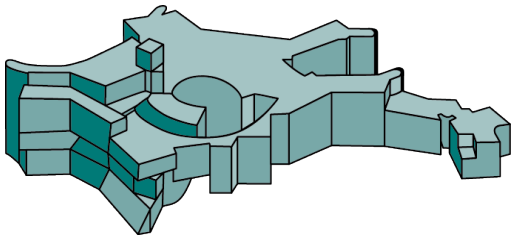
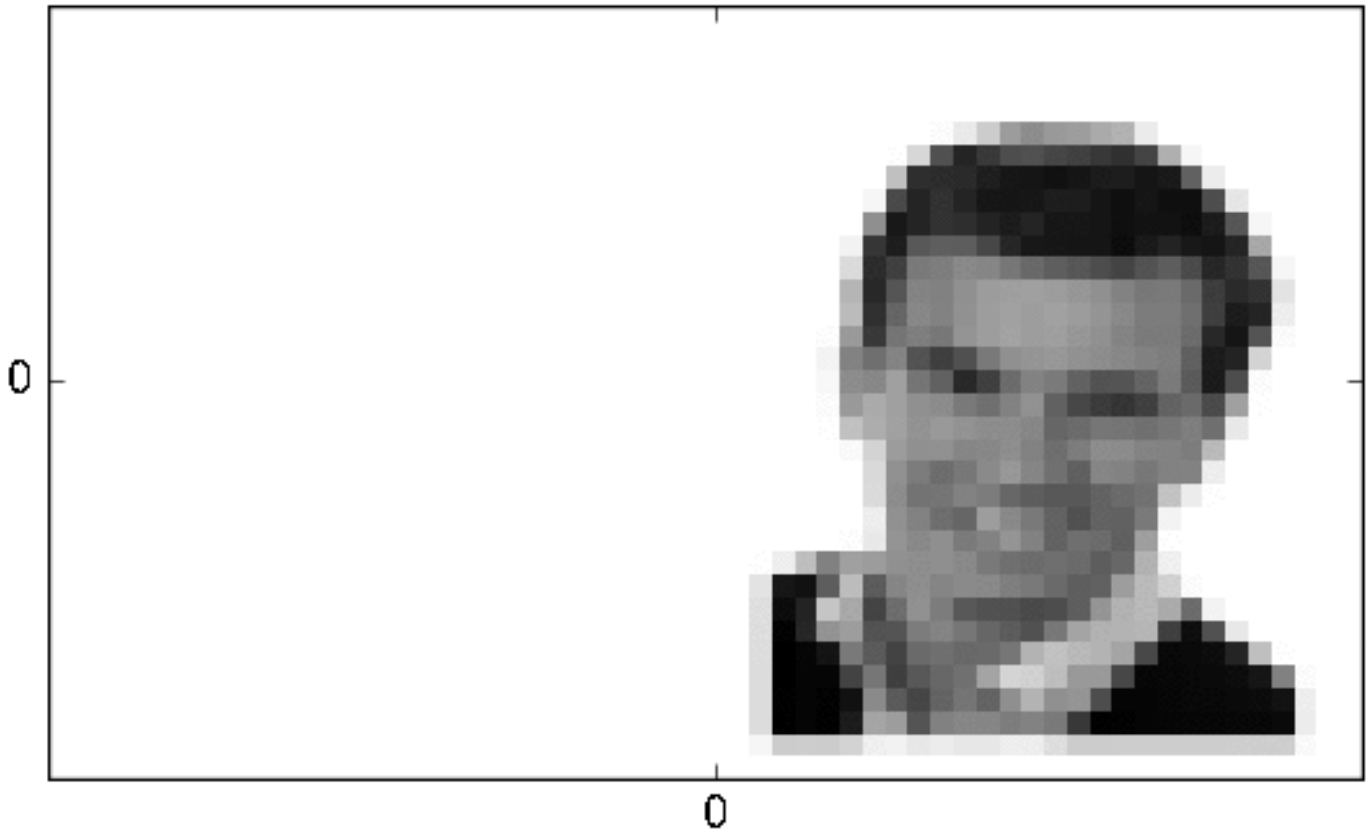
IFT, NIFTy and D3PO - a set of toolboxes for data analysis in astrophysics

Daniel Pumpe, Mahsa Ghaempanah, Maksim Greiner, Marco Selig, Niels Oppermann, Sebastian Dorn, Valentina Vacca, Theo Steininger, Torsten Enßlin

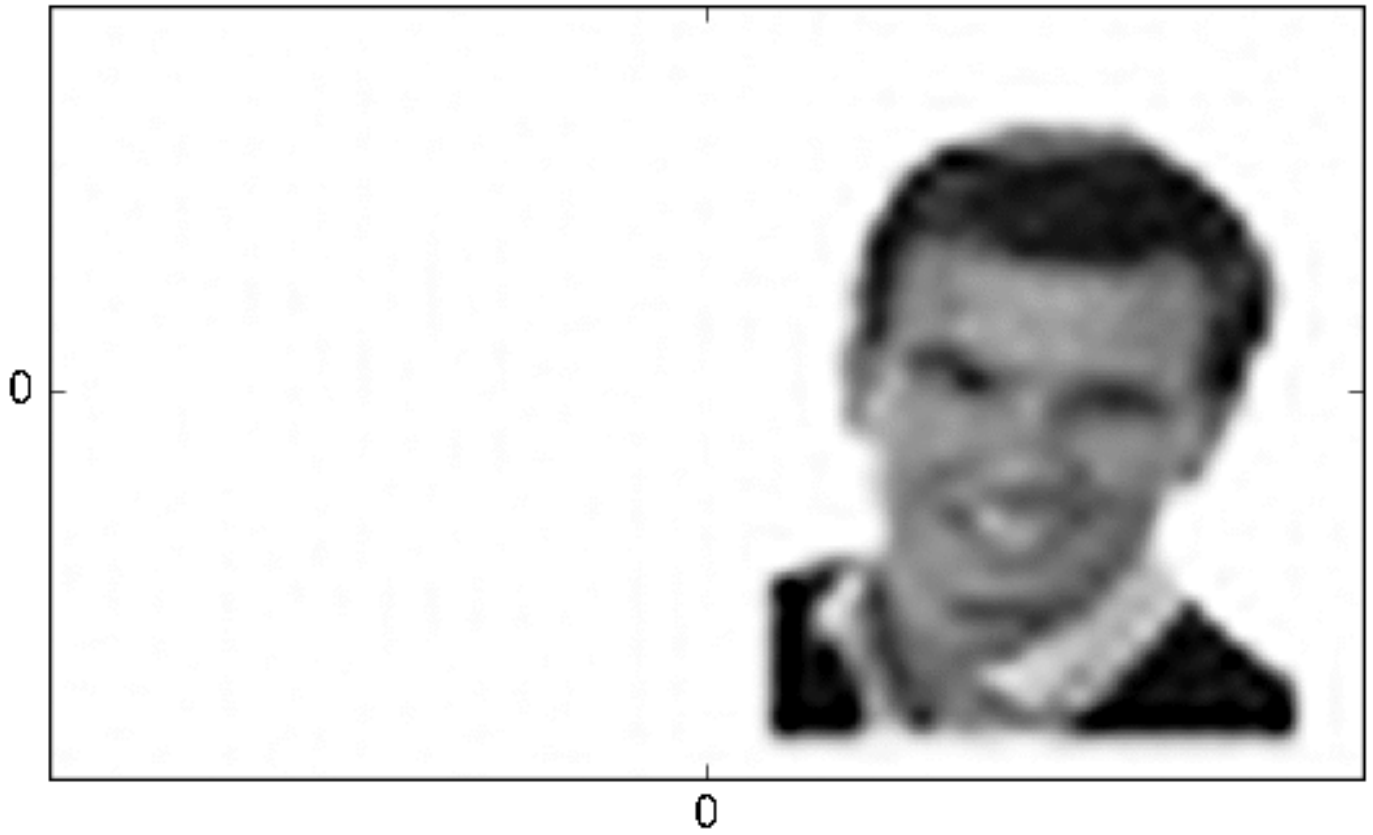
MAX-PLANCK-INSTITUT FOR ASTROPHYSICS



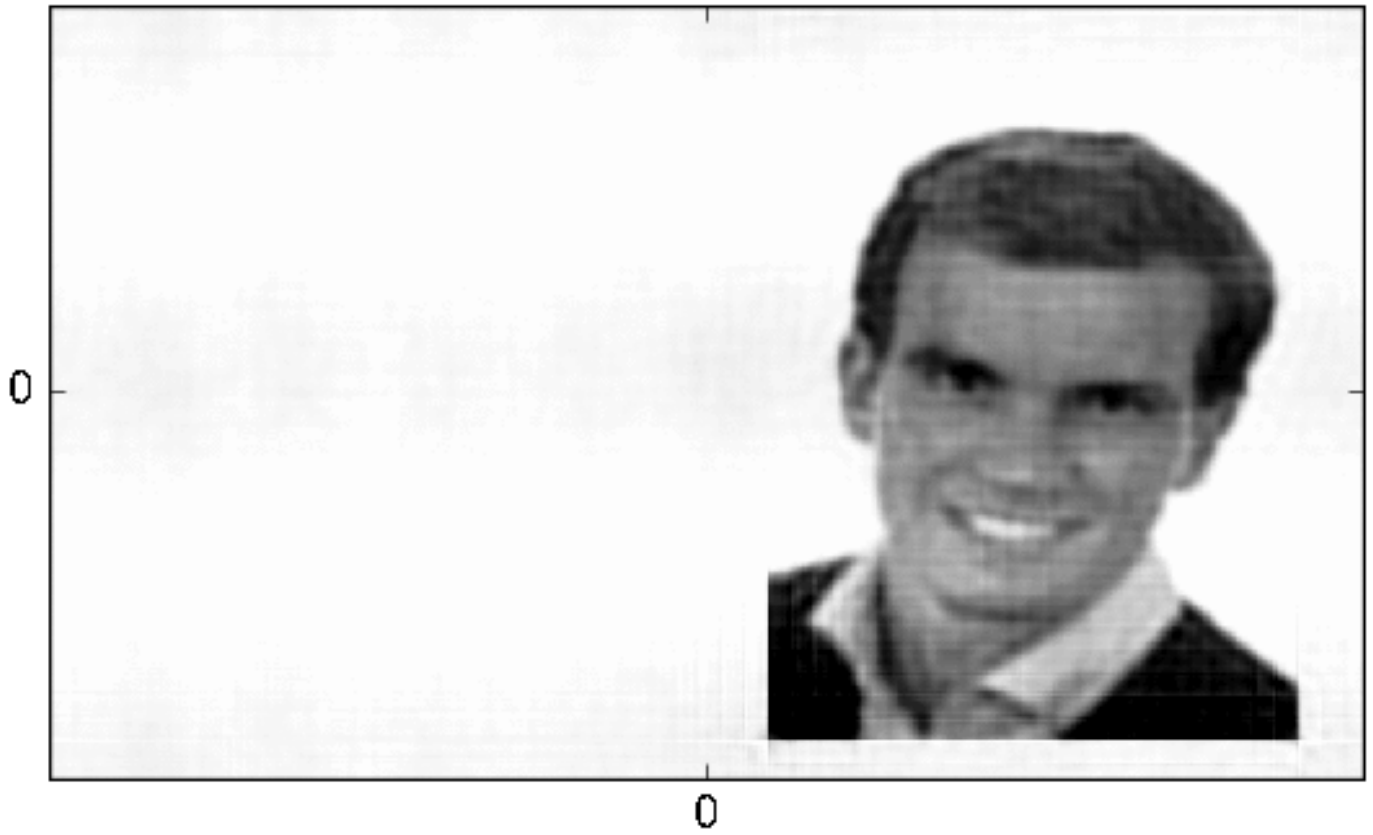
The Cookie Theft



The Cookie Theft



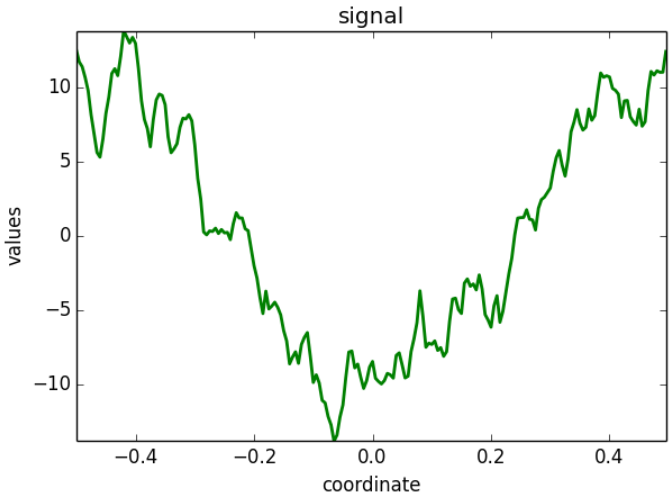
The Cookie Theft

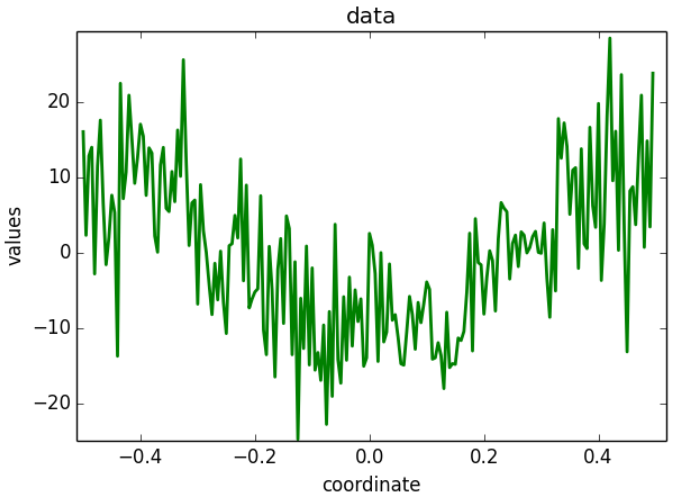


Outline

- Information Field Theory
- Numerical Information Field Theory- NIFTy
- D3PO-algorithm

Motivation





?

signal —observation→ data —inference→ filtered map

Motivation

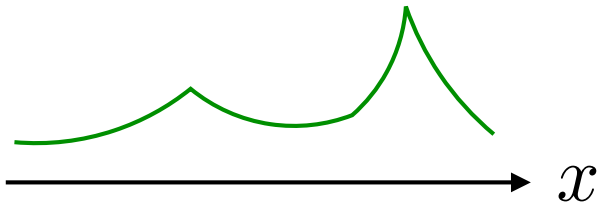
Data ⟵ observation ⟶ Signal estimate

inference

- data vector
- finite set of numbers

$$\vec{d} = (d_1, d_2, d_3, \dots)^T$$

- signal field
- infinite number of degrees of freedom



$$\vec{s} = s(x)\ , \quad x \in \Omega$$

Conditional probabilities

$$P\left(\text{event} \mid \text{condition}\right)$$

- product rule $$P\left(a, b\right) = P\left(a\right) P\left(b \mid a\right) = P\left(b\right) P\left(a \mid b\right)$$
- sum rule $$P\left(a \mid b\right) + P\left(\bar{a} \mid b\right) = 1$$
- marginalization $$P\left(a\right) = \sum_{\text{all } b} P\left(a, b\right)$$

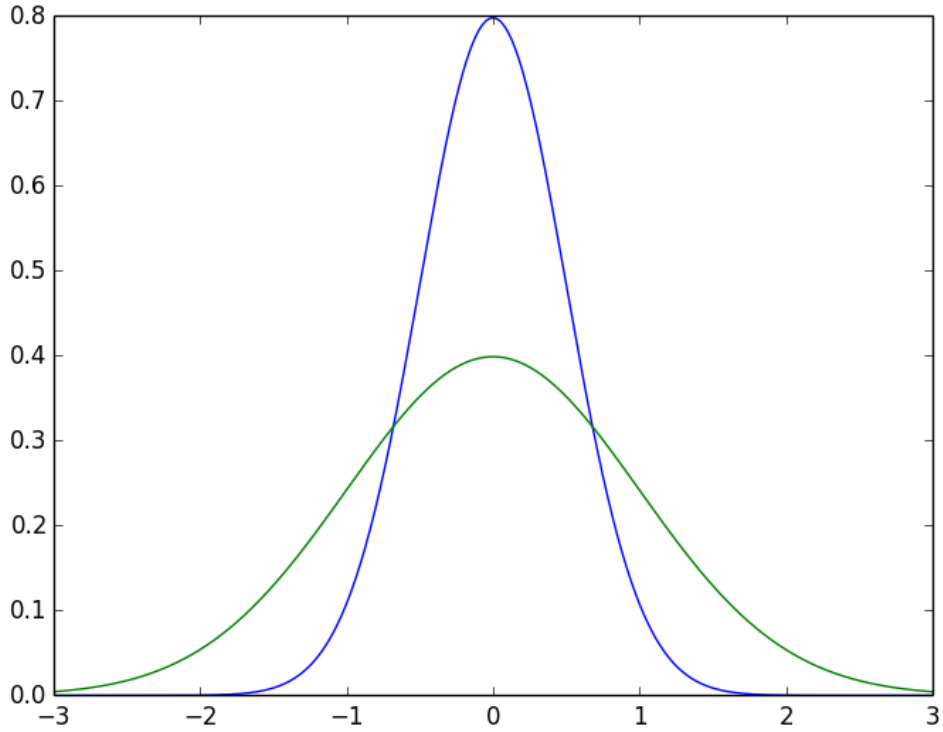
Bayes Theorem

"Information is what forces a change of rational beliefs"

by Ariel Caticha

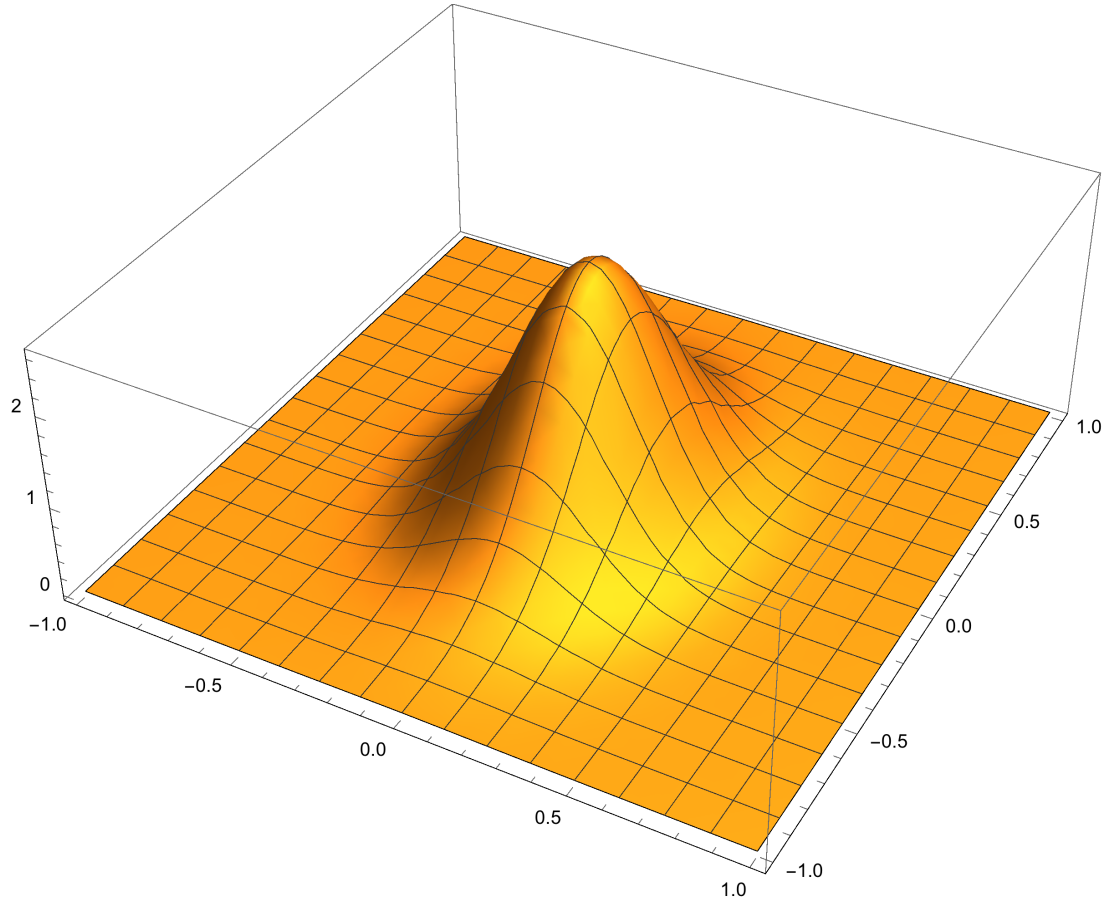
$$\underbrace{P(s \mid d)}_{\text{posterior}} = \frac{\overbrace{P(d \mid s)}^{\text{likelihood}}\overbrace{P(s)}^{\text{prior}}}{\underbrace{P(d)}_{\text{evidence}}}$$

Statistical Field Theory



$$\mathscr{G}(\phi, \sigma) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\phi^2}{2\sigma^2}\right)$$

Statistical Field Theory



$$\mathscr{G}\left(\varphi,\Phi\right) := \frac{1}{\sqrt{\det\left[2\pi\Phi\right]}}\exp\left(-\frac{1}{2}\varphi^{\dagger}\Phi^{-1}\varphi\right)$$

Correlation structure

Statistical homogeneity

$$S_{x,y} = \langle s_x s_y \rangle_{(s)} = C_s\left(x - y\right)$$

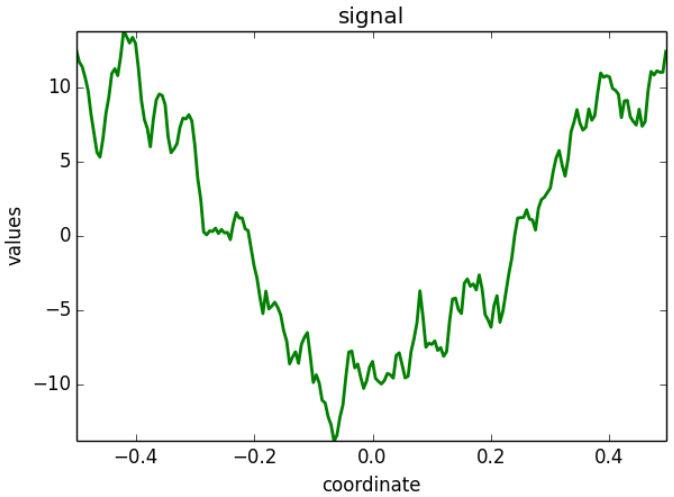
Fouriertransformation

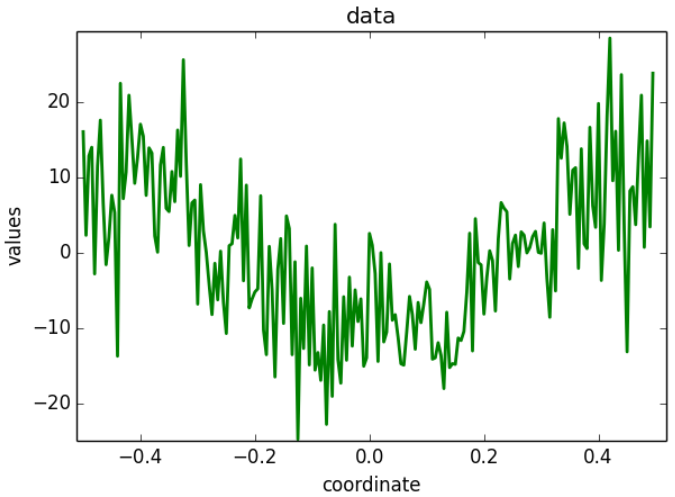
$$S_{k,k'} = (2\pi)^{\dim} \delta\left(k - k'\right) C_s\left(k\right)$$

Power Spectrum

$$C_s\left(k\right) = P\left(k\right)$$

Wiener Filter





?

signal —observation→ data —inference→ filtered map

$$d = \boldsymbol{R}s + n \qquad s \hookleftarrow \mathscr{G}\left(s, S\right) \qquad n \hookleftarrow \mathscr{G}\left(n, N\right)$$

Wiener Filter

$$d = \boldsymbol{R}s + n \qquad s \hookleftarrow \mathscr{G}\,(s, S) \qquad n \hookleftarrow \mathscr{G}\,(n, N)$$

deterministic calculus

$$P\,(d \mid s) = \delta\,(d - \boldsymbol{R}s - n)$$

probabilistic calculus

$$P\,(d \mid s) = \int \mathcal{D}n\,\delta\,(d - \boldsymbol{R}s - n)\,\mathscr{G}\,(n, N)$$

Wiener Filter

The information Hamiltonian:

$$\begin{aligned} H(d,s) &= -\ln\left[P(d \mid s)\, P(s)\right] \\ &= -\ln\left[\mathscr{G}(d - Rs, N)\, \mathscr{G}(s, S)\right] \end{aligned}$$

Wiener Filter

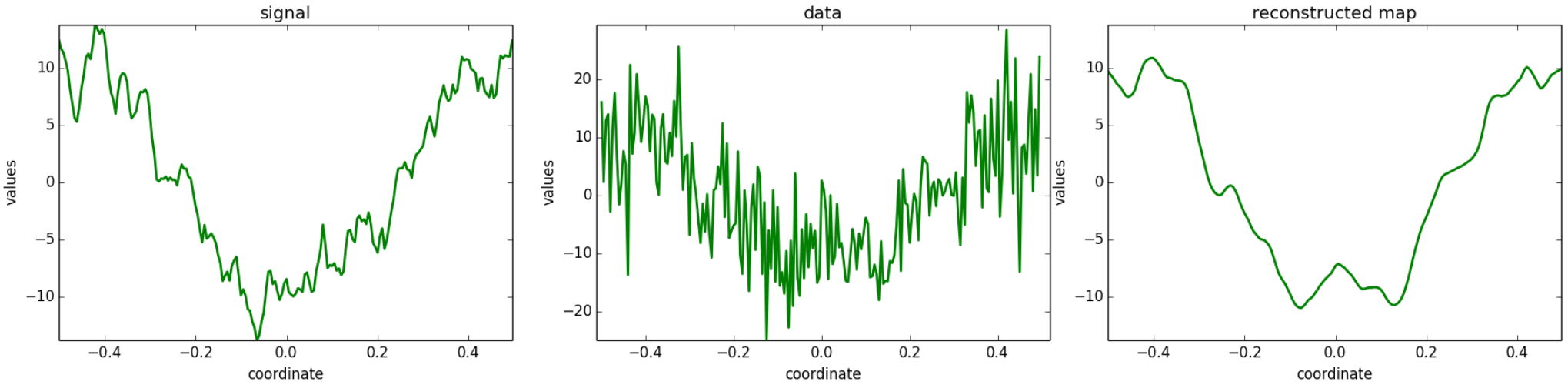
Ansatz for field inference: *Maximum a posteriori*

$$0 \stackrel{!}{=} \left. \frac{\delta H\left(d, s\right)}{\delta s} \right|_{s=m_{MAP}}$$

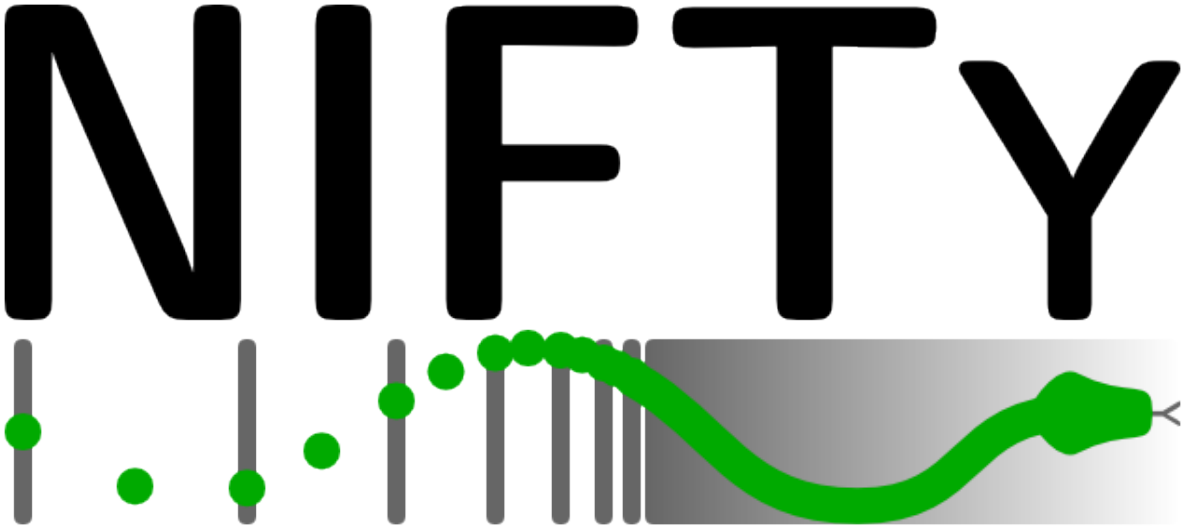
$$m_{MAP} = \underbrace{\left(S^{-1} + R^\dagger N^{-1} R\right)^{-1}}_{D} \underbrace{R^\dagger N^{-1} d}_{j}$$

$$P\left(s \mid d\right) \propto \mathscr{G}\left(s - m_{\mathrm{MAP}}, D\right)$$

Wiener Filter



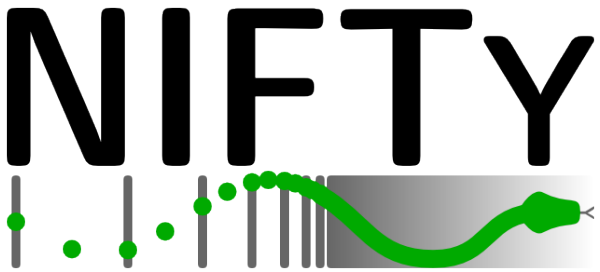
signal —observation→ data —inference→ filtered map



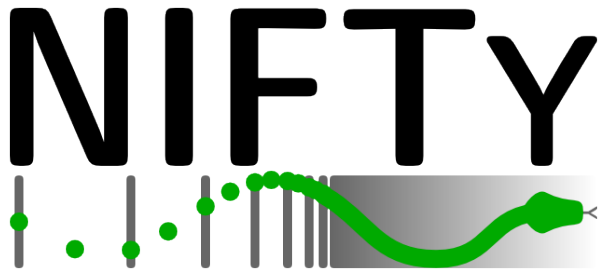
Numerical Information Field Theory

Designed to enable the development of signal inference algorithms

Selig et al. (2013)
arXiv: 1301.4499



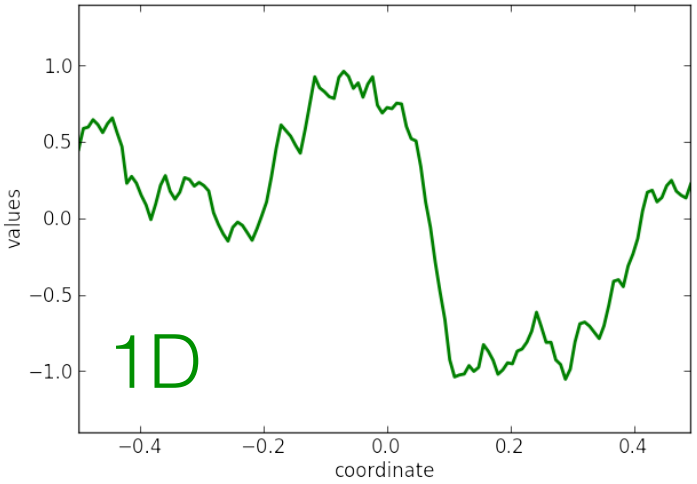
- is a versatile PYTHON library incorporating CYTHON, C++, and C libraries

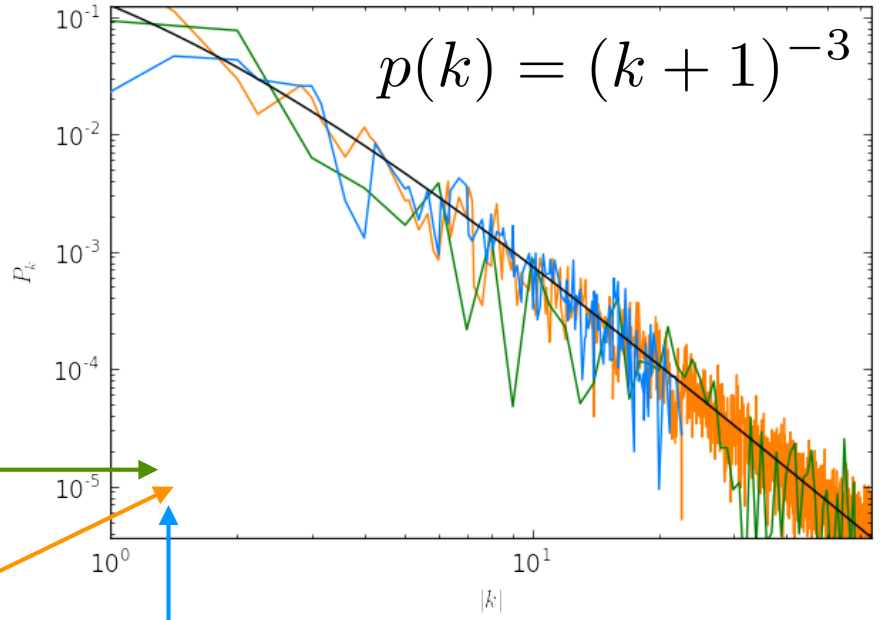


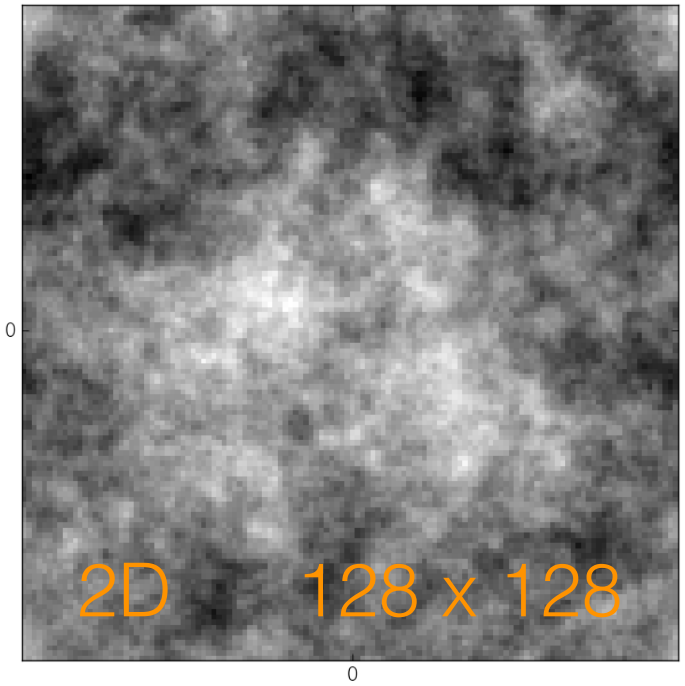
- is a versatile PYTHON library incorporating CYTHON, C++, and C libraries

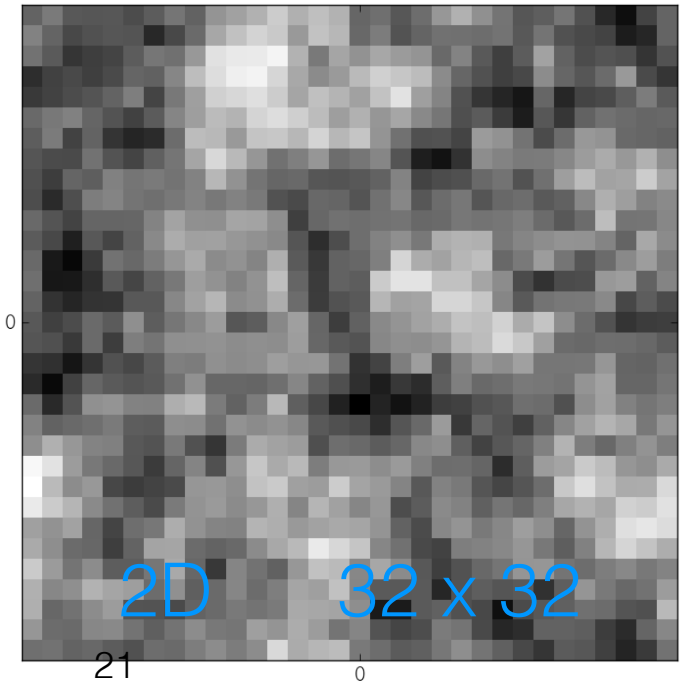
- operates regardless of the underlying spatial grid and its resolution

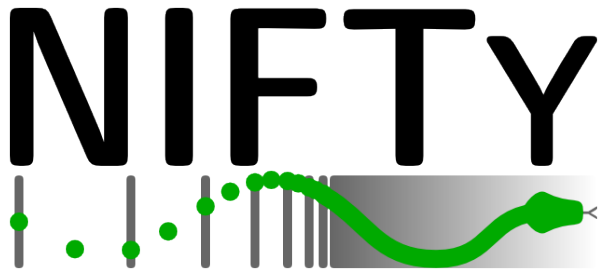
Grid independence





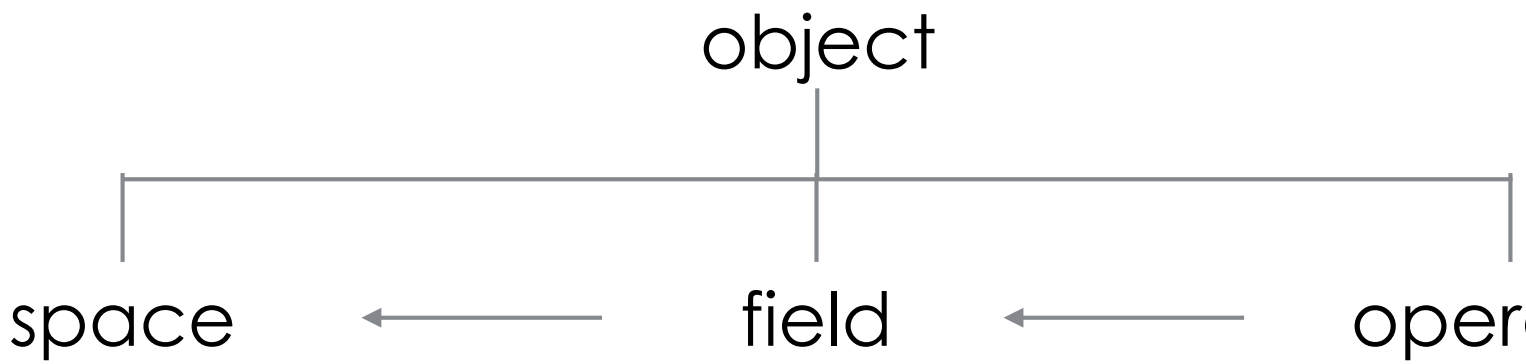






- is a versatile PYTHON library incorporating CYTHON, C++, and C libraries
- operates regardless of the underlying spatial grid and its resolution
- abstracts spaces, fields, and operators into an object oriented framework

NIFTy classes

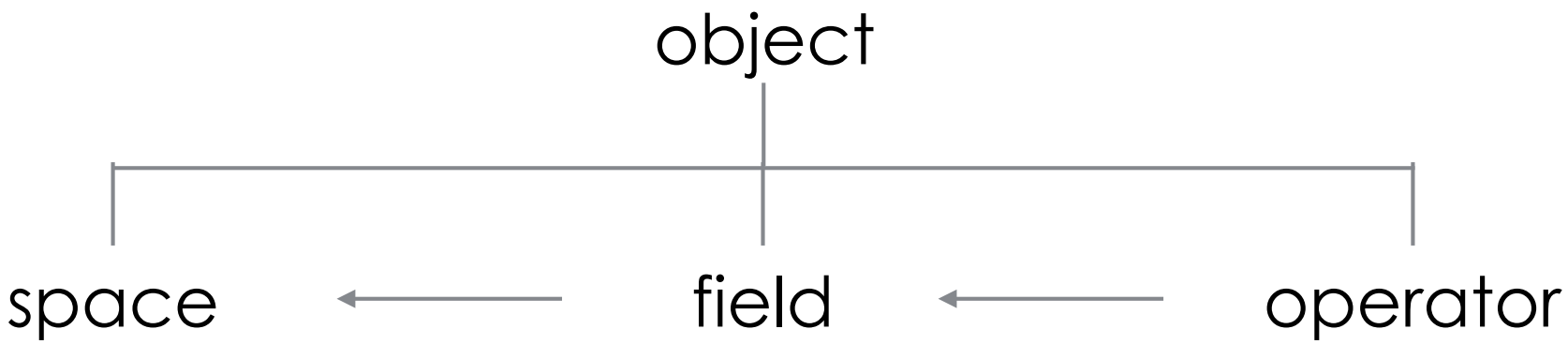


- parameters

- domain space
- field values

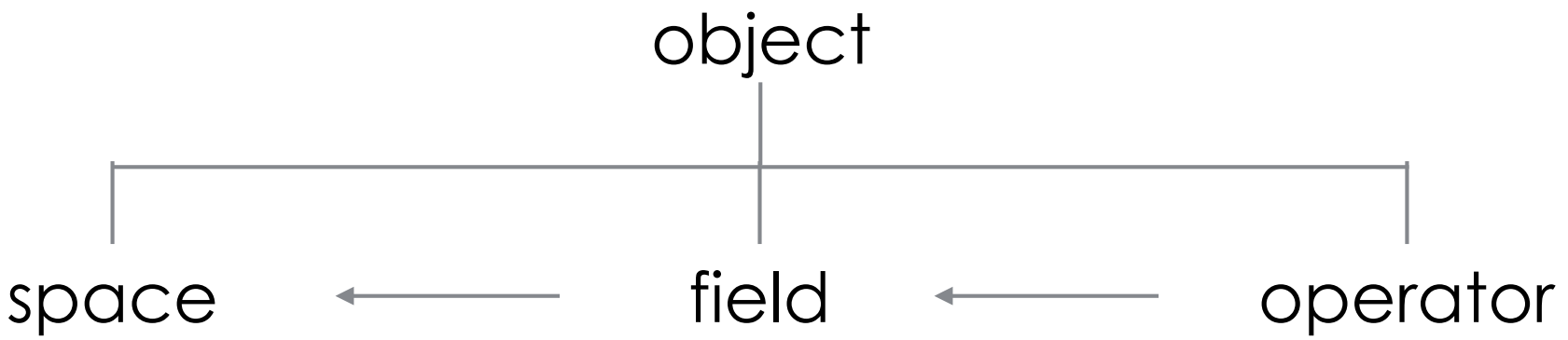
- domain space
- target space
- instance methods applied to fields

NIFTy classes



- **point_space** • unstructured list of points
- **rg_space** • n-dimensional regular grid
- **lm_space** • spherical harmonics
- **hp_space** • Gauss-Legendre grid
- **gl_space** • HEALPix grid
- **nested_space** • arbitrary product of grids

NIFTy classes



point_space

rg_space

lm_space

hp_space

gl_space

nested_spac

diagonal_operator

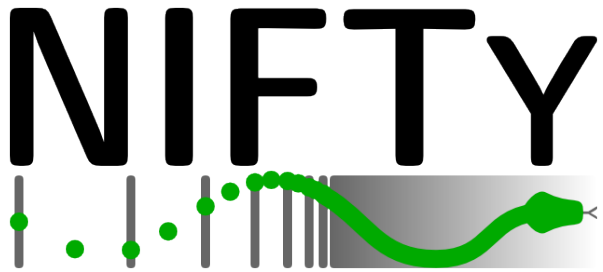
projection_operator

vecvec_operator

response_operator

invertible_operator

propagator_operator



- is a versatile PYTHON library incorporating CYTHON, C++, and C libraries
- operates regardless of the underlying spatial grid and its resolution
- abstracts spaces, fields, and operators into an object oriented framework
- allows the user the abstract formulation and programming of signal inference algorithms

Test it yourself

get NIFTy:

- https://github.com/information-field-theory/nifty
- wwwmpa.mpa-garching.mpg.de/ift/nifty/wmpa.mpa-garching.mpg.de/ift/

Installation:

- pip install ift_nifty
- python setup.py install

Dependencies:

- neccessary: Numpy, Scipy, Matplotlib, multiprocessing
- optional: GFFT, Healpy, libsharp-wrapper

Hands-on: Wiener Filter

- Create your own mock data in a space of your choice
- Infer the most plausible signal from the mock data
- Study the influence of the Prior (S, N) on the reconstruction
- Infer the power spectrum of the signal (advanced)

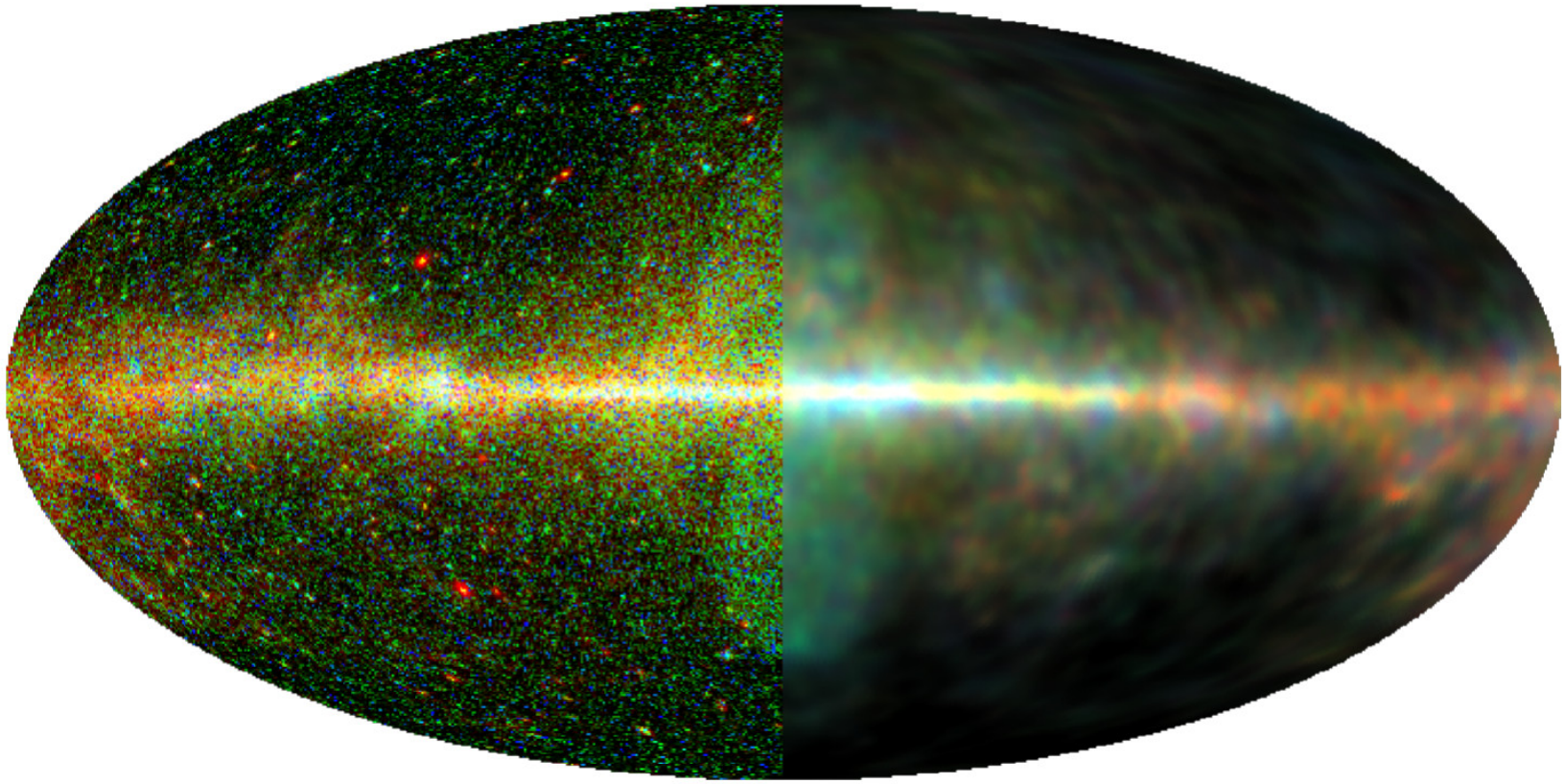
$$d = \boldsymbol{R}s + n \qquad s \hookleftarrow \mathscr{G}\,(s, S) \qquad n \hookleftarrow \mathscr{G}\,(n, N)$$

$$m_{\mathrm{MAP}} = Dj$$

$$D^{-1} = \boldsymbol{R}^\dagger N^{-1} \boldsymbol{R} + S^{-1} \quad j = \boldsymbol{R} N^{-1} d$$

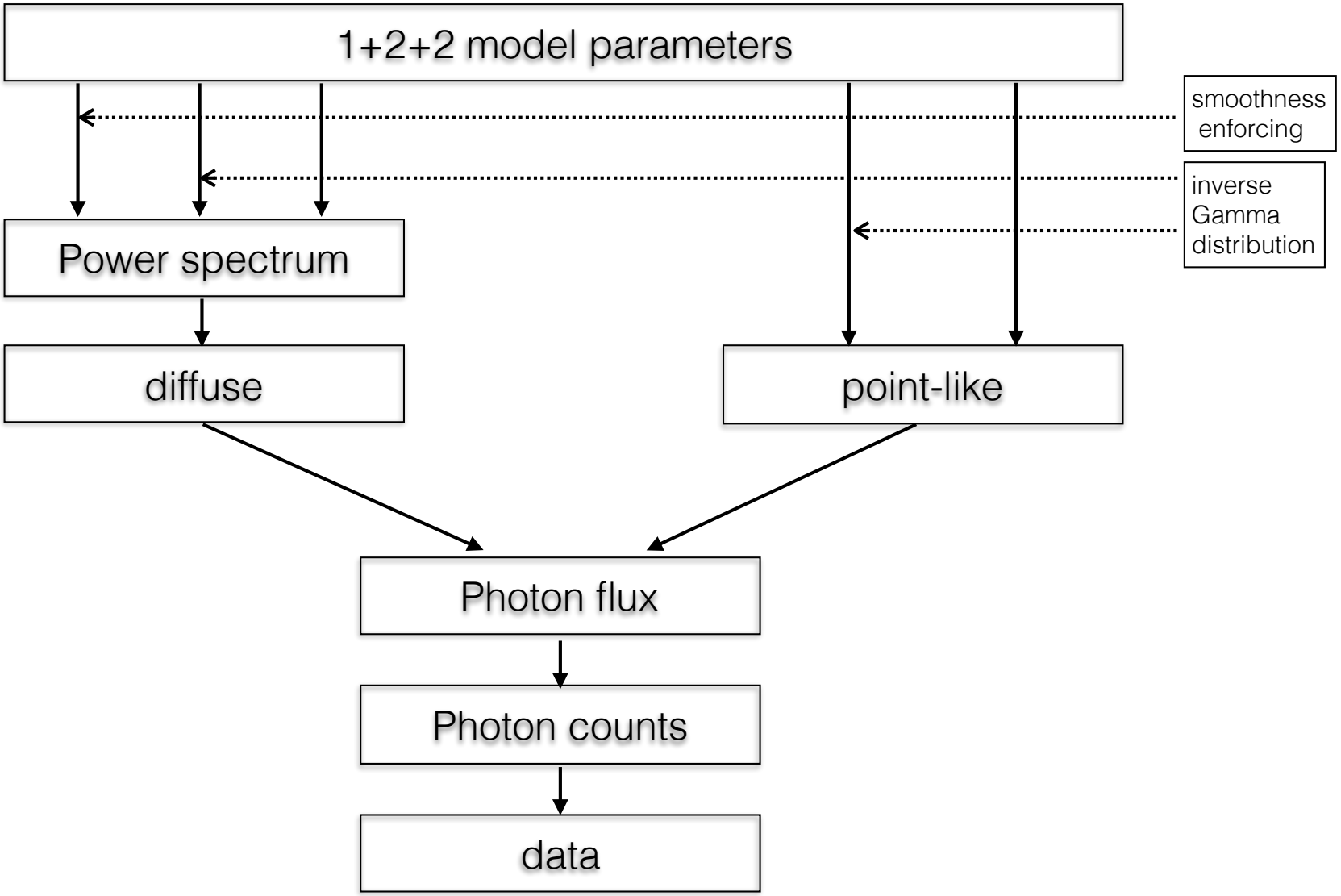
D3PO

Denoising, Deconvolving, and Decomposing Photon Observations



Selig et al. (2014)
arXiv:1311.1888

D3PO Algorithm



Theoretical background

To reconstruct **point-like** and **diffuse sources** from one photon count image

$$\boldsymbol{\rho} = \boldsymbol{\rho}_{\text{diffuse}} + \boldsymbol{\rho}_{\text{point-like}}$$

Poissonian likelihood:

$$P\left(\boldsymbol{d} \mid \boldsymbol{R\rho}\right) = \prod_{\mathrm{i}} \frac{1}{d_{\mathrm{i}}!} \left(\boldsymbol{R\rho}\right)_{\mathrm{i}}^{d_i} \, \mathrm{e}^{-\left(\boldsymbol{R\rho}\right)_{\mathrm{i}}}$$

Prior Assumptions

- Address the morphology of point-like and diffuse flux
- Try to break degeneracy of the likelihood

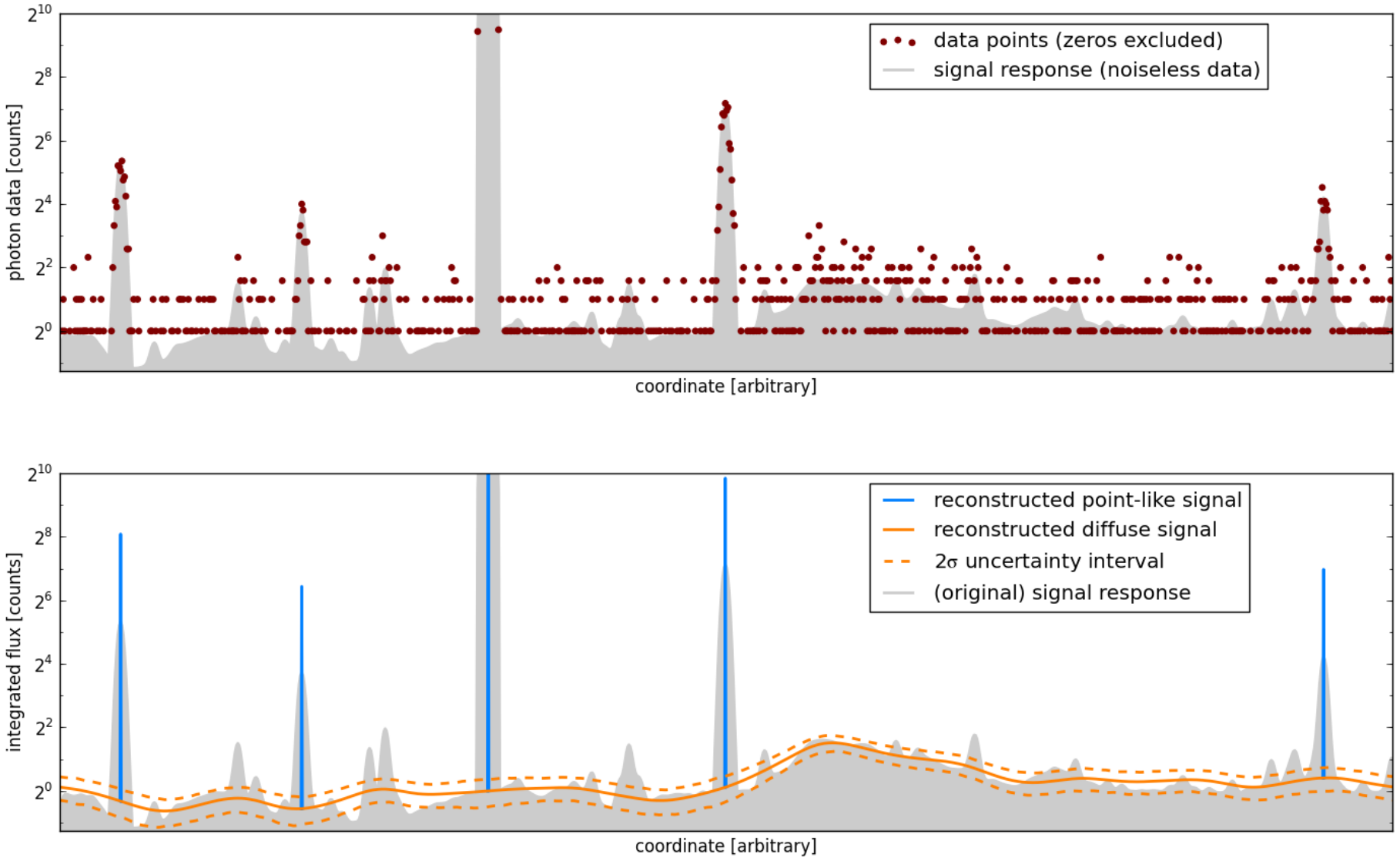
diffuse component:

$$P\left(s \mid \tau\right) P_{\mathrm{un}}\left(\tau \mid \alpha, q\right) P_{\mathrm{sm}}\left(\tau \mid \sigma\right)$$

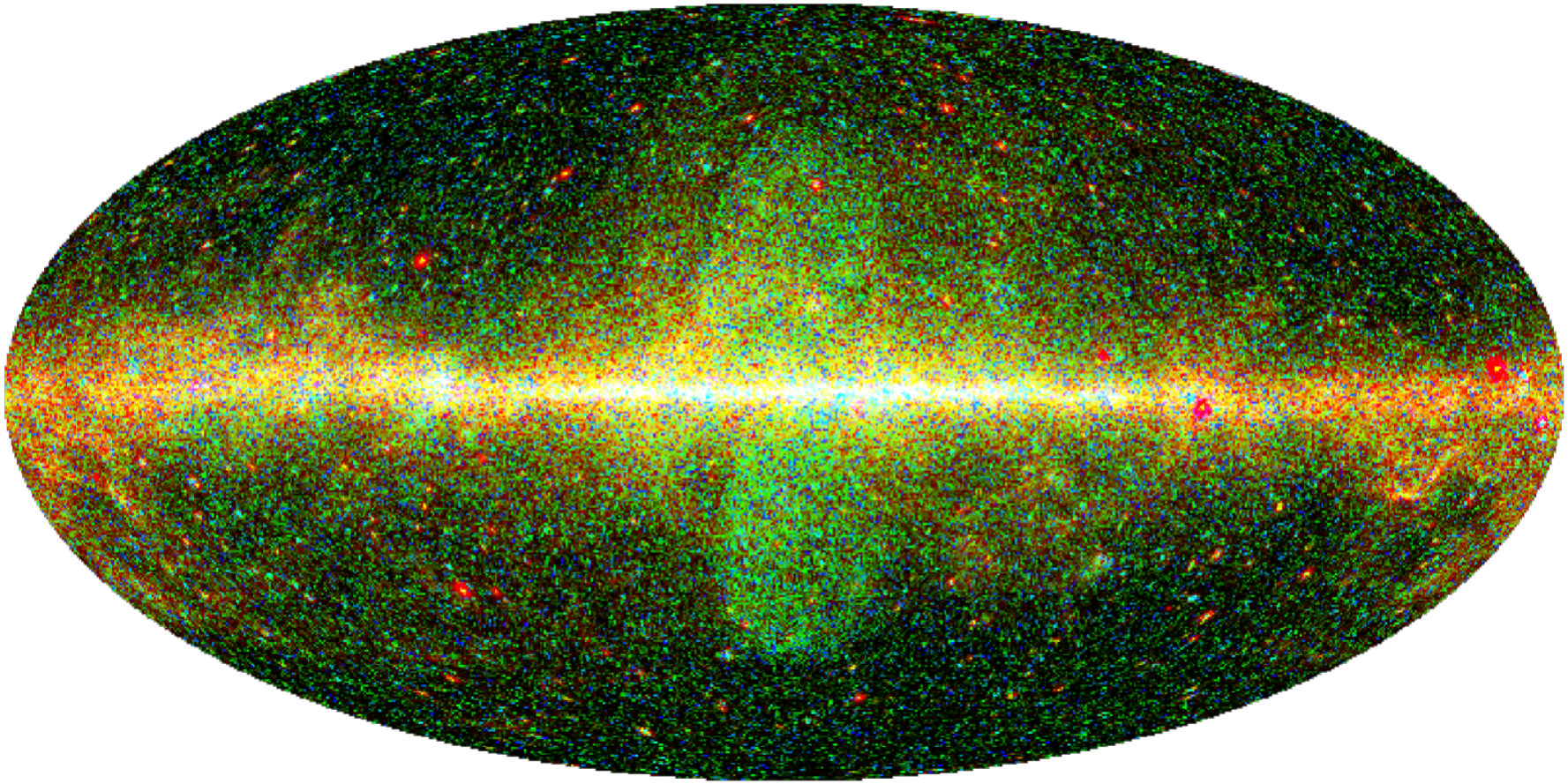
point-like component:

$$P\left(u \mid \beta, \eta\right)$$

D3PO 1D scenario

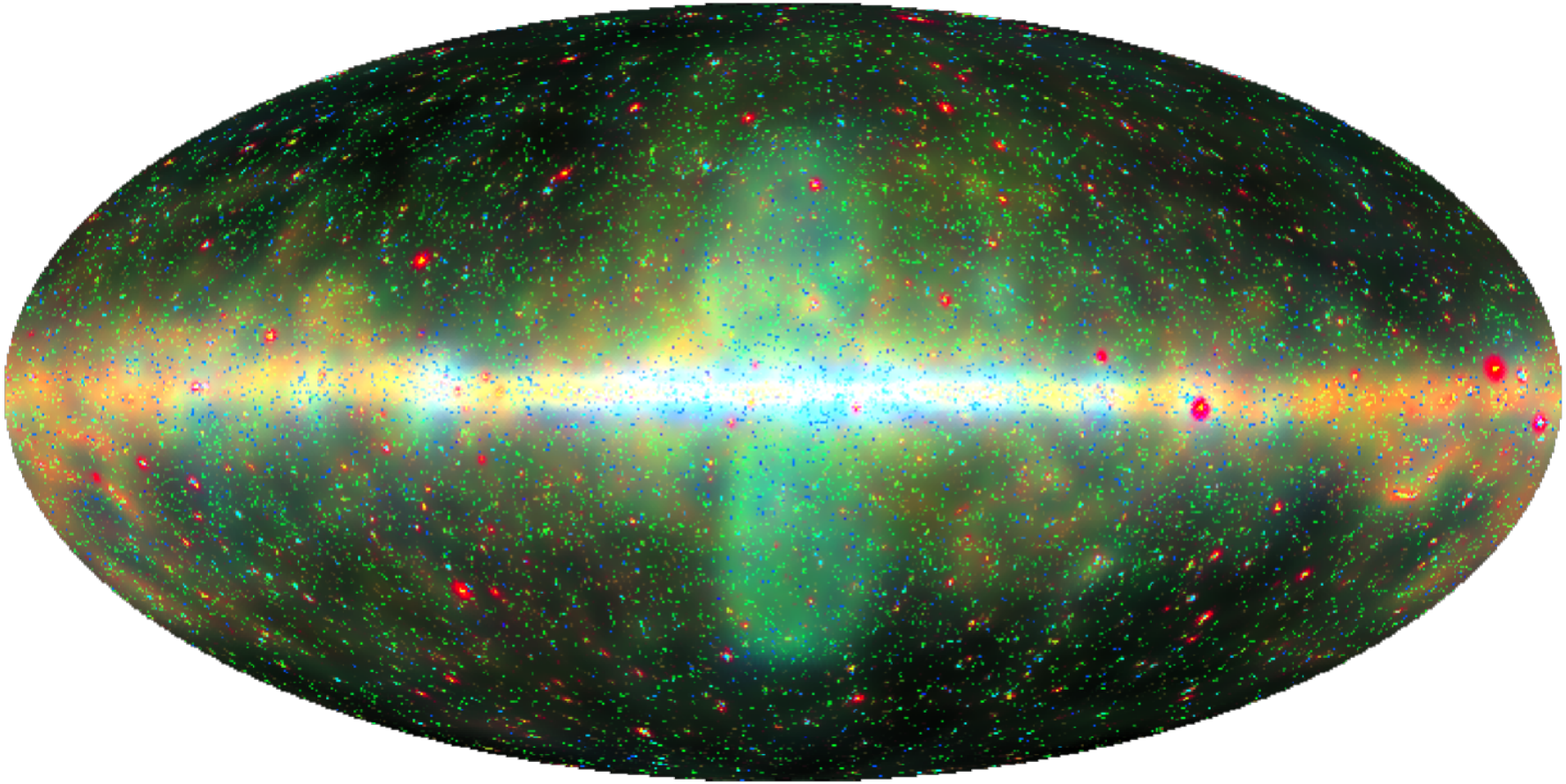


Fermi all sky map



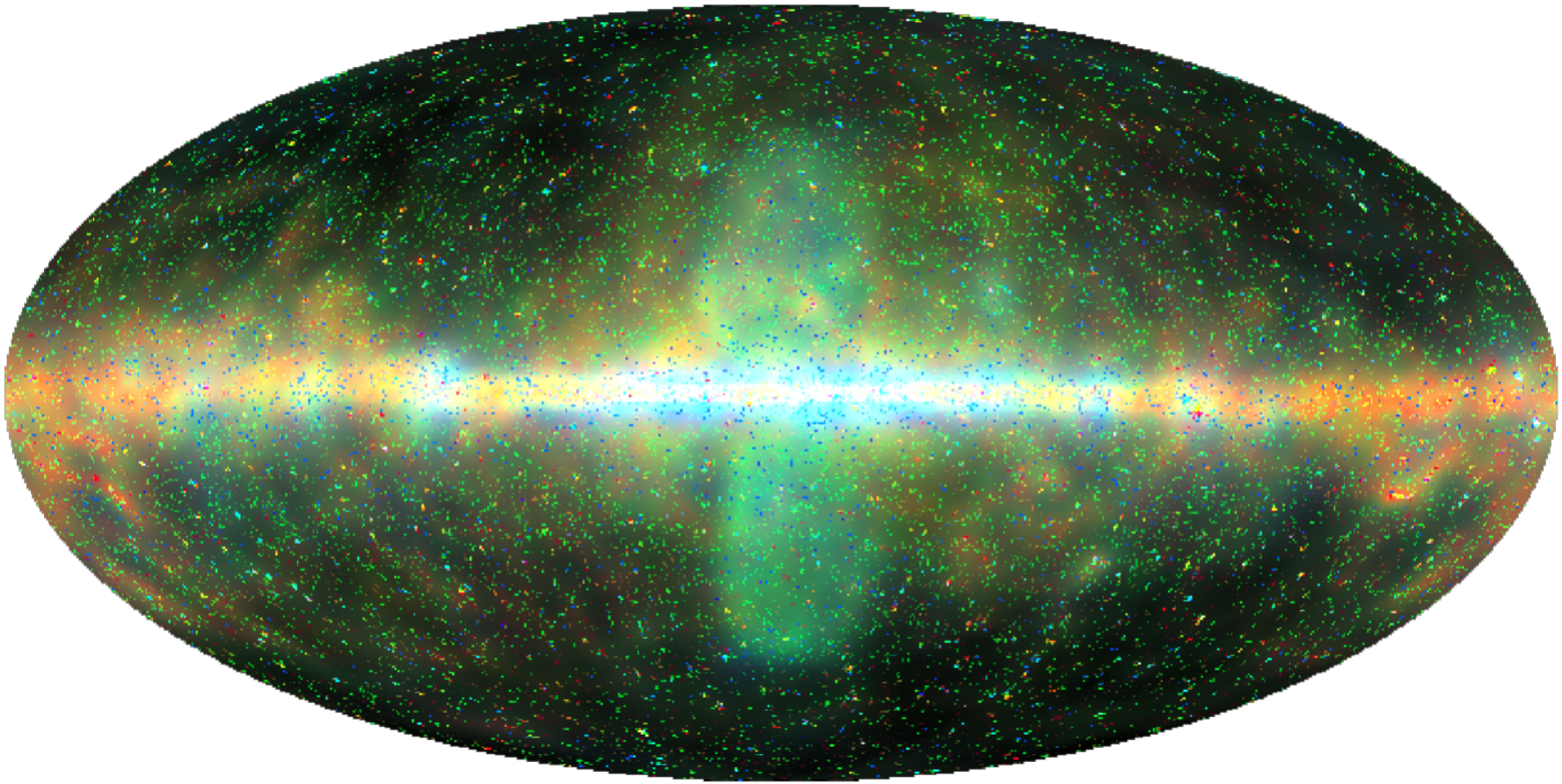
6.5 years of data

Fermi all sky map



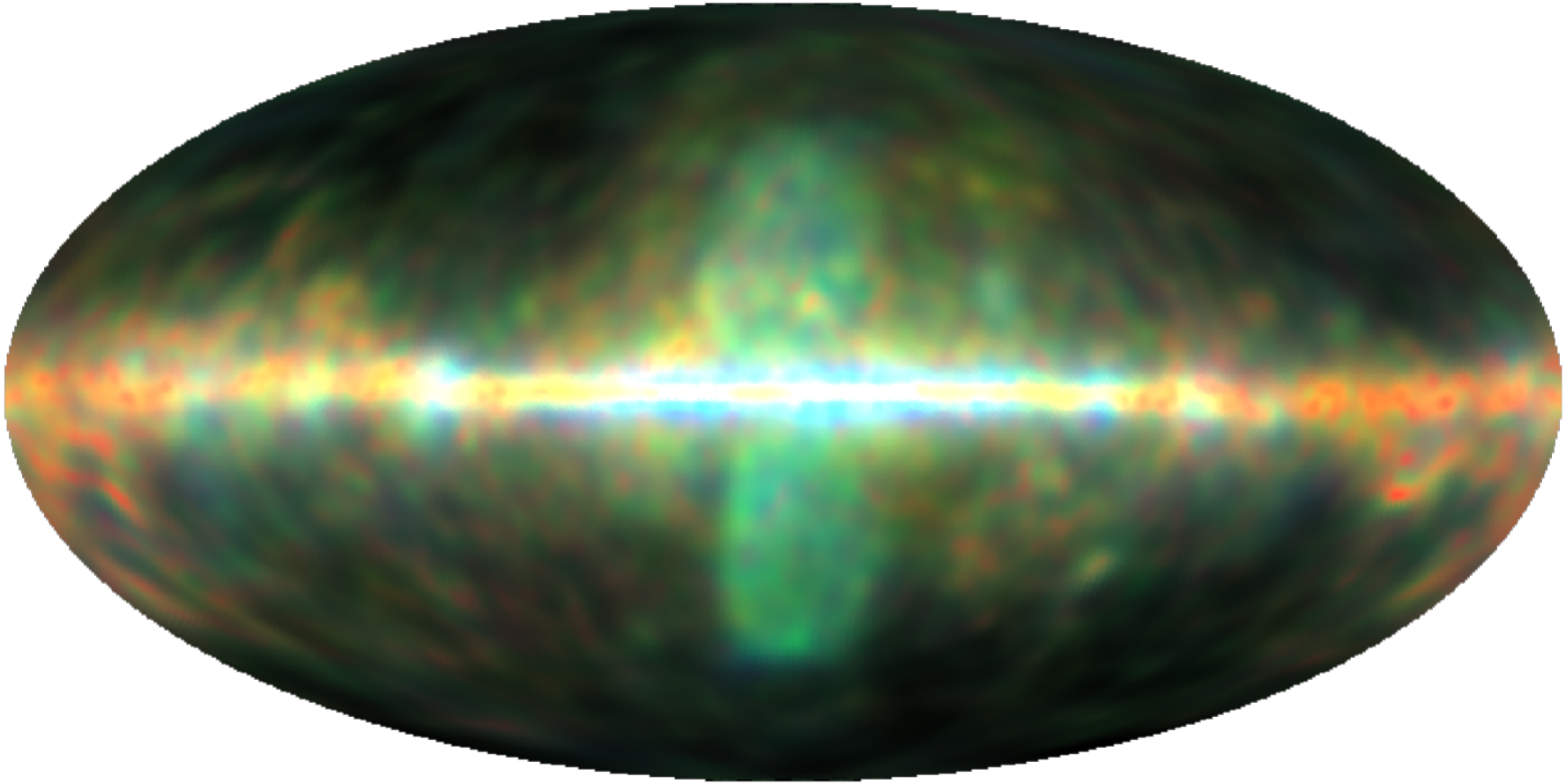
6.5 years of data … denoised

Fermi all sky map



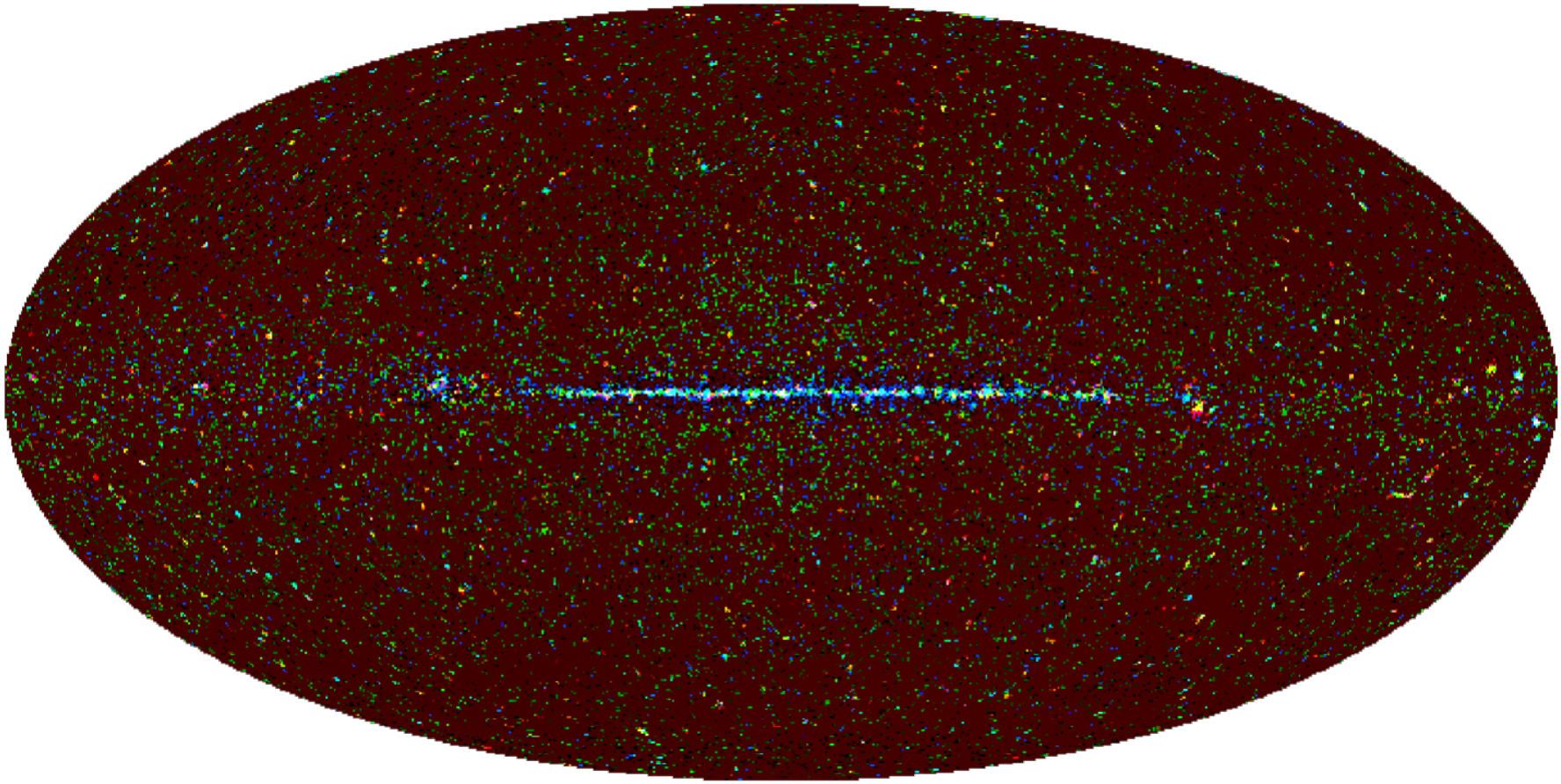
6.5 years of data … denoised, deconvolved

Fermi all sky map



6.5 years of data … denoised, deconvolved, decomposed

Fermi all sky map



6.5 years of data … denoised, deconvolved, decomposed

Test it yourself

get D3PO:

- https://github.com/information-field-theory/d3po
- http://wwwmpa.mpa-garching.mpg.de/ift/d3po/

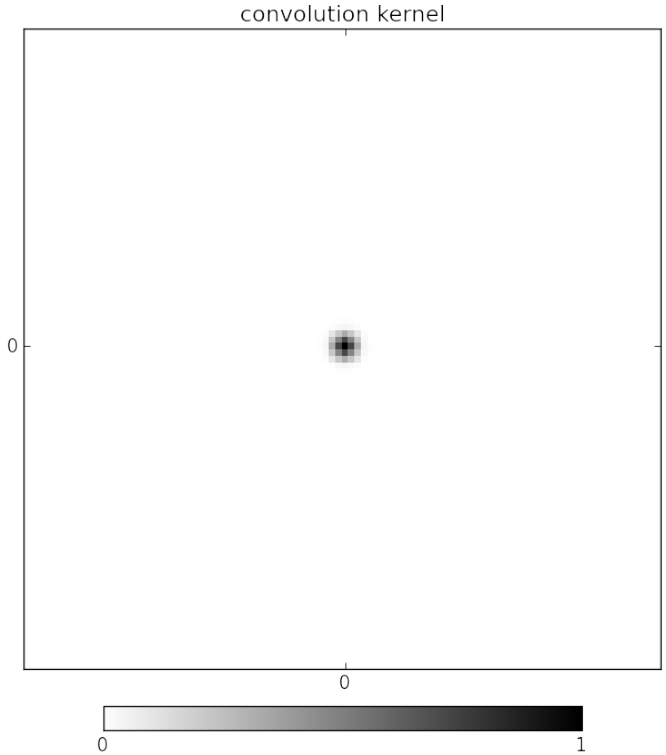
Installation:

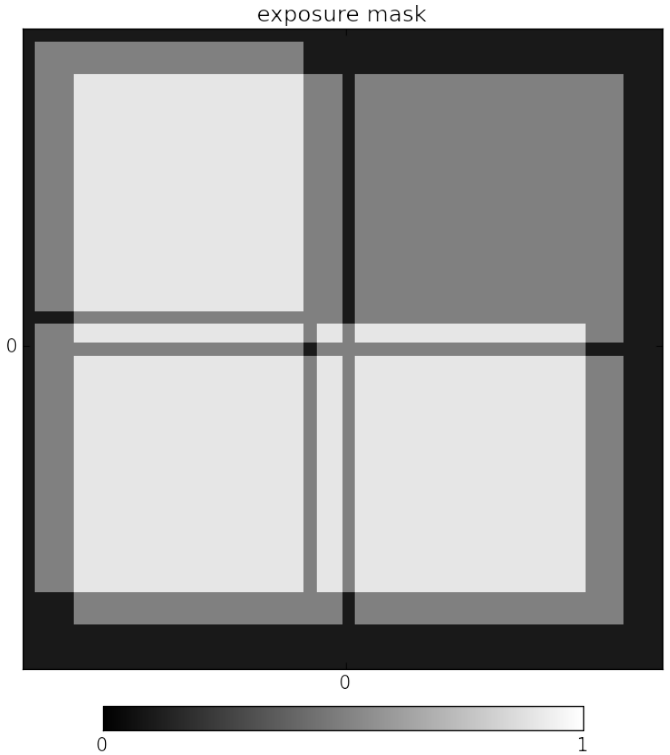
- pip install ift_d3po
- python setup.py install

D3PO a guided demo

The Response operator

$$\lambda = \boldsymbol{R}\left(\mathrm{e}^{s} + \mathrm{e}^{u}\right)$$





D3PO a guided demo

The algorithms configuration

```
##=============================================================================
## model parameters
##=============================================================================

alpha                = 1
q                    = 1E-12
sigma                = 10
beta                 = 1.5
eta                  = 1E-4

##=============================================================================
## primary settings
##=============================================================================

## flags
MAP_s                = True
MAP_t                = True
MAP_u                = None
NO_t                 = False
NO_u                 = False
notes                = True
saves                = False

## convergence tolerance
map_tol              = 1E-3
tau_tol              = 5E-2

## postprocess uncertainty
aftermath            = True

## multiprocessing
ncpu                 = 8
nper                 = 1

## random seed
seed                 = 42
```

D3PO a guided demo

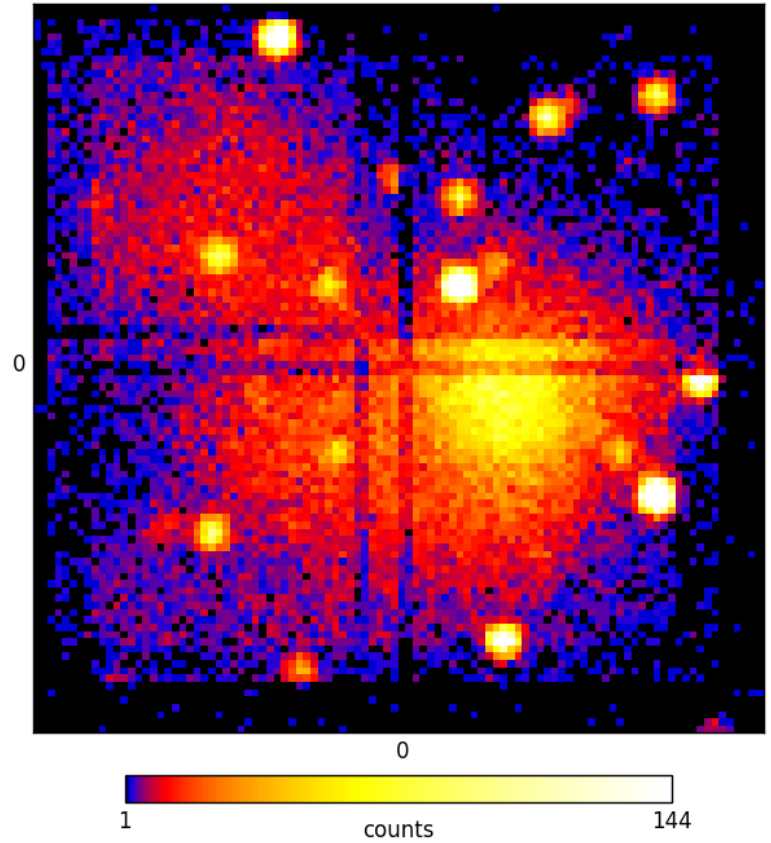
Solve the problem

```
##set up demo
demodirectory= "./" ##modify do D3PO install directory
R, d =load_demo(demodirectory)

## set up problem
problem= d3po.problem(R, configfile="demo_config",
                      workingdirectory="./de3po_demo/")

##solve problem
problem.solve

##get results
s,u,rho_s,rho_u = problem.get(s=True,u=True,rho_s=True,rho_u=True)
```

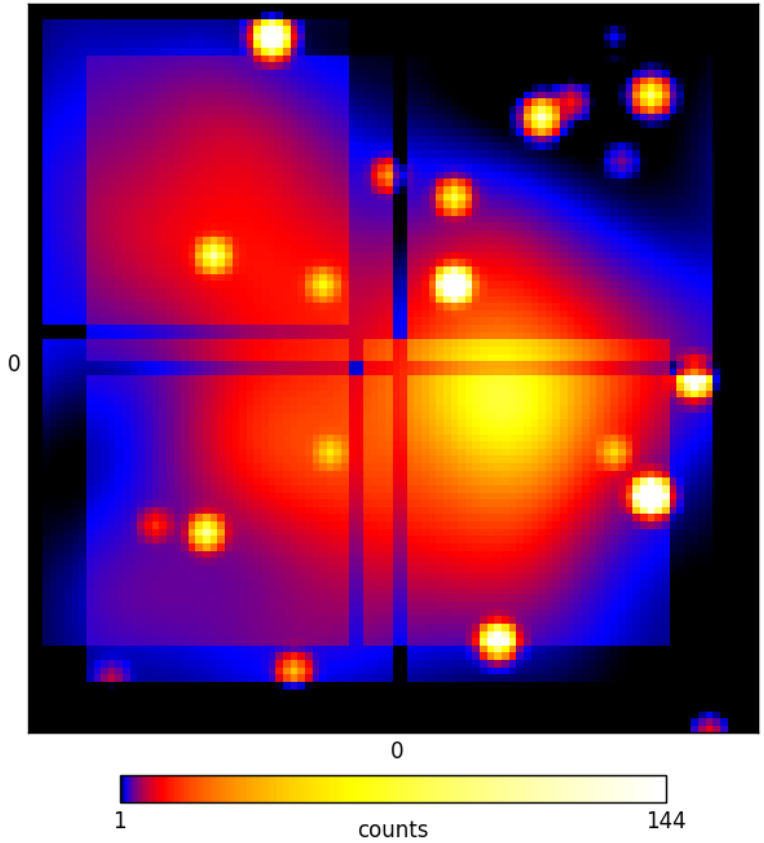
D3PO a guided demo

D1PO

raw input data



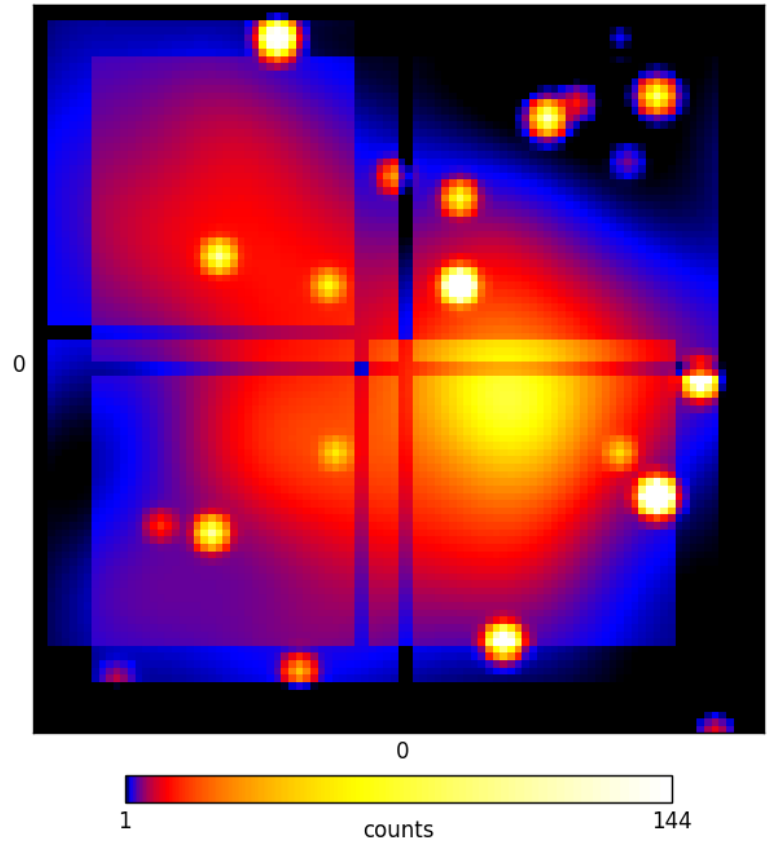
denoised observations



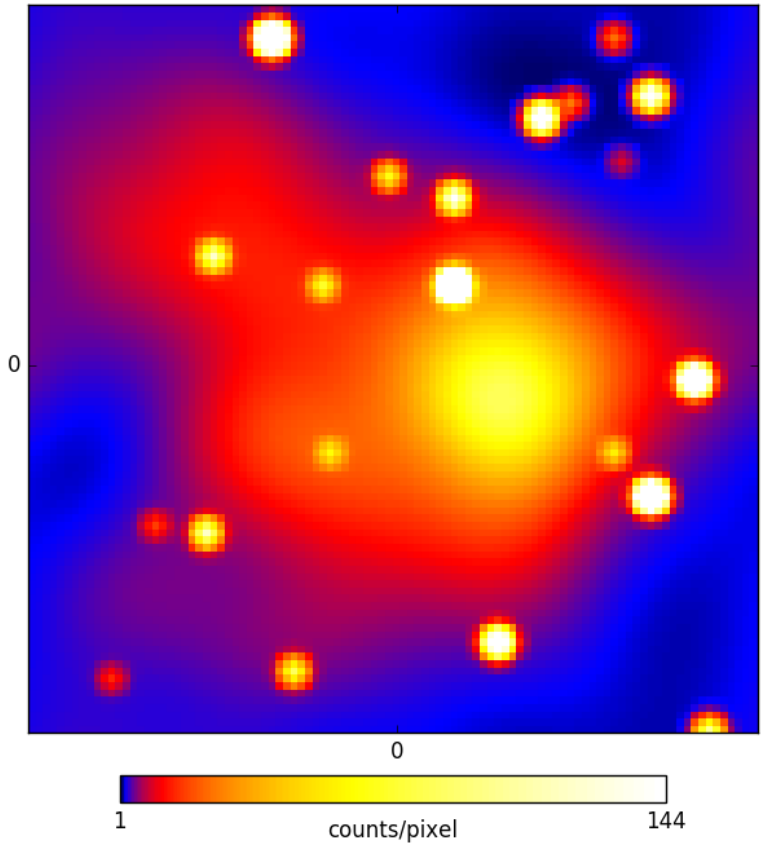
D3PO a guided demo

D12PO

denoised observations



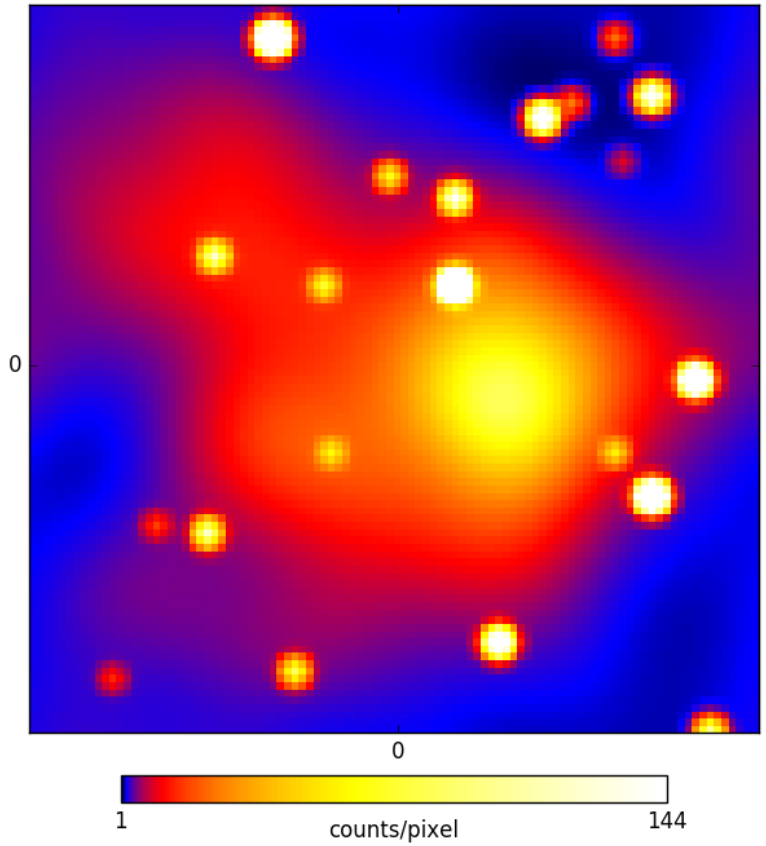
demasked photon flux



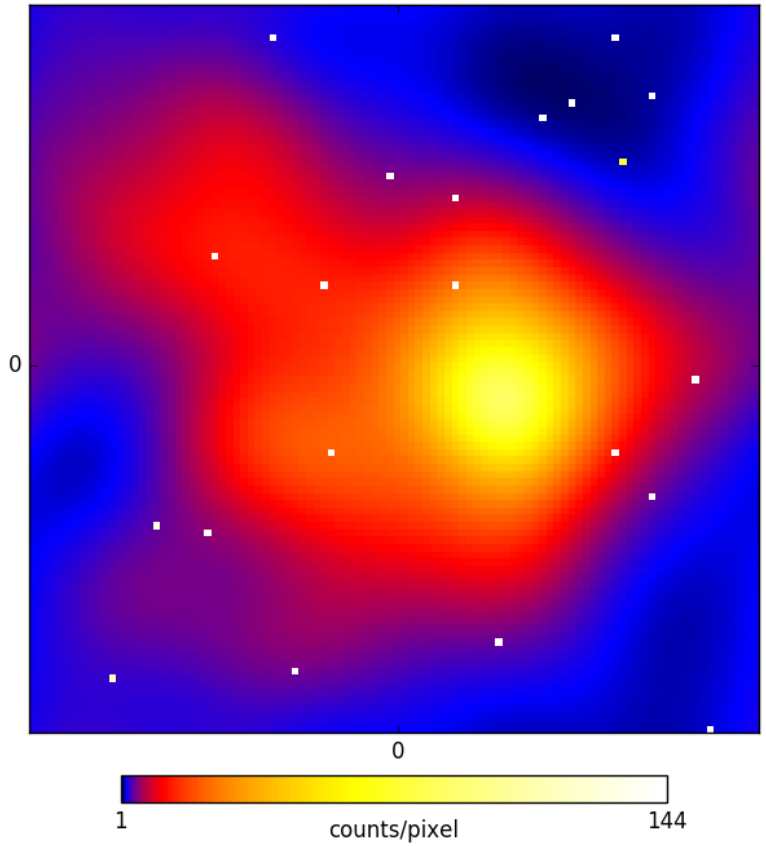
D3PO a guided demo

D123PO

reconvolved photon flux



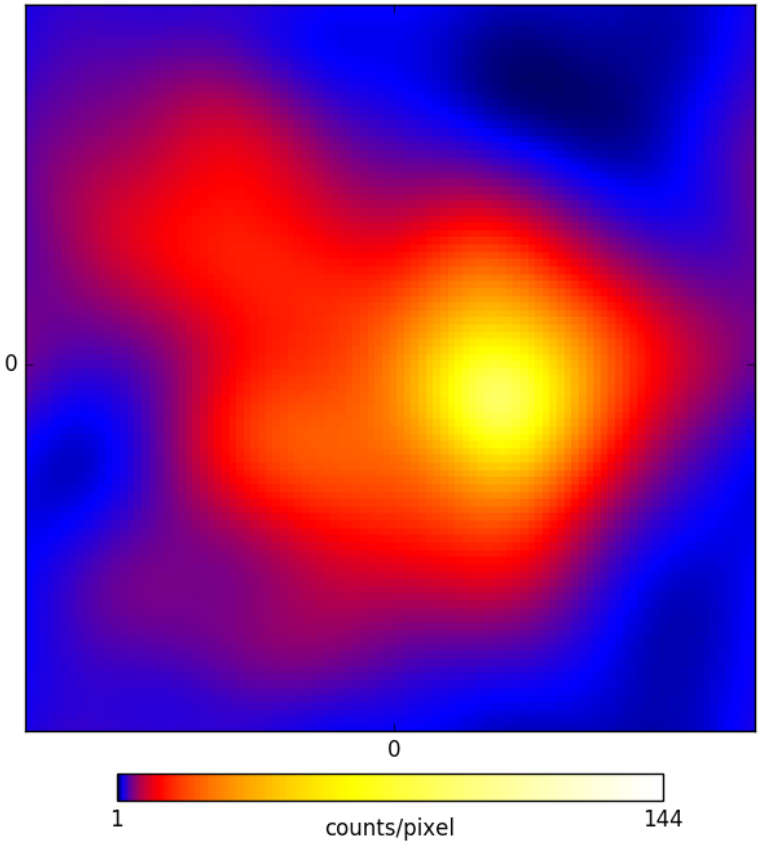
total photon flux



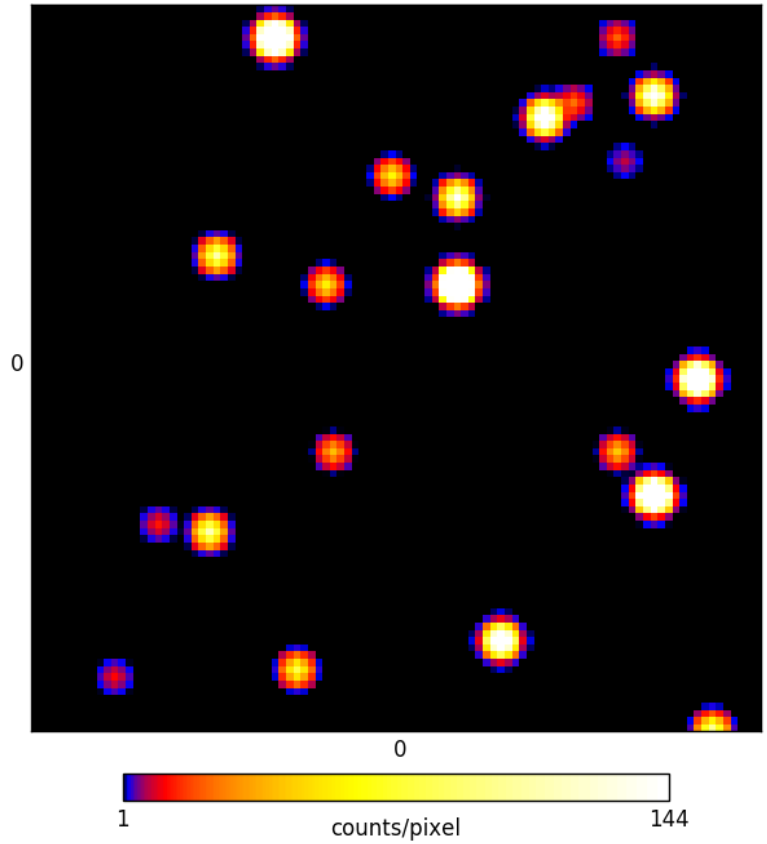
D3PO a guided demo

D123PO

diffuse photon flux



point like photon flux



D3PO package

An informal step-by-step guide

1. Data & Signal
    - Signal & data space
    - Response
2. Configuration
    - Model parameters
    - Secondary settings
    - numerical settings
3. Solve your problem