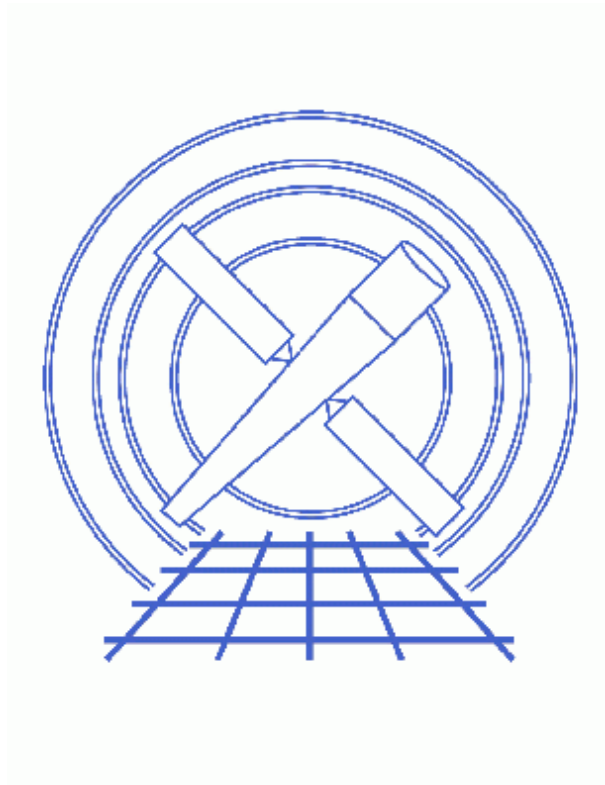


Using A Pileup Model



Sherpa Threads (CIAO 3.4)

Table of Contents

- **Background Information**
 - ◆ Remove the acis detect afterglow Correction
- **Getting Started**
- **Reading in Data & Instrument Responses**
- **Defining the Models**
 - ◆ Multi-Component Source Model
 - ◆ Pileup Model
- **Fitting with the Powell and Monte-Powell Optimization Methods**
 - ◆ Fit using the Powell method
 - ◆ Fit again using the Monte-Powell method
 - ◆ Calculate the parameter uncertainties
- **Examining the Pileup Fraction**
- **Saving the Fit Results**
- **History**
- **Images**
 - ◆ Source spectrum
 - ◆ Energy-filtered source spectrum
 - ◆ Plot of the fit and residuals

Using A Pileup Model

Sherpa Threads

Overview

Last Update: 1 Dec 2006 – reviewed for CIAO 3.4: no changes

Synopsis:

This thread describes how to include a pileup model in the model expression when fitting data in *Sherpa*. The [Background Information](#) section contains information on how the data should be processed in order to run this thread.

Related Links:

- [The Chandra ABC Guide to Pileup](#) (PS, 23 pages)
- [Why Topics on pileup](#)
- [A comparison, for low pileup fractions, of the pileup models in *Sherpa* and Xspec with that in ISIS.](#)

Proceed to the [HTML](#) or [hardcopy \(PDF: A4 | letter\)](#) version of the thread.

Background Information

Experience with the `jdpileup` model has chiefly been for on-axis point sources, and in this thread, this condition is assumed. Even when analyzing piled data from a point-source, keep in mind that events in the core of the Point Spread Function (PSF) may be severely piled up while events in the PSF wings may be essentially unpiled, so generally, the fraction of the measuring aperture deemed to be piled up (f model parameter) is less than 1.

The standard procedure for `jdpileup` is:

1. **Remove the `acis_detect_afterglow_correction`**, if applicable. [The next section](#) explains this step in more detail.
2. Reprocess the data with a [new bad pixel file](#) and apply the standard event filtering to create a level=2 event file; this is illustrated in the [Create a New Level=2 Event File](#) thread.
3. Extract a source spectrum from a circular region, usually about 2 arcsec in radius (e.g. by following the [psextract thread](#)).
4. Use the [pileup model in *Sherpa*](#) to fit a spectral model.

For a detailed discussion of the model, see [Davis \(2001\)](#).

Remove the `acis_detect_afterglow` Correction

An afterglow is the residual change from the interaction of a cosmic ray in a CCD. [Standard data processing](#) (SDP, aka "the pipeline") uses the tool `acis_detect_afterglow` to flag possible cosmic ray events in the level 1 event file; these are then filtered out in the level 2 event file. It has been determined, however, that 3–5 % of the valid source photons may be rejected from diffracted spectra. These rejections, though a small fraction of the total events, are systematic and non-uniform.

In order to accurately model the pileup, it is necessary to remove this correction so that no source photons are eliminated from the event file. Read the [Pileup Talk](#) for more information, in particular [Data Preparation and Caveats](#). The [Remove the `acis_detect_afterglow` correction thread](#) shows how to undo the afterglow filtering.

Important Note

A new, more precise method for identifying afterglow events was introduced to SDP at version DS 7.4.0. If your data was processed with DS 7.4.0 or later, `acis_detect_afterglow` was not run in the pipeline. *The new hot pixel tools are more judicious with respect to throwing away piled source events. Users should be able to analyze the data without having to unfilter events first.*

The [Get Started section](#) of the [Remove the `acis_detect_afterglow` correction thread](#) shows how to check the processing version used for your data.

Getting Started

Sample ObsID used: 1618 (ACIS-S, NGC 4258)

The files used in this example were created by following these [CIAO threads](#):

- [Extract ACIS Spectra for Pointlike Sources and Make RMFs and ARFs](#)
- [Grouping PHA Data before Fitting](#)

Here is a list of all the necessary files:

```
source_bin10.pi
background.pi
arf.fits
rmf.fits
```

The data files are available in [sherpa.tar.gz](#), as explained in the [Sherpa Getting Started thread](#).

Reading in Data & Instrument Responses

The spectra that will be used in this session have already been binned by a factor of 10. The dataset is input into *Sherpa* with the `DATA` command:

```
sherpa> data source_bin10.pi
```

Using A Pileup Model – Sherpa

```
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ ERRORS ">filename<[cols CHANNEL,STAT_ERR]" fitsbin
Background data are being input from:
/data/sherpa/pileup/thread/background.pi
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ BERRORS ">filename<[cols CHANNEL,STAT_ERR]" fitsbin
RMF is being input from:
/data/sherpa/pileup/thread/rmf.fits
ARF is being input from:
/data/sherpa/pileup/thread/arf.fits
```

Since the response files are defined in the header of `source_pha.fits`, the instrument response model is automatically established for use:

```
sherpa> show

..output omitted..

-----
Defined analysis model stacks:
-----

instrument source 1 = AutoReadResponse
instrument back 1 = AutoReadResponse

-----
Defined instrument model components:
-----

rsp1d[AutoReadResponse]
  Param   Type   Value
  -----  ---
  1   rmf string: "/data/sherpa/pileup/thread/rmf.fits" (N_E=1077,N_PHA=1024)
  2   arf string: "/data/sherpa/pileup/thread/arf.fits" (N_E=1077)
```

If this is not the case for your data, you will need to manually set the instrument model:


```
sherpa> instrument = rsp[instname](rmf.fits, arf.fits)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
```

The background is subtracted from the data, rather than fit simultaneously:

```
sherpa> subtract
```

The input dataset may be plotted:

```
sherpa> lp data
```

The data are plotted in energy space since the instrument model is providing the information necessary for the computation of the number of predicted counts in each detector bin. [Figure 1](#)  shows the resulting plot.

A filter may be applied to the data before proceeding:

```
sherpa> ignore source 1 all
sherpa> notice source 1 energy 1:7
sherpa> lp data
```

Remove the `acis_detect_afterglow` Correction

Note that the pileup correction will include the entire available energy information *regardless of the specified filter*. The fit statistics, however, are calculated only on the specified bins.

Figure 2.6 shows the filtered data, which has an energy range of 1 – 7 keV. See the [Choosing an Energy Filter](#) why topic for help in picking a range.

Defining the Models

Multi-Component Source Model

Since we have background-subtracted the data (rather than fitting it simultaneously), it is only necessary to create a source model expression. We model this source with a power law ([xspowerlaw](#)) absorbed by the interstellar medium ([xswabs](#)).

First, we set up each model component. The absorption model will be referred to as "abs", and the broken power law will be "power".

```
sherpa> xswabs[abs]
abs.nH parameter value [0.1]

sherpa> xspowerlaw[power]
power.PhIndx parameter value [1]
power.norm parameter value [0.000167764]
```

Note that since a dataset has already been input, *Sherpa* estimates the initial parameter values for the models based on the data. If you know the Galactic column density for your source, you may set `w.nh` and freeze it:

```
sherpa> w.nh = <value>
sherpa> freeze w.nh
```

In this thread, we allow it to vary during the fit.

Now that the model components have been established, the product of the two is assigned as the source model for the dataset:

```
sherpa> source 1 = (abs * power)

sherpa> show source
Source 1: (abs * power)
xswabs[abs] (XSPEC model name: wabs) (integrate: off)
  Param  Type      Value      Min      Max      Units
  ----  -
  1     nH thawed    1e-01    1e-07    10      10^22/cm^2
xspowerlaw[power] (XSPEC model name: powerlaw) (integrate: on)
  Param  Type      Value      Min      Max      Units
  ----  -
  1PhIndx thawed    1         -3       10
  2     norm thawed 1.6776e-04 1.6776e-06 1.6776e-02 photons/keV/cm**2/s at 1 keV
```

Pileup Model

It is also necessary to define the pileup model (`jdpileup`):

```
sherpa> pileup = jdpileup[jdp]
jdp.alpha parameter value [0.5]
jdp.g0 parameter value [1]
jdp.f parameter value [0.95]
jdp.n parameter value [1]
jdp.ftime parameter value [3.241]
jdp.fracexp parameter value [0.987]

sherpa> show jdp
jdpileup[jdp] (integrate: off)
  Param  Type      Value      Min      Max      Units
  -----
  1 alpha thawed    0.5       0       1
  2   g0 frozen     1       1e-120  1
  3    f thawed    0.95      0.9     1
  4    n frozen     1       1e-120  100
  5 ftime frozen    3.241     1e-120  5      sec
  6 fracexp frozen  0.987     0       1
```

Read carefully the following information on how to set the pileup model parameters:

- ***alpha***

`alpha` parameterizes "grade migration" in the detector, and represents the probability, per photon count greater than one, that the piled event is not rejected by the spacecraft software as a "bad event". Specifically, if `n` photons are piled together in a single frame, the probability of them being retained (as a single photon event with their summed energy) is given by $\alpha^{(n-1)}$.

alpha is the parameter most likely to be allowed to vary in a fit.

- ***g0:***

Grade correction for single photon detection, i.e. a fraction `g0` of single photon events will be retained as good grades.

In practice, this should be frozen to unity (the default) in any fit.

- ***f:***

The fraction of events in the source extraction region to which pileup will be applied. A typical value is approximately 0.95, and it should always be > 0.85 . `f` is the second most likely parameter, after `alpha`, to be allowed to vary in a fit.

It is recommended that the lower limit of this parameter be set to 0.85 before fitting (the default is 0.9):

```
sherpa> jdp.f.min=0.85
```

- ***n:***

Divide the model counts among `n` regions, to which the pileup model will be applied independently. This should be approximately the number of 3x3 pixel islands in the extracted spectra; for point sources, it is unity (the default).

Using A Pileup Model – Sherpa

This should remain a frozen parameter in any fit.

- ***ftime***

The frame time parameter (*ftime*) should be set to the good exposure time per frame, which is recorded in the header keyword *EXPTIME* of the *event file*. (Note that *EXPTIME* is used instead of *TIMEDEL* because the latter includes the transfer time, which *ftime* should not.)

The CIAO command may be executed from within *Sherpa*, as long as it is escaped with the dollar sign (\$):

```
sherpa> $dmkeypar acisf01618_evt2.fits EXPTIME echo+
3.2

sherpa> jdp.ftime=3.2
```

- ***fracexp***

This parameter is a fraction ≤ 1 that multiplies the frame exposure time to create a shorter, effective frame exposure time.

Users should set and freeze *fracexp* to unity, and set the frame exposure time via the *ftime* parameter only:

```
sherpa> jdp.fracexp=1
```

Information on using this parameter to model novel deadtime effects is available in the [The Chandra ABC Guide to Pileup](#).

The pileup model parameters are now set as shown:

```
sherpa> show jdp
jdpileup[jdp] (integrate: off)
  Param  Type      Value      Min      Max      Units
  -----
1 alpha thawed    0.5        0        1
2 g0 frozen      1          1e-120   1
3 f thawed      0.95       0.85     1
4 n frozen      1          1e-120  100
5 ftime frozen   3.2       1e-120   5      sec
6fracexp frozen   1          0         1
```

Fitting with the Powell and Monte–Powell Optimization Methods

When fitting data, it is possible to use a combination of the [montecarlo](#) and [powell optimization methods](#) to find the minimum, running a series of fits that alternate between the methods. In *Sherpa* there is a method called [monte-powell](#) which does these steps for you.

Monte–Powell is much more likely to find a global minimum for pile–up, but takes a *much* longer time. Depending on the complexity of the model and data, the fit can run for as long as 24 hours. If you have the time to let the fit run, Monte–Powell is the recommended method for performing this fitting. Otherwise, use a simpler method to find a local minimum, such as the Powell optimization method.

Using A Pileup Model – Sherpa

Here the fit is first run with the Powell method, then refined by switching to Monte–Powell and refitting.

Fit using the Powell method

As mentioned above, using the Monte–Powell method is effective, but may take a long time for the fit to find a global minimum for pileup. First we use the Powell method, which will speed up the fit, but may require some "hands–on" work afterwards.

We set the optimization to Powell and start the fit:

```
sherpa> method powell
sherpa> fit
powll: v1.2
powll:  initial statistic value =      4.28866E+03
powll:   converged to minimum =      5.36253E+01 at iteration =      45
powll:  final statistic value =      5.36253E+01
      abs.nH  6.13085  10^22/cm^2
      power.PhoIndx  1.42201
      power.norm  0.00194221  photons/keV/cm**2/s at 1 keV
      jdp.alpha  0.550389
      jdp.f  0.9057
```

By using the Powell method, we are settling for finding a *local* minimum; now one should examine the solution, check the error bars, find a new minimum, refit, etc. Doing an hour of work of this sort of work may be more useful to the user than waiting for a Monte–Powell fit to converge.

Fit again using the Monte–Powell method

Now we change the method to Monte–Powell and re–run the fit to see how the results change. Note that we start at the results obtained by Powell, but you could reset the model parameters before continuing, if desired.

The following fit took approximately 17 hours to run on a SunBlade100 (384 MB main memory):

```
sherpa> method monte-powell
sherpa> fit
monpo: v1.2
monpo:  initial statistic value =      5.36253E+01
powll: v1.2
powll:  initial statistic value =      7.76375E+05
powll:   converged to minimum =      5.36253E+01 at iteration =      31
powll:  final statistic value =      5.36253E+01
powll: v1.2
powll:  initial statistic value =      2.12974E+03

... (output omitted) ...
powll: v1.2
powll:  initial statistic value =      2.72390E+05
powll:   converged to minimum =      5.36256E+01 at iteration =      37
powll:  final statistic value =      5.36256E+01
monpo:  final statistic value =      5.36251E+01
      abs.nH  6.13484  10^22/cm^2
      power.PhoIndx  1.42369
      power.norm  0.00261819  photons/keV/cm**2/s at 1 keV
      jdp.alpha  0.371895
      jdp.f  0.9478
```

This table compares the results from the two runs with different optimization methods:

Using A Pileup Model – Sherpa

parameter	powell	monte-powell
abs.nH	6.13085	6.13484
power.PhoIndx	1.42201	1.42369
power.norm	0.00194221	0.00261819
jdp.alpha	0.550389	0.371895
jdp.f	0.9057	0.9478

Calculate the parameter uncertainties

At this point, we continue with the fit results obtained by using the Monte-Powell method. For the best-fit values to really mean something, we need to find the uncertainties on the parameters. The `uncertainty`, `covariance`, and `projection` commands can be used to estimate confidence intervals for the thawed parameters:


```
sherpa> projection
Projection: optimization reset to LM.
Projection complete for parameter: abs.nH
Projection complete for parameter: power.PhoIndx
Projection complete for parameter: power.norm
WARNING: upper projection bound not found for jdp.alpha
Projection complete for parameter: jdp.alpha
Projection complete for parameter: jdp.f

Computed for sherpa.proj.sigma = 1
-----
Parameter Name      Best-Fit Lower Bound   Upper Bound
-----
abs.nH              6.13484 -0.647054            +0.683536
power.PhoIndx       1.42369 -0.265149            +0.270204
power.norm          0.00261819 -0.00154002          +0.0127003
jdp.alpha           0.371895 -0.225414            +0.628105
jdp.f               0.9478 -0.254523            +0.0141215
```

The table gives the same best-fit values as before, as well as providing upper and lower bounds on each of them. The warning ("upper projection bound not found") may be ignored and is explained in [this FAQ](#).

Finally, plot the fit and the residuals of the fit:

```
sherpa> lp 2 fit delchi
```

The final plot is show in [Figure 3](#) .

Examining the Pileup Fraction

It is now possible to see what fraction of the events are calculated as piled:

```
sherpa> show pileup
1: 0.365245 0.789452
```

Using A Pileup Model – Sherpa

```
2: 0.2054 0.179825
3: 0.0770064 0.0273074
4: 0.0216528 0.0031101
5: 0.00487069 0.000283373
6: 0.000913032 2.15159e-05
7: 0.000146702 1.40028e-06
*** pileup fraction: 0.210548
```

This command prints the pileup fractions from the most recent fit. From left to right, the columns report the number of piled photons, percentage of that number of photons in the pileup region, and percentage of that number of piled photons in the total events. The fraction of pileup in the observation is printed after the breakdown. The next paragraph analyzes the above results, which should clarify this explanation.

In this example, the first row indicates that 36.5% of the frames contained a single photon in the pileup region and 78.9% of the events were single-photon events. The second row shows that 20.5% of the frames contained 2 photons in the pileup region and 18.0% of the events were due to 2 photons. And so forth. For this observation, the total pileup fraction is 21.1%.

Saving the Fit Results

Before exiting *Sherpa*, save the fit results to a file:

```
sherpa> save all pileup_fit.shp
```

These results may then be restored in a later session with the use command.

History

17 Jan 2005 updated for CIAO 3.2: added [Changes in CIAO 3.2 section](#)

05 Jul 2005 added links to new why topics and pileup manual

11 Jul 2005 thread updated to use new data (ObsID 1618, NGC 4258)

21 Dec 2005 reviewed for CIAO 3.3: no changes

01 Dec 2006 reviewed for CIAO 3.4: no changes

URL: <http://cxc.harvard.edu/sherpa/threads/pileup/>

Last modified: 1 Dec 2006

Image 1: Source spectrum

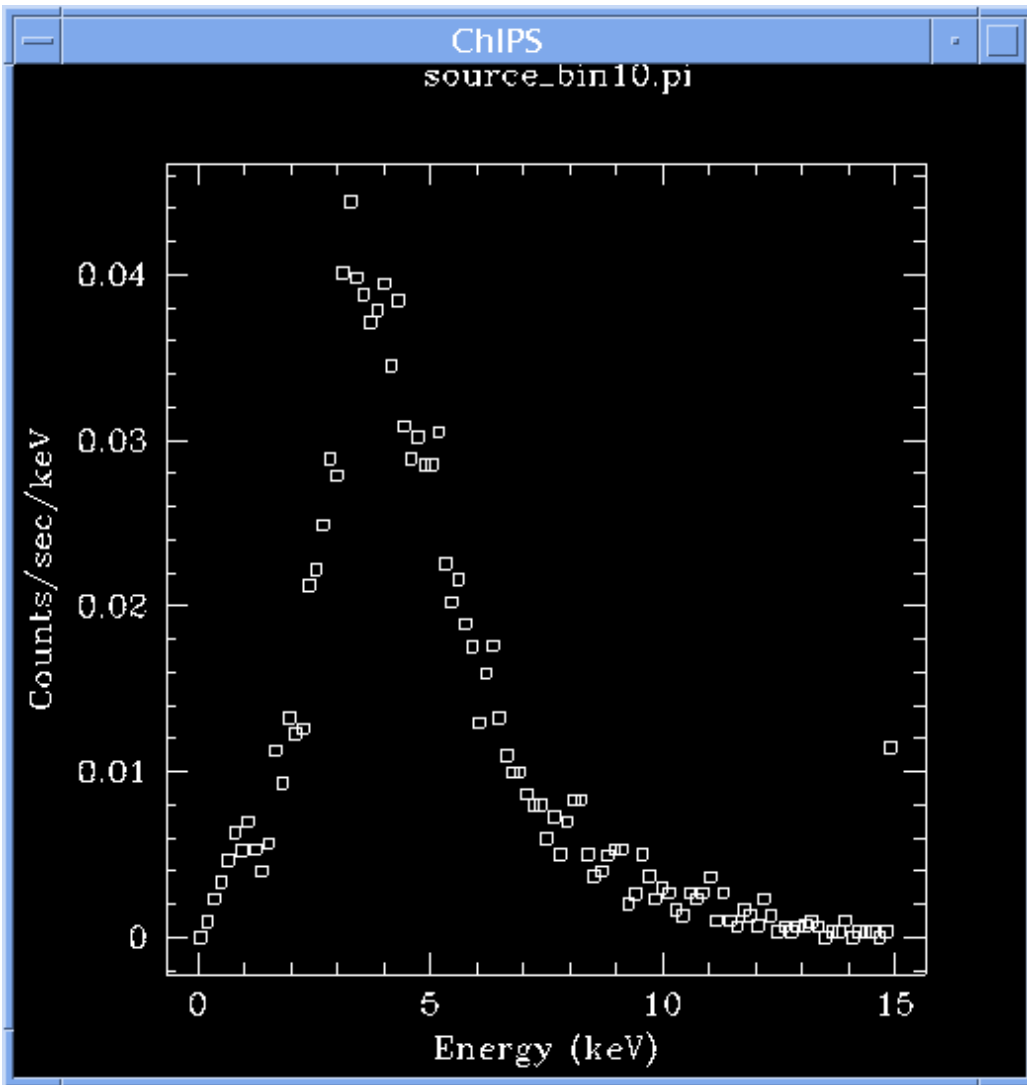


Image 2: Energy-filtered source spectrum

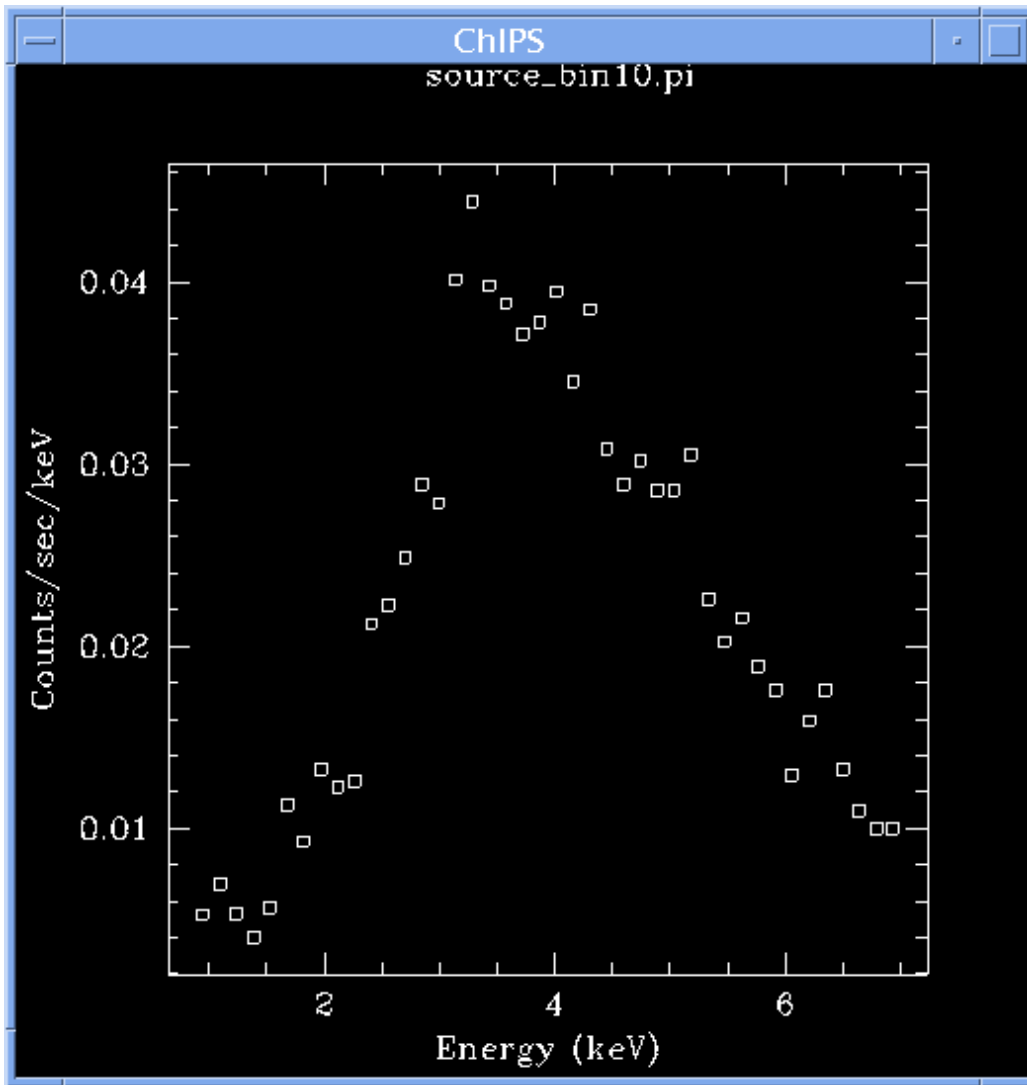


Image 3: Plot of the fit and residuals

