Richardson-Lucy bandpass correction

September 27, 2013

This is the documentation of the Python software package version 1.9x for Richardson-Lucy deconvolution of bandpass data. A list of changes compared to the last published version (1.8x) can be found in "*list of changes.txt*". The most important changes are a detailed documentation of all relevant Python methods (see *index.html*) and the support of the EXCEL Add-In pyXLL (downloadable from www.pyxll.com) to use the RichLucy methods from within Microsoft EXCEL.

The algorithm used here is based on Richardson-Lucy deconvolution as described in

Eichstaedt S., Schmaehling F., Wuebbeler G., Anhalt K., Buenger L., Krueger U. & Elster C. *Comparison of the Richardson-Lucy method and a classical approach for spectrometer bandwidth correction* **Metrologia**, vol. 50, 2013

1 Software requirements:

- Python 2.7
- matplotlib
- Pyqt4 for the GUI version

For most operating systems these requirements are satisfied by any standard Python installation.
The software can be used in three different ways: 1. from the command line 2. using the graphical user interface (GUI) 3. from a Python shell

2 Basic Assumptions

1. It is assumed that the input files contain the data column-wise. The first column is assumed to contain the wavelength values, the second the measured values and the third column (if existing) is taken as the standard uncertainties at the corresponding wavelengths.
2. It is assumed that the wavelength scale is strictly monotonically increasing.

Error messages are thrown when these assumptions are not met.
A conceptual assumption of the Richardson-Lucy method is that the provided bandpass function fully explains the measured data. That is, any other influences or pre-processings of the measured spectrum may result in biased results and under-estimated uncertainties.

3 Command Line Usage

Make sure that the path to the software is in your system search path or copy the software to the location of the data you would like to process. From the command line invoke:

- the command
  python RichLucy.py [bandpass file][measurement file] [output file]
  to process the measured spectrum data in [measurement file] with the bandpass data in [bandpass file] and store the results in [output file]
- the command
  python RichLucy.py [bandpass file][measurement file] [output file][maxiter]
  with an integer value [maxiter] denoting the maximum number of Richardson-Lucy iterations
- the command
  python RichLucy.py [bandpass file][measurement file] [output file][maxiter] [MCruns]

to carry out [MCruns] Monte Carlo trials for the evaluation of uncertainties.

Uncertainty evaluation is carried out only when the input [MCruns] is given. Note that for uncertainty evaluation a value of [maxiter] has to be specified.
If interpolation of the bandpass or the measured spectrum is required, these are stored in [bandpass file]*interp and [measurement file]*interp, respectively.
If interpolation of the measured spectrum was necessary, the Richardson-Lucy iterations are called using the interpolated spectrum and the RichLucy estimate is transformed back to the original wavelength scale afterwards. Both results are stored, except when uncertainties are calculated.

4 Using the GUI

For beginners, the GUI version of the software might be the most easiest way to try out the Richardson-Lucy method. It uses the same functions as the command line version, but encapsulates loading of data files, setting parameters and plotting of results in a graphical user interface. The important parts of the GUI which are required to start playing around are described using tooltips. Please note:

> `The GUI uses the command window (either the one from which it is called or the one that opens when double-clicking >RichLucyGUI.py< in a file browser). That is, any additional information and warnings are printed there. Hence, it is strongly advised to keep an eye on this window to see what the program is doing and whether there are any problems!`

A good start to use the software is as follows:

1. Load a text-file (.txt, .csv, ...) containing the bandpass values (see Basic Assumptions)
2. Load a text-file containing the measured spectrum values
3. Hit the 'Start' button.
4. Plot the results using the corresponding buttons at the bottom of the window.

The 'Continue' button can be used to take a look at the progress of the Richardson-Lucy deconvolution:

1. Load the data as above.
2. Enter in 'max. iterations' a small number (1 to 10)
3. Uncheck the checkboxes 'automatic stopping' and 'evaluate uncertainties'

4. Hit the 'Start' button once and then the 'Continue' button.

With each click on the 'Continue' button the Richardson-Lucy iteration is continued, starting from the last result of the previous number (='max. iterations') of RichLucy iterations. The progress can then be seen in the bottom figure of the GUI.

5 Usage from a Python shell

The *RichLucy.py* module can be imported from any other Python module or at a Python shell, such as **IPhython**. After importing the module, the *help()* command can be used to obtain the description of the module at the Python shell. Only basic knowledge of Python programming is required in order to use these methods. In particular, the user shall be familiar with the Python concept of arguments and keyword-arguments.
From the module the following methods (in alphabetical order) can be used from the Python shell or by importing the module from another Python module.

- *EstMeas(Shat, b, lambda_S, lambda_b, display=False)*

  ```
  Calculate the measurement result estimated from an estimated
  input spectrum and the bandpass function.
  If necessary, the bandpass is interpolated to obtain an
  equidistantly sampled sequence. If necessary, the estimated
  input spectrum Shat is interpolated to obtain the same step
  size as for the bandpass
  ```

- *Interpolate2Equidist(lamb_axis, values, display=True, interp_kind='cubic', tol_equi=1e-05)*

  ```
  Interpolation of measured values if wavelength scale axis is
  not equidistant. Input of this function are the original
  measured values and the corresponding wavelength scale.
  The output is the interpolated version with the new scale, a
  flag whether interpolated or not and a tuple (numNaN,numNeg)
  with the number of NaN and negative values after interpolation.
  ```

- *Interpolate2SameStep(M, lambda_M, delta_b, display=True, interp_kind='cubic', tol_equi=1e-05)*

  ```
  Interpolation of measured spectrum if its wavelength step
  size is not equal to that of the bandpass function or if
  it is not equidistant. The output is the interpolated
  version with the new scale, a flag whether interpolated
  or not and a tuple (numNaN,numNeg) with the number of NaN
  and negative values after interpolation.
  ```

- *LoadFromFile(filename)*

  ```
  Load wavelength scale axis, measured values and uncertainties
  from file. Output of this method is the matrix "data".
  ```

- *RichLucy(b, delta, M, maxiter=500, autostop=1, display=1, optFun=calcOptCurv, calcCrit=True, returnAll=False, initialShat=None)*

  ```
  Richardson Lucy deconvolution

  Output:
      Shat ........ estimated spectrum
      critVals1 ... result values returned from optFun
      critVals2 ... result values returned from optFun

  Input variables:
      b ........... bandpass function values
      delta........ wavelength step size for b and M
      M ........... measured spectrum
      maxiter ..... maximum number of iterations
      autostop..... if equal to 1, use optimal number of iterations
      display ..... if larger than zero, provide text output
      optFun ...... function to calculate number of iterations
      calcCrit .... if true critVals is returned
      returnAll ... if true returns full matrix of intermediate results
      initialShat.. estimated spectrum from a previous run of this routine
  ```

- *RichLucyUncertainty(bhat, b_unc, lambda_b, Mhat, M_unc, lambda_M, runs=100, maxiter=500, autostop=1, display=1, optFun=calcOptCurv, calcCrit=True, returnAll=False, plotRes=False)*

  ```
  This function carries out uncertainty evaluation for the Richardson-Lucy
  deconvolution by using the GUM Supplement 2 Monte Carlo method.
  ```

- *calcOptCurv(SH, M, btilde, autostop=1, display=False, orig_mode=False)*

  ```
  Calculate the root-mean-squared progress in the RichLucy iterations.
  Determine the optimal number of iterations from the curvature of the
  rms values.
  ```

- *calcOptDiff(SH, M, bdtilde, autostop=1, display=False)*

  ```
  function to calculate optimal iteration from weighted rms difference of
  estimated measurement and actual measured spectrum
  ```

- *demo(bandwidth=40, skew=1, FWHM=10, maxiter=2000)*

  ```
  Demonstrate usage of the Richardson-Lucy method for bandpass correction.
  A triangular bandpass of given bandwidth and skewness is used to simulate
  a measurement of a Gaussian-shaped spectrum of given FWHM value.

  The maximum number of Richardson-Lucy iterations is given by maxiter.

  Two different automatic stopping criteria are employed and the results
  can be compared.
  ```

- *sim_bandpass(bandwidth, sampledist, skew=1, noise=0, noise_low=0)*

  ```
  Generate a triangular bandpass function with given bandwidth
  and skewnes (skew) on the wavelength axis [-bandwidth,bandwidth]

  Relative measurement noise can be simulated using noise and noise_low
  as maximum and minimum relative noise levels.

  The output of the method is
      hat - the triangular bandpass function
      lamb- the wavelength axis
  ```

- *sim_measurement(Sfun, lamb, b, bandwidth, delta_b, noise_low=0, noise_up=0)*

  ```
  Generate a Gaussian-shaped spectrum by simulating a measurement of a
  clean Gaussian-shaped function Sfun defined on the wavelength axis lamb
  with a triangular bandpass function b of given bandwidth and wavelength
  step size delta_b.

  Note that Sfun needs to be a function object.

  Measurement noise can be simulated using the relative noise levels
  noise_low and noise_up

  The output of the method is
      M - the simulated measured spectrum
  ```