ixpeobssim: a simulation and analysis framework for IXPE

Michela Negro — Louisiana State University
Niccolò Di Lalla — Stanford University
Stefano Silvestri — INFN Pisa

Based on work done by a larger group
(see L. Baldidni et al., 2022, SoftwareX, Volume 19, id. 101194)

Simulation and Analysis

- Simulation and analysis framework:
  - Based on `python` programming language and the associated scientific ecosystem
  - Designed to produce fast and realistic simulated IXPE observations
  - Complemented by a suite of post-processing applications to select, bin and analyze simulated and real IXPE data

- Output data are:
  - Event lists in FITS format, containing a strict superset of the information included in the publicly released IXPE data products
  - Fully compliant with the visualization and analysis tools commonly used by the X-ray community (`XSPEC`, `Sherpa`, `3ML`, `DS9`, `HENDRICS`).

Simulation - Several example provided

Installation, overview, documentation and tutorials are available:

If you download the package, you'll find examples of simulations at

`ixpeobssim/ixpeobssim/examples/`

- cena.py
- instrumental_bkg.py
- ngc1068.py
- opp_sequence.py
- sgra.py
- toy_binary.py
- toy_casa.py
- toy_disk.py
- toy_gauss_disk.py
- toy_livetime_phase.py
- toy_livetime_time.py
- toy_magnetar.py
- toy_ms_pulsar.py
- toy_multiple_sources.py
- toy_offaxis.py
- toy_periodic_source.py
- toy_point_source_bkg.py
- toy_point_source_grayfilter.py
- toy_point_source.py
- toy_pollin.py
- toy_radial_disk.py
- toy_softwarex.py
- toy_tangential_disk.py
- toy_xspec_point_source.py
- tycho_3d.py

Simulations with ixpeobssim

When is it recommended to use ixpeobssim simulations?

- Your source is **faint**
- Your source is **extended**
- your source is **variable** (or flux decaying in time) during the observation

Otherwise, webPIMMs should give a good enough MDP estimate

New release of ixpeobssim

Current version: ixpeobssim (33.2.1) - Fri, 26 Jun 2026 16:57:34 +0200

New IRFs

Adding new response files for calendar years 2025 and 2026, that should match precisely what is available on the HEASARC CALDB for the same time intervals (modulo the fact that ixpeobssim uses the arf files, while ixpecalcarf recalculates everything starting from the modulation factor and the quantum efficiency).

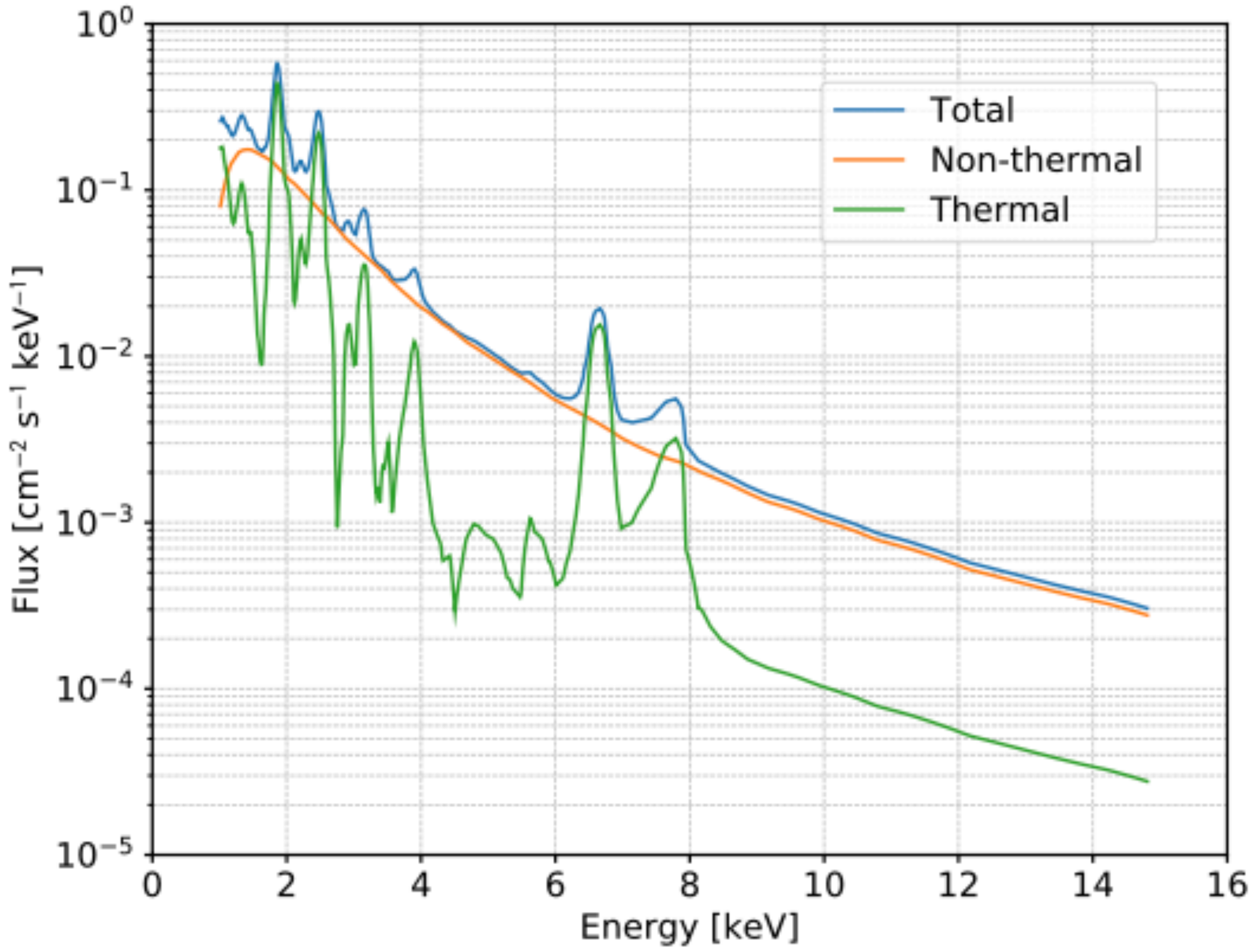
User has to select the appropriate set of IRFs (e.g. `irfname=ixpe:obssim20260701:v13`*)*

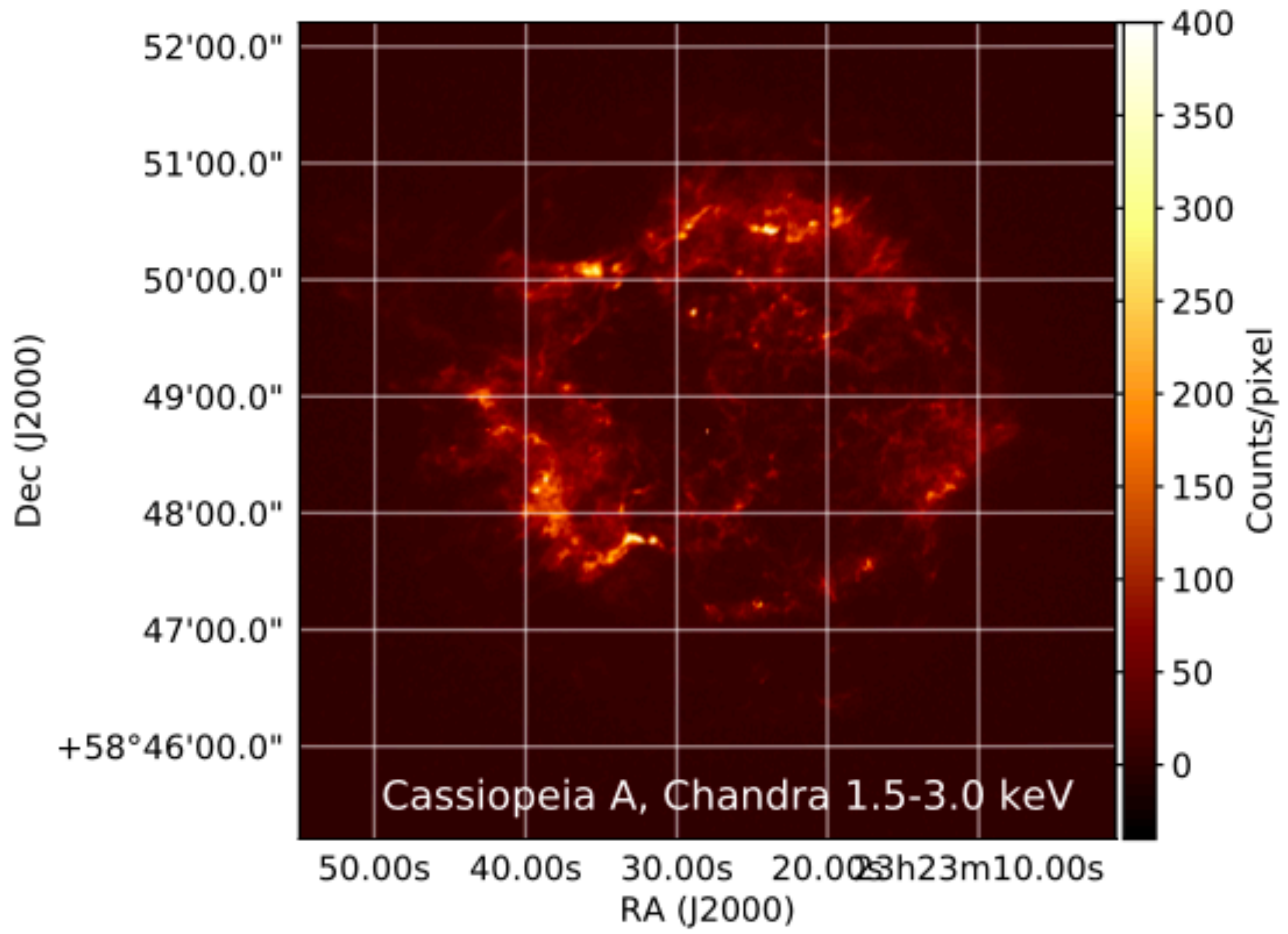
New background

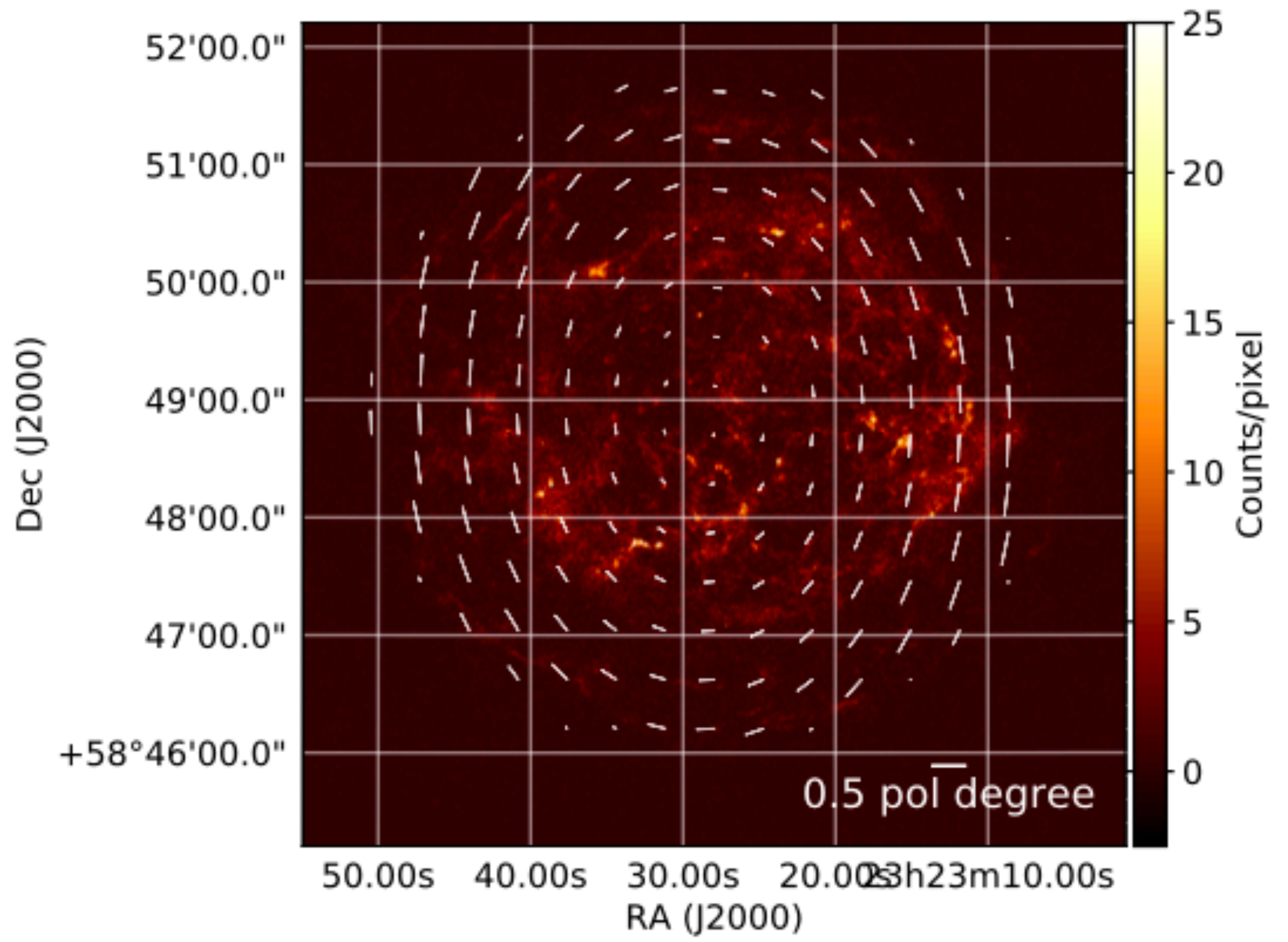
All examples pointing to the new background template that only uses in-eclipse data.

Simulation – Source Model definition

- Need to define essentially three properties:
  - Morphology (point sources, disks, annuli, generic extended sources from FITS images)
  - Energy spectrum in units of $[\mathrm{cm}^{-1}\,\mathrm{s}^{-1}\,\mathrm{keV}^{-1}]$
  - Polarization model (degree and angle, or Stokes parameters Q and U)
- Can use a Chandra photon list in lieu of defining morphology and spectrum
- Can overlay several components in the same model
- Support for time-dependent transient and periodic sources







Simulation – Configuration file example

Example of a point source with background

```python
import numpy

from ixpeobssim.config import file_path_to_model_name, bootstrap_display
from ixpeobssim.srcmodel.bkg import xTemplateInstrumentalBkg
from ixpeobssim.srcmodel.polarization import constant
from ixpeobssim.srcmodel.roi import xPointSource, xROIModel
from ixpeobssim.srcmodel.spectrum import power_law
from ixpeobssim.utils.fmtaxis import fmtaxis
from ixpeobssim.utils.matplotlib_ import plt, setup_gca


__model__ = file_path_to_model_name(__file__)


# Source coordinates, in decimal degrees.
SRC_NAME = 'EX Hydrae'
SRC_RA, SRC_DEC = 193.101, -29.2489

# Pointing coordinates
PNT_RA, PNT_DEC = SRC_RA, SRC_DEC

# Spectral and polarimetric parameters
PL_NORM = 1.8e-2
PL_INDEX = 1.65
PD = 0.08
PA = -30.
SPEC = power_law(PL_NORM, PL_INDEX)
POL_DEG = constant(PD)
POL_ANG = constant(numpy.radians(PA))

# Definition of the sources and the region of interest.
SRC = xPointSource('EX_Hydrae', SRC_RA, SRC_DEC, SPEC, POL_DEG, POL_ANG)
BKG = xTemplateInstrumentalBkg()
ROI_MODEL = xROIModel(PNT_RA, PNT_DEC, SRC, BKG)
```

Define Source name and coordinates

Define the spectral and polarization parameters

Define the Region of Interest (RoI): Source + background

Simulation – Configuration file example

Example of a point source with background

```python
def run():
    """Run all.
    """
    simulate()
    select()
    bin_()


if __name__ == '__main__':
    pipeline.bootstrap_pipeline('toy_point_source_bkg')
```

This simulated the fits files

This runs basic analyses to see the results

Simulation – Run script

Example of a point source with background

```python
import numpy

import ixpeobssim.core.pipeline as pipeline
import ixpeobssim.config.toy_point_source_bkg as input_model
from ixpeobssim.binning.polarization import xBinnedPolarizationCube


DURATION = 500000.

def simulate():
    """Run the simulation.
    """
    pipeline.xpobssim(duration=DURATION, saa=False, occult=False)
```

- Simulate an observation starting from an arbitrary source model:
  - Calculate the expected number of events by convolving the source spectrum with the effective area and extract the event times based on the light curve
  - Extract the true energies and sky positions and smear them with energy dispersion and PSF
  - Generate the angular distribution of the photoelectrons according to the polarization model
- With more components the simulation is separately performed for each one and the resulting photon lists are then merged together.

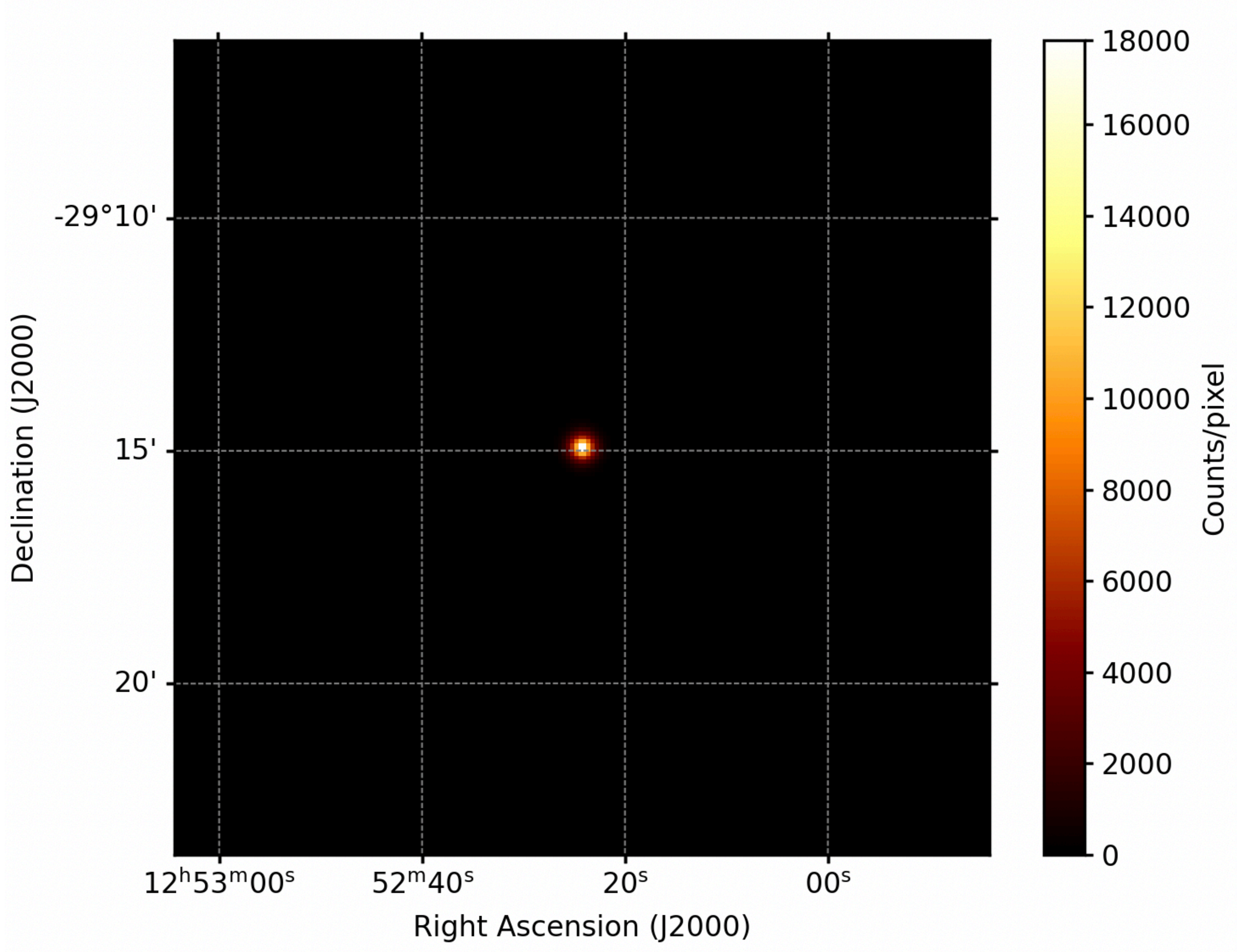
Simulation – Analyze

Example of a point source with background

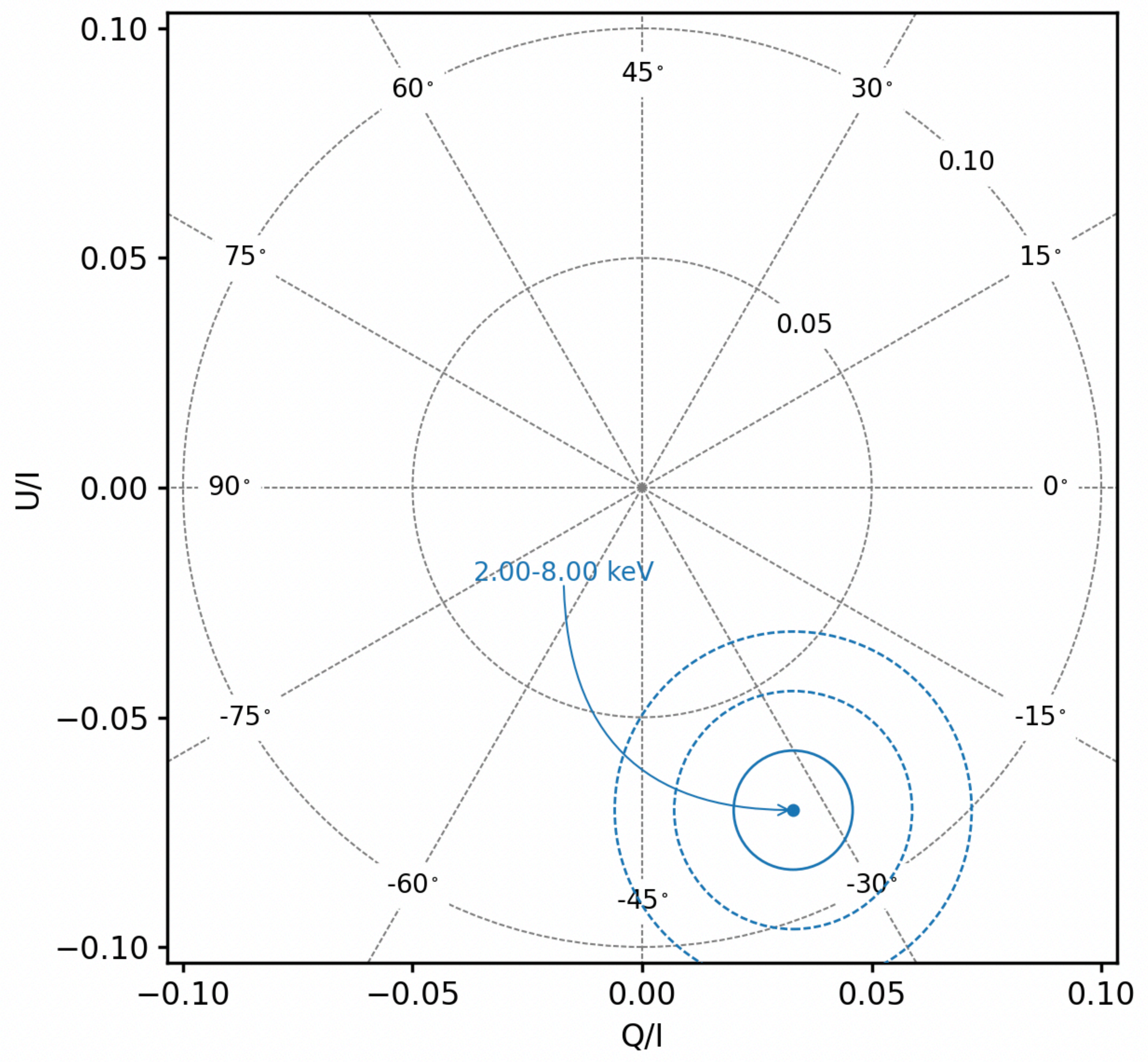
```
def select(src_rad=0.75, bkg_inner_rad=1.5, bkg_outer_rad=3.):
    """Select the photon lists.
    """
    file_list = pipeline.file_list()
    pipeline.xpselect(*file_list, rad=src_rad, suffix='src')
    pipeline.xpselect(*file_list, innerrad=bkg_inner_rad, rad=bkg_outer_rad, suffix='bkg')


ENERGY_BINNING = numpy.array([2., 3., 4., 5.5, 8.])

def bin_():
    """Create the necessary binned files.
    """
    pipeline.xpbin(*pipeline.file_list(), algorithm='CMAP')
    pipeline.xpbin(*pipeline.file_list('src'), algorithm='CMAP')
    pipeline.xpbin(*pipeline.file_list('bkg'), algorithm='CMAP')
    kwargs = dict(algorithm='PCUBE', ebinalg='LIST', ebinning=ENERGY_BINNING)
    pipeline.xpbin(*pipeline.file_list('src'), **kwargs)
    pipeline.xpbin(*pipeline.file_list('bkg'), **kwargs)
```

Results of the simulations

CMAP



PCUBE



Quantity	2.00--8.00 keV
E_MEAN	3.8810063644059603
COUNTS	254479.0
MU	0.336022381301422
W2	2626.732421875
N_EFF	105470.5859375
FRAC_W	0.4144569333324164
MDP_99	0.03931184263898911
I	16644.609375
I_ERR	51.25165939331055
Q	547.6119384765625
Q_ERR	215.69568516734992
U	-1168.50732421875
U_ERR	215.67226517942316
QN	0.032900258898735046
QN_ERR	0.012958891392868745
UN	-0.07020334899425507
UN_ERR	0.012957484331435273
QUN_COV	2.1899076509449324e-08
PD	0.07753023505210876
PD_ERR	0.012957476699416206
PA	-32.44511293504224
PA_ERR	4.788678605149939
P_VALUE	1.6831174953030958e-08
CONFID	0.9999999831688251
SIGNIF	5.641769071481441

Smulated: PD = 8%, PA = -30 deg

New Instrumental background based on data

instrumental_bkg_template.py 8 ●

Users > mnegro > opt > anaconda3 > envs > threeML > lib > python3.9 > site-packages > ixpeobssim > config > instrumental_bkg_template.py > ...

```python

from ixpeobssim import IXPEOBSSIM_SRCMODEL
from ixpeobssim.utils.fmtaxis import fmtaxis
from ixpeobssim.config import file_path_to_model_name
from ixpeobssim.utils.matplotlib_ import plt, setup_gca
from ixpeobssim.srcmodel.roi import xROIModel
from ixpeobssim.srcmodel.bkg import xTemplateInstrumentalBkg


__model__ = file_path_to_model_name(__file__)

RA = 45.
DEC = 45.
file_path = os.path.join(IXPEOBSSIM_SRCMODEL, 'ascii', 'instrumental_bkg_template.txt')

bkg = xTemplateInstrumentalBkg(file_path = file_path)

ROI_MODEL = xROIModel(RA, DEC, bkg)


def display(emin=0., emax=15.):
    """Display the source model.
    """
    energy = numpy.linspace(emin, emax, 250)
    # Energy spectrum
    plt.figure('%s spectrum' % __model__)
    plt.plot(energy, bkg.photon_spectrum(energy))
    setup_gca(xmin=emin, xmax=emax, ymin=bkg.photon_spectrum(emax),
              logy=True, grids=True, **fmtaxis.spec)



if __name__ == '__main__':
    from ixpeobssim.config import bootstrap_display
    bootstrap_display()
```

credits: Stefano Silvestri

1

2

3

4

- **New method**: average counts (full IXPE band) from years of occultation data, filtered for in-eclipse, plus particle cut (Di Marco et al. 2023)
- Background appears stable, unpolarized; slight rate decline over years possible
- Always check it doesn't overshoot the data

Simulation – Configuration file example

Example of an extended source

```python
# Coordinates of the pointing.
RA_PNT = 350.8664167
DEC_PNT = 58.8117778
# Maximum polarization degree, and corresponding radius.
MAX_POL_DEG = 0.5
MAX_RADIUS = arcmin_to_degrees(2.8)

def _load_spec(file_name, emin=1., emax=15.):
    """Convenience function to load a spectral csv file.
    """
    file_path = os.path.join(IXPEOBSSIM_CONFIG_ASCII, file_name)
    return load_spectral_spline(file_path, emin=emin, emax=emax, delimiter=',')

# Read in the spectral models from the proper files.
total_spec_spline = _load_spec('casa_total_spectrum.csv')
non_therm_spec_spline = _load_spec('casa_nonthermal_spectrum.csv')
therm_spec_spline = total_spec_spline - non_therm_spec_spline

def therm_spec(E, t):
    """Definition of the photon spectrum for the thermal component.
    """
    return therm_spec_spline(E)

def non_therm_spec(E, t):
    """Definition of the photon spectrum for the non-thermal component.
    """
    return non_therm_spec_spline(E)
```

you can build your own spectrum ascii file

1.0154972516429583	0.2600210491838398
1.0283397977608348	0.2654538391443833
1.0305009791403983	0.27100013994125344
1.0435302195220733	0.27381659612225306
1.0501049510845077	0.27381659612225306
1.0611457106148563	0.2654538391443833
1.0678220611554607	0.25734649292415696
1.076795545992024	0.2520796282683438
1.0835798647688821	0.2520796282683438
1.097270588027409	0.2469205554964901
1.1041806806012615	0.2443807498945191
1.1134629622683097	0.2418670685351525
1.127537843733924	0.2418670685351525
1.1346335687298978	0.23569540207102993
1.1417772864645506	0.23206825257419877
1.151372256955922	0.22731873349845377
1.1586196685366936	0.22266641828924652
1.1732584983632384	0.21810931756796426
1.1855975557948553	0.21364548267070463
1.198069885144096	0.21144794187612884
1.2157639696064315	0.215865862115309 3
1.2285644224762402	0.22037608885351068
1.2389068246423216	0.22731873349845377
1.2519582638043738	0.23691700646048355
1.2704556458483907	0.2469205554964901
1.2838394405293694	0.25734649292415696
1.294650948929021	0.26821265360916674

Simulation – Configuration file example

Example of an extended source

```
# The thermal component is unpolarized.
therm_pol_ang = constant(0.)
therm_pol_deg = constant(0.)

def non_therm_radial_profile(r, E=None, t=None):
    """Radial profile of the polarization degree.

    This is going from zero at the center of the field to the target maximum value
    at the specified radius. (Note the return value is clipped between 0 and 1
    to avoid unphysical values.)
    """
    return numpy.clip((MAX_POL_DEG * r / MAX_RADIUS), 0., 1.)

non_therm_pol_field = xTangentialPolarizationField(RA_PNT, DEC_PNT, non_therm_radial_profile)
non_therm_pol_deg = non_therm_pol_field.polarization_degree_model()
non_therm_pol_ang = non_therm_pol_field.polarization_angle_model()
```

you can us your own extended polarization model

Simulation – Configuration file example

Example of an extended source

```
# Finally, the Chandra images for the thermal and non thermal components.
le_img_file_path = os.path.join(IXPEOBSSIM_CONFIG_FITS, 'casa_1p5_3p0_keV.fits')
he_img_file_path = os.path.join(IXPEOBSSIM_CONFIG_FITS, 'casa_4p0_6p0_keV.fits')

# Create the model components.
therm_comp = xExtendedSource('Cas A thermal', le_img_file_path, therm_spec,
    therm_pol_deg, therm_pol_ang)
non_therm_comp = xExtendedSource('Cas A non-thermal', he_img_file_path, non_therm_spec,
    non_therm_pol_deg, non_therm_pol_ang)
instrumental_bkg = xTemplateInstrumentalBkg()

# Create the actual ROI object.
ROI_MODEL = xROIModel(RA_PNT, DEC_PNT)
ROI_MODEL.add_sources(therm_comp, non_therm_comp, instrumental_bkg)
```

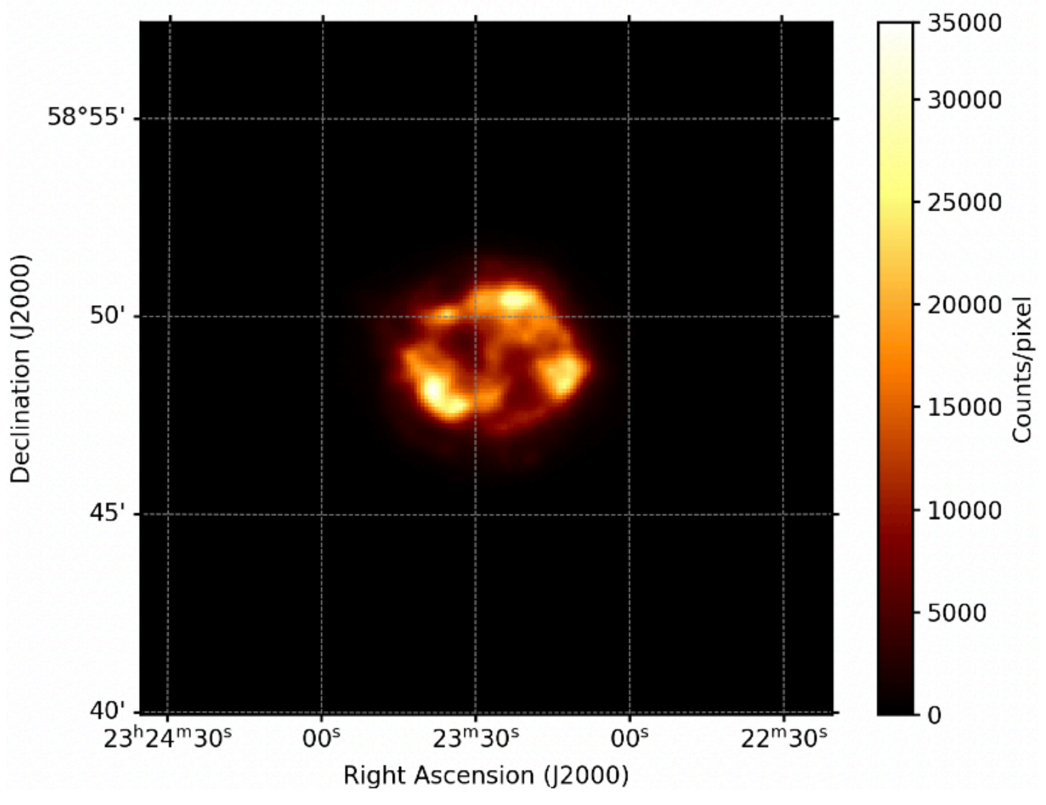
Use Chandra images for Stokes-I

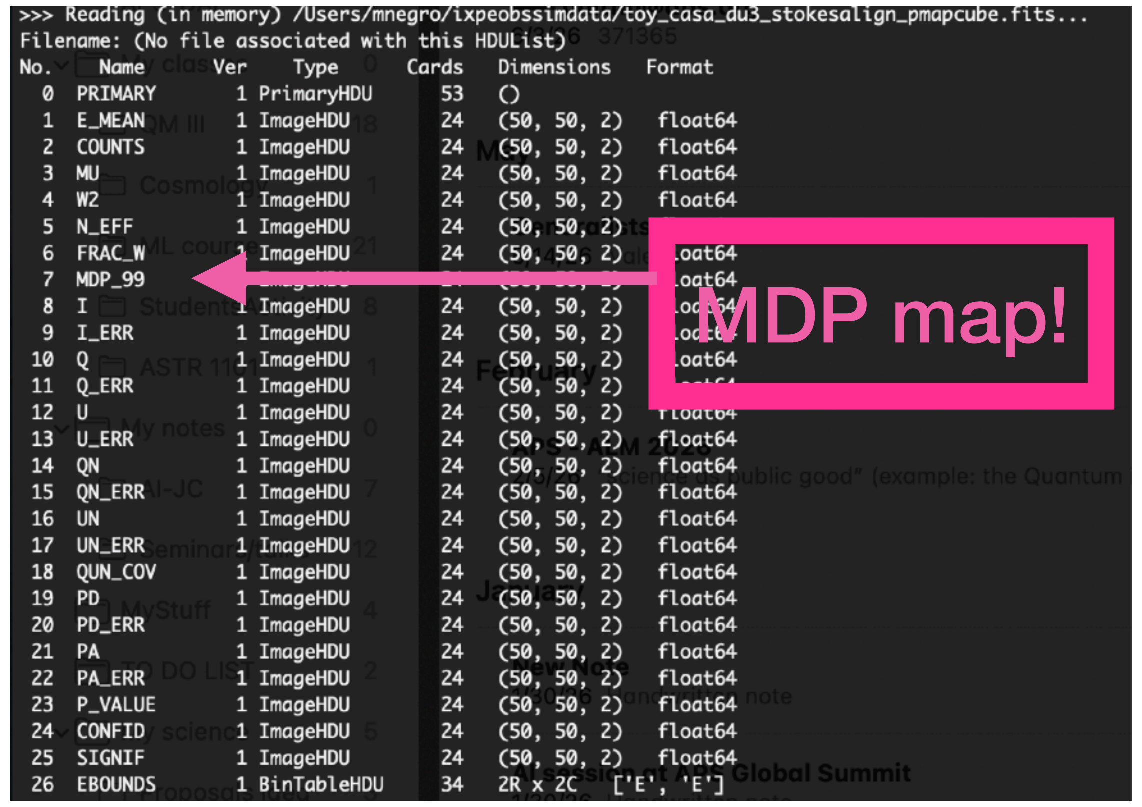
Build your ROI: thermal comp + non-thermal comp + background

Simulation – run file + analysis

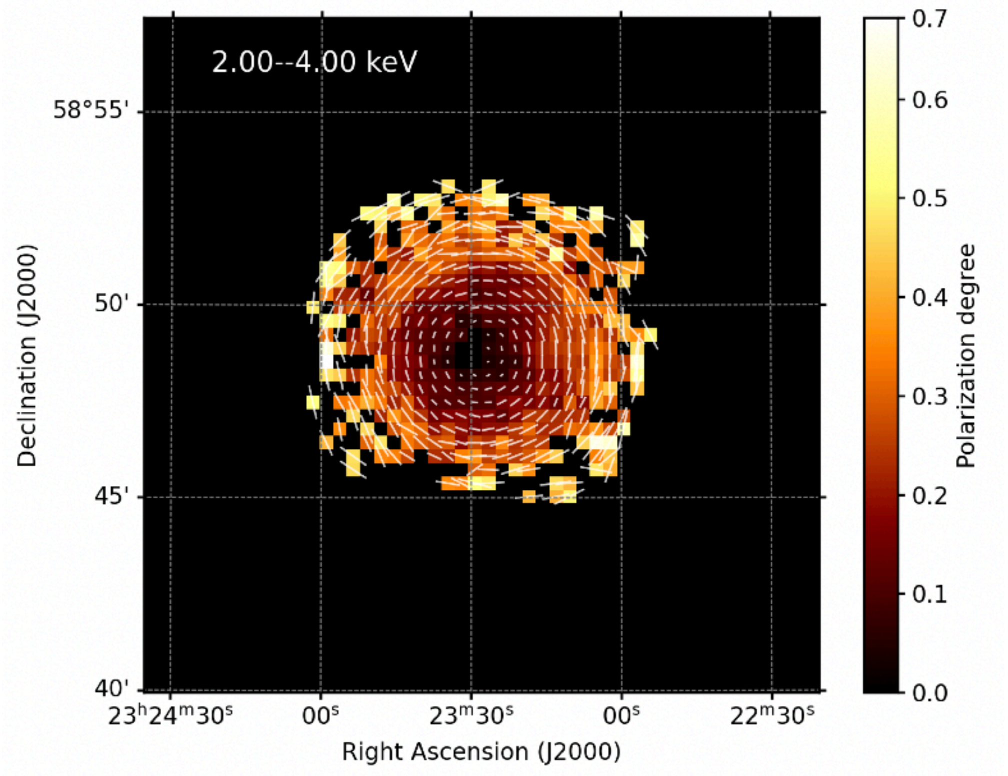
Example of an extended source

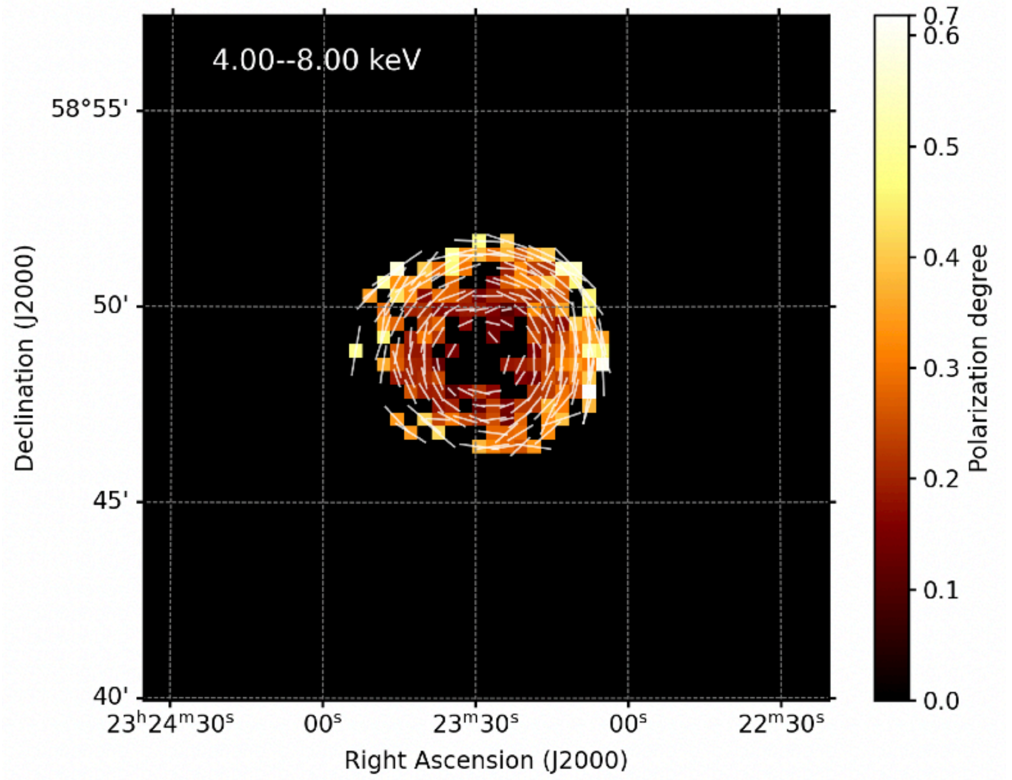
```python
DURATION = 1500000.
ENERGY_BINNING = [2., 4., 8.]


def simulate():
    """Run the simulation and align the stokes parameters tangentially.
    """
    file_list = pipeline.xpobssim(duration=DURATION)
    pipeline.xpstokesalign(*file_list, mode='TAN')

def bin_():
    """Create the counts map and the polarization map cubes, in both the original
    and the rotated (via xpstokealign) flavors.
    """
    kwargs = dict(npix=50, ebinalg='LIST', ebinning=ENERGY_BINNING)
    file_list = pipeline.file_list()
    pipeline.xpbin(*file_list, algorithm='CMAP')
    pipeline.xpbin(*file_list, algorithm='PMAPCUBE', **kwargs)
    file_list = pipeline.file_list('stokesalign')
    pipeline.xpbin(*file_list, algorithm='PMAPCUBE', **kwargs)
```





```
>>> Reading (in memory) /Users/mnegro/ixpeobssimdata/toy_casa_du3_stokesalign_pmapcube.fits...
Filename: (No file associated with this HDUList)
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      53   ()
  1  E_MEAN        1 ImageHDU        24   (50, 50, 2)   float64
  2  COUNTS        1 ImageHDU        24   (50, 50, 2)   float64
  3  MU            1 ImageHDU        24   (50, 50, 2)   float64
  4  W2            1 ImageHDU        24   (50, 50, 2)   float64
  5  N_EFF         1 ImageHDU        24   (50, 50, 2)
  6  FRAC_W        1 ImageHDU        24   (50, 50, 2)
  7  MDP_99
  8  I             1 ImageHDU        24   (50, 50, 2)
  9  I_ERR         1 ImageHDU        24   (50, 50, 2)
 10  Q             1 ImageHDU        24   (50, 50, 2)
 11  Q_ERR         1 ImageHDU        24   (50, 50, 2)
 12  U             1 ImageHDU        24   (50, 50, 2)
 13  U_ERR         1 ImageHDU        24   (50, 50, 2)   float64
 14  QN            1 ImageHDU        24   (50, 50, 2)   float64
 15  QN_ERR        1 ImageHDU        24   (50, 50, 2)   float64
 16  UN            1 ImageHDU        24   (50, 50, 2)   float64
 17  UN_ERR        1 ImageHDU        24   (50, 50, 2)   float64
 18  QUN_COV       1 ImageHDU        24   (50, 50, 2)   float64
 19  PD            1 ImageHDU        24   (50, 50, 2)   float64
 20  PD_ERR        1 ImageHDU        24   (50, 50, 2)   float64
 21  PA            1 ImageHDU        24   (50, 50, 2)   float64
 22  PA_ERR        1 ImageHDU        24   (50, 50, 2)   float64
 23  P_VALUE       1 ImageHDU        24   (50, 50, 2)   float64
 24  CONFID        1 ImageHDU        24   (50, 50, 2)   float64
 25  SIGNIF        1 ImageHDU        24   (50, 50, 2)   float64
 26  EBOUNDS       1 BinTableHDU     34   2R x 2C   ['E', 'E']
```

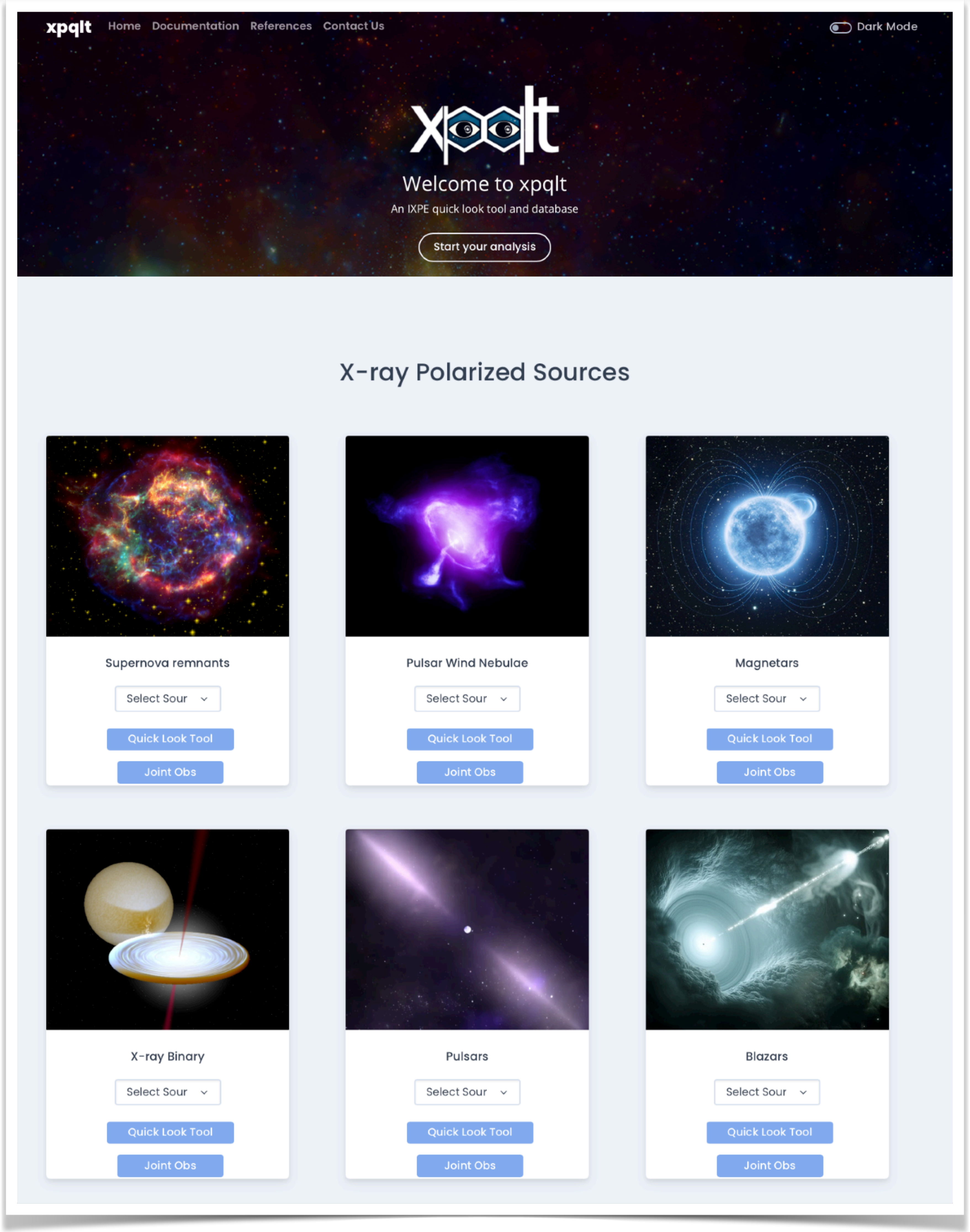





For more analysis tools see Niccolò DI Lalla's presentation to the "Friends of IXPE" forum last week:

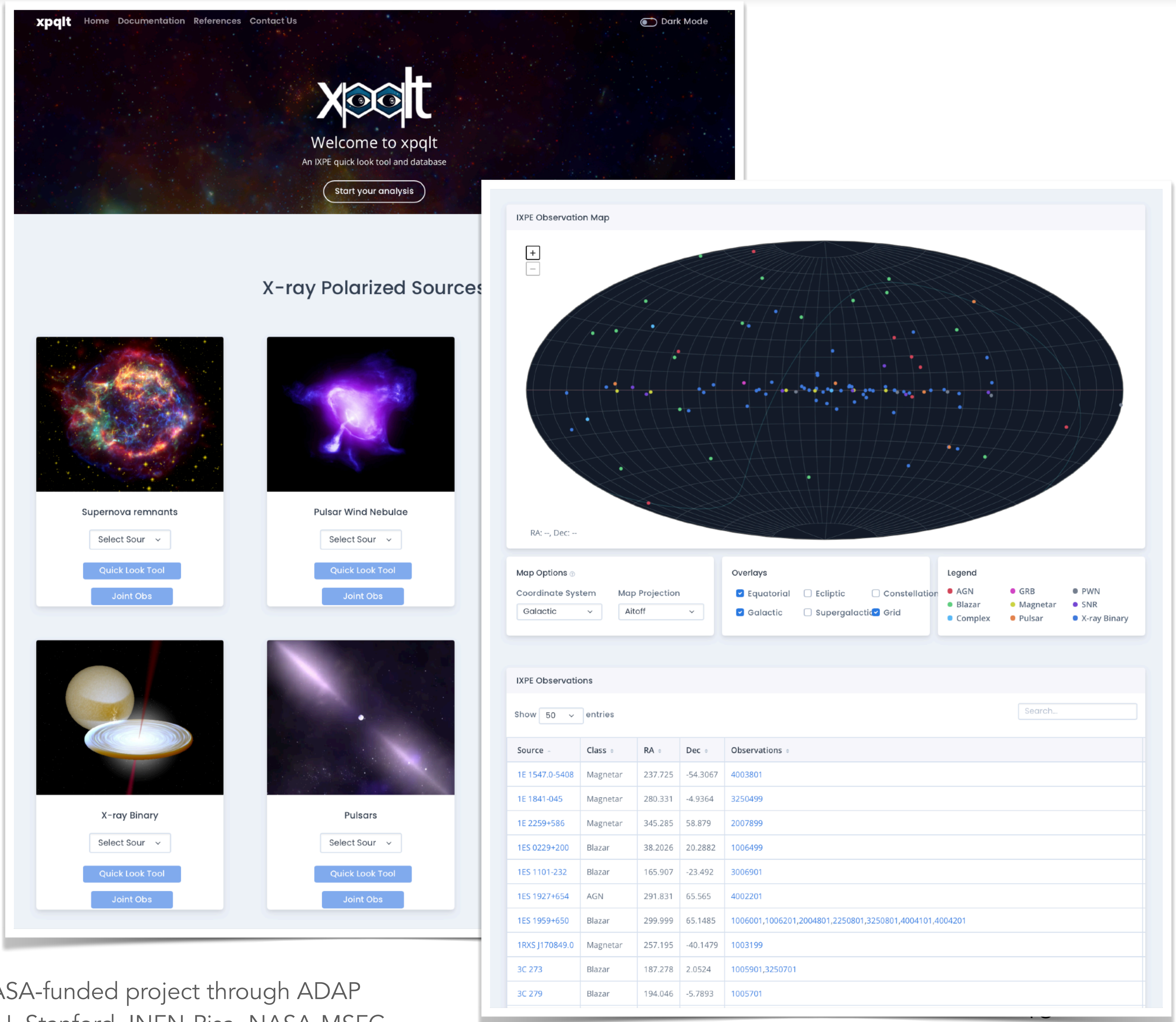
https://heasarc.gsfc.nasa.gov/docs/ixpe/friendsofixpe/20260618-NDL.pdf

xpqlt.com – a quick-look tool and database



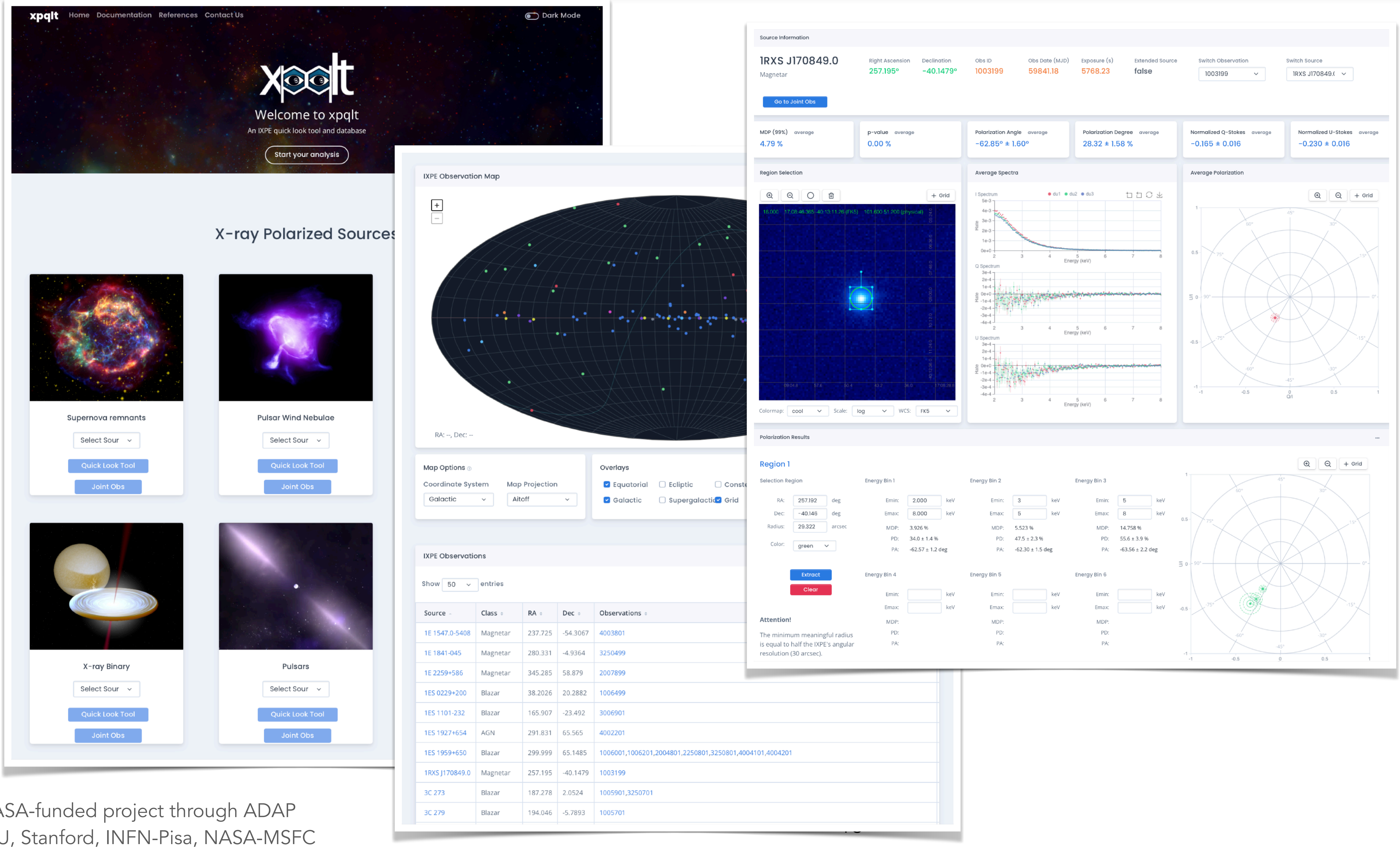
NASA-funded project through ADAP
LSU, Stanford, INFN-Pisa, NASA-MSFC

xpqlt.com – a quick-look tool and database



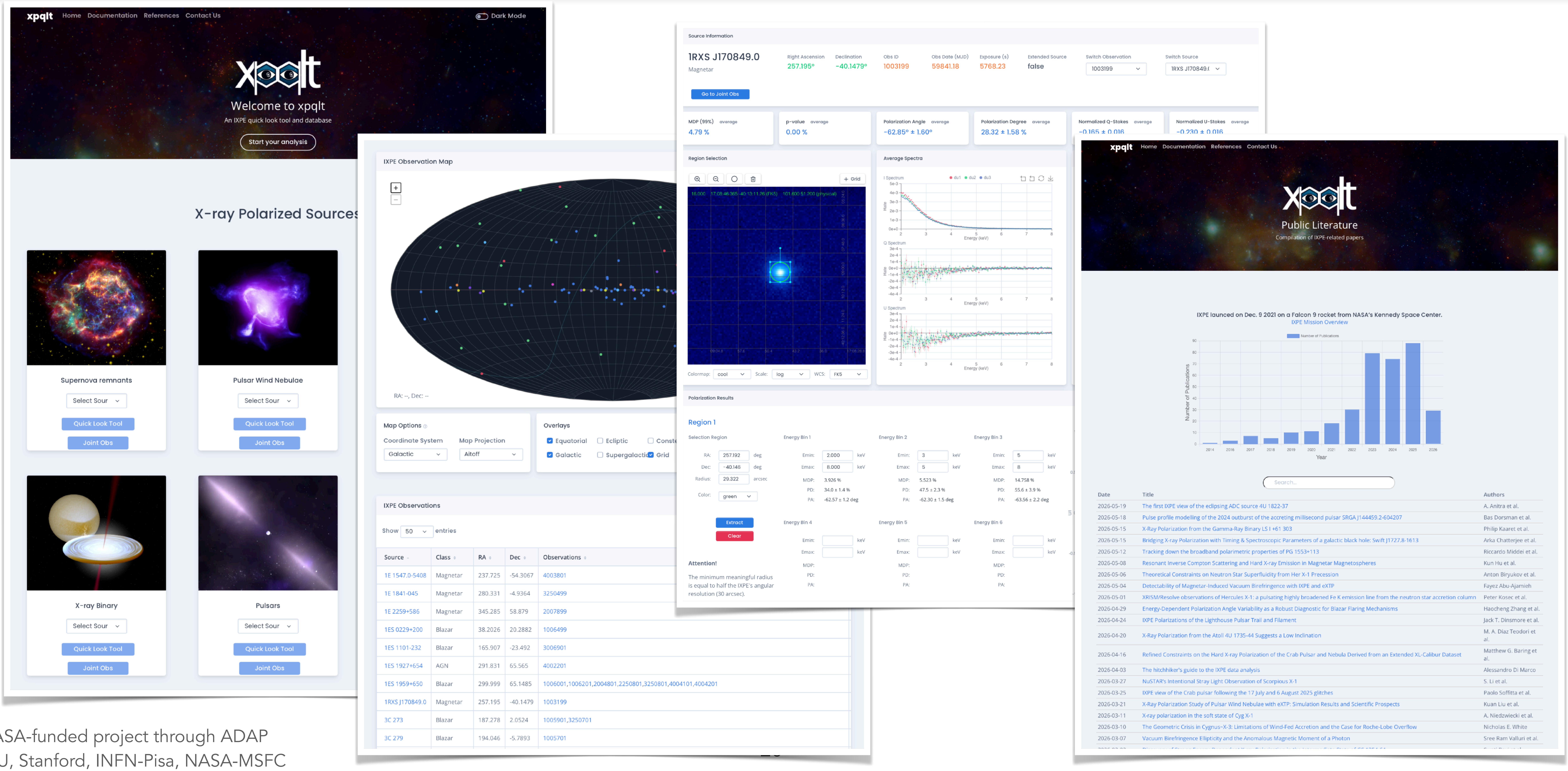
NASA-funded project through ADAP
LSU, Stanford, INFN-Pisa, NASA-MSFC

xpqlt.com – a quick-look tool and database



NASA-funded project through ADAP
LSU, Stanford, INFN-Pisa, NASA-MSFC

xpqlt.com – a quick-look tool and database



NASA-funded project through ADAP

LSU, Stanford, INFN-Pisa, NASA-MSFC

References

Installation, overview, documentation, tutorials

Github repository

Please, cite the Paper
L. Baldini et al. 2022

Visit the ixpeobssim GitHub repository to contribute and submit issues you may find while using the software.

Take a look to the ixpeobssim tutorial material that was presented at HEAD20 meeting (March 2023)

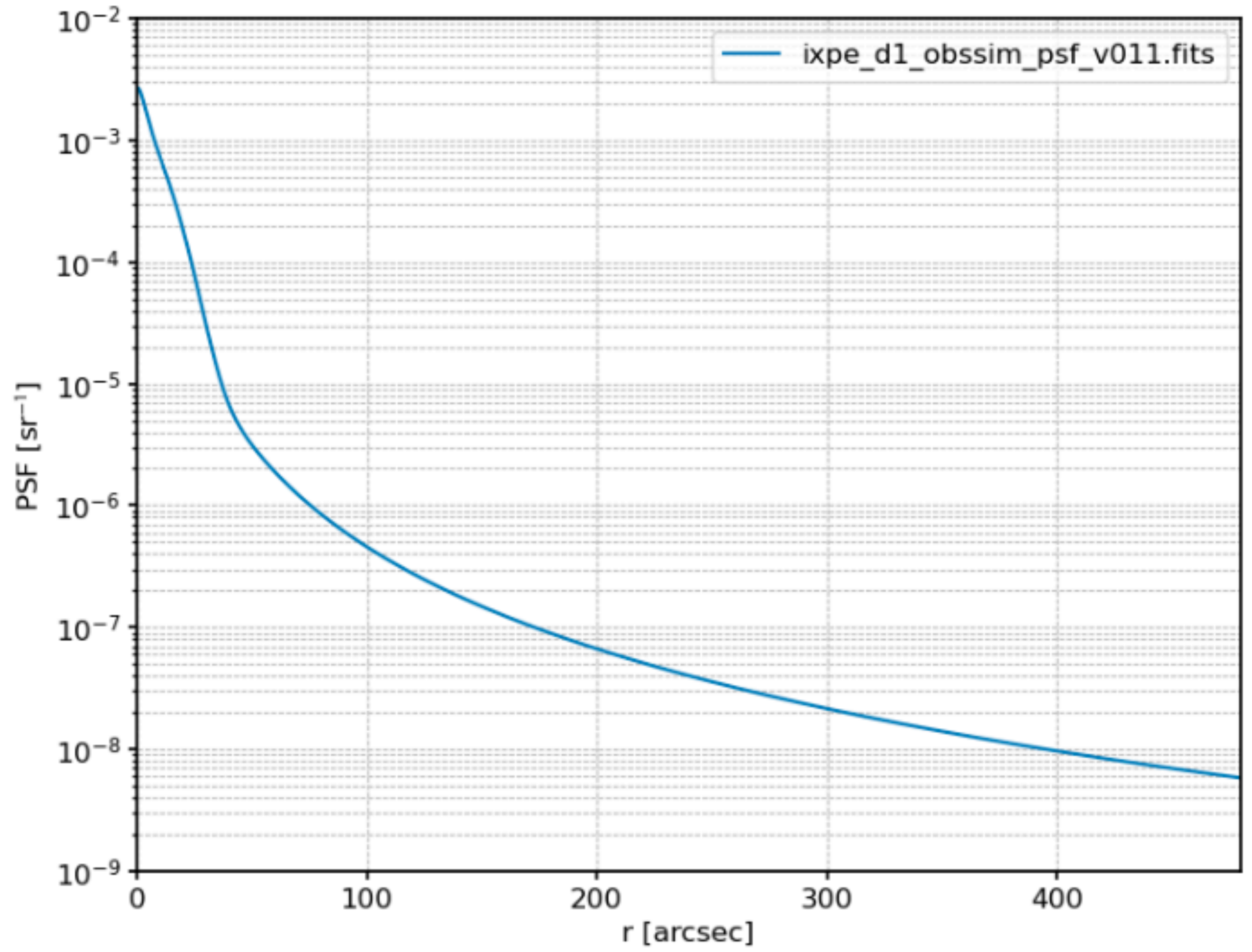

ixpeobssim: a simulation
and analysis framework for IXPE
Back-up

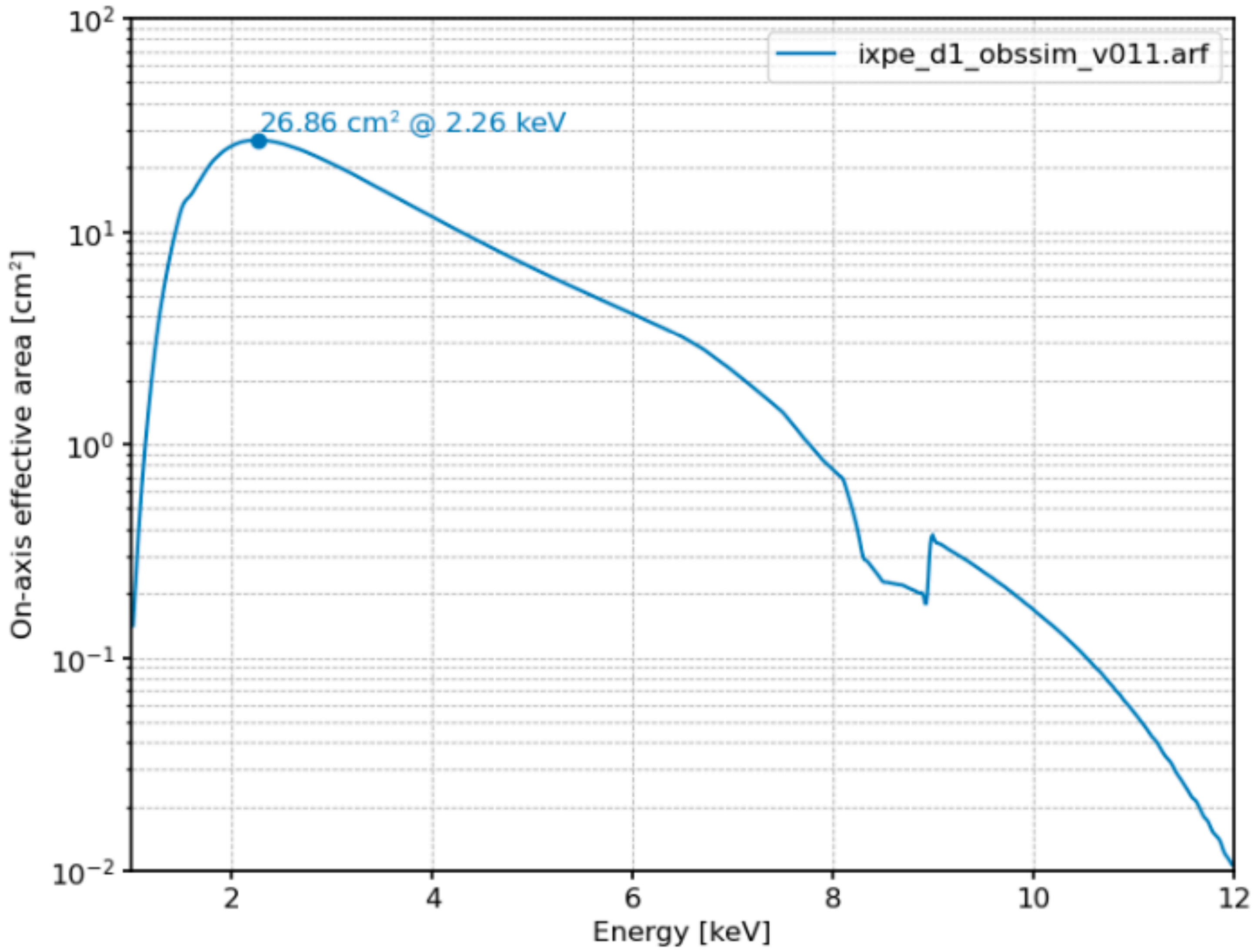
The project

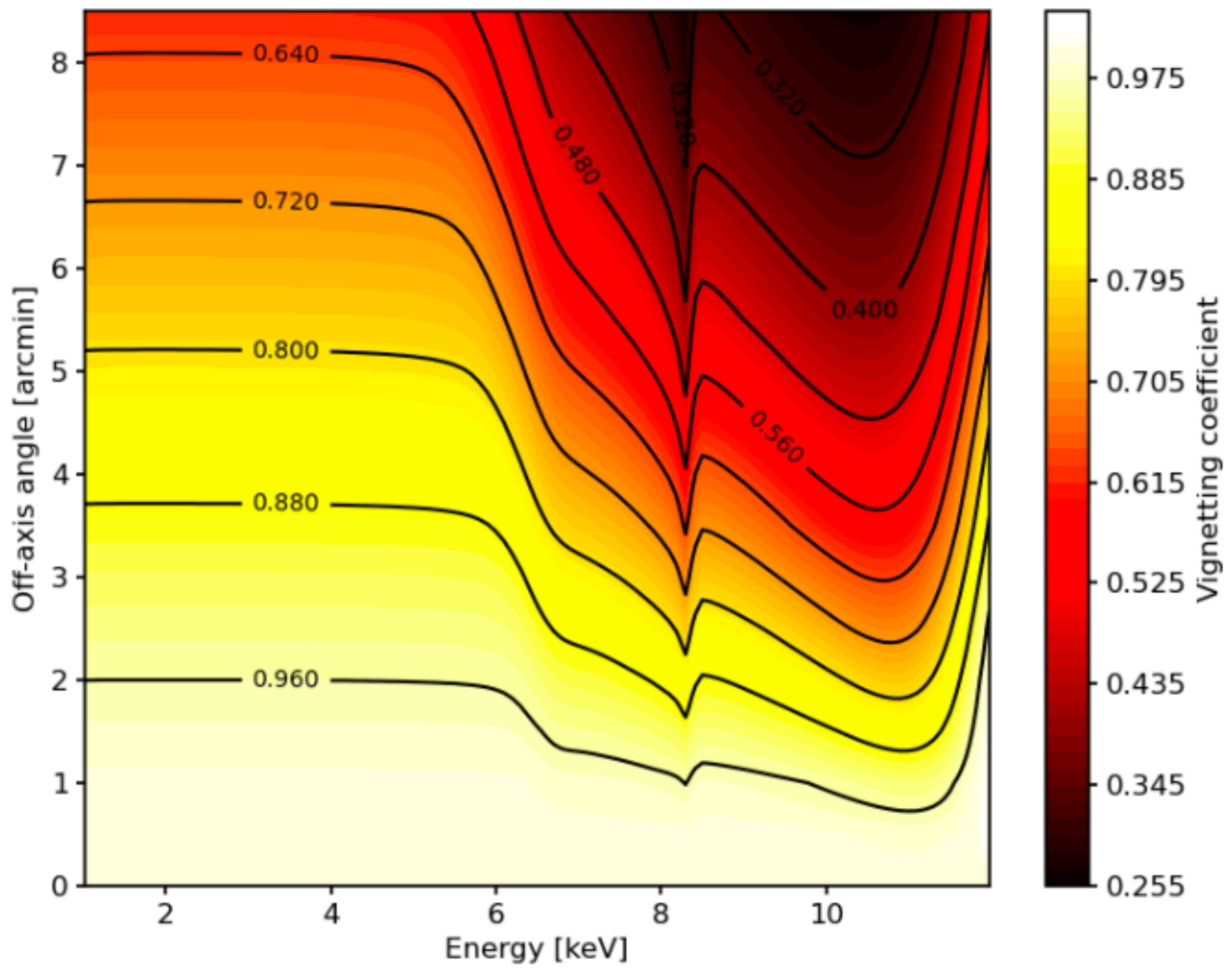
- Project started in 2015 under the name of `XIMPOL`:
  - Initially not tied to any specific mission or instrument design
  - After IXPE selection in 2017, it was renamed and progressively tailored in preparation for the new mission
  - Publicly released in 2022 to support the analysis of public IXPE data and engage the broader community in anticipation of the upcoming General Observer program

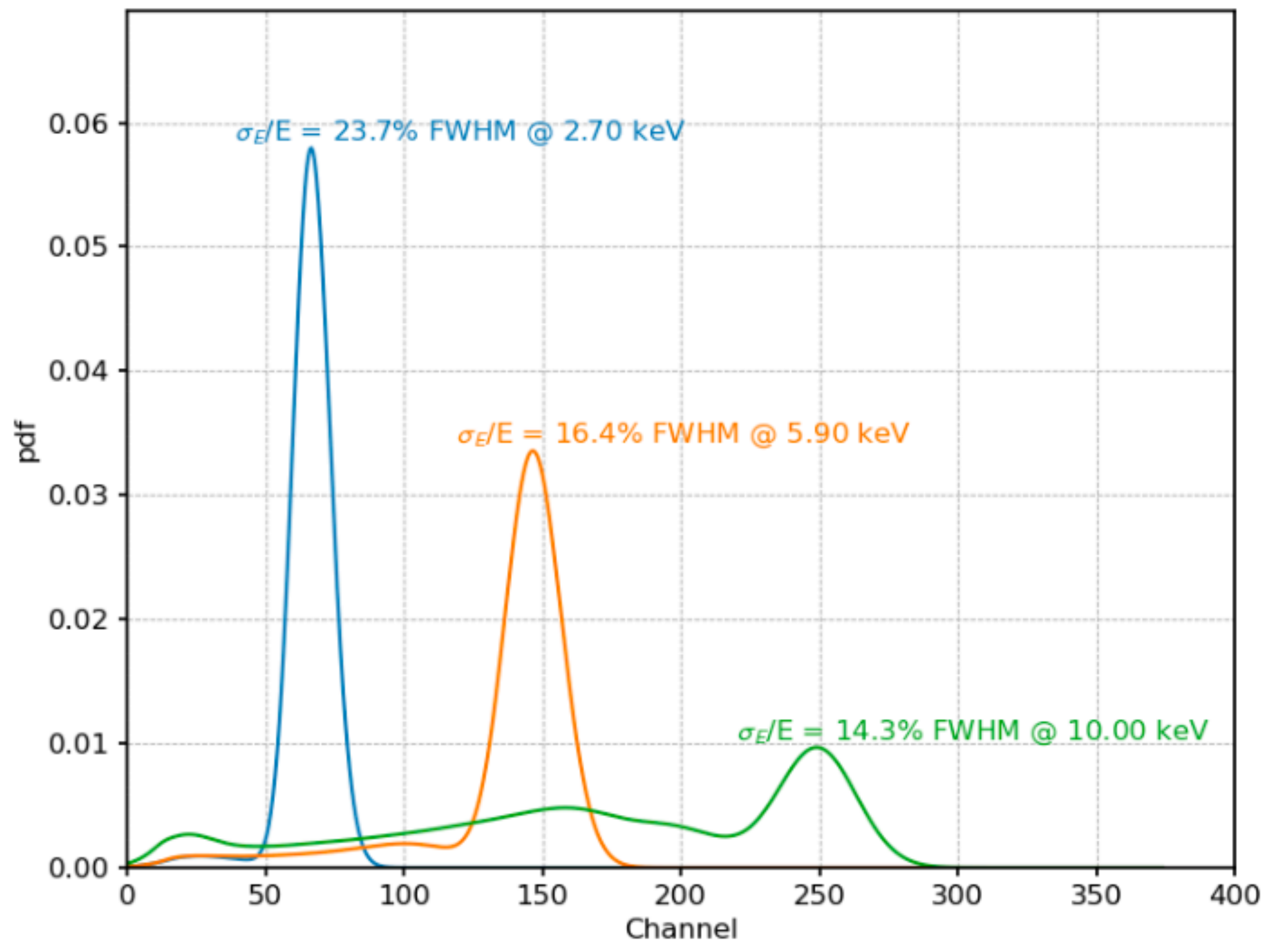
Instrument Response Functions

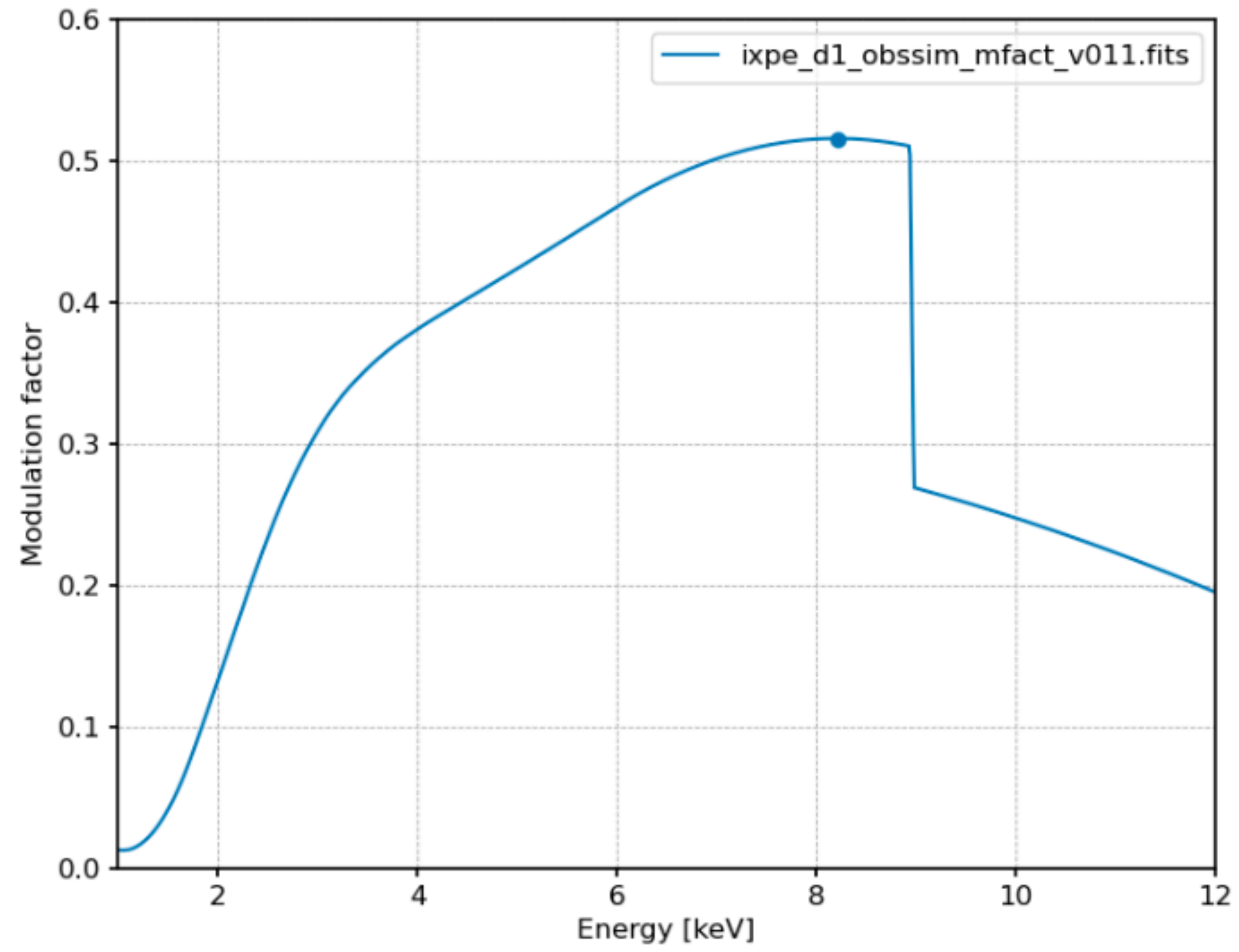
- ▷ FITS files in OGIP format
- ▷ Each of the three IXPE DUs has its own set of IRFs
- ▷ Generated and stored in a local CALDB, then submitted to HEASARC



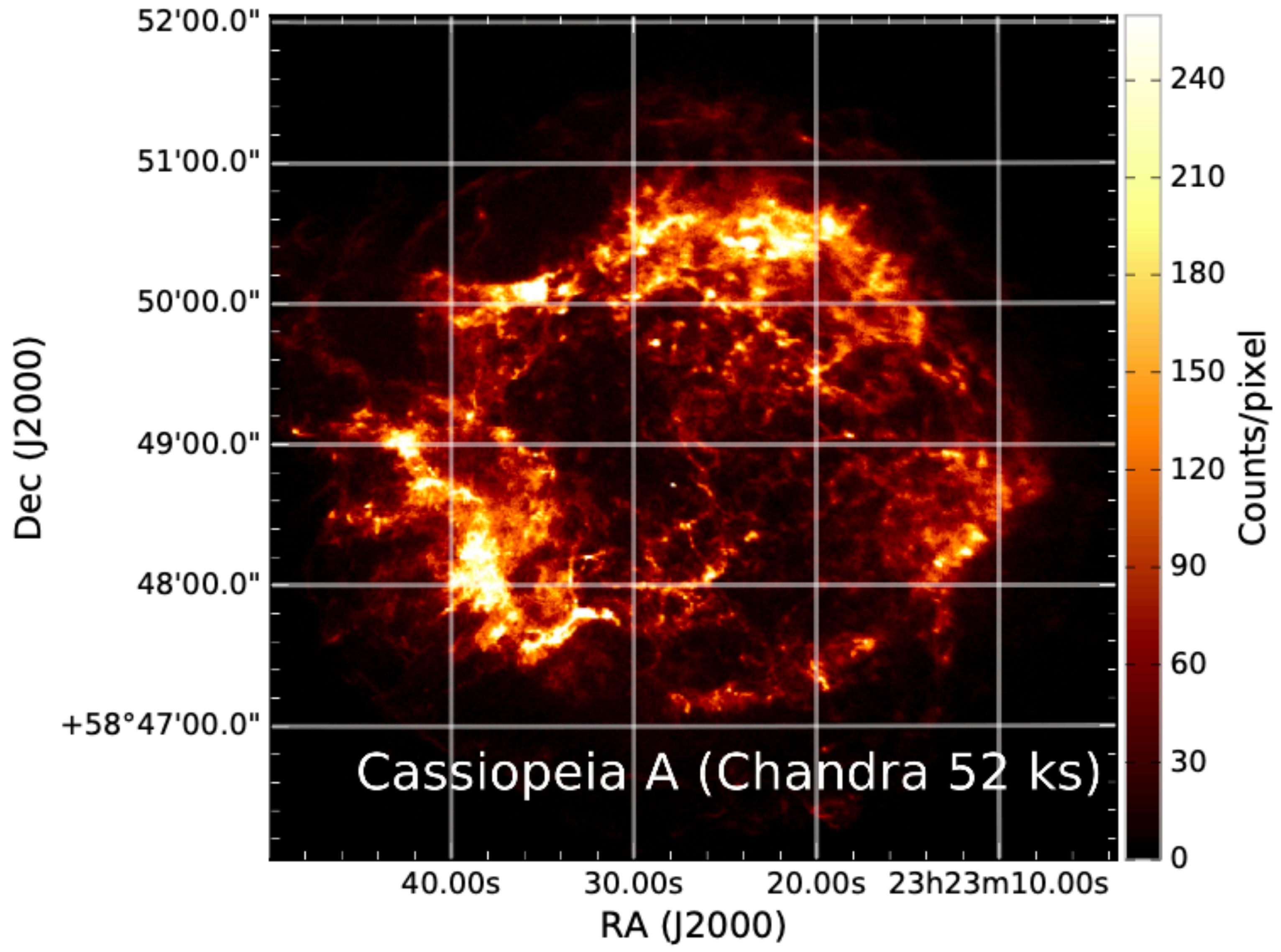


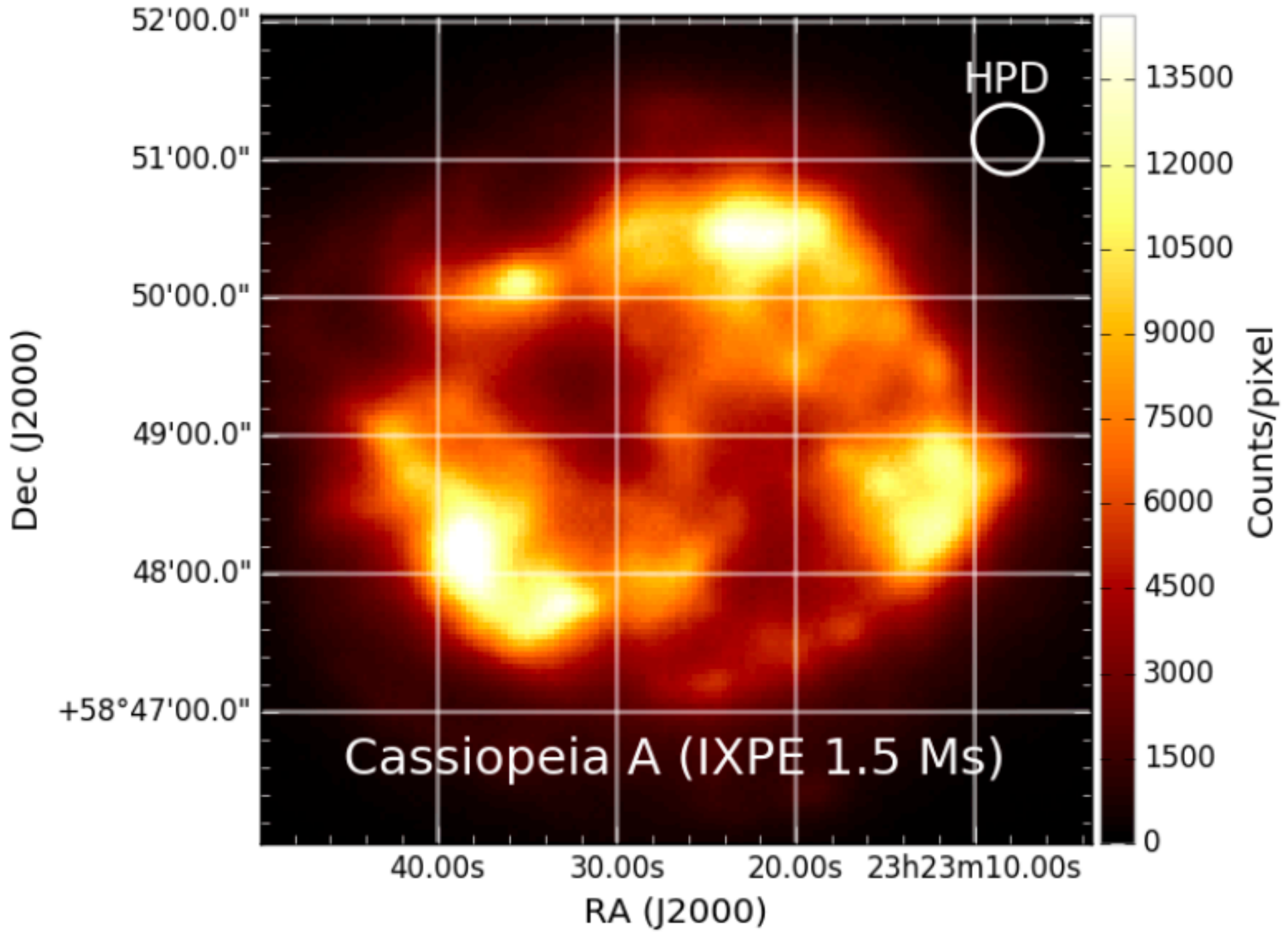


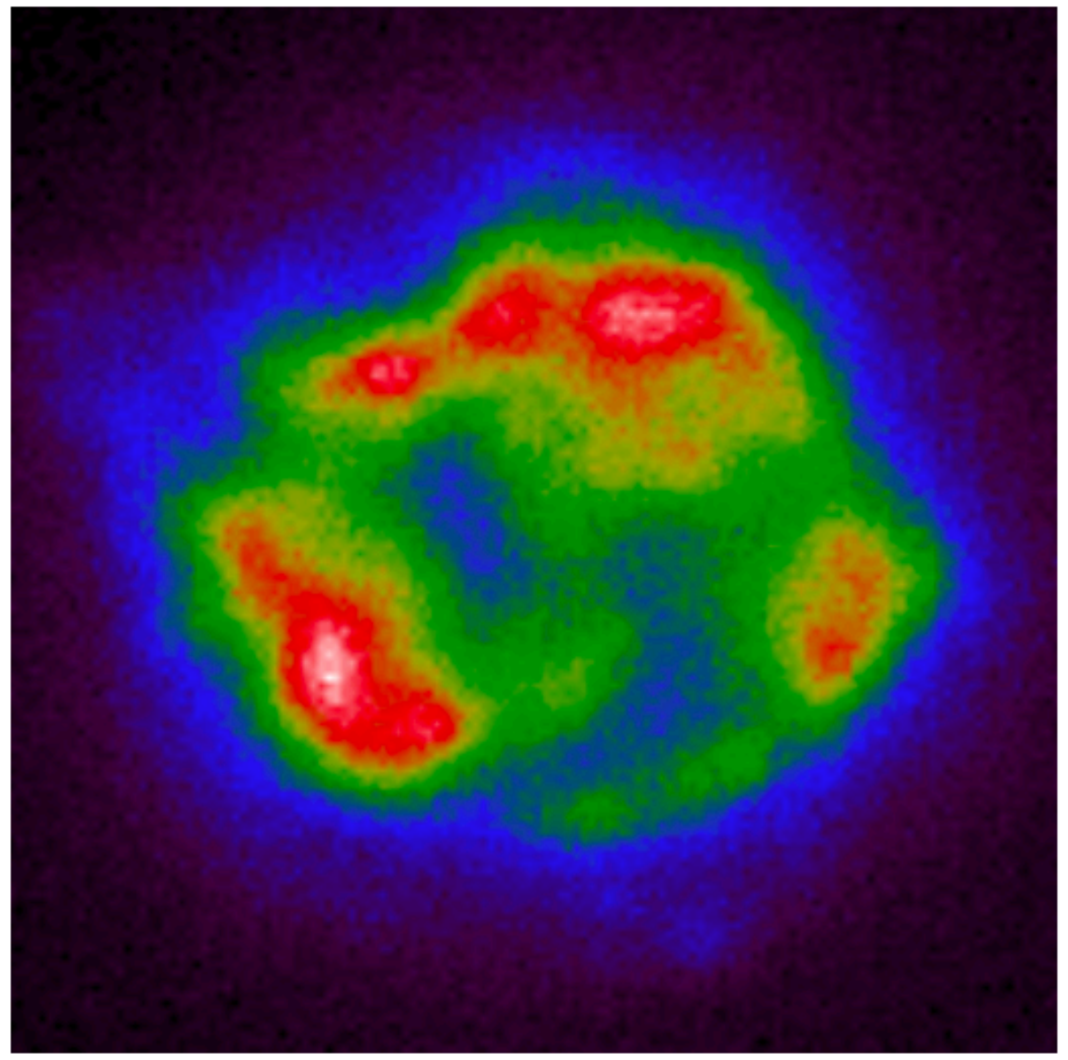




Simulation – extended source, from Chandra image







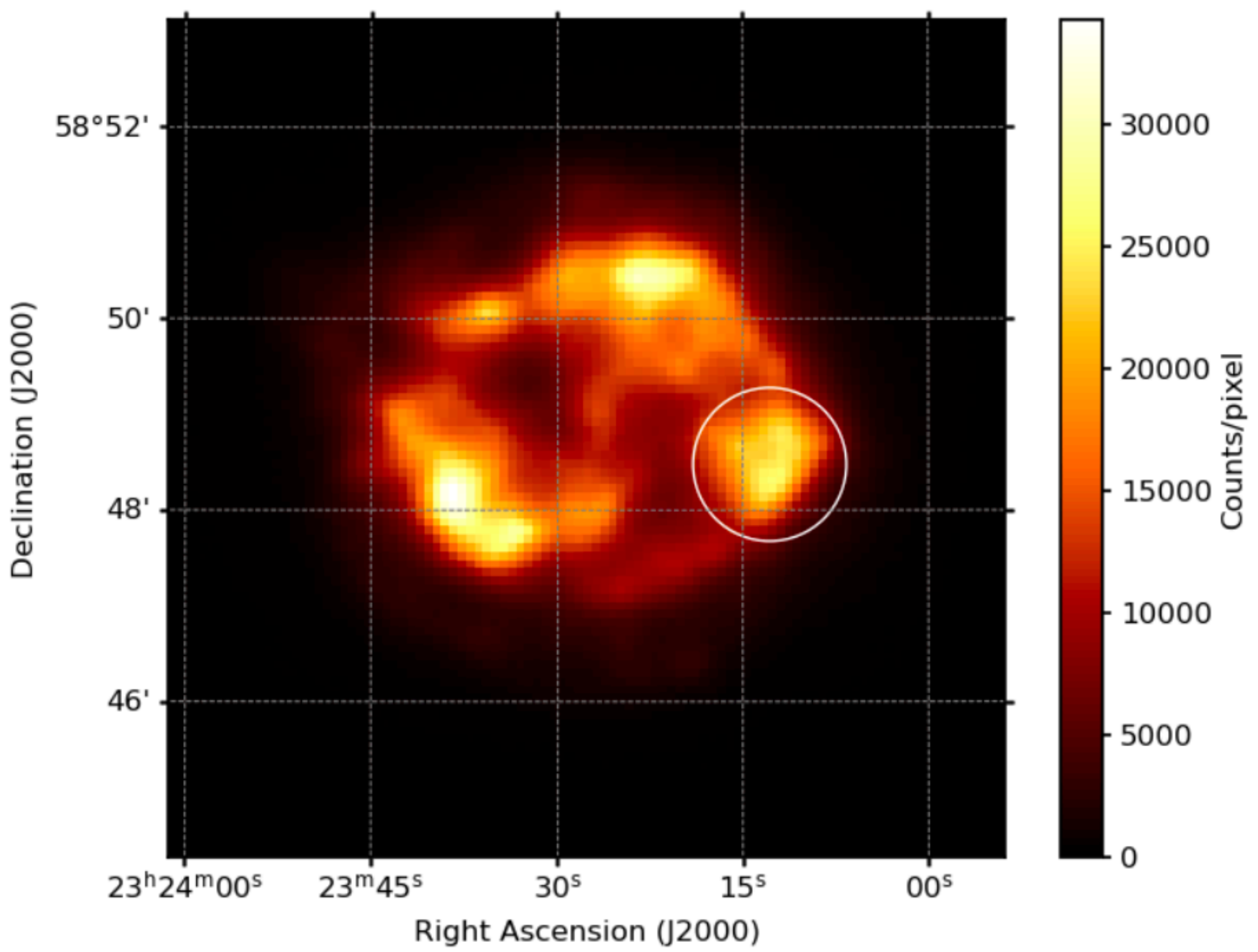
The simulation pipeline will generate 3 fits files, one for each DU.

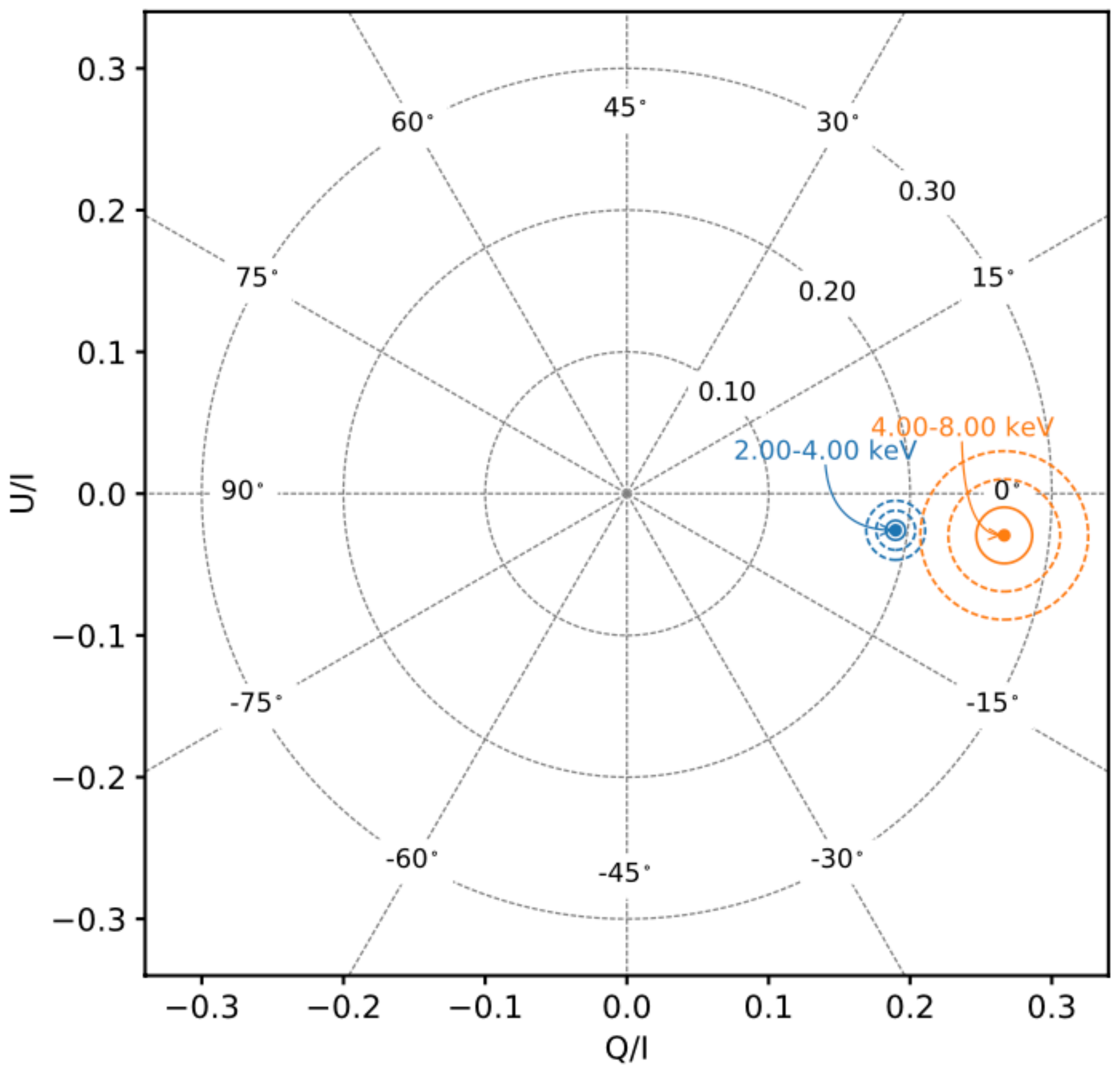
Each file contains the simulated photon list and is analogous to the real-data files.

Analysis – xpbin PCUBE

Polarization cube

- ▷ The simplest possible data structure holding polarization information
- ▷ Table listing I, Q, U, polarization degree and angle with the associated uncertainties in multiple energy bins:
    - ▷ Provided with methods to rescale and subtract the background contribution

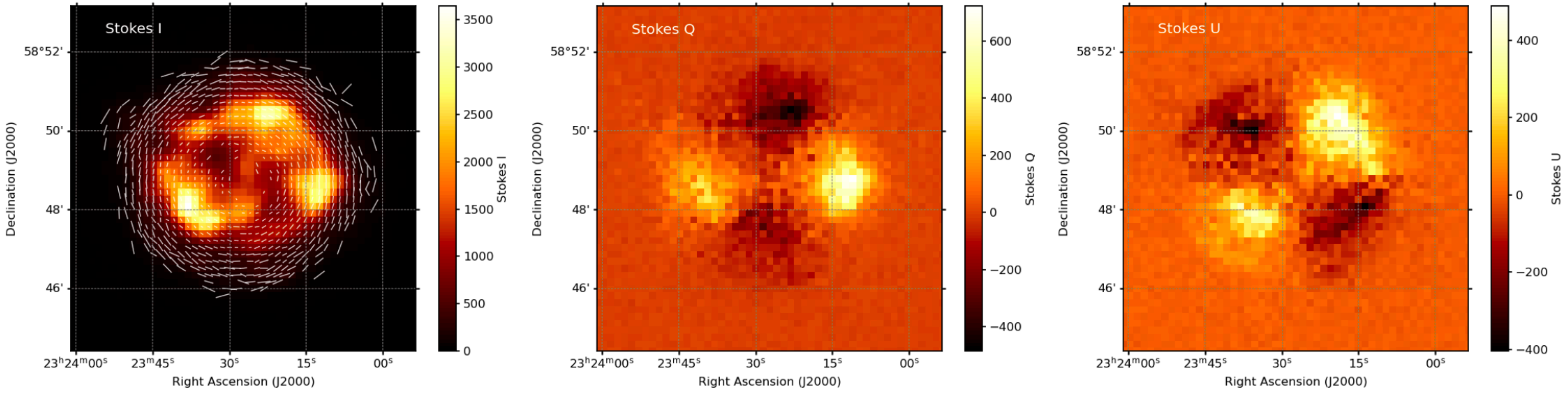




Quantity	2.00--8.00 keV
E_MEAN	5.0117599361664364
COUNTS	5762.0
MU	0.3921964538001015
W2	170.2692108154297
N_EFF	1664.503662109375
FRAC_W	0.28887602605160967
MDP_99	0.2681089106047689
I	532.3661499023438
I_ERR	13.048724174499512
Q	-67.53325653076172
Q_ERR	47.02301654580805
U	-61.832847595214844
U_ERR	47.02772717795586
QN	-0.12685489654541016
QN_ERR	0.08832833671794095
UN	-0.11614722013473511
UN_ERR	0.08833718519966481
QUN_COV	-8.85179443077969e-06
PD	0.17199519276618958
PD_ERR	0.0883654765108721
PA	-68.76152928032351
PA_ERR	14.73511598093247
P_VALUE	0.15021762132159794
CONFID	0.849782378678402
SIGNIF	1.035500479458798

Analysis – xpbin PMAP
Polarization map

- ▷ Hold the exact same information as polarization cubes, but binned in sky-coordinates
- ▷ Maps of I, Q, U, polarization degree and angle in multiple energy layers
  - ▷ Provided with methods to convolve the map with a generic binned kernel and overlay the arrows of the polarization information



with the command `xpbinview` it is possible to quickly visualize all the outputs from xpbin

Analysis – xpbin PHA1, PHA1Q, PHA1U

Polarization map

- Main interface to spectro-polarimetric fitting in `XSPEC`, `Sherpa` and `3ML`
- Generalization of the standard PHA spectra:
  - PHA1, PHA1Q and PHA1U
- Lightweight python wrapper dubbed *xpxspec* shipped with *ixpeobssim*:
  - Together with a few simple, multiplicative polarimetric models provided by HEASARC through the page hosting XSPEC additional models