The Astrophysical Journal Supplement Series, 274:43 (9pp), 2024 October https://doi.org/10.3847/1538-4365/ad7bab

Sherpa: An Open-source Python Fitting Package

Aneta Siemiginowska[1], Douglas Burke[1], Hans Moritz Günther[2], Nicholas P. Lee[1], Warren McLaughlin[1], David A. Principe[2], Harlan Cheer[1], Antonella Fruscione[1], Omar Laurino[1], Jonathan McDowell[1], and Marie Terrell[1]
[1] Center for Astrophysics | Harvard & Smithsonian, 60 Garden St., Cambridge, MA 02138, USA; asiemiginowska@cfa.harvard.edu
[2] MIT Kavli Institute for Astrophysics and Space Research, Massachusetts Institute of Technology, Cambridge, MA, USA
Received 2024 August 23; revised 2024 September 11; accepted 2024 September 15; published 2024 October 11

Abstract

We present an overview of Sherpa, an open-source Python project, and discuss its development history, broad design concepts, and capabilities. Sherpa contains powerful tools for combining parametric models into complex expressions that can be fit to data using a variety of statistics and optimization methods. It is easily extensible to include user-defined models, statistics, and optimization methods. It provides a high-level user interface for interactive data analysis, such as within a Jupyter notebook, and it can also be used as a library component, providing fitting and modeling capabilities to an application. We include a few examples of Sherpa applications to multiwavelength astronomical data.

Unified Astronomy Thesaurus concepts: Astronomy software (1855)

1. Introduction

Data-processing software developed by astronomical observatories typically generates standard data products which constitute the basis for scientific analysis. Example data products include spectra and images, data cubes, or for the case of high-energy astrophysics, lists of detected photons called event lists. Generally, these software packages focus on raw data reduction leaving scientific inference—which requires appropriate statistical methodology and theoretical understanding of the astrophysics—to other specialized packages. While the underlying astrophysics theory must be specific to the nature of the observed source, the statistical methodology is broader. This methodology involves applying the theory to the observations and making scientific inference based of the quality and characteristics of the data.

Sherpa is a general modeling and fitting application with a library of models, statistics, and optimizers. (P. Freeman et al. 2001; B. L. Refsdal et al. 2009). It was developed by the Chandra X-ray Center (CXC) with a focus on fitting X-ray spectra and images. It is distributed as part of the Chandra Interactive Analysis of Observations (CIAO) software package (A. Fruscione et al. 2006) and as a stand-alone Python package under the open-source GNU General Public License (D. Burke et al. 2024). While Sherpa was originally developed with the specific requirements of X-ray data modeling in the Poisson regime, which use calibration information in forward-fitting methods, it is designed to handle more generic data such as optical spectra or spectral energy distributions (e.g., C. Nigro et al. 2022; D. Ilić et al. 2023). In the case of 2D-image analysis, Sherpa can incorporate ancillary data from background maps, exposure maps, and point-spread function (PSF) images as part of 2D model expressions. Accounting for these factors is necessary for robust scientific inference. Sherpa has been used for analysis of multiwavelength images, spectra, and time series from many telescopes, including ground-based optical telescopes (e.g., K. M. Leighly et al. 2022; L. Barquín-González et al. 2024; H. Fan et al. 2024), the Hubble Space Telescope (HST; K. S. Green et al. 2023), the Spitzer Infrared Spectrograph infrared telescope (A. Ruiz et al. 2013), Chandra, XMM-Newton, and NuStar in X-rays, the Fermi Space Telescope for high-energy γ-rays (F. Aharonian et al. 2024), and H.E.S.S. for TeV data (H. E. S. S. Collaboration et al. 2012). It can also be used easily with nonastronomical data (e.g., T. Aldcroft 2010). Sherpa is flexible, modular, and extensible. It has an IPython user interface and it is also an importable Python module. Sherpa models, optimization, and statistic functions are available via both C++ and Python for software developers using such functions directly in their own code (e.g., the recent `fantasy` tool for fitting optical spectra by D. Ilić et al. 2023).

Many of Sherpa's users are in the X-ray community, but its Python foundations and compatibility with multiwavelength data makes it an attractive and viable option for astronomers working at other wavelengths. Sherpa handles world coordinate system (WCS) information and calibration products, has the fit statistics appropriate for both Poisson and Gaussian data, and contains extensively tested robust optimization methods.

In the following sections we present a brief history of the Sherpa development (Section 2), a quick study of its use by citation numbers (Section 3), an overview of the design (Section 4), its current capabilities with examples (Sections 5 and 6), Sherpa's place in the Python ecosystem (Section 7), and the open-source management of the project (Section 8). This paper provides an overview of the project and it is not intended to be software documentation. The open-source code and details of the releases are available on GitHub.[3] The user documentation, reference/application programming interface with class diagrams are located on the ReadTheDocs[4] site.

2. A Brief History of Sherpa

Sherpa has been developed over the past two decades by the CXC (S. Doe et al. 1997, 1998; P. Freeman et al. 2001) as a complement to the CIAO software (A. Fruscione et al. 2006).

[3] https://github.com/sherpa/sherpa
[4] https://sherpa.readthedocs.io/en/latest/

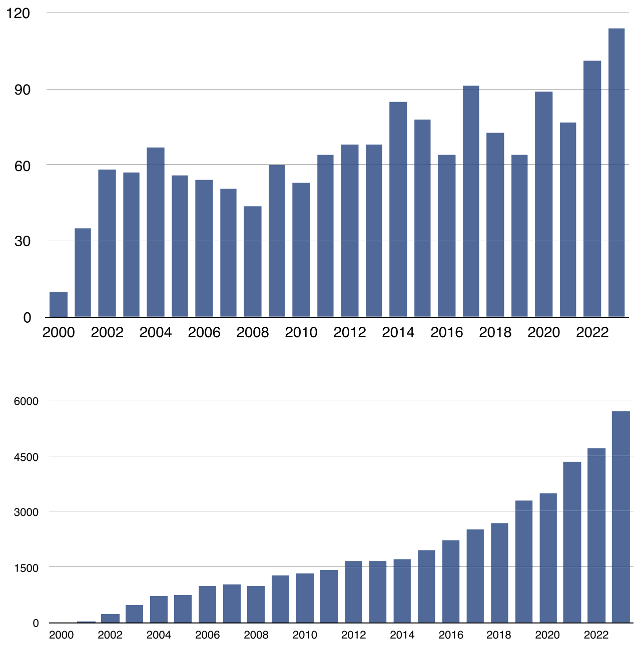

Figure 1. Publications that used Sherpa (top panel) and the citations of these papers (bottom panel) over time until the end of 2023 (from the ADS library). The *x*-axis shows years and the vertical axis shows the number of publications.

The main goal was to provide Chandra users with a general fitting application with flexibility in the selection of statistics and optimization methods and capabilities for building complex model expressions. Although the initial focus was on fitting X-ray data, the intent was to develop a general application and promote a multiwavelength analysis of diverse data sets.

Sherpa was originally written in a combination of C, C++, and Fortran. To adapt the user interface to the evolving needs of the astronomical community, a major rewrite of the Sherpa software was undertaken (S. Doe et al. 2006). This rewrite, initially released as a beta version in 2007, reimplemented Sherpa in Python with some modules in C++ for improved speed. (S. Doe et al. 2007; B. L. Refsdal et al. 2009; O. Laurino et al. 2015, 2019). Since 2014, Sherpa follows an open-development model with the source code and all development occurring on GitHub. It has had a total of 23 public open-source software releases to date (D. Burke et al. 2024).[3] Its open-source code is licensed under the GNU General Public license version 3. Sherpa can be downloaded as part of the CIAO software distribution or as a Python package from GitHub or Pypi.

3. Publications

A full-text search in the Astrophysics Data System (ADS[5]) returns more than 1600 references to Sherpa being used in scientific publications. Figure 1 (top panel) presents yearly science publications (until 2023 December) that used Sherpa and indicates that the program's usage has been steady with an increasing number of publications in the past 2–3 yr. The bottom panel in Figure 1 shows the increasing citations of these papers with a total of 36,417 cited papers.

Areas of research covered by the scientific publications are diverse. Some recent examples of Sherpa applications include:

(1) analysis of emission lines in optical spectra of an active galactic nucleus (AGN) sample (L. Barquín-González et al. 2024);

(2) modeling the high-resolution optical spectral lines from molecular clouds and synthetic spectra, in the EDIBLES Very Large Telescope survey (H. Fan et al. 2024);

(3) fitting complex models to X-ray spectra of AGN, a supernova remnant, or γ-ray source classifications (L. Godinaud et al. 2024; E. Pouliasis et al. 2024; H. Yang et al. 2024) together with the Bayesian X-ray Analysis package (J. Buchner et al. 2014);

(4) modeling clusters of galaxies using optical, radio, and X-ray images of gravitationally lensed clusters of galaxies (B. Beauchesne et al. 2024; H. Ebeling et al. 2024);

(5) modeling γ-rays and TeV spectra with the `gammapy` software using Sherpa's `simplex` optimization method (F. Aharonian et al. 2024).

Chandra observers use Sherpa to fit low- and high-spectral-resolution X-ray spectra, images, and radial profiles. For advanced analysis tasks Sherpa can be and has been incorporated into processing scripts and pipelines. At the CXC Sherpa was used in data processing to fit >1.3 million individual source detections and performed approximately one million additional image-fitting operations to compile the Chandra Source Catalog[6] (I. N. Evans et al. 2010, 2024). Sherpa was embedded into Iris, a Virtual Observatory application for modeling spectral energy distributions (O. Laurino et al. 2014). Outside the CXC, B. Beauchesne et al. (2024) recently incorporated Sherpa into a pipeline for modeling the mass of a gravitational lens with X-rays, lens parameters, and galaxy kinematics. In this case, Sherpa was used to fit a photoionization model to obtain a temperature map of an X-ray cluster. In another example, T. Plsek et al. (2024) fitted a 2D X-ray image of a galaxy with a smooth 2D β-model and used the 2D-image residuals to detect cavities to produce a training data set for the machine learning application CaDeT.

4. Design and Documentation

Sherpa's design has two main components: a user interface (UI) session and a Python object layer (S. Doe et al. 2007). This provides flexibility and allows Sherpa to be used in Python scripts, Python and IPython shells, Jupyter notebooks, and in other applications, such as, e.g., `ds9 DAX` (K. J. Glotfelty et al. 2011).

The Sherpa UI was designed more than two decades ago to simplify user interactions on the command line, within a Sherpa interactive session (see Figure 2). It was based on the standard style used by IRAF (D. Tody 1993) and XSPEC (K. A. Arnaud 1996) at that time. As a result, there are many command-line functions that can be used directly within the session or included in command-line scripts. This classical user interface allows CIAO users to open a standard Sherpa session in the IPython environment and have user-friendly access to the defined functions for accessing data, selecting models, statistics, and optimization methods. This interface also provides many standard visualization options making it easy to generate and display standard plots and images. The session

[5] https://ui.adsabs.harvard.edu/

[6] https://cxc.cfa.harvard.edu/csc/

```
Python
------
>>> from sherpa.models import Polynom1D
>>> from sherpa.fit import Fit
>>> from sherpa.stats import LeastSq
>>> from sherpa.optmethods import LevMar

>>> mdl = Polynom1D()
>>> mdl.c2.thaw()
>>> fit = Fit(d1, mdl, stat=LeastDq(), method=LevMar())
>>> res1 = fit.fit()
>>> print(res1.format())


Sherpa UI
----------
>>> from sherpa.ui import *

>>> load_data(1,'test.dat')
>>> set_source('polynom1d.mdl')
>>> thaw(mdl.c2)
>>> set_stat('leastsq')
>>> fit()
>>> print(get_fit_results())
```

Figure 2. An example of Sherpa Python commands for fitting a polynomial to a 1D data array, `d1`, accessing Python objects (top) or using the UI (bottom).

default settings can be customized to access additional options for input/output (I/O), plotting statistics etc.

The Python object layer is designed for power users who want to access Python modules directly, write complex Python scripts which include Sherpa, or use parts of Sherpa in their own Python package.

This bimodal design also impacted the design of the user documentation. Originally, the Sherpa documentation was developed within the CIAO environment. In this system, XML files contain descriptions of UI functions and form a base for online help (`ahelp`) in the CIAO–Sherpa session. The CIAO–Sherpa website documentation is based on the same XML files and provides an extended description of concepts and examples in form of analysis threads. The CIAO–Sherpa website has been available to users since the early days of the Chandra mission. Although it focuses on specific X-ray analysis issues, it also contains information and directions for modeling and fitting astronomical data in general.

Documentation of the Sherpa Python package is built directly from the content included in the Sherpa code and documentation files in the `docs` directory on GitHub. The standard Python `sphinx` system is used to generate Sherpa `ReadTheDocs` pages. The content was initially derived from the CIAO–Sherpa documentation and it evolved to expand content at the code level and improve code accessibility. More recently, several Jupyter Notebooks have been added to provide interactive examples.

5. Capabilities

Figure 3 displays the main Sherpa components with options for data, models, optimization methods, Bayesian samplers, and returned parameter distributions, which can be visualized. The details are available on the Sherpa ReadTheDocs documentation website.

The following sections describes the main capabilities of Sherpa.

5.1. Data

Sherpa can create, store, filter, and display data. It distinguishes between data that is binned (such that the value

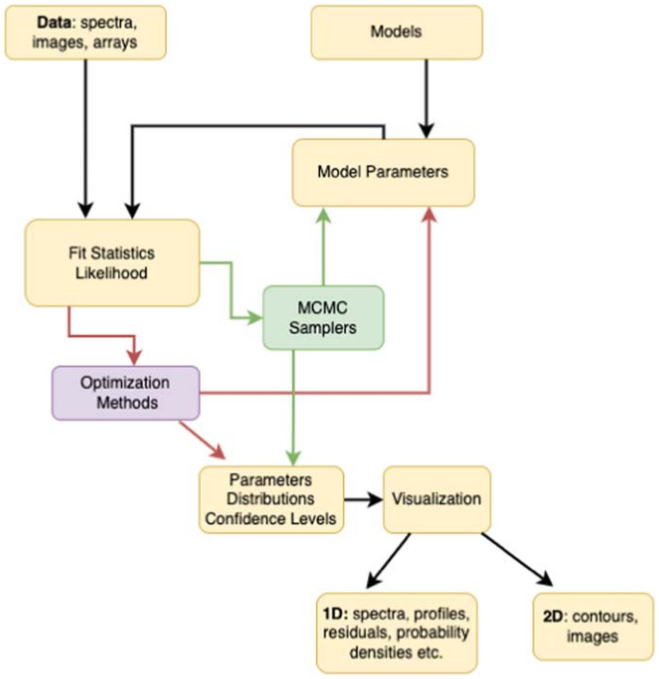
Data: spectra, images, arrays
Models
Model Parameters
Fit Statistics Likelihood
MCMC Samplers
Optimization Methods
Parameters Distributions Confidence Levels
Visualization
1D: spectra, profiles, residuals, probability densities etc.
2D: contours, images

Figure 3. Sherpa forward fitting. 1D or 2D data and complex models are defined. The best-fit parameters are found by a classic optimization method (violet box and red connectors) or model parameters are sampled from the posterior distribution with a Bayesian MCMC sampler (green box and connectors). The parameters with the confidence levels and the distributions can be visualized with the plotting and imaging options.

y represents an observation or model in an interval of *x* values ranging from xlo to xhi) and data where *y* is the value at point *x*. There are three main data classes defined in the `sherpa.data` module: Data1D, Data1DInt, and Data2D to handle, respectively, (*x*, *y*), (xlo, xhi, *y*), and (*x*1, *x*2, *y*) data. Dynamic filtering and grouping allows users to select a subset of the data range, or group the data for model evaluation on different grids.

The `sherpa.astro.io` module contains routines to read and write Flexible Image Transport System (FITS[7]) and ASCII format data. Sherpa supports a choice of two I/O backend packages: `Astropy` (Astropy Collaboration et al. 2018, 2022) and `crates` (in CIAO). Standard astronomy data files, such as spectra, images, and calibration files (e.g., ancillary response file, ARF or redistribution matrix file, RMF, in the X-ray regime; see OGIP/92-007[8] and CAL/GEN/92-002[9] for file-format definitions) are read or written using the selected backend package.

5.1.1. Astronomical Data

The `sherpa.astro.data` module is tailored for astronomy data sets and supports two types of data: 2D images (DataIMG) and X-ray spectra (DataPHA), along with associated calibration information (DataARF and DataRMF). Both types of data extend the module's capabilities to meet the specific requirements of astronomical data and support the following:

[7] https://heasarc.gsfc.nasa.gov/docs/heasarc/fits.html
[8] https://heasarc.gsfc.nasa.gov/docs/heasarc/ofwg/docs/spectra/ogip_92_007/ogip_92_007.html
[9] https://heasarc.gsfc.nasa.gov/docs/heasarc/caldb/docs/memos/cal_gen_92_002/cal_gen_92_002.html

1. image filtering using geometric shapes (regions);
2. different coordinate units for filtering images (logical, physical, and WCS), depending on the available metadata;
3. different analysis units (channels, energy, and wavelengths) for filtering and displaying X-ray spectral files in the pulse-height analyzer (PHA) format;
4. dynamical rebinning of PHA data to improve the signal to noise (grouping and quality).

5.2. Building Complex Models

The `sherpa.models.model` module allows models to be defined and combined using allowed expressions, such as addition, subtraction, and multiplication. The model parameters can be linked across different models and data sets.

The `sherpa.models` and `sherpa.astro.models` modules contain a library of 1D and 2D models. Additionally, the specialized models are provided in `sherpa.astro.optical`, `sherpa.astro.xspec`, `sherpa.instrument`, and `sherpa.astro.instrument`.

The instrument modules contain definitions of specific calibration models, which are typically input from files such as ARF and RMF for X-ray spectral fitting. By default, these calibration models are convolved with the physical models during the forward fitting of the data.

All models specific to X-ray spectra and contained in the `sherpa.astro.xspec` module are provided by the XSPEC model library (K. A. Arnaud 1996), which can be linked to Sherpa at build time.

5.3. Simulations

Sherpa provides several methods to generate simulated data sets using the `numpy.random` library. The `sherpa.ui.fake` function generates the synthetic 1D or 2D data for a model, assuming a default Poisson distribution. The `sherpa.astro.ui.fake_pha` function generates X-ray spectra for a given spectral model, exposure time, and the instrument response files.

The `sherpa.sim` module provides the Bayesian MCMC algorithm for low-count Poisson data. It also includes the `sherpa.sim.simulate` module with classes to support simulations for the posterior predictive p-values and likelihood ratio tests for spectral models (see R. Protassov et al. 2002).

5.4. Fitting

The `sherpa.fit` module represents the core of Sherpa's fitting functionality. The `Fit` class combines data and model expression to be fit and uses an optimizer to minimize the selected statistics function.

Sherpa uses the forward-fitting method to determine the best-fit parameters. The statistical functions for modeling Poisson and Gaussian data are available in the `sherpa.stats` module. Several robust optimizers specifically designed for fitting nonlinear models are provided in the `sherpa.optmethods` module, including Levenberg–Marquardt (K. Levenberg 1944; D. W. Marquardt 1963), Nelder–Mead simplex (J. A. Nelder & R. Mead 1965) and Monte Carlo/differential evolution (R. Storn & K. Price 1997). These optimizers work by optimizing the fit statistics function and returning the best-fit model.

Several methods are available in the `sherpa.estmethods` module for computing the uncertainties and confidence level for the best-fit parameters. The `confidence` method computes confidence bounds by varying a parameter along the grid and optimizing the model for the other parameters to find the best-fit statistics, though it can be relatively slow due to the large number of fitting operations required. The `covar` method, on the other hand, is based on the covariance matrix and does not account for correlations between the parameters, but is much faster.

5.5. Bayesian Analysis

Sherpa's implementation of Bayesian analysis with Poisson likelihood and priors is based on the model described in D. A. van Dyk et al. (2001) and implemented in the `pybloxcs` package (A. Siemiginowska et al. 2011). It uses the Metropolis or Metropolis–Hastings algorithm in the Markov Chain Monte Carlo (MCMC) to sample the posterior density. The main function which runs the MCMC is defined in `sherpa.astro.ui.get_draws`. Calibration uncertainties can be incorporated into the sampling by using the pragmatic or full Bayes approach, as described in H. Lee et al. (2011) and J. Xu et al. (2014) using calibration files, such as ARF and RMF in the X-ray case, which include uncertainties.

5.6. Visualization

Sherpa has a generic plotting abstraction and is currently released with three interfaces: (1) `matplotlib` (J. D. Hunter 2007) for publication-quality static plots, (2) `bokeh` for interactive plots, and (3) `ds9` for visualizing data using the interactive `ds9` application (W. A. Joye & E. Mandel 2003).

The `sherpa.plot` module defines the main plot classes, such as `plot`, `histogram`, `point`, `contour`, and `image`. It also contains many options for common display options for data, model fit and residuals, and other predefined options to support, for example, confidence contours or probability density histograms.

A separate `sherpa.astro.plot` module defines classes for plotting astronomical data and includes specific options for plotting PHA data or displaying FITS images.

6. Fitting Examples

In this section, we present some examples of Sherpa usage.

6.1. Fitting 1D Data

Sherpa can fit 1D data (simultaneously or individually), binned or unbinned, including spectra, surface-brightness profiles, light curves, and general ASCII arrays. The standard PHA type files for X-ray spectra and the associated calibration files are handled in a special manner, as the required information is automatically read from the files' headers. Multiple data sets can be fit simultaneously with model parameters varied independently or linked across data sets. The Sherpa documentation provides many examples of analysis threads and usage to guide users.

Figure 4 illustrates a fit to an optical spectrum (a simple ASCII table). A small section of the Sloan Digital Sky Survey spectrum of a starburst galaxy, SDSS J151806.13+424445.0[10] ($z = 0.0403$) is fitted assuming a constant model and multiple

[10] https://skyserver.sdss.org/dr18/VisualTools/quickobj

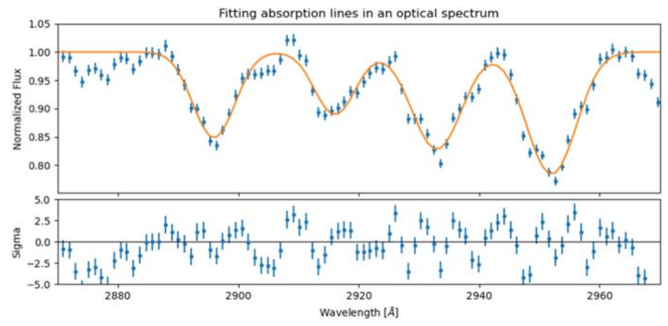

Figure 4. An example of fitting multiple absorption lines to the optical spectrum. The model consisted of a polynomial (`polynom1d`) and four independent absorption lines (`opticalgaussian` parameterized by the position, width, and optical depth). The top panel shows the data points (blue) overplotted with the best-fit model with the residuals shown in the bottom panel.

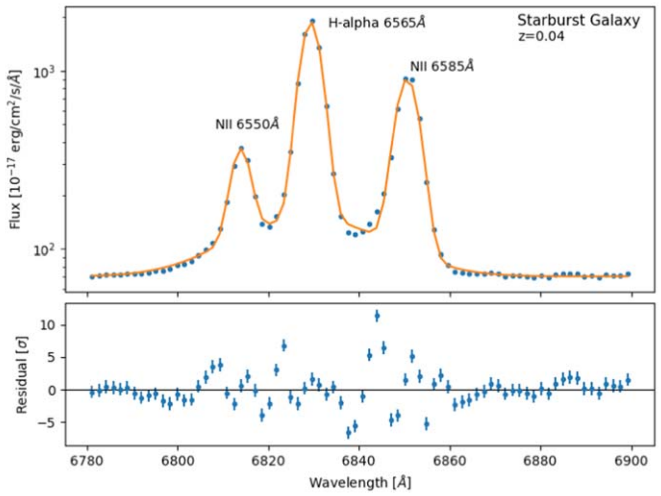

Figure 5. An optical spectrum (SDSS) fit with a model combined with a constant model (`const1d`) and a set of four Gaussian emission lines (`gauss1d`). The top panel shows the data (blue points) and the model fit (orange). The lower panel shows the residuals.

Gaussian profiles for the emission lines in Figure 5. The resulting data and model spectrum displayed the redshifted emission lines at the observed wavelength position for N II 6568 Å, Hα 6565 Å, and N II 6583 Å.

An example of a linear model fit to data with asymmetric errors is shown in Figure 6. This is often the case when the data are provided as a result of the previous analysis. In this example high-redshift quasars were fit assuming an absorbed power-law model (F. Shaban et al. 2022). The best-fit photon index has asymmetric uncertainties, so the lower and upper bounds have different values. In order to evaluate a dependence of the photon index on the X-ray luminosity for this quasar's sample a constant model was fit to the data assuming measurement errors given by an average of upper and lower errors (shown as the orange line in Figure 6). However, a bootstrap procedure (`resample_data` in Sherpa), can be used to simulate the data assuming a skewed distribution for the measurement errors (e.g., a sum of two Gaussians with different width set by the lower and upper errors). The resulting median value of the photon index and the uncertainties obtained through bootstrap methods indicate a slight bias in fitting these data assuming an average error as illustrated in the right panel in Figure 6.

Figure 7 shows a typical use of Sherpa-fitting 1D data, in this case X-ray spectra observed with the the Neutron Star Interior Composition Explorer (NICER; K. Gendreau & Z. Arzoumanian 2017) from the flaring K giant HD 251108. NICER started observing the source after a flare peak and observed for about 20 days as the flare cooled down. In a first step all 127 spectra are fit independently with the same model (a sum of two collisionally excited, optically thin plasma models from APEC, A. R. Foster et al. 2012). At later times, when the flare is faint, the fit uncertainties are larger, but Sherpa fits indicate that we observe the cooling of the flare plasma over the entire range shown in Figure 7; see H. M. Günther & D. Pasham et al. (2024) for details of this data set.

6.2. Fitting 2D Data

Sherpa can handle 2D images and includes more than 10 spatial models. X-ray images are typically sparse and require Poisson likelihood, which is one of the available choices of the statistical method. Imaging calibration data, such as PSF,

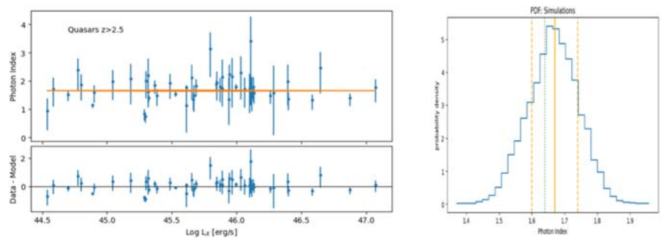

Figure 6. Fitting luminosity vs. photon index relation for a sample of quasars (F. Shaban et al. 2022). Photon index data have asymmetric errors. Left: The upper panel displays the photon index (blue points with asymmetric errors for each source at a given luminosity) and the best-fit linear model (orange line). The lower panel shows the residuals. Right: Probability distribution from the bootstrap. The orange lines mark the median and 1σ uncertainties. The green dotted line marks the best-fit model assuming an average measurement error given by upper and lower errors.

exposure maps, and background maps can be included in Sherpa's 2D forward fitting.

Figure 8 shows an example of 2D fitting of the X-ray cluster A2495 observed with Chandra (L. Rosignoli et al. 2024). The system is known for the triple offsets between the center of the intercluster medium, the central galaxy, and the warm gas. The 2D model for the surface brightness consisted of two `beta2d` model components with the same location of the center. The residual image in Figure 8 shows an additional peak offset from the center of the model indicating the presence of offset emission.

An example of a user-defined 2D model function fit to the trap map of a CCD detector is displayed in Figures 9 and 10. Such models are used to gather information about the characteristics of a detector and can be fit to the data over time, for example to monitor changes in a detector performance.

7. Sherpa and Python Ecosystem

Sherpa's place in the scientific Python ecosystem is shown in Figure 11. Sherpa has only a small number of required dependencies, most importantly `numpy` (C. R. Harris et al. 2020), the foundational numerical array-processing library for

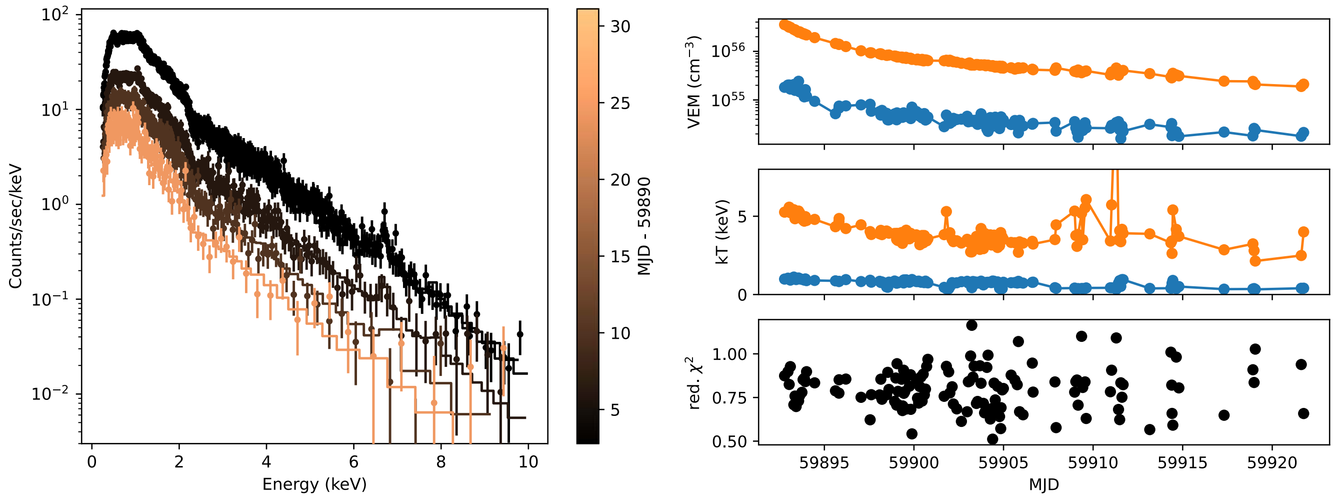

Figure 7. NICER observations of a large stellar flare from a K-type giant star with Sherpa fits. Left: Four selected examples from the set of 127 spectra. Each spectrum is fitted with the same type of model (two thermal emission components), but each fit is independent with four parameters (temperature and volume emission measure, VEM, for both components). The color bar on the right shows the time of observation and indicates the flare decay over time (black to orange). Right: Fitted VEM (top) and temperature kT (center) over time (with k being the Boltzmann constant) for the two emission components (orange and blue). The fit quality (reduced χ^2) for each individual fit is shown in the bottom panel. For clarity, error bars are not shown.

all of the scientific Python stack. Sherpa also makes use of a number of common Python packages for infrastructure that are used for the installation, for running tests (`pytest` and some of its plug-ins), and for building the documentation (`sphinx`). As described in Section 5.6 Sherpa interfaces with visualization packages `matplotlib`, `bokeh`, and `ds9` and also connects via a plug-in into `ds9`'s analysis menu (K. J. Glotfelty et al. 2011).

In turn, Sherpa provides functionality for other more specialized packages, which are used for specific astronomical analysis tasks. Those packages may call on Sherpa to, for example, read spectra in specific file formats, use the Sherpa model library and interface to XSPEC models, or perform numerical model fitting. The most well-known packages that depend on Sherpa are (1) `gammapy` (A. Donath et al. 2023), a Python package for γ-ray astronomy used as a core-library for the Cherenkov Telescope Array Observatory but that also supports other high-energy facilities; (2) BXA (Bayesian X-ray analysis, J. Buchner et al. 2014), a package which connects Sherpa (or XSPEC) to a Bayesian nested-sampling algorithm; (3) `naima`, a package to derive nonthermal particle distributions from relativistic electron populations; and (4) `xija`, a package used to model complex time-series data using a network of coupled nodes (one particular application of this package is the thermal modeling of the Chandra spacecraft).

8. Project Management

8.1. Governance and Development Model

The Sherpa source code is developed on GitHub to facilitate an open-development model. Issues and bugs are tracked and discussed publicly, opening the opportunity for contributions. Individual code contributions are submitted as pull requests on GitHub, where their utility is reviewed. The Sherpa project requires a minimum of one approval by a core developer before a pull request is merged into the code base. Sherpa is released twice a year.

8.2. Collaboration with Related Projects

Sherpa interfaces with the XSPEC model library (K. A. Arnaud 1996) which provides a set of crucial astrophysical models. Sherpa is one of the largest users of the library outside of XSPEC. The Sherpa team often identifies issues or inconsistencies and provides feedback to XSPEC team.

The `astropy.modeling` subpackage, designed like Sherpa, distinguishes data, models, statistics, and numerical optimizers. However, Sherpa has a far larger library of model functions, more advanced optimizers, and more specialized statistics functions. In collaboration with core developers from `astropy`, the Sherpa team developed a package called `saba`[11] to make Sherpa models, statistics, and optimizers available as plug-ins to the `astropy` community.

8.3. Inclusive Practices and Sustainability

Beyond the adoption of standard tools like `pytest`, `sphinx`, `pip`, and `conda`, a large fraction of the code is covered by the test suite. This assures that a contributed code change does not introduce bugs and break Sherpa. These efforts make it easier to contribute to Sherpa development.

Code sustainability, as demonstrated by `astropy` (Astropy Collaboration et al. 2018), requires growing the community of contributors and increasing the size and impact of contributions from outside the core team. Currently, a large fraction of Sherpa questions, bug reports, and code contributions are initiated through the CXC help desk, with a smaller number of such contributions via GitHub. Sherpa follows GitHub's list of recommended community standards[12] to align the project with the best practices for the open development. Additionally, developing maintainable and well-documented code with good test coverage, and focusing on timely updates to Python and other dependencies, are necessary for long-term sustainability. Growing the community of Sherpa users who become

[11] https://saba.readthedocs.io/en/latest/
[12] https://github.com/sherpa/sherpa/community

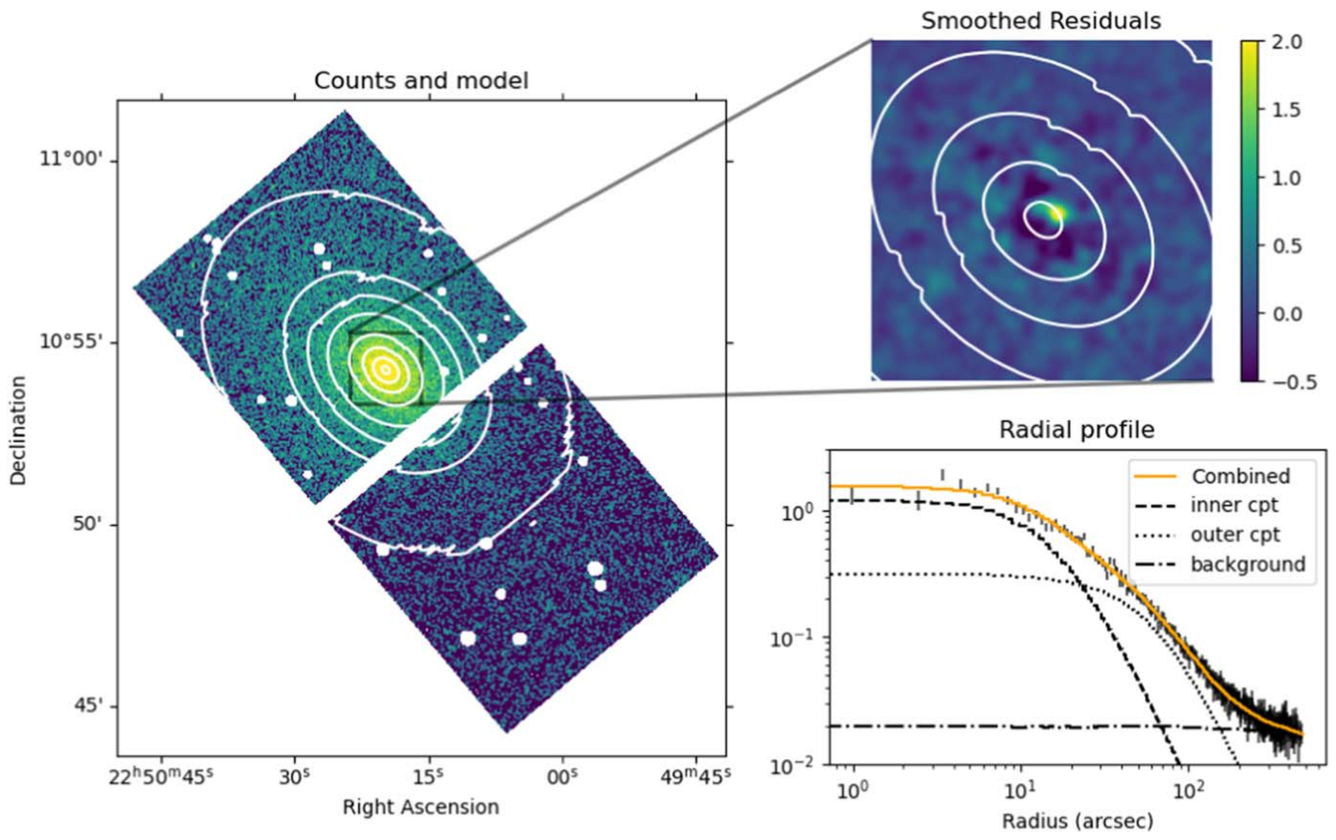

Figure 8. Left: Chandra image of an X-ray cluster A2495 overlaid with the contours of the 2D fit assuming two `beta2d` model components with the same location of the center. Top right: the smoothed residuals with visible offset from the central location of the model. This cluster is known to have offsets between the central galaxy and the hot gas. Bottom right: the radial profiles of the surface brightness based on the fitted model components and the sum of the two components overlaid over the data and the background level.

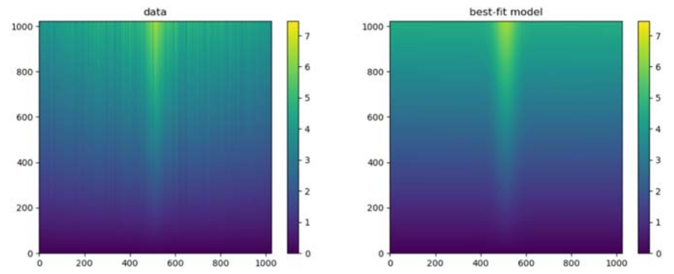

Figure 9. Modeling-trap map of an ACIS CCD with a 2D model defined as $T(x, y) = (a + bx)y + cy{*}\exp^{\frac{-(x-x0)^2}{2\sigma^2}}$. The left panel shows the data and the right panel shows the model image, with the color scale on the right in units of counts.

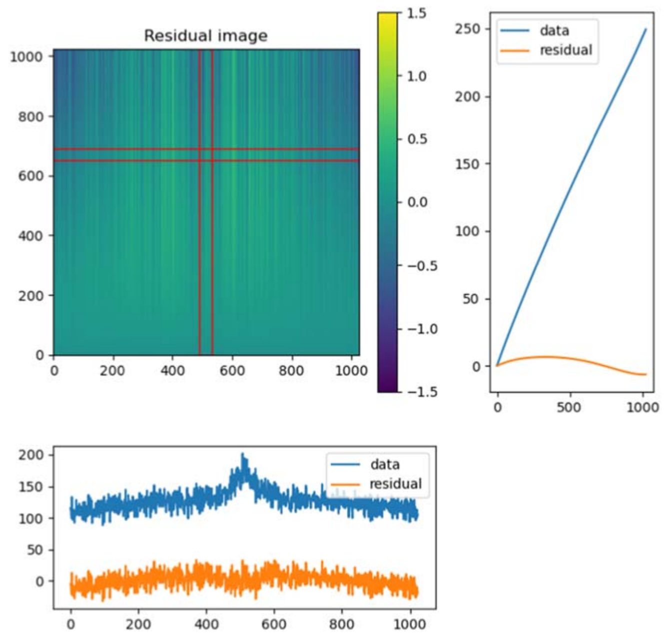

Figure 10. The color image of residuals from the best model shown in Figure 9. The color scale indicates σ values. The red boxes show the regions used for the 1D comparison plots shown in the right and bottom panels. The blue lines indicate the data and the orange lines the residuals.

contributors, by bringing new ideas and feature requests, will increase sustainability of the project.

9. Summary

Sherpa was designed and developed to support traditional forward-fitting methodology. Recent advances in techniques and open-source software availability bring new ideas for modeling large complex data sets, including simulation-based inference (e.g., D. Huppenkothen & M. Bachetti 2022; D. Barret & S. Dupourqué 2024) and machine learning methods (e.g., C. Rhea et al. 2021, 2023). These emerging methods may be able to handle multidimensional data, such as spatial–spectral–temporal data available in modern observations (see e.g., C. Xu et al. 2021). Sherpa future development plans include extensions to multidimensional data and data cubes from integral field units obtained by modern optical telescopes, such as, for example, James Webb Space Telescope

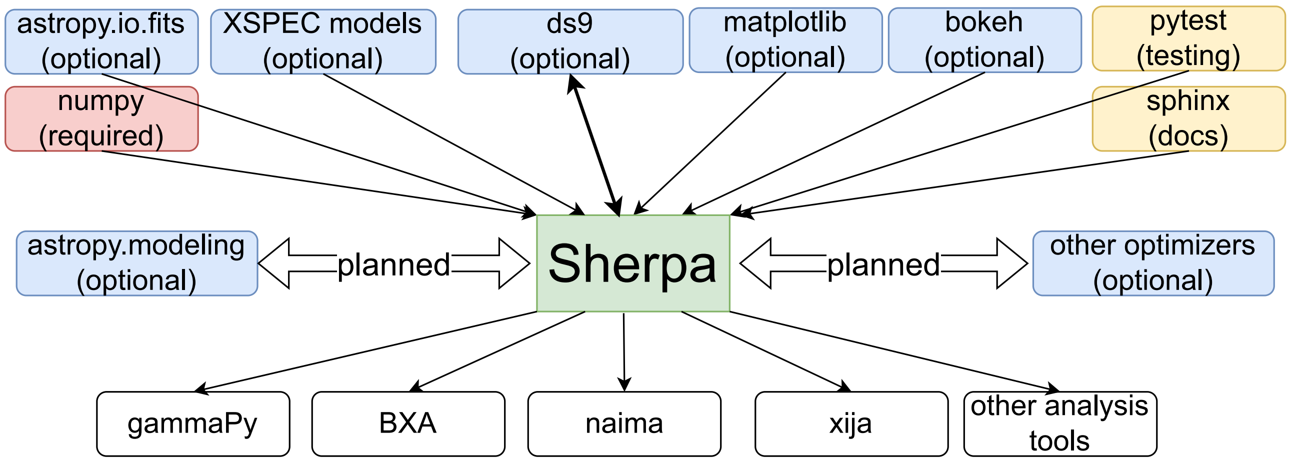

Figure 11. Sherpa's place in the scientific Python ecosystem. Sherpa requires a numeric library (red box), and interfaces with several optional libraries for I/O, external model libraries, and visualization (blue boxes). It enables the fitting and modeling in several domain-specific packages (white boxes). The two large white arrows mark future implementation of interoperability with `astropy.modeling` and external optimizers. Infrastructure packages needed for developers, but not users, are marked with yellow boxes.

(JWST) and the Giant Magellan Telescope. Sherpa flexible framework allows also for incorporating new emerging methods, e.g., simulation-based inference for working with complex models. We note that Sherpa's open-source development model provides a great way for the community contributions to bring new ideas into this modeling framework.

Sherpa is a Python package for modeling and fitting astronomical data. Although it has been developed for modeling X-ray data from the beginning, it was designed to model any type of data and it is well suited for multiwavelength analysis. Sherpa has a solid foundation. It has been widely used and tested in many applications over the past two decades. It provides robust methodology to model general astronomical data and its flexibility allows users to incorporate Sherpa in Python scripts or larger data analysis pipelines.

Acknowledgments

We acknowledge contributions to Sherpa by many people in addition to the present authors who have been involved in the project at different stages over the years. The initial design in the late 1990s involved Maureen Conroy, Stephen Doe, Malin Ljungberg, and William Joye. Stephen Doe was with the project as the lead system developer until 2013. Peter Freeman focused on developing the scientific modeling and statistical inference methodology (2000–2003), Dan Nguyen (2005–2022) developed optimization methods including implementation of Nelder–Mead and differential evolution algorithms, Chris Stawarz (2005–2006) and Brian Refsdal (2006–2011) worked on the Python version of the code. We also acknowledge discussions and input by current and former members of the Science Data System team Nina Bonaventura, John Davis, Martin Elvis, Liz Galle, Kenny Glotfelty, John Houck, David P. Huenemoerder, and Michael Nowak. Tom Aldcroft, Jamie Budynkiewicz, Johannes Buchner, Christoph Deil, Axel Donath, Iva Laginja, Katrin Leinweber, and Brigitta Sipcz contributed to the open-source code via GitHub pull requests. We also thank Janet Evans, Pepi Fabbiano, Margarita Karovska, Vinay Kashyap, Fabrizio Nicastro, Rafael Martinez Galarza, and Andreas Zezas for discussions of Sherpa's applications to a variety of data problems.

This work was supported NASA contract to the CXC NAS8-03060; H.M.G. and D.A.P. specifically were supported by contract SV3-73016 to MIT for support of the CXC, which is operated by SAO for and on behalf of NASA.

This research has made use of NASA's Astrophysics Data System.

Facilities: CXO, NICER, Sloan

Software: Sherpa, matplotlib.

ORCID iDs

Aneta Siemiginowska https://orcid.org/0000-0002-0905-7375
Douglas Burke https://orcid.org/0000-0003-4428-7835
Hans Moritz Günther https://orcid.org/0000-0003-4243-2840
Nicholas P. Lee https://orcid.org/0009-0006-5274-6439
David A. Principe https://orcid.org/0000-0002-7939-377X
Antonella Fruscione https://orcid.org/0000-0002-6414-3954
Omar Laurino https://orcid.org/0000-0001-9697-4659
Jonathan McDowell https://orcid.org/0000-0002-7093-295X

References

Aharonian, F., Ait Benkhali, F., Aschersleben, J., et al. 2024, A&A, 686, A308
Aldcroft, T. 2010, in Proc. of the 9th Python in Science Conf., ed. S. van der Walt & J. Millman (SciPy), 1
Arnaud, K. A. 1996, in ASP Conf. Ser. 101, Astronomical Data Analysis Software and Systems V, ed. G. H. Jacoby & J. Barnes (San Francisco, CA: ASP), 17
Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, ApJ, 935, 167
Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123
Barquín-González, L., Mateos, S., Carrera, F. J., et al. 2024, A&A, 687, A159
Barret, D., & Dupourqué, S. 2024, A&A, 686, A133
Beauchesne, B., Clement, B., Hibon, P., et al. 2024, MNRAS, 527, 3246
Buchner, J., Georgakakis, A., Nandra, K., et al. 2014, A&A, 564, A125
Burke, D., Laurino, O., McLaughlin, W., et al. 2024, sherpa/sherpa: Sherpa 4.16.1, v4.16.1, Zenodo, doi:10.5281/zenodo.593753
Doe, S., Ljungberg, M., Siemiginowska, A., & Joye, W. 1998, in ASP Conf. Ser. 145, Astronomical Data Analysis Software and Systems VII, ed. R. Albrecht, R. N. Hook, & H. A. Bushouse (San Francisco, CA: ASP), 157
Doe, S., Nguyen, D., Stawarz, C., et al. 2006, in ASP Conf. Ser. 351, Astronomical Data Analysis Software and Systems XV, ed. C. Gabriel et al. (San Francisco, CA: ASP), 77

Doe, S., Nguyen, D., Stawarz, C., et al. 2007, in ASP Conf. Ser. 376, Astronomical Data Analysis Software and Systems XVI, ed. R. A. Shaw, F. Hill, & D. J. Bell (San Francisco, CA: ASP), 543
Doe, S., Siemiginowska, A., Joye, W., & McDowell, J. 1997, in ASP Conf. Ser. 125, Astronomical Data Analysis Software and Systems VI, ed. G. Hunt & H. Payne (San Francisco, CA: ASP), 492
Donath, A., Terrier, R., Remy, Q., et al. 2023, A&A, 678, A157
Ebeling, H., Richard, J., Beauchesne, B., et al. 2024, MNRAS, submitted (arXiv:2404.11659)
Evans, I. N., Evans, J. D., Martínez-Galarza, J. R., et al. 2024, ApJS, 274, 22
Evans, I. N., Primini, F. A., Glotfelty, K. J., et al. 2010, ApJS, 189, 37
Fan, H., Rocha, C. M. R., Cordiner, M., et al. 2024, A&A, 681, A6
Foster, A. R., Ji, L., Smith, R. K., & Brickhouse, N. S. 2012, ApJ, 756, 128
Freeman, P., Doe, S., & Siemiginowska, A. 2001, Proc. SPIE, 4477, 76
Fruscione, A., McDowell, J. C., Allen, G. E., et al. 2006, Proc. SPIE, 6270, 62701V
Gendreau, K., & Arzoumanian, Z. 2017, NatAs, 1, 895
Glotfelty, K. J., Miller, J., & Chen, J. 2011, in ASP Conf. Ser. 442, Astronomical Data Analysis Software and Systems XX, ed. I. N. Evans et al. (San Francisco, CA: ASP), 629
Godinaud, L., Acero, F., Decourchelle, A., & Ballet, J. 2024, A&A, submitted (arXiv:2404.17296)
Green, K. S., Gallagher, S. C., Leighly, K. M., et al. 2023, ApJ, 953, 186
Günther, H. M., Pasham, D., et al. 2024, ApJ, submitted (arXiv:2410.03616)
Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Natur, 585, 357
H. E. S. S. Collaboration, Abramowski, A., Acero, F., et al. 2012, A&A, 541, A5
Hunter, J. D. 2007, CSE, 9, 90
Huppenkothen, D., & Bachetti, M. 2022, MNRAS, 511, 5689
Ilić, D., Rakić, N., & Popović, L. Č. 2023, ApJS, 267, 19
Joye, W. A., & Mandel, E. 2003, in ASP Conf. Ser. 295, Astronomical Data Analysis Software and Systems XII, ed. H. E. Payne, R. I. Jedrzejewski, & R. N. Hook (San Francisco, CA: ASP), 489
Laurino, O., Budynkiewicz, J., D'Abrusco, R., et al. 2014, A&C, 7, 81
Laurino, O., Burke, D., Evans, J., et al. 2019, in ASP Conf. Ser. 521, Astronomical Data Analysis Software and Systems XXVI, ed. M. Molinaro, K. Shortridge, & F. Pasian (San Francisco, CA: ASP), 479
Laurino, O., Siemiginowska, A., Evans, J., et al. 2015, in ASP Conf. Ser. 495, Astronomical Data Analysis Software an Systems XXIV, ed. A. R. Taylor & E. Rosolowsky (San Francisco, CA: ASP), 313
Lee, H., Kashyap, V. L., van Dyk, D. A., et al. 2011, ApJ, 731, 126
Leighly, K. M., Choi, H., DeFrancesco, C., et al. 2022, ApJ, 935, 92
Levenberg, K. 1944, QApMa, 2, 164
Marquardt, D. W. 1963, SIAM, 11, 431
Nelder, J. A., & Mead, R. 1965, CompJ, 7, 308
Nigro, C., Sitarek, J., Gliwny, P., et al. 2022, A&A, 660, A18
Plsek, T., Werner, N., Topinka, M., & Simionescu, A. 2024, MNRAS, 527, 3315
Pouliasis, E., Ruiz, A., Georgantopoulos, I., et al. 2024, A&A, 685, A97
Protassov, R., van Dyk, D. A., Connors, A., Kashyap, V. L., & Siemiginowska, A. 2002, ApJ, 571, 545
Refsdal, B. L., Doe, S. M., Nguyen, D. T., et al. 2009, in Proc. of the 8th Python in Science Conf., ed. G. Varoquaux, S. Walt, & J. Millman (SciPy), 51
Rhea, C., Hlavacek-Larrondo, J., Kraft, R., Bogdan, A., & Geelen, R. 2021, RNAAS, 5, 113
Rhea, C., Hlavacek-Larrondo, J., Kraft, R., et al. 2023, arXiv:2311.18014
Rosignoli, L., Ubertosi, F., Gitti, M., et al. 2024, ApJ, 963, 8
Ruiz, A., Risaliti, G., Nardini, E., Panessa, F., & Carrera, F. J. 2013, A&A, 549, A125
Shaban, F., Siemiginowska, A., Suleiman, R. M., El-Nawawy, M. S., & Ali, A. 2022, JHEAp, 36, 152
Siemiginowska, A., Kashyap, V., Refsdal, B., et al. 2011, in ASP Conf. Ser. 442, Astronomical Data Analysis Software and Systems XX, ed. I. N. Evans et al. (San Francisco, CA: ASP), 439
Storn, R., & Price, K. 1997, J. Glob. Optim., 11, 341
Tody, D. 1993, in ASP Conf. Ser. 52, Astronomical Data Analysis Software and Systems II, ed. R. J. Hanisch, R. J. V. Brissenden, & J. Barnes (San Francisco, CA: ASP), 173
van Dyk, D. A., Connors, A., Kashyap, V. L., & Siemiginowska, A. 2001, ApJ, 548, 224
Xu, C., Günther, H. M., Kashyap, V. L., Lee, T. C. M., & Zezas, A. 2021, AJ, 161, 184
Xu, J., van Dyk, D. A., Kashyap, V. L., et al. 2014, ApJ, 794, 97
Yang, H., Hare, J., & Kargaltsev, O. 2024, ApJ, 971, 180