

# Richardson-Lucy bandpass correction

This is the documentation of the Matlab software package version 2.0 for Richardson-Lucy deconvolution of bandpass data.

Copyright S. Eichstädt, F. Schmähling (Physikalisch-Technische Bundesanstalt) 2014

This software is licensed under a BSD-like license:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

This software was developed at Physikalisch-Technische Bundesanstalt. The software is made available 'as is' free of cost. PTB assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, safety, suitability or any other characteristic. In no event will PTB be liable for any direct, indirect or consequential damage arising in connection with the use of this software.

When publishing results procuded by this software, cite:

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

## Software requirements:

Matlab (developed and tested with Matlab R2013a) with toolboxes:

- Statistics Toolbox (for uncertainty evaluation)
- Parallel Toolbox (for faster uncertainty evaluation)

## Basic Assumptions

1. 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 underestimated uncertainties.

## Usage from the Matlab shell

**At first, please take a look at the examples.** You can find examples in **demoRichLucy.m**, **demoRichLucy\_interactive.m**, **example\_1.m** and **example\_2**. The example\_1.m demonstrate uncertainty evaluation of the deconvolution via Monte-Carlo. The example demoRichLucy\_interactive.m will show you the behaviour of the deconvolution using the Richardson-Lucy approach stepwise. The example demo\_RichLucy.m represent the normally case or usage. In example\_2.m the usage of a user defined stopping criterium and regularization rule is figured out.

**The usage of the new object orientated software design for the Richardson-Lucy deconvolution is completely figured out in these example files.**

The calling of the function RichLucy.m (see demoRichLucy.m) is the same as in previous version of the software package for deconvolution using the Richardson-Lucy approach. Users of older version of the software package should not notice/realize that a new version of the software is running.

- the command

```
my_RichLucy = RichLucy_OOP(lambda,M,u_M,bandwidth,delta,b,u_b,[],[])
```

call the constructor of the RichLucy\_OOP object. When  $u_M$  and/or  $u_b$  (uncertainties for the measurement and the bandpass) are unknown you should use []. The last two input parameters are optional and describing options for the bandpass object (i.e. class BPOptions.m) and for the Richardson-Lucy object (i.e. class RichLucyOptions.m). When putting [] inside, default behaviour will be occur.

- the command `Shat = my_RichLucy.Start_RL`

will start a deconvolution process using Richardson-Lucy approach and the curvature criterion to stop the iteration. A user defined stopping rule can add to the RichLucyOption object via

- the command `my_RLO = RichLucyOptions`  
`my_RLO.Set_StoppingRule(@your_stopping_rule);`

the stopping rule should be implemented as a Matlab function and (the interface) must have 5 input parameters. For example please take a look at `StoppingRule_CurvatureCrit.m` or `StoppingRule_EEM_Crit.m`. The same procedure is carried out for the integration of some smoothing/regularization algorithms to smooth every step inside the Richardson-Lucy iteration. (in comparison to early version's these both things are the minor changes)

- the command

```
my_RLO.Set_Regularization(@(new_RLstep,old_RLstep)your_regularization_rule(new_RLstep,old_RLstep,varargin));
```

The regularization routine should be saved as a matlab-function (please keep in mind the required interface (new\_RLstep,old\_RLstep,varargin) when you are implementing your own regularization rule).

- the command `Shat = my_RichLucy.Start_RichLucyUncertainty`

will start uncertainty evaluation via Monte-Carlo. The number of runs can be controlled via member property of the class RichLucyOptions.m

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.