],Q,Qw);
```

Inner filter correction and blank subtraction (`XcQS, QS_RU outputted as NaN`):

```
> [XcRU Arp IFCmat BcRU XcQS QS_RU]=FDOMcorrect(X,Ex,Em,Emcor,Excor,W,[],A,B,[],[],[]);
```

Blank subtraction and QS calibration (`IFCmat outputted as NaN`):

```
> [XcRU Arp IFCmat BcRU XcQS QS_RU]=FDOMcorrect(X,Ex,Em,Emcor,Excor,W,[],[],B,[],Q,Qw);
```

If normalizing to user-specified parameters for Raman area, all 12 input variables and 6 output variables must be specified and irrelevant variables appearing on the left hand side will be outputted as NaNs. For example, to normalize fluorescence intensities to the *height* of the Raman peak at 275/303 nm and perform QS calibration (`IFCmat, BcRU outputted as NaN`):

```
>RamOpt=[275 303 303]
>[XcRU Arp IFCmat BcRU XcQS QS_RU]=FDOMcorrect(X,Ex,Em,Emcor,Excor,W,RamOpt,[],[],[],Q,
Qw);
```

Troubleshooting FDOMcorrect

Method statements are outputted to screen and saved to file. These document the steps implemented when generating the data in the rest of the file. If different wavelengths were used to calculate Raman areas than were specified by the user, this will also be documented.

Note that old versions of OutData_FDOM.xls are NOT deleted automatically when you run the program. If you run the program again the new data will overwrite some or all of the data in the existing file. You can delete old versions of the file manually or from the current directory by typing at the command prompt:

```
> delete OutData_FDOM.xls
```

Error messages alert the user to incorrectly formatted input files or to incompatibilities between input data and the function request statement. In the final example above using Raman *height* normalization, if Em = 303 nm is not among the measurement wavelengths in each of the inputted Raman files (in this case, W and Q_w), the program will abort by error. When performing Raman *area* normalization, however, if one or more Raman files do not include the exact wavelengths requested as upper and lower emission wavelengths, the program will automatically seek an alternative wavelength range, print it to screen, then continue processing. Note that if this occurs, it is advisable to run the program again specifying an emission range that is suitable for all Raman files, so that Raman areas are calculated consistently across the entire dataset.

6. DETERMINING THE RAMAN PEAK INTEGRATION RANGE USING RAMANINTEGRATIONRANGE.M

This function takes a matrix of water Raman scans and a vector of corresponding excitation wavelengths and calculates the upper and lower limits to the Raman peaks. The method is described in Murphy et al. (2011, Applied Spectroscopy, 65 (2)). The integration range determined by this method can be used as input parameters to the function FDOMcorrect.m.

Settings for tolerance, sequence and halfwidth can be user-specified or else the default values will be used (0.01, 8, 1800cm^{-1} respectively).

For further information on this function, type:

```
>help RamanIntegrationRange
```

Examples

For a matrix of 10 Raman scans (RAMmat) each obtained at an excitation of 350 nm, produce a list of start and end values for peaks (IR), the median range (IRmed) and a difference matrix for each individual scan relative to the median (IRdiff). Use the default settings for tolerance, halfwidth and sequence.

```
>[IR IRmed IRdiff] = RamanIntegrationRange(RAMmat,1:10,350);
```

For a matrix of 3 Raman scans each obtained at different excitation wavelengths, produce a list of start and end values for peaks (IR).

```
>[IR] = RamanIntegrationRange(RAMmat,1:3,[300 320 350]);
```

For a matrix of 2 Raman scans called 'a' and 'b', each obtained at different excitation wavelengths, produce a list of start and end values for peaks (IR). Use sequence =6 and default settings for tolerance and halfwidth.

```
>[IR] = RamanIntegrationRange(RAMmat,['a','b'],[300 320],[],6);
```

For a matrix of 3 Raman scans called 'a', 'b', and 'c', each obtained at 300 nm excitation, produce a list of start and end values for peaks (IR), the median range (IRmed) and a difference matrix for each individual scan relative to the median (IRdiff). Use tolerance = 3% and default settings for sequence and halfwidth.

```
>[IR IRmed IRdiff] = RamanIntegrationRange(RAMmat,['a','b','c'],300,[],[],0.03);
```

7. FAQs

Error Messages

I am getting error messages. Where can I get help on solving these?

See the next section of this document (trouble shooting).

Basic help on using MATLAB can be found at http://www.models.kvl.dk/IntroMatlab

Importing data files containing text

I am using ReadInEEMs to import Varian data files. Why arent the excitation wavelengths imported automatically?

Although in earlier versions of ReadInEEMs, only the numerical data were imported automatically, in version 1.1 the Ex wavelengths are extracted automatically from the first header row. If this process is not working, check you have the right copy of ReadInEEMs (to obtain the path, type: > which ReadInEEMs) and also check the EEM file producing the error to make sure that Ex wavelengths appearing in row 1 match with the numerical data in rows 3 onward.

My data files are in ascii .txt format but contain a mixture of numbers and text. How can I import these?

There is no built-in Matlab function that will do this although there are some very good programs donated by the MATLAB user community designed for this purpose. They can be downloaded freely from the mathworks file exchange (e.g., http://www.mathworks.com/matlabcentral/fileexchange/10946-readtext). Another option is to open your files and save them each as comma delimited text (.csv) or excel (.xls). It is possible to read in text data from .xls and .xlsx files, and to exclude cells with text from the imported range when reading in .csv files.

Coping with method variations

I have been sampling over a period of time during which I adapted my methods for collecting EEMs, absorbance and Raman scans. For example, some of my EEMs were in 2-nm wavelength intervals and others were in 4-nm intervals. How can I use the toolbox with my dataset?

You have two options for coping with method changes in your raw data files. The first is to create separate templates to run each consistent dataset separately. This might be the best choice if the datasets were created for different projects and you are happy to keep them separate. Also, if lots of aspects of your methods changed (e.g. EEM wavelengths, correction factors, Abs wavelengths, Raman wavelengths) then you could end up with a long and unwieldy program. Processing datasets separately also does not prevent you from merging the corrected datasets if later on you want to analyze them together, and in many cases this would involve less steps than if you merge lots of different types of data-files at the outset.

The second option is to load different datasets separately, but join them together before correcting the data, producing a single output dataset. This option might be preferable if the method changes were minor and affected a small number of file types, and you don't want to produce separate outputs. For this, you need to repeat the code in the data loading steps as many times as there are different Ex and Em wavelengths in the raw data files and follow this up with interpolation and merging steps.

For example, imagine you have an EEM dataset in which Ex was always the same but some of the Em wavelengths were in 2-nm intervals and the others were in 4-nm intervals. You want your output to be in 2-nm intervals. The two types of EEM are in separate folders on your hard drive, called method1 and method2:

```
%load the datasets for method1 and method2 separately
>cd 'C:\EEMdata\method1
>type=2;format ='csv'; range='A3..CD113';headers=[1 1];display=1;outdat=1;
>[X1,Emmat1,Exmat1,filelist_eem1,outdata1]=ReadInEEMs(type,format,range,headers,display,outdat);
>Ex1=Exmat(1,:); Em1=Emmat(:,1); %Em1 = (280:2:500)'
```

```
>cd 'C:\EEMdata\method2
>type=2;format ='csv'; range='A3..CD56';headers=[1 1];display=1;outdat=1;
>[X2,Emmat2,Exmat2,filelist_eem2,outdata2]=ReadInEEMs(type,format,range,headers,display,outdat);
>Ex2=Exmat(1,:); Em2=Emmat(:,1); %Em2 = (280:4:500)'

%interpolate the Em2 wavelengths to agree with Em1=(280:2:500)'
>X2i = interpn(1:size(X2,1),Em2,Ex2,X2,1:size(X2,1),Em1,Ex2);

%merge the EEM datasets creating a single X, filelist, etc.
>filelist_eem = char(filelist_eem1,filelist_eem2)
>X=[X1;X2i];
>outdata=[outdata1;outdata2(3:end,:)];
>Ex=Ex1; Em=Em1;
```

We collect EEMs using a consistent methodology but update our Excor correction files and dilution series approximately once a month, which means that a particular dataset may include samples with a range of calibration and correction factors. What is the best way to deal with this?

The FDOMcorrect.m program allows you to specify a range of dilution series for various samples, however, it expects only one set of Ex and Em correction files per dataset. If Excor (and/or Emcor) varies within a dataset but there is no other reason to divide up the dataset, consider doing the following:

Set up a single template where the correction step is repeated several times, each time using a different unique combination of Excor and Emcor. Don't automatically save the output data, since it will be overwritten each time FDOMcorrect is run. The final output is cherry-picked from the various versions of corrected data, in order that the right Excor and Emcor are used for each sample. The information on which correction file to use in each case will need to come from somewhere, e.g. the log file. Here is an example of how this could be implemented for a 5 sample dataset that used three different Excor factors:

```
%From the log file, import ExCorID which is a vector of filenames indicating which Excor file
should be used for each sample. This will be turned into a logical matrix (ones & zeros)
matching each Excor file with each sample.

> ExCorID=char('October5','Apr10','October5','Jan15','Apr10','Apr10')%Excor files for 5 samples

> uExCorID=unique(ExCorID,'rows') % Excor1-3 ='Apr10','Jan15','October5',

%In the next correction step, you MUST define Excor1-3 in the same order as uExCorIDs, so that
the first row of uExcorID corresponds to Excor1, and so on. Otherwise, you will use the wrong
correction files. In the example below, Excor1 is the correction factors from April 10.

>[XcRU1 Arp1 IFCmat1 BcRU1 XcQS1 QS_RU1]=FDOMcorrect(X,Ex,Em,Emcor,Excor1,W,[],A,[],[],Q,Qw);

>[XcRU2 Arp2 IFCmat2 BcRU2 XcQS2 QS_RU2]=FDOMcorrect(X,Ex,Em,Emcor,Excor2,W,[],A,[],[],Q,Qw);

>[XcRU3 Arp3 IFCmat3 BcRU3 XcQS3 QS_RU3]=FDOMcorrect(X,Ex,Em,Emcor,Excor3,W,[],A,[],[],Q,Qw);

%Merge the three corrected datasets, cherry-picking the right data. The code below demonstrates
how to obtain XcRU. Use similar lines of code for the other variables (e.g. XcQS). Three lines
must be replicated and modified for each new variable (indicated with purple comments).

> XcRU = NaN*ones(size(XcRU1)); %Create the output file.

> for j=1:size(uExCorID,1)

>      ExCl=strcmp({uExCorID (Yclasses(j,:))}, ExCorID);  % 1s = matches on the jth Excor file

>      XcRUi=eval(['XcRU' num2str(j)]); %XcRU1, XcRU2, XcRU3,...

>      XcRU(ExCl,:,:) = XcRUi(ExCl,:,:); %insert the data for the jth correction file into XcRU

>      ExC_log(:,j)= ExCl; %5 rows x 3 columns of 1s (match) or zeros (no match)

> end
```

RAMAN NORMALIZATON

I collected blank EEMs but no Raman scans. How can I calculate Raman Areas?

If you do not have separate water Raman scans, you can extract them from your blank matrix. The steps are as follows:

(1) Read in the blanks with ReadInEEMs.m to define X_b, Exb and Emb as demonstrated in the FDOMcorr template.

(2) Match the blanks to your samples with MatchSamples.m

(3) Extract the Raman scans at Ex=350 with the following code:

```
> W = [Emb'; squeeze(X_b(:,:,Exb==350))];
```

IFE CORRECTION

Why does the template include a switch that prevents IFE correction of samples which did not have absorbance and fluorescence pathlengths of 1 cm?

The simplified IFE correction algorithm used in the program is valid only when both the absorbance and fluorescence measurements for a sample were made in a cell with path length of 1 cm. Similarly, the algorithm is inaccurate for highly concentrated samples (the program produces a warning if A254 > 0.3). For further information, see Ohno (2002, *Fluorescence Inner-Filtering Correction for Determining the Humification Index of Dissolved Organic Matter, Environ. Sci. Technol. 2002, 36, 742-746*).

I only want to perform IFE correction on part of my dataset. How do I switch it off for samples that don't need IFE correction?

You can either make dummy absorbance scans with zero absorbance and match these to the samples that dont need IFE correction, or you can apply a switch that sets some of the absorbance scans in matrix A to zero before using A in FDOMcorr.

Samples that have zero absorbance are not affected by IFE correction.

To make a switch, create a column in your sample log which specifies which samples need IFE correction and which do not. See the template where path length data imported from the sample log is used to activate just such a switch.

SPECTRAL CORRECTION

I dont want to perform spectral correction since my fluorometer does it automatically. How do I turn it off?

Run code to specify correction factors of 1 at each wavelength in Emcor and Excor, i.e.

Emcor=[Em ones(size(Em))]

Excor=[Ex' ones(size(Ex))']

MISSING DATA FILES

How can I deal with the fact that I occasionally have missing blanks?

You have two choices:

(1) Use a replacement file collected with identical instrument settings as close in time as possible

(2) Use a fake EEM (called, e.g. dummyEEM.csv) which looks the same as the other blanks but has zeros where the data should be, and treat it exactly the same way as a real data file.

It is recommended that you track samples where you have introduced dummy files using a comment and/or flag in your sample log. Note that it would not work to use a blank made of zeros if you need to use the blank to calculate Raman areas.

How can I deal with occasional missing Absorbance scans? Some samples in my dataset were optically dilute so I did not collect absorbance scans for these.

Similar to above, you can create a dummy scan that duplicates a real scan except that the data are replaced with zeros. Samples that have zero absorbance are not affected by IFE correction.

How can I account for the fact that some of my samples were diluted?

The FDOMcorr template demonstrates how to do this, using a column of dilution factors read in from your sample log. The column contains 1s if samples were undiluted and the dilution factor otherwise. For example, a sample diluted to 50% strength would have a dilution factor of 2 (50% =1/2).

SAVING DATA

I dont want to automatically save my data to disk when running FDOMcorrect.m. How do I turn this off?

Go to lines 82-86 of the program where it describes your File Export preferences, then set:

```
ExportFiles = false.
```

What is the best way to save my data?

The safest way is to delete any unwanted variables then save your entire workspace, especially because automatically saved variables may have different names to the ones you gave them in your workspace.

If you chose to automatically output data from ReadInEEMs and FDOMcorrect, note the following:

1. variables saved in RawData_FDOM.mat are always named: X Emmat Exmat filelist outdata
2. variables saved in CorrData_FDOM.mat are always named: XcRU XcQS Arp IFCmat BcRU QS_RU X Ex Em Emcor Excor W RamOpt A B T Q Qw
3. each time these m-files are run they will overwrite any existing data files in the same folder with the same name (i.e. CorrData_FDOM and RawData_FDOM).
4. if you used ReadInEEMs to import both samples and blanks, then a file called RawData_FDOM.mat will be written to each the sample and the blank directory, and each will contain variables with the same names
5. automatically saved variables may have different names to the ones you gave them in this workspace. For example, if you used A_1cm in FDOMcorrect, it will be called A in CorrData_FDOM.mat

Variables you might want that are not automatically saved include: XcRU100, XcQS100, A100, A_1cm, A_val and the matched up filelists: filelist_eem, PLr, PLabs, PLbk, PLqs, PLqs_R, PLbk_R

8. TROUBLE SHOOTING ERROR MESSAGES IN THE FDOMCORR TOOLBOX FOR MATLAB

(1) M-File not found

Problem: When you call a function, MATLAB produces and error like this:

```
??? Undefined function or method 'ReadInEEMs'
```

Solution: The function you are trying to call is not in your current directory nor specified in your path, so MATLAB does not know how to locate it. Add the folder containing the function to the MATLAB path. Type "help addpath" for more information. See FAQs for basic help with MATLAB.

(2) File matching error

Problem: When you run FDOMcorrect.m, MATLAB produces and error like this:

```
Error - Check that your VECTOR 1 (e.g. correction file) encompasses
 the full range of wavelengths in VECTOR 2 (e.g. emission scan)
Hit any key to continue...
 VECTOR 1 range is 200.07 to 600.01 in increments of 0.95.
 VECTOR 2 range is 200 to 400 in increments of 5.
 This error can usually be resolved by restricting the wavelength range of VECTOR 2,

 Be aware of rounding errors (e.g. 200.063 is NOT equivalent to 200 nm)
??? Error using ==> FDOMcorrect>MatchWave at 592
Error - abandoning calculations.
```

Solution: VECTOR 1 does not encompass the full range of wavelengths in VECTOR 2, because the smallest value in VECTOR 1 (200.07) is larger than the smallest value in VECTOR 2 (200). Essentially, you are trying interpolate the data in VECTOR 1 outside of the range of your data points.

To fix this, your choices are to (1) round the first wavelength in VECTOR 1 to 200 (you can round all of the wavelengths, although it is not necessary, since the data will be interpolated to match the wavelengths in VECTOR 2), or (2) restrict the wavelength range in VECTOR 2 (so VECTOR 2 becomes e.g. 205:5:400).

(3) Errors produced when writing Excel data to disk

Occasional problems exporting Excel files have been reported. It may be related to the version of MATLAB and/or the operating system in use. To test whether xlswrite works on your computer, see: `>help xlswrite`. If it does not, you can switch off the option to automatically export data to Excel (see FAQs). You can also try deleting old versions of OutData_FDOM.xls before running the script over again.

(4) Error when implementing Raman height normalization with FDOMcorrect.

Check that you have read in the right number of variables and placed them in the right locations (with commas separating each variable). Use the empty set [] as an input variable if you wish to use the default value for that variable.

See the help for FDOMcorrect which has examples and trouble-shooting tips.

object returned error code 0x800A03EC

(5) XLSWRITE error: "Object returned error code 0x800A03EC"

A number of things can cause this error, as documented in http://www.mathworks.com/support/solutions/en/data/1-3QJ5I3/index.html?solution=1-3QJ5I3. In particular, it occurs when attempting to write more than 256 columns of data to Excel.

With ReadInScans, if the input scans have data for more than 256 wavelengths, the wavelength resolution of data exported to Excel is automatically decreased by ignoring every second column of data. This has a no effect on subsequent data steps since all the wavelengths are retained in MATLAB. In most cases, omitting these data would also have negligible effects on data visualization and interpretation of the Excel data.

To prevent the omission of data when exporting to Excel, use input scans that have no more than 256 wavelengths. If your input scans are larger than this, one option might be to exclude unnecessary data points when calling ReadInScans. For example, exclude from the input range any absorbance data collected at wavelengths above and below the wavelength range that was used for collecting EEMs.