# Pylira: deconvolution of images in the presence of Poisson noise

Axel Donath[‡*], Aneta Siemiginowska[‡], Vinay Kashyap[‡], Douglas Burke[‡], Karthik Reddy Solipuram[§], David van Dyk[¶]

**Abstract**—All physical and astronomical imaging observations are degraded by the finite angular resolution of the camera and telescope systems. The recovery of the true image is limited by both how well the instrument characteristics are known and by the magnitude of measurement noise. In the case of a high signal to noise ratio data, the image can be sharpened or “deconvolved” robustly by using established standard methods such as the Richardson-Lucy method. However, the situation changes for sparse data and the low signal to noise regime, such as those frequently encountered in X-ray and gamma-ray astronomy, where deconvolution leads inevitably to an amplification of noise and poorly reconstructed images. However, the results in this regime can be improved by making use of physically meaningful prior assumptions and statistically principled modeling techniques. One proposed method is the LIRA algorithm, which requires smoothness of the reconstructed image at multiple scales. In this contribution, we introduce a new python package called *Pylira*, which exposes the original C implementation of the LIRA algorithm to Python users. We briefly describe the package structure, development setup and show a Chandra as well as Fermi-LAT analysis example.

**Index Terms**—deconvolution, point spread function, poisson, low counts, X-ray, gamma-ray

## Introduction

Any physical and astronomical imaging process is affected by the limited angular resolution of the instrument or telescope. In addition, the quality of the resulting image is also degraded by background or instrumental measurement noise and non-uniform exposure. For short wavelengths and associated low intensities of the signal, the imaging process consists of recording individual photons (often called "events") originating from a source of interest. This imaging process is typical for X-ray and gamma-ray telescopes, but images taken by magnetic resonance imaging or fluorescence microscopy show Poisson noise too. For each individual photon, the incident direction, energy and arrival time is measured. Based on this information, the event can be binned into two dimensional data structures to form an actual image.

As a consequence of the low intensities associated to the recording of individual events, the measured signal follows Poisson statistics. This imposes a non-linear relationship between the measured signal and true underlying intensity as well as a coupling of the signal intensity to the signal variance. Any statistically correct post-processing or reconstruction method thus requires a careful treatment of the Poisson nature of the measured image.

To maximise the scientific use of the data, it is often desired to correct the degradation introduced by the imaging process. Besides correction for non-uniform exposure and background noise this also includes the correction for the "blurring" introduced by the point spread function (PSF) of the instrument. Where the latter process is often called "deconvolution". Depending on whether the PSF of the instrument is known or not, one distinguishes between the "blind deconvolution" and "non blind deconvolution" process. For astronomical observations, the PSF can often either be simulated, given a model of the telescope and detector, or inferred directly from the data by observing far distant objects, which appear as a point source to the instrument.

While in other branches of astronomy deconvolution methods are already part of the standard analysis, such as the CLEAN algorithm for radio data, developed by [Hog74], this is not the case for X-ray and gamma-ray astronomy. As any deconvolution method aims to enhance small-scale structures in an image, it becomes increasingly hard to solve for the regime of low signal-to-noise ratio, where small-scale structures are more affected by noise.

## The Deconvolution Problem

### *Basic Statistical Model*

Assuming the data in each pixel $d_i$ in the recorded counts image follows a Poisson distribution, the total likelihood of obtaining the measured image from a model image of the expected counts $\lambda_i$ with $N$ pixels is given by:

$$\mathscr{L}(\mathbf{d}|\lambda) = \prod_i^N \frac{\exp -d_i \lambda_i^{d_i}}{d_i!} \tag{1}$$

By taking the logarithm, dropping the constant terms and inverting the sign one can transform the product into a sum over pixels, which is also often called the *Cash* [Cas79] fit statistics:

$$\mathscr{C}(\lambda|\mathbf{d}) = \sum_i^N (\lambda_i - d_i \log \lambda_i) \tag{2}$$

Where the expected counts $\lambda_i$ are given by the convolution of the true underlying flux distribution $x_i$ with the PSF $p_k$:

$$\lambda_i = \sum_k x_i p_{i-k} \tag{3}$$

This operation is often called "forward modelling" or "forward folding" with the instrument response.

* Corresponding author: axel.donath@cfa.harvard.edu
‡ Center for Astrophysics | Harvard & Smithsonian
§ University of Maryland Baltimore County
¶ Imperial College London

### *Richardson Lucy (RL)*

To obtain the most likely value of $\mathbf{x}_n$ given the data, one searches a maximum of the total likelihood function, or equivalently a of minimum $\mathscr{C}$. This high dimensional optimization problem can e.g., be solved by a classic gradient descent approach. Assuming the pixels values $x_i$ of the true image as independent parameters, one can take the derivative of Eq. 2 with respect to the individual $x_i$. This way one obtains a rule for how to update the current set of pixels $\mathbf{x}_n$ in each iteration of the optimization:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \cdot \frac{\partial \mathscr{C}(\mathbf{d}|\mathbf{x})}{\partial x_i} \tag{4}$$

Where $\alpha$ is a factor to define the step size. This method is in general equivalent to the gradient descent and backpropagation methods used in modern machine learning techniques. This basic principle of solving the deconvolution problem for images with Poisson noise was proposed by [Ric72] and [Luc74]. Their method, named after the original authors, is often known as the *Richardson & Lucy* (RL) method. It was shown by [Ric72] that this converges to a maximum likelihood solution of Eq. 2. A Python implementation of the standard RL method is available e.g. in the *Scikit-Image* package [vdWSN+14].

Instead of the iterative, gradient descent based optimization it is also possible to sample from the posterior distribution using a simple Metropolis-Hastings [Has70] approach and uniform prior. This is demonstrated in one of the *Pylira* online tutorials (Introduction to Deconvolution using MCMC Methods).

### *RL Reconstruction Quality*

While technically the RL method converges to a maximum likelihood solution, it mostly still results in poorly restored images, especially if extended emission regions are present in the image. The problem is illustrated in Fig. 1 using a simulated example image. While for a low number of iterations, the RL method still results in a smooth intensity distribution, the structure of the image decomposes more and more into a set of point-like sources with growing number of iterations.

Because of the PSF convolution, an extended emission region can decompose into multiple nearby point sources and still lead to good model prediction, when compared with the data. Those almost equally good solutions correspond to many narrow local minima or "spikes" in the global likelihood surface. Depending on the start estimate for the reconstructed image $\mathbf{x}$ the RL method will follow the steepest gradient and converge towards the nearest narrow local minimum. This problem has been described by multiple authors, such as [PR94] and [FBPW95].

### *Multi-Scale Prior & LIRA*

One solution to this problem was described in [ECKvD04] and [CSv+11]. First, the simple forward folded model described in Eq. 3 can be extended by taking into account the non-uniform exposure $e_i$ and an additional known background component $b_i$:

$$\lambda_i = \sum_k \left(e_i \cdot (x_i + b_i)\right) p_{i-k} \tag{5}$$

The background $b_i$ can be more generally understood as a "baseline" image and thus include known structures, which are not of interest for the deconvolution process. E.g., a bright point source to model the core of an AGN while studying its jets.

Second, the authors proposed to extend the Poisson log-likelihood function (Equation 2) by a log-prior term that controls

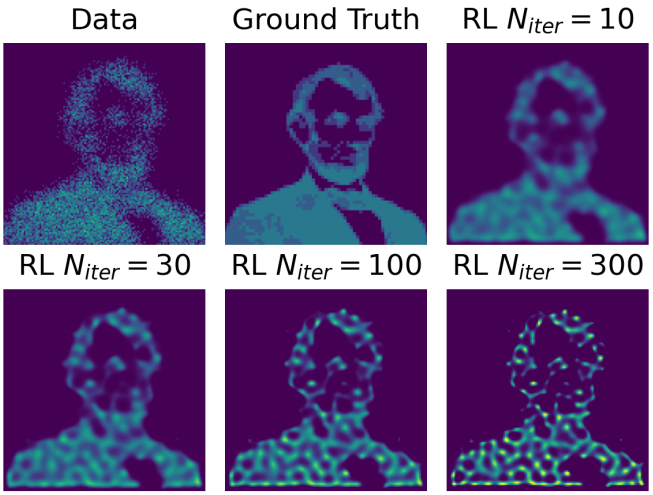


***Fig. 1:*** *The images show the result of the RL algorithm applied to a simulated example dataset with varying numbers of iterations. The image in the upper left shows the simulated counts. Those have been derived from the ground truth (upper mid) by convolving with a Gaussian PSF of width $\sigma = 3$ pix and applying Poisson noise to it. The illustration uses the implementation of the RL algorithm from the Scikit-Image package [vdWSN+14].*

the smoothness of the reconstructed image on multiple spatial scales. Starting from the full resolution, the image pixels $x_i$ are collected into 2 by 2 groups $Q_k$. The four pixel values associated with each group are divided by their sum to obtain a grid of "split proportions" with respect to the image down-sized by a factor of two along both axes. This process is repeated using the down sized image with pixel values equal to the sums over the 2 by 2 groups from the full-resolution image, and the process continues until the resolution of the image is only a single pixel, containing the total sum of the full-resolution image. This multi-scale representation is illustrated in Fig. 2.

For each of the 2x2 groups of the re-normalized images a Dirichlet distribution is introduced as a prior:

$$\phi_k \propto \text{Dirichlet}(\alpha_k, \alpha_k, \alpha_k, \alpha_k) \tag{6}$$

and multiplied across all 2x2 groups and resolution levels $k$. For each resolution level a smoothing parameter $\alpha_k$ is introduced. These hyper-parameters can be interpreted as having an information content equivalent of adding $\alpha_k$ "hallucinated" counts in each grouping. This effectively results in a smoothing of the image at the given resolution level. The distribution of $\alpha$ values at each resolution level is the further described by a hyper-prior distribution:

$$p(\alpha_k) = \exp\left(-\delta\alpha^3/3\right) \tag{7}$$

Resulting in a fully hierarchical Bayesian model. A more complete and detailed description of the prior definition is given in [ECKvD04].

The problem is then solved by using a Gibbs MCMC sampling approach. After a "burn-in" phase the sampling process typically reaches convergence and starts sampling from the posterior distribution. The reconstructed image is then computed as the mean of the posterior samples. As for each pixel a full distribution of its values is available, the information can also be used to compute the associated error of the reconstructed value. This is another main advantage over RL or Maxium A-Postori (MAP) algorithms.

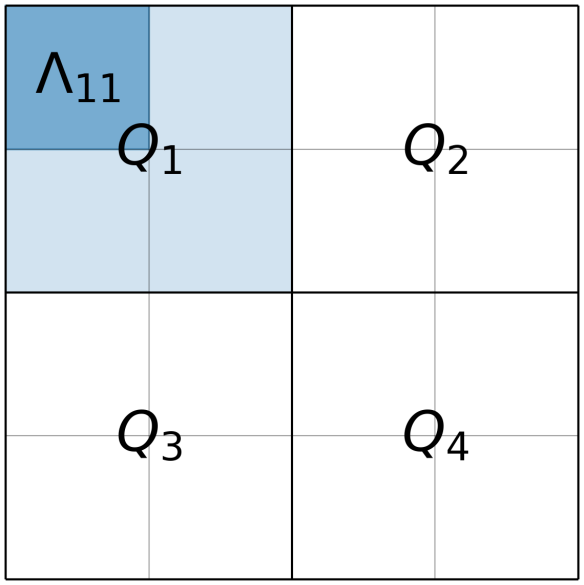


***Fig. 2:*** *The image illustrates the multi-scale decomposition used in the LIRA prior for a 4x4 pixels example image. Each quadrant of 2x2 sub-images is labelled with $Q_N$. The sub-pixels in each quadrant are labelled $\Lambda_{ij}$. .*

## The Pylira Package

### *Dependencies & Development*

The *Pylira* package is a thin Python wrapper around the original *LIRA* implementation provided by the authors of [CSv+11]. The original algorithm was implemented in *C* and made available as a package for the *R Language* [R C20]. Thus the implementation depends on the *RMath* library, which is still a required dependency of *Pylira*. The Python wrapper was built using the *Pybind11* [JRM17] package, which allows to reduce the code overhead introduced by the wrapper to a minimum. For the data handling, *Pylira* relies on *Numpy* [HMvdW+20] arrays for the serialisation to the *FITS* data format on *Astropy* [Col18]. The (interactive) plotting functionality is achieved via *Matplotlib* [Hun07] and *Ipywidgets* [wc15], which are both optional dependencies. *Pylira* is openly developed on Github at https://github.com/astrostat/pylira. It relies on *GitHub Actions* as a continuous integration service and uses the *Read the Docs* service to build and deploy the documentation. The online documentation can be found on https://pylira.readthedocs.io. *Pylira* implements a set of unit tests to assure compatibility and reproducibility of the results with different versions of the dependencies and across different platforms. As *Pylira* relies on random sampling for the MCMC process an exact reproducibility of results is hard to achieve on different platforms; however the agreement of results is at least guaranteed in the statistical limit of drawing many samples.

### *Installation*

*Pylira* is available via the Python package index (pypi.org), currently at version 0.1. As *Pylira* still depends on the *RMath* library, it is required to install this first. So the recommended way to install Pylira is on *MacOS* is:

```
$ brew install r
$ pip install pylira
```

On *Linux* the *RMath* dependency can be installed using standard package managers. For example on Ubuntu, one would do

```
$ sudo apt-get install r-base-dev r-base r-mathlib
$ pip install pylira
```

For more detailed instructions see Pylira installation instructions.

### *API & Subpackages*

*Pylira* is structured in multiple sub-packages. The `pylira.src` module contains the original C implementation and the *Pybind11* wrapper code. The `pylira.core` sub-package contains the main Python API, `pylira.utils` includes utility functions for plotting and serialisation. And `pylira.data` implements multiple pre-defined datasets for testing and tutorials.

## Analysis Examples

### *Simple Point Source*

*Pylira* was designed to offer a simple Python class based user interface, which allows for a short learning curve of using the package for users who are familiar with Python in general and more specifically with *Numpy*. A typical complete usage example of the *Pylira* package is shown in the following:

```
import numpy as np
from pylira import LIRADeconvolver
from pylira.data import point_source_gauss_psf

# create example dataset
data = point_source_gauss_psf()

# define initial flux image
data["flux_init"] = data["flux"]

deconvolve = LIRADeconvolver(
    n_iter_max=3_000,
    n_burn_in=500,
    alpha_init=np.ones(5)
)

result = deconvolve.run(data=data)

# plot pixel traces, result shown in Figure 3
result.plot_pixel_traces_region(
    center_pix=(16, 16), radius_pix=3
)

# plot pixel traces, result shown in Figure 4
result.plot_parameter_traces()

# finally serialise the result
result.write("result.fits")
```

The main interface is exposed via the `LIRADeconvolver` class, which takes the configuration of the algorithm on initialisation. Typical configuration parameters include the total number of iterations `n_iter_max` and the number of "burn-in" iterations, to be excluded from the posterior mean computation. The data, represented by a simple Python `dict` data structure, contains a `"counts"`, `"psf"` and optionally `"exposure"` and `"background"` array. The dataset is then passed to the `LIRADeconvolver.run()` method to execute the deconvolution. The result is a `LIRADeconvolverResult` object, which features the possibility to write the result as a *FITS* file, as well as to inspect the result with diagnostic plots. The result of the computation is shown in the left panel of Fig. 3.

### *Diagnostic Plots*

To validate the quality of the results *Pylira* provides many built-in diagnostic plots. One of these diagnostic plot is shown in the right panel of Fig. 3. The plot shows the image sampling trace

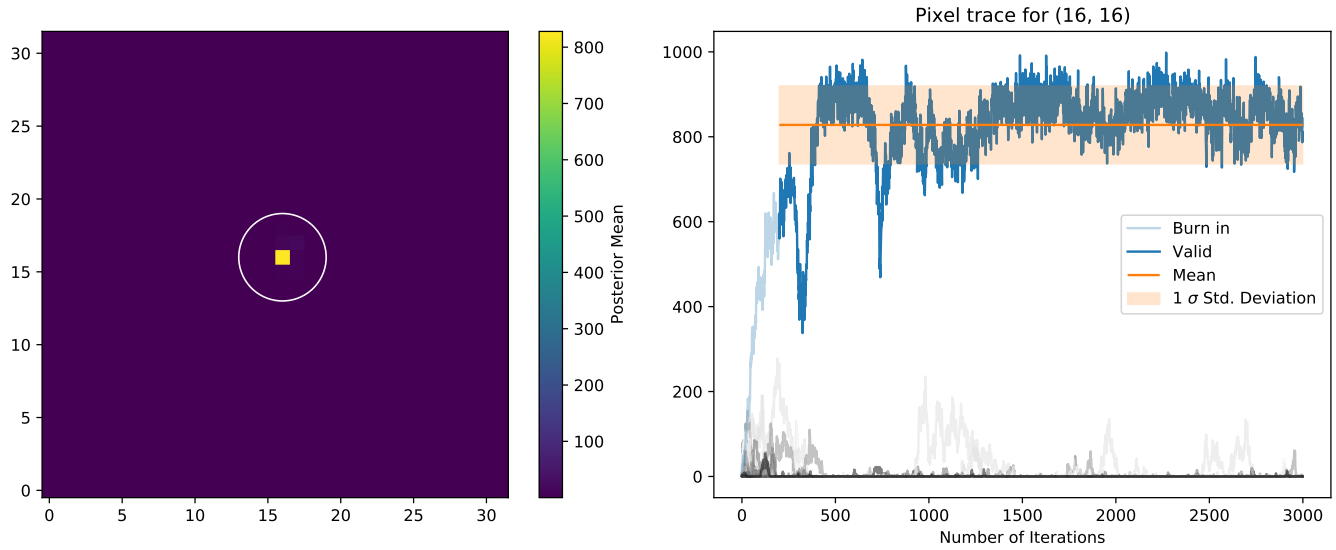


***Fig. 3:*** *The curves show the traces of value the pixel of interest for a simulated point source and its neighboring pixels (see code example). The image on the left shows the posterior mean. The white circle in the image shows the circular region defining the neighboring pixels. The blue line on the right plot shows the trace of the pixel of interest. The solid horizontal orange line shows the mean value (excluding burn-in) of the pixel across all iterations and the shaded orange area the* $1\ \sigma$ *error region. The burn in phase is shown in transparent blue and ignored while computing the mean. The shaded gray lines show the traces of the neighboring pixels.*

for a single pixel of interest and its surrounding circular region of interest. This visualisation allows the user to assess the stability of a small region in the image e.g. an astronomical point source during the MCMC sampling process. Due to the correlation with neighbouring pixels, the actual value of a pixel might vary in the sampling process, which appears as "dips" in the trace of the pixel of interest and anti-correlated "peaks" in the one or mutiple of the surrounding pixels. In the example a stable state of the pixels of interest is reached after approximately 1000 iterations. This suggests that the number of burn-in iterations, which was defined beforehand, should be increased.

*Pylira* relies on an MCMC sampling approach to sample a series of reconstructed images from the posterior likelihood defined by Eq. 2. Along with the sampling, it marginalises over the smoothing hyper-parameters and optimizes them in the same process. To diagnose the validity of the results it is important to visualise the sampling traces of both the sampled images as well as hyper-parameters.

Figure 4 shows another typical diagnostic plot created by the code example above. In a multi-panel figure, the user can inspect the traces of the total log-posterior as well as the traces of the smoothing parameters. Each panel corresponds to the smoothing hyper parameter introduced for each level of the multi-scale representation of the reconstructed image. The figure also shows the mean value along with the $1\ \sigma$ error region. In this case, the algorithm shows stable convergence after a burn-in phase of approximately 200 iterations for the log-posterior as well as all of the multi-scale smoothing parameters.

### *Astronomical Analysis Examples*

Both in the X-ray as well as in the gamma-ray regime, the Galactic Center is a complex emission region. It shows point sources, extended sources, as well as underlying diffuse emission and thus represents a challenge for any astronomical data analysis.

*Chandra* is a space-based X-ray observatory, which has been in operation since 1999. It consists of nested cylindrical paraboloid and hyperboloid surfaces, which form an imaging optical system for X-rays. In the focal plane, it has multiple instruments for different scientific purposes. This includes a high-resolution camera (HRC) and an Advanced CCD Imaging Spectrometer (ACIS). The typical angular resolution is 0.5 arcsecond and the covered energy ranges from 0.1 - 10 keV.

Figure 5 shows the result of the *Pylira* algorithm applied to Chandra data of the Galactic Center region between 0.5 and 7 keV. The PSF was obtained from simulations using the *simulate_psf* tool from the official Chandra science tools *ciao 4.14* [FMA+06]. The algorithm achieves both an improved spatial resolution as well as a reduced noise level and higher contrast of the image in the right panel compared to the unprocessed counts data shown in the left panel.

As a second example, we use data from the Fermi Large Area Telescope (LAT). The Fermi-LAT is a satellite-based imaging gamma-ray detector, which covers an energy range of 20 MeV to >300 GeV. The angular resolution varies strongly with energy and ranges from 0.1 to >10 degree[1].

Figure 6 shows the result of the *Pylira* algorithm applied to Fermi-LAT data above 1 GeV to the region around the Galactic Center. The PSF was obtained from simulations using the *gtpsf* tool from the official *Fermitools v2.0.19* [Fer19]. First, one can see that the algorithm achieves again a considerable improvement in the spatial resolution compared to the raw counts. It clearly resolves multiple point sources left to the bright Galactic Center source.

## Summary & Outlook

The *Pylira* package provides Python wrappers for the LIRA algorithm. It allows the deconvolution of low-counts data following

1. https://www.slac.stanford.edu/exp/glast/groups/canda/lat_Performance.htm

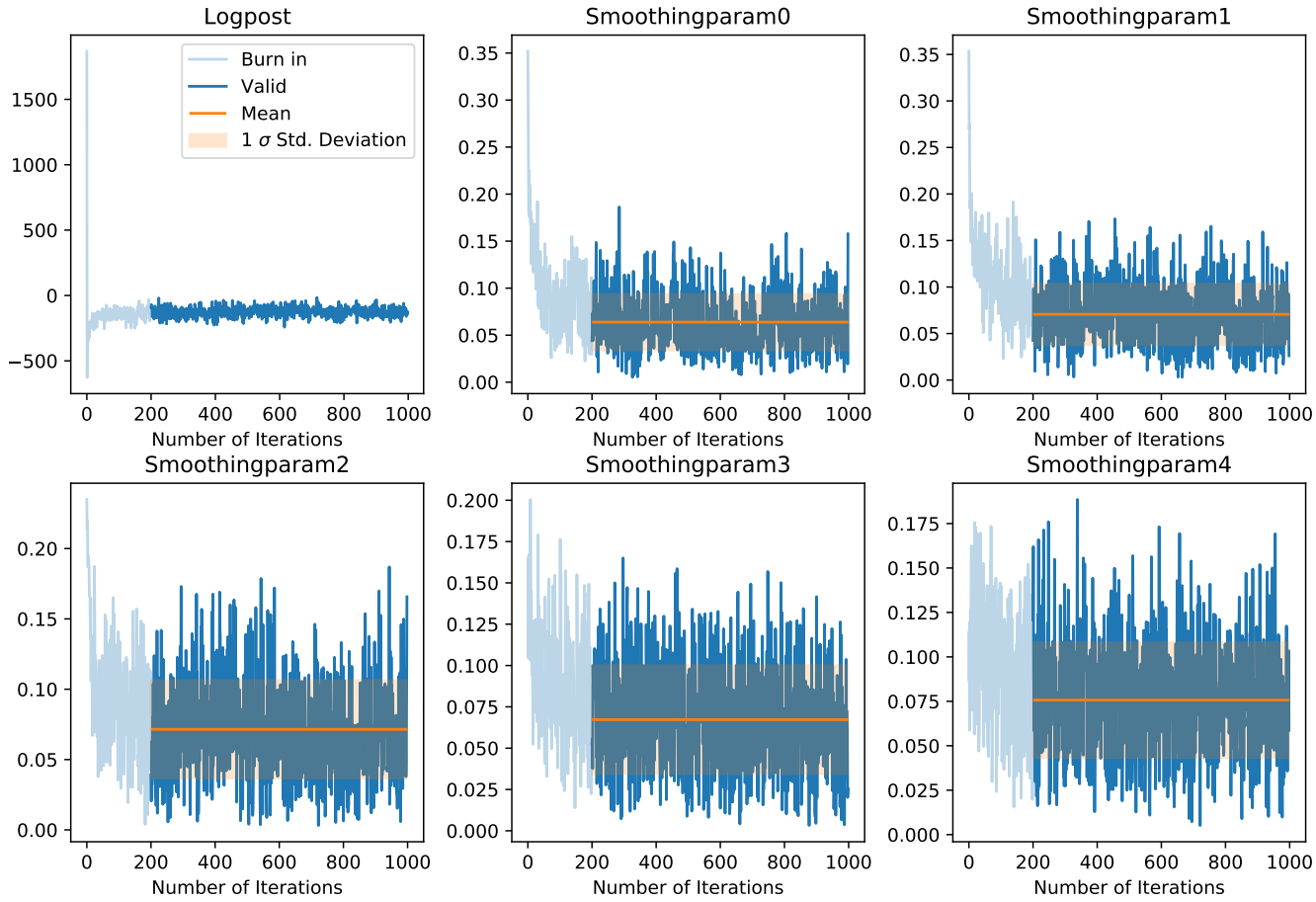


***Fig. 4:*** *The curves show the traces of the log posterior value as well as traces of the values of the prior parameter values. The* SmoothingparamN *parameters correspond to the smoothing parameters* $\alpha_N$ *per multi-scale level. The solid horizontal orange lines show the mean value, the shaded orange area the* 1 $\sigma$ *error region. The burn in phase is shown transparent and ignored while estimating the mean.*

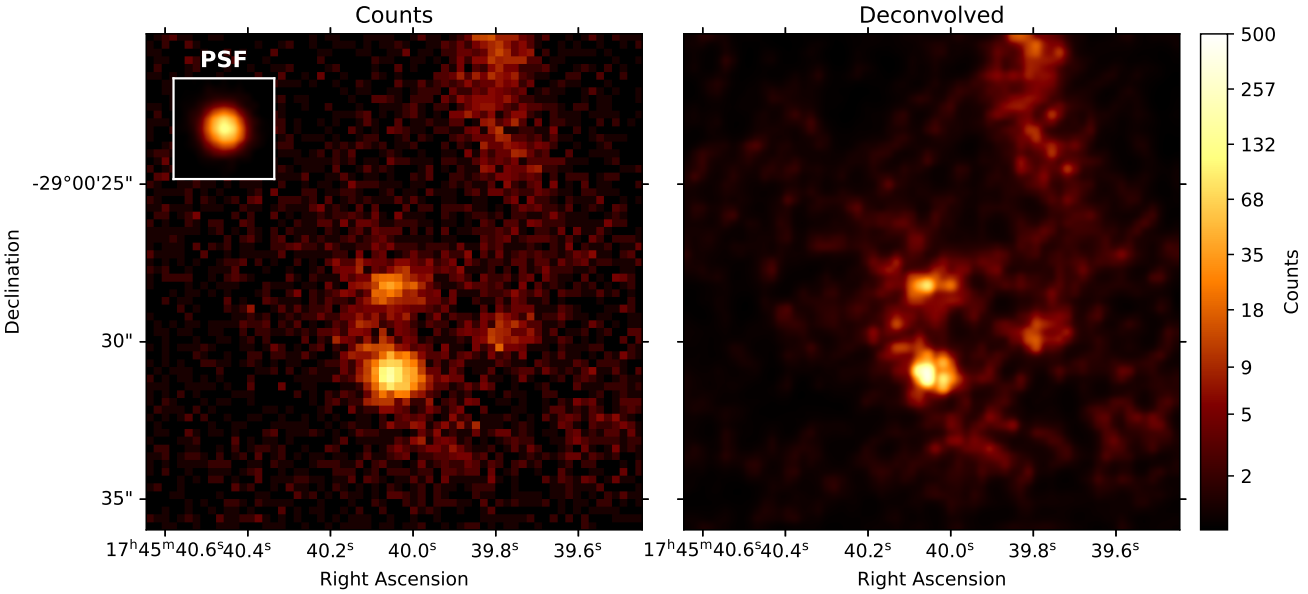


***Fig. 5:*** *Pylira applied to Chandra ACIS data of the Galactic Center region, using the observation IDs* 4684 *and* 4684*. The image on the left shows the raw observed counts between 0.5 and 7 keV. The image on the right shows the deconvolved version. The LIRA hyperprior values were chosen as* ms_al_kap1=1, ms_al_kap2=0.02, ms_al_kap3=1. *No baseline background model was included.*

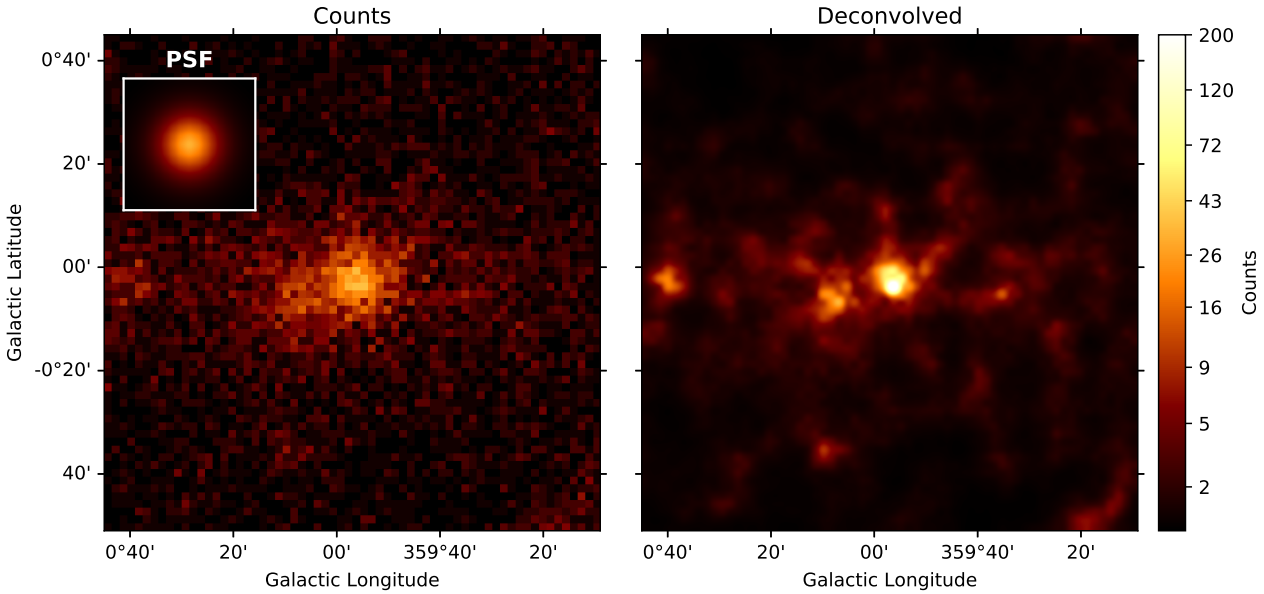


***Fig. 6:*** *Pylira applied to Fermi-LAT data from the Galactic Center region. The image on the left shows the raw measured counts between 5 and 1000 GeV. The image on the right shows the deconvolved version. The LIRA hyperprior values were chosen as* ms_al_kap1=1, ms_al_kap2=0.02, ms_al_kap3=1. *No baseline background model was included.*

Poisson statistics using a Bayesian sampling approach and a multi-scale smoothing prior assumption. The results can be easily written to FITS files and inspected by plotting the trace of the sampling process. This allows users to check for general convergence as well as pixel to pixel correlations for selected regions of interest. The package is openly developed on GitHub and includes tests and documentation, such that it can be maintained and improved in the future, while ensuring consistency of the results. It comes with multiple built-in test datasets and explanatory tutorials in the form of Jupyter notebooks. Future plans include the support for parallelisation or distributed computing, more flexible prior definitions and the possibility to account for systematic errors on the PSF during the sampling process.

### Acknowledgements

This work was conducted under the auspices of the CHASC International Astrostatistics Center. CHASC is supported by NSF grants DMS-21-13615, DMS-21-13397, and DMS-21-13605; by the UK Engineering and Physical Sciences Research Council [EP/W015080/1]; and by NASA 18-APRA18-0019. We thank CHASC members for many helpful discussions, especially Xiao-Li Meng and Katy McKeough. DvD was also supported in part by a Marie-Skodowska-Curie RISE Grant (H2020-MSCA-RISE-2019-873089) provided by the European Commission. Aneta Siemiginowska, Vinay Kashyap, and Doug Burke further acknowledge support from NASA contract to the Chandra X-ray Center NAS8-03060.

## References

[Cas79] W. Cash. Parameter estimation in astronomy through application of the likelihood ratio. *The Astrophysical Journal*, 228:939–947, March 1979. doi:10.1086/156922.

[Col18] Astropy Collaboration. The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package. *The Astrophysical Journal*, 156(3):123, September 2018. arXiv:1801.02634, doi:10.3847/1538-3881/aabc4f.

[CSv+11] A. Connors, N. M. Stein, D. van Dyk, V. Kashyap, and A. Siemiginowska. LIRA — The Low-Counts Image Restoration and Analysis Package: A Teaching Version via R. In I. N. Evans, A. Accomazzi, D. J. Mink, and A. H. Rots, editors, *Astronomical Data Analysis Software and Systems XX*, volume 442 of *Astronomical Society of the Pacific Conference Series*, page 463, July 2011.

[ECKvD04] David N. Esch, Alanna Connors, Margarita Karovska, and David A. van Dyk. An image restoration technique with error estimates. *The Astrophysical Journal*, 610(2):1213–1227, aug 2004. URL: https://doi.org/10.1086/421761, doi:10.1086/421761.

[FBPW95] D. A. Fish, A. M. Brinicombe, E. R. Pike, and J. G. Walker. Blind deconvolution by means of the richardson–lucy algorithm. *J. Opt. Soc. Am. A*, 12(1):58–65, Jan 1995. URL: http://opg.optica.org/josaa/abstract.cfm?URI=josaa-12-1-58, doi:10.1364/JOSAA.12.000058.

[Fer19] Fermi Science Support Development Team. Fermitools: Fermi Science Tools. Astrophysics Source Code Library, record ascl:1905.011, May 2019. arXiv:1905.011.

[FMA+06] Antonella Fruscione, Jonathan C. McDowell, Glenn E. Allen, Nancy S. Brickhouse, Douglas J. Burke, John E. Davis, Nick Durham, Martin Elvis, Elizabeth C. Galle, Daniel E. Harris, David P. Huenemoerder, John C. Houck, Bish Ishibashi, Margarita Karovska, Fabrizio Nicastro, Michael S. Noble, Michael A. Nowak, Frank A. Primini, Aneta Siemiginowska, Randall K. Smith, and Michael Wise. CIAO: Chandra's data analysis system. In David R. Silva and Rodger E. Doxsey, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6270 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 62701V, June 2006. doi:10.1117/12.671760.

[Has70] W. K. Hastings. Monte Carlo Sampling Methods using Markov Chains and their Applications. *Biometrika*, 57(1):97–109, April 1970. doi:10.1093/biomet/57.1.97.

[HMvdW+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. URL: https://doi.org/10.1038/s41586-020-2649-2, doi:10.1038/s41586-020-2649-2.

[Hog74] J. A. Hogbom. Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines. *Astronomy and Astrophysics Supplement*, 15:417, June 1974.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi:10.1109/MCSE.2007.55.

[JRM17] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between c++11 and python, 2017. https://github.com/pybind/pybind11.

[Luc74] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 79:745, June 1974. doi:10.1086/111605.

[PR94] K. M. Perry and S. J. Reeves. Generalized Cross-Validation as a Stopping Rule for the Richardson-Lucy Algorithm. In Robert J. Hanisch and Richard L. White, editors, *The Restoration of HST Images and Spectra - II*, page 97, January 1994. doi:10.1002/ima.1850060412.

[R C20] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL: https://www.R-project.org/.

[Ric72] William Hadley Richardson. Bayesian-Based Iterative Method of Image Restoration. *Journal of the Optical Society of America (1917-1983)*, 62(1):55, January 1972. doi:10.1364/josa.62.000055.

[vdWSN+14] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. URL: https://doi.org/10.7717/peerj.453, doi:10.7717/peerj.453.

[wc15] Jupyter widgets community. ipywidgets, a github repository. Retrieved from https://github.com/jupyter-widgets/ipywidgets, 2015.